Structured convolutional networks

Jacobsen, J.H.

Publication date
2018
Document Version
Final published version
License
Other

Structured Convolutional Networks

by Jörn-Henrik Jacobsen

Structured Convolutional Networks

Jörn-Henrik Jacobsen

This book was typeset by the author using latex.

Cover art: Martin Grosz - ”LUX” (Courtesy of the artist)
Printed by: Ridderprint BV — www.ridderprint.nl

ISBN: 978-94-6375-032-5

Structured Convolutional Networks

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties
ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op dinsdag 3 juli 2018 te 10:00 uur

door

Jörn-Henrik Jacobsen

geboren te Braunschweig, Bondsrespubliek Duitsland

Promotiecommissie

Promotor:	Prof. dr. ir. A. W. M. Smeulders	Universiteit van Amsterdam
Co-promotor:	Prof. dr. C. G. M. Snoek	Universiteit van Amsterdam
Overige leden:	Prof. dr. M. Bethge	Eberhard Karls Universität Tübingen
	Prof. dr. R. Turner	Universiteit van Amsterdam
	Prof. dr. M. Welling	Universiteit van Amsterdam
	Dr. A. Habibian	Qualcomm
	Dr. T.E.J. Mensink	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The work described in this thesis has been carried out at the Intelligent Sensory Information Systems lab of the University of Amsterdam. The research is supported by the STW ImaGene project.

CONTENTS

1 INTRODUCTION

In this chapter, we introduce the main ideas underlying this thesis, raise the 4 main research questions and introduce the core mathematical concepts.

1.1 TOWARDS A NEW APPROACH TO SCIENCE

At the core of our existence, we are basically clueless. We have no idea where we come from, why our universe came to life, and how some of the most intriguing aspects of our world work. Heidegger calls this state of affairs "das geworfen sein", the state of being thrown into existence. After all, no one asked us if we want to be alive and no one told us how life works, but yet here we are. To get a grip on this most fundamental of all issues, humanity developed the scientific method to shed light on the workings and meaning of life, or to be more precise: to help us rule out conceptual mistakes about how the universe functions. By construction, the scientific method is limited through the quality of experimental or theoretical questions one can ask. To increase the precision of such questions, science is mostly concerned with reducing problems to their essence and modeling them explicitly in this reduced complexity space. However, many phenomena are so entangled and non-deterministic, it is questionable if they can ever be understood with the help of reductionist mechanistic models of the world.

From this perspective, surrendering and giving up hope to fully understand dynamics of complicated non-linear systems like the human brain, embodied agents interacting with the real world, or large-scale social systems seems perfectly sensible. Indeed, recent advances in artificial intelligence have empirically shown exceptional performance of tools that embrace the fact that it is hopeless to understand some of the most interesting problems in their full complexity from first principles. There has been a transition in empirical sciences and engineering from hard-coded heuristics, like Bag-of-Words in Computer Vision, GLM-based approaches in Neuroimage Analysis and logic-based rules in natural language processing, to fully learned generic systems in the form of Deep Neural Networks, constituting the state-of-the-art in many problems to date.

Deep Neural Networks learn the heuristics to reduce the complexity of a problem by themselves, "all" that is needed is a suitable architectural prior and a large dataset that sufficiently describes the problem of interest.

Thus, we shift the problem of identifying our desired reduced complexity space to a learning algorithm that can efficiently approximate a good solution and automatically find a good trade-off between generalization and simplification. It has been empirically shown that Deep Networks have the power to capture highly non-linear, hierarchical interactions between extremely complex variations in the data, while having a relatively simple structure in their low-level building blocks. However, even though they are seemingly simple, a large network of such basic elements is not simple anymore as it can, in essence, learn any function. Thus, by replacing the tedious, and often even impossible design of rules to solve complex real-world problems with a learner, we end up with a new problem: A superstar in high-dimensional problem-solving that is an entire black-box.

Understanding how and what these algorithms learn from data remains a challenge. Overcoming this major disadvantage of current deep neural networks would open up the possibility to have these algorithms reveal their heuristics to us. Imagine an algorithm that can be trained to classify MRI brain scans of elderly humans as healthy or Alzheimer's diseased from a small dataset and not only get the guarantee of robust predictions but also being able to extract the anatomical model identified by the algorithm that causes the prediction of the disease. We would have found an algorithm that not only robustly predicts which subject is diseased, we would also get a model hypothesis about what Alzheimer's actually is. To achieve this, we need a new breed of structured deep networks that allow to open up the black-box, overcome data-inefficiency and obtain robust predictions. Even though there is still a long way to go to establish robust learning algorithm as an alternative to scientific methods, there is a lot of work done in this direction and once we get there, it might possibly change the toolbox of experimental scientists fundamentally and will be tremendously helpful to obtain novel empirical insights in many relevant areas.

In this thesis, we introduce various flavors of structured deep networks, that aim to shed light on different aspects of the learned models. The chapters are steps towards structured Deep Neural Networks, whose inner workings can be easier understood to reveal the structure of the learned representations, introduce new anchor points for regularization and improve data-efficiency in limited sample scenarios.

1.2 DEEP CONVOLUTIONAL NEURAL NETWORKS

Deep convolutional neural networks are amongst the most successful types of deep learning algorithms and the ones this thesis focuses on. They are the current state-of-the-art for many classification and regression tasks. Beyond their superior performance on tasks they have been explicitly trained for, they have also been shown to perform very well in transfer-learning scenarios, substantiating the generic nature of their learned representations. Domains they excel in include images, audio, medical images, chemistry data, physics data and the range of applications is constantly expanding [1]. However, their mathematical properties remain mysterious and there is little understanding of how their learned structure relates to properties of the problems they have been trained on [2], [3]. Understanding the regularities of the data these algorithms exploit and how they discover them during learning to achieve their excellent generalization, is thus of great scientific interest.

In this thesis, we focus on the setting of classifying natural images with convolutional networks, as an example of a domain where deep networks excel. The goal of the classification task is to learn a mapping from high-dimensional inputs $x_i \in \mathcal{R}^D$ to low-dimensional outputs $y_i \in \mathcal{R}^d$, where $D >> d$. Classification thus requires to eliminate a lot of non-informative variabilities, and hence to contract space in appropriate directions. Natural images are high-dimensional signals that exhibit many complex variabilities while mathematical tools are mostly limited to the case of low-dimensional geometry. However, the directions to contract can be very high dimensional, are typically unknown a priori and thus have to be learned from data. This is done by optimizing the convolutional operators W_j of a J-layer network to minimize a task-specific loss function:

$$\Phi_J(x) = \rho W_J(...(\rho W_j(...\rho W_1(x_0)))). \tag{1}$$

ρ is a point-wise non-linearity, e.g. ReLU $\triangleq max(x, 0)$ and operators W_j convolutions of n-dimensional signals $x(u, v)$ with translation variable u and channel index v with filters $w(u, v) \in S$, where S is the filters support, defined as:

$$W_{j+1}x_j = \sum_v \sum_{u' \in S} w_{j+1}(u', v)x_j(u - u', v). \tag{2}$$

For natural image classification, u is 2-dimensional and v 1-dimensional. The weights w_j are often trained by minimizing the cross-entropy between the ground truth label distribution and the estimated softmax probability distribution:

$$\mathcal{L}(f_0(\Phi_J(x_i)), y_i) = -\sum_i y_i log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}). \tag{3}$$

Here $f_0(.)$ is a final linear projection onto the logits s_j and a subsequent softmax function.

The variabilities to be reduced by this mapping can be defined as the symmetries of the classification task [3]. The transformation $\mathcal{T}$ is said to be a symmetry of the classification function $f : \mathbb{R}^D \to \mathbb{R}^d$, if it preserves the label for all $x_i \in X$:

$$f(x_i) = f(\mathcal{T}x_i). \tag{4}$$

We informally partition the set of symmetries of the classification task into two categories:

$\mathcal{T}_G$: Well-understood transformation groups. To some extent, they can be tackled by imposing structure on the learned representation a priori.

$\mathcal{T}_A$: Transformations that are not well-defined and hard to estimate. They lack mathematical structure that can be tackled efficiently and have to be learned from data.

Chapter 3 and 4 focus on strategies to incorporate knowledge about $\mathcal{T}_G$ into deep networks. Chapter 5 focuses on structuring $\mathcal{T}_A$ by approximating them with known transformations. And finally, chapter 6 answers the question if invariance with respect to any $\mathcal{T}$ has to be built progressively, or if powerful deep representations can be learned that do not discard any information about the data and thus build no global invariance up until the very last layer.

1.3 THE LOCAL STRUCTURE OF IMAGES

CNNs are designed to exploit translation invariance of locally stationary signals which natural images are prime examples of. This is one case of a well-understood symmetry $\mathcal{T}_G$ of the classification task. A locally translated image of a cat is still an image of a cat and so is a blurred or slightly rotated version of it. More generally, we raise research question 1.

RESEARCH QUESTION 1: Can we encode the structure of natural images into CNNs by design?

In chapter 3 we increase the amount of local symmetries the network can easily build invariance against. We do so by equipping each layer of a CNN with a geometrically meaningful front-end. Inspired by scale-space theory [4], this front-end is a content-agnostic operator designed to respect and emphasize key properties of natural images in the following sense:

NO LOCATION PREFERENCE: Convolutional operator

NO SCALE PREFERENCE: Multi-scale representation

NO ORIENTATION PREFERENCE: Steerability with respect to rotation

GEOMETRICALLY MEANINGFUL: Reveals complete set of differential invariants

It turns out, that the multi-scale Njet [5] obeys all the above properties. It is the local Taylor expansion up to Nth order at each location of the image, computed on multiple Gaussian-smoothed scales $\{\sigma_s = 2^s\}_{s=0}^{S}$:

$$\text{Njet}_s x \triangleq \{G_{s_1,\ldots,s_n}(u, \sigma_s) * x = \frac{\partial_n G(u, \sigma_s)}{\partial u} * x\}_{n=0}^{N}. \tag{5}$$

Note, that here observation and differentiation are the same operation, expressed as convolution with a Gaussian derivative of order n and scale σ_s. The choice of an Njet is well-validated. It has its roots in the early days of visual perception research [4] and has been extensively used in former state-of-the-art computer vision systems, such as SIFT [6].

The front-end is incorporated into deep convolutional networks by removing the standard pixel basis and expressing each individual filter w_j in every layer as a weighted linear combination over the multi-scale Njet coefficients:

$$w_j(u) = \sum_{i=0}^{S \times N} c_i \text{Njet}_i(u), \tag{6}$$

where the weights c_i are learned via backpropagation. We call such CNNs "Structured Receptive Field Networks" (RFNNs), in honor of Koenderinks work on receptive field families and the Njet. In chapter 3, we show how this multi-scale representation lends itself well to re-structure a deep CNN in a principled way and is easily extended to higher dimensional images. We also show, that RFNNs obtain state-of-the-art results in various limited data scenarios, outperforming strong baselines like the Scattering network. This work was the first to successfully connect handcrafted basis parametrizations with learned CNNs without loss of generality. The proposed framework makes it straightforward to achieve rotation- or approximately scale-invariant layers with subpixel accuracy and similar parametrizations have recently been proven useful in equivariant networks [7], [8] as well.

1.4 FRAME-BASED CNNS AND STEERABILITY

In chapter 4 we formalize the idea of alternative CNN parametrizations and their transformation properties. The multi-scale Njet is not a basis in the classical sense, as its coefficients are overcomplete and non-orthogonal. Thus, we need a more general notion of spanning sets that allows for overcompleteness and non-orthogonality.

RESEARCH QUESTION 2: Can we generalize alternative parametrizations and exploit their transformation properties explicitly?

In the signal processing literature [9], the natural generalization of a basis of a vector space is called a frame. In frame terminology, an orthonormal basis is a Parseval-tight frame with unit norm. Every *tight frame* preserves the norm and exhibits perfect reconstruction. Frames can be seen as a superset of orthogonal bases in the sense that every basis is a frame, but not the reverse.

Any finite spanning set of vectors $V = \{\phi_n\}_{n=1}^{N}$ in a Hilbert space $\mathbb{H}$ is a frame. This means there must exist two constants $A, B > 0$, such that for all x in $\mathbb{H}$:

$$A||x||^2 \leq \sum_{n=1}^{N} |\langle x, \phi_n \rangle|^2 \leq B||x||^2. \tag{7}$$

If $A = B$, the frame is a tight frame and if $A = B = 1$, it is a Parseval tight frame. Frames allow us to represent signals in another domain where salient properties can be emphasized more. Further, it gives us the right framework to make sure we preserve the norm of the signals. For instance, by moving A into the sum, any tight frame can be converted into a Parseval tight frame with unit norm.

We show that various spanning sets vary significantly in their performance when used as an alternative parametrization in CNNs. At the same time, parametrizations that emphasize key aspects of the data perform superior to the naive pixel basis. If the chosen frame is steerable [10], [11], it also enables us to separate a features pose from its canonical appearance. Steering an arbitrary filter $w(u, v)$ under a k-parameter Lie group $\mathcal{T}_G$, for the transformation $g(\tau) \in \mathcal{T}_G$, can be written as follows:

$$g(\tau)w(u, v) = \sum_{n=1}^{N} c_n g(\tau) e_n = \sum_{m=1}^{M} b_m \alpha_m(\theta) \phi_m. \tag{8}$$

Thus, it is sufficient to first transform the frame at each location of the input and apply the weights of the canonical feature afterwards. This *disentangles* the canonical filter from its transformed k-parameter variants, i.e. c_n and b_m respectively govern the weight of each coefficient to form a filter $w(u, v)$ and $\alpha_m(\theta)$ are the steering functions governing the transformation of $g(\tau)$ acting on $w(u, v)$ as a whole. As frames may be overcomplete, a frame-parametrized filter can potentially have $m > n$ coefficients. This means transforming features amounts to a point-wise multiplication of frame coefficients with cos, sin and exp activation functions, which is suitable for learning in a CNN.

The steering equations that determine α_m can be derived a priori, based on the frame of choice by solving a linear system determined by the infinitesimal generator and applying the exponential map. However, the pose parameters θ_k either have to be pre-defined based on some heuristic or estimated from data. To avoid such heuristics, we learn a small neural network that dynamically estimates a vector of pose parameters at each location, conditioned on the input:

$$\theta(u, v) = \text{PoseNet}(x(u, v)). \tag{9}$$

This pose estimation network can be understood as a learned pooling function, which does data-dependent pooling on the orbit of poses, conditioned on the input. The depth of this network determines the amount of local context around the filtered image patch we want to include into the estimation, an important hyper-parameter as we show in chapter 4. Disentangling pose and canonical appearance this way gives rise to deep networks with the ability to dynamically adapt their filters under pre-defined transformations. It also provides a simple way to regularize the pose variables to obtain local invariants.

In summary, we introduced frame-based CNNs, that give us means to explicitly encode knowledge about salient signal properties and priors about well-understood symmetries $\mathcal{T}_G$ of the classification problem into generically learned CNNs. Having pose as an explicit parameter in the network allows us to disentangle pose and appearance of each filter and pre-define structures in the CNN to learn geometrically meaningful representations from fewer data and dedicate parameters explicitly to geometric tasks (e.g. by incorporating them into the PoseNet).

1.5 STRUCTURING THE CHANNEL DOMAIN OF CNNS

In chapter 3 and chapter 4 we have structured and exploited spatial regularities of the classification problem. As explained earlier, this can be done if we understand the transformations $\mathcal{T}_G$ we want to encode into our network well. However, in most cases we also deal with transformations $\mathcal{T}_A$ that are hard to describe mathematically and not known a priori. In the following we will highlight the role of the channel dimension in CNNs for such transformations and discuss methods to tackle the third research question.

RESEARCH QUESTION 3: Can we structure and reveal unknown symmetries of the classification problem?

[3] introduces multi-scale hierarchical convolutional networks, a novel class of deep networks. They approximate the symmetries of the classification task with factorized groups of symmetries who's size grows with depth while the network progressively builds larger local invariants. The one-dimensional channel index v of a vanilla CNN is replaced by a multidimensional vector of attribute indices $v = (u, v_1, ..., v_j)$ and every layers linear operators W_j are generalized convolutions along (u, v). Each v_j represents an attribute of the data and indexes a finite-dimensional Lie group, potentially sampled in discrete intervals. A parallel transport is defined as an action $g_j \in \mathcal{T}_A$ along the index space (u, v) of x_j [3]:

$$\forall v_j \in v,\ g_j.x_j(v_j) \triangleq x_j(g_j.v_j). \tag{10}$$

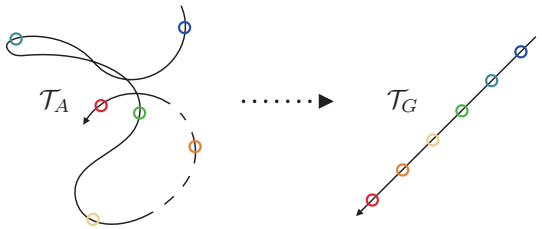


Figure 1: Hierarchical Attribute CNNs learn a function that locally linearizes unknown non-linear transformations $\mathcal{T}_A$ and approximates them with euclidean translations $\mathcal{T}_G$ that can be structured and analyzed much easier.

The dimensionality of local symmetries H_j is growing with depth and can be factorized as follows:

$$\forall j \geq 0,\ G_j = G_{j-1} \rtimes H_j. \tag{11}$$

In chapter 5 we show that the operators W_j can be implemented via linear multi-dimensional convolutions. The group of symmetry we employ to approximate the symmetries of the classification problem, are n-dimensional Euclidean translations $G_j = \mathbb{R}^d,\ d \in \mathbb{N}$. Thus, G_j factorizes as:

$$\forall j \geq 0,\ G_j = \mathbb{R}^{d_{j-1}} \rtimes \mathbb{R}^{d_j}. \tag{12}$$

This can be seen as a special case of the general multi-scale hierarchical networks introduced in [3], as it does not permit to embed non-commutative groups without additional effort. To allow invariance with respect to non-commutative groups, we integrate earlier attribute dimensions out and thus create explicit invariants with respect to all attribute transformations except the last three. Further, this helps to keep the number of parameters stable, which would otherwise grow exponentially. The layer $j+1$ is thus computed via multi-dimensional convolution operators W_{j+1} as follows:

$$W_{j+1}x_j = \int \sum_{u',v'_j,v'_{j-1} \in S} w_j(u', v'_j, v'_{j-1}) x_j(u - u', v_j - v'_j, v_{j-1} - v'_{j-1}, v_{j-2})\ dv_{j-2}. \tag{13}$$

Hierarchical Attribute CNNs aim to locally linearize the unknown symmetries $\mathcal{T}_A$ and map them to well-defined Euclidean translations. This corresponds to mapping possibly high-dimensional curved manifolds to flat multi-dimensional translations, as illustrated in figure 1 for a 1-dimensional curved manifold-like structure. We propose an efficient training algorithm for such networks, analyze the learned translations and show that Hierarchical Attribute CNNs learn highly structured representations with an order of magnitude fewer parameters compared to vanilla CNNs.

1.6 DEEP INVERTIBLE NETWORKS

The previous research questions have focused on building and analyzing flexible discriminative invariants efficiently. In contrast, chapter 6 asks the question if it is necessary at all to progressively build invariance and discard seemingly un-informative variability about the input signal.

RESEARCH QUESTION 4: Is invariance and loss of information a necessity for deep networks to generalize well?

Several lines of work suggest that progressive loss of information and invariance are essential properties of representations that generalize well to unseen data [12], [13]. And the difficulty to recover inputs from hidden representations in many common deep network architectures [14], [15], [16] supports this claim.

In chapter 6 we show that this loss of information is indeed just a sufficient, but not a necessary condition to learn powerful representations. We do so, by designing an invertible model that ensures, that the mutual information between representations is constantly equal to one for any depth:

$$I(x_i, x_j) = 1, \ \forall i, j \leq J. \tag{14}$$

To avoid trivial invertibility, we design a so-called *i*-RevNet. It is a cascade of homeomorphic layers and thus ensures that the model can not just copy the signal and store one part for a perfect reconstruction and work on the other part for good classification accuracy. In a bijective net, such a duplication is not possible by definition as we have a unique 1-to-1 mapping of the form:

$$f(x) = y \ \Leftrightarrow \ f^{-1}(y) = x. \tag{15}$$

Typically such an inverse is ill-defined, as the condition number of trained neural networks is large. Thus, a small change in the input can cause a large change in the output and vice versa. In an *i*-RevNet we define the inverse explicitly, inspired by work on RevNets, Real-NVP and NICE [17], [18], [19]. *i*-RevNets are the first fully-invertible deep network, that performs well on large-scale problems like Imagenet.

However, to achieve good generalization performance, the network should progressively separate and contract the data with appropriate change of variables [3]. A deep network that generalizes well should contract the inner-class distances:

$$||\Phi_j(x_i) - \Phi_j(x_k)|| \leq ||x_i - x_k||, \text{ if } f(x_i) = f(x_k), \tag{16}$$

and at the same time separate the classes by at least an ϵ-margin to not collapse them into one another:

$$\exists \epsilon > 0, f(x_i) \neq f(x_j) \Rightarrow ||\Phi_j(x_i) - \Phi_j(x_j)|| > \epsilon. \tag{17}$$

We empirically identify such a contraction and separation phenomenon in *i*-RevNets trained on Imagenet and propose to explain the good generalization performance with these progressive properties. Our results illustrate, that the Euclidean metric becomes progressively more meaningful with depth and that deep networks can build progressively more task-related representations without discarding any information.

2 LIST OF PUBLICATIONS

Jörn-Henrik Jacobsen, Jan van Gemert, Zhongyou Lou, and Arnold W.M. Smeulders. "Structured Receptive Fields in CNNs" In Proceedings of CVPR, 2016.

Jörn-Henrik Jacobsen, Bert de Brabandere, and Arnold W.M. Smeulders. "Dynamic Steerable Blocks in Deep Residual Networks" In Proceedings of BMVC, 2017.

Jörn-Henrik Jacobsen, Edouard Oyallon, Stéphane Mallat, and Arnold W.M. Smeulders. "Hierarchical Attribute CNNs" In Workshop for Principled Approaches to Deep Learning ICML, 2017.

Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. "*i*-RevNet: Deep Invertible Networks" In Proceedings of ICLR, 2018.

Jörn-Henrik Jacobsen has contributed to all aspects of the above publications. Arnold Smeulders has provided insight and supervision for all publications above. Jan van Gemert has provided guidance and advice and Zhongyou Lou technical advice in the publication about "Structured Receptive Fields in CNNs". Bert de Brabandere has provided technical guidance in the publication about "Dymamic Steerable Block in Deep Residual Networks". Stéphane Mallat and Edouard Oyallon have provided guidance and advice in the publication about "Hierarchical Attribute CNNs". Edouard Oyallon has provided guidance and advice in the publication about "*i*-RevNet: Deep Invertible Networks".

3 STRUCTURED RECEPTIVE FIELDS IN CNNS

3.1 INTRODUCTION

Where convolutional networks have appeared enormously powerful in the classification of images when ample data are available [20], we focus on smaller image datasets. We propose structuring receptive fields in CNNs as linear combinations of basis functions to train them with fewer image data.

The common approach to smaller datasets is to perform pre-training on a large dataset, usually ImageNet [21]. Where CNNs generalize well to domains similar to the domain where the pre-training came from [22], [23], the performance decreases significantly when moving away from the pre-training domain [23], [24]. We aim to make learning more effective for smaller sets by restricting CNNs parameter spaces. Since *all images* are spatially coherent and human observers are considered to only cast local variations up to a certain order as meaningful [4], [25] our key assumption is that it is unnecessary to learn these properties in the network. When visualizing the intermediate layers of a trained network, see e.g. [26] and Figure 3, it becomes evident that the filters as learned in a CNN are locally coherent and as a consequence can be decomposed into a smooth compact filter basis [27].

We aim to maintain the CNN's capacity to learn general variances and invariances in arbitrary images. Following from our assumptions, the demand is posed on the filter set that i) a linear combination of a finite basis set is capable of forming any arbitrary filter necessary for the task at hand, as illustrated in Figure 2 and ii) that we preserve the full learning capacity of the network. For i) we choose the family of Gaussian filters and its smooth derivatives for

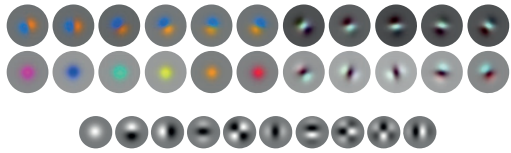


Figure 2: A subset of filters of the first structured receptive field CNN layer as trained on 100-class ILSVRC2012 and the Gaussian derivative basis they are learned from. The network learns scaled and rotated versions of zero, first, second and third order filters. Furthermore, the filters learn to recombine the different input color channels which is a crucial property of CNNs.

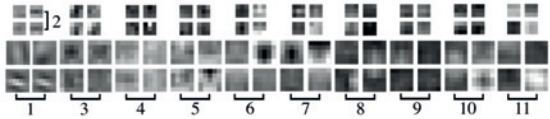


Figure 3: Filters randomly sampled from all layers of the GoogLenet model [29], from left to right layer number increases. Without being forced to do so, the model exhibits spatial coherence (seen as smooth functions almost everywhere) after being trained on ILSVRC2012. This behaviour reflects the spatial coherence of the input feature maps even in the highest layers.

which it has been proven [27] that 3-rd or 4-th order is sufficient to capture all local image variation perceivable by humans. According to scale-space theory [4], [28], the Gaussian family constitutes the Taylor expansion of the image function which guarantees completeness. For ii) we maintain backpropagation parameter optimization in the network, now applied to learning the weights by which the filters are summed into the effective filter set.

Similarly motivated, the Scattering Transform [30], [31], [32], a special type of CNN, uses a complete set of wavelet filters ordered in a cascade. However, different from a classical CNN, the filters parameters are not learned by backpropagation but rather they are fixed from the start and the whole network structure is motivated by signal processing principles. In the Scattering Network the choice of local and global invariances are tailored to the type of images specifically. In the Scattering Transform invariance to group actions beyond local translation and deformation requires explicit design [31] with the regards to the variability encountered in the target domain such as translation [30], rotation [32] or scale. As a consequence, when the desired invariance groups are known a priori, Scattering delivers very effective networks.
Our paper takes the best of two worlds. On the one hand, we adopt the Scattering principle of using fixed filter bases as a function prior in the network. But on the other hand, we maintain from plain CNNs the capacity to learn arbitrary effective filter combinations to form complex invariances and equivariances.

Our main contributions are:

- Deriving the structured receptive field network (RFNN) from first principles by formulating filter learning as a linear decomposition onto a filter basis, unifying CNNs and multiscale image analysis in a learnable model.
- Combining the strengths of Scattering and CNNs. We do well on both domains: i) small datasets where Scattering is best but CNNs are weak; ii) complex datasets where CNNs excel but Scattering is weak.

- State-of-the-art classification results on a small dataset where pre-training is infeasible. The task is Alzheimer's disease classification on two widely used brain MRI datasets. We outperform all published results on the ADNI dataset.

3.2 RELATED WORK

3.2.1 Scale-space: the deep structure of images

Scale-space theory [28] provides a model for the structure of images by steadily convolving the image with filters of increasing scale, effectively reducing the resolution in each scale step. While details of the image will slowly disappear, the order by which they do so will uniquely encode the deep structure of the image [4]. Gaussian filters have the advantage in that they do not introduce any artifacts [25] in the image while Gaussian derivative filters form a complete and stable basis to decompose locally any realistic image. The set of responses to the derivative filters describing one patch is called the N-jet [5].

In the same vein, CNNs can be perceived to also model the deep structure of images, this time in a non-linear fashion. The pooling layers in a CNN effectively reduce resolution of input feature maps. Viewed from the top of the network down, the spatial extent of a convolution kernel is increased in each layer by a factor 2, where a 5x5 kernel at the higher layer measures 10x10 pixels on the layer below. The deep structure in a CNN models the image on several discrete levels of resolution simultaneously, precisely in line with Scale-space theory.

Where CNNs typically reduce resolution by max pooling in a non-linear fashion, Scale-space offers a linear theory for continuous reduction of resolution. Scale-space theory treats an image as a function of the mathematical apparatus to reveal the local image structure. In this paper, we exploit the descriptive power of Scale-space theory to decompose the image locally on a fixed filter basis of multiple scales.

3.2.2 CNNs and their parameters

CNNs [33] have large numbers of parameters to learn [34]. This is their strength as they can solve extremely complicated problems [34], [35]. At the same time, their number of unrestricted parameters is a limiting factor in terms of the large amounts of data needed to train. To prevent overfitting, which is an issue even when training on large datasets like the million images of the ILSVRC2012 challenge [21], usually regularization is imposed with methods like dropout [36] and weight decay [37].

Regularization is essential to achieving good performance. In cases where limited training data are available, CNN training quickly overfits regardless and the learned representations do not generalize well. Transfer learning from models pre-trained in similar domains to the new domain is necessary to achieve competitive results [38]. One thing pre-training on large datasets provides is knowledge about properties inherent to all natural images, such as spatial coherence and robustness to uninformative variability. In this paper, we aim to design these properties into CNNs to improve generalization when limited training data are available.

3.2.3 The Scattering representation

To reduce model complexity we draw inspiration from the elegant convolutional Scattering Network [30], [31], [32]. Scattering uses a multi-layer cascade of a pre-defined wavelet filter bank with nonlinearity and pooling operators. It computes a locally translation-invariant image representation, stable to deformations while avoiding information loss by recovering wavelet coefficients in successive layers. No learning is used in the image representation: all relevant combinations of the filters are fed into an SVM-classifier yielding state-of-the-art results on small dataset classification. Scattering is particularly well-suited to small datasets because it refrains from feature learning. Since all filter combinations are pre-defined, their effectiveness is independent of dataset size. In this paper, we also benefit from a fixed filter bank. In contrast to Scattering, we *learn* linear combinations of a filter basis into effective filters and non-linear combinations thereof.

The wavelet filterbank of Scattering is carefully designed to sample a range of rotations and scales. These filters and their properties are grounded in wavelet theory [9] and exhibit precisely formulated properties. By using interpretable filters, Scattering can design invariance to finite groups such as translation [30], scale and rotation [32]. Hard coding the invariance into the network is effective when the problem and its invariants are known precisely, but for many applications this is rarely the case. When the variability is unknown, additional Scattering paths have to be computed, stored and processed exhaustively before classification. This leads to a well-structured but very high dimensional parameter space. In this paper, we use a Gaussian derivatives basis as the filter bank, firmly grounded in scale-space theory [4], [25], [28]. Our approach incorporates learning effective filter combinations from the very beginning, which allows for a compact representation of the problem at hand.

3.2.4 Recent CNNs

Restriction of parameter spaces has led to some major advances in recent CNNs performance. Network in Network [39] and GoogleNet [29] illustrate that fully connected layers, which constitute most of Alexnet's parameters, can be replaced by a global average pooling layer reducing the number of parameters in the fully connected layers to virtually zero. The number of parameters in the convolution layers is increased to enhance the expressiveness of each layers features. Overall the total number of parameters is not necessarily decreased, but the function space is restricted, allowing for bigger models while classification accuracy improves [29], [39].

The VGG Network [40] improves over Alexnet in a different way. The convolution layers parameter spaces are restricted by splitting each 5x5 convolution layer into two 3x3 convolution layers. 5x5 convolutions and 2 subsequent 3x3 convolutions have the same effective receptive field size while each receptive field has 18 instead of 25 trainable parameters. This regularization enables learning larger models that are less prone to overfitting. In this paper, we follow a different approach in restricting the free parameter space without reducing filter size.

3.3 DEEP RECEPTIVE FIELD NETWORKS

3.3.1 Structured receptive fields

In our structured receptive field networks we make the relationship between Scale-space and CNNs explicit. Whereas normal CNNs treat images and their filters as pixel values, we aim for a CNN that treats images as functions in Scale-space. Thus, the learned convolution kernels become functions as well. We therefore approximate an arbitrary CNN filter $F(x)$ with a Taylor expansion around a up to order M

$$F(x) = \sum_{m=0}^{M} \frac{F^m(a)}{m!}(x-a)^m. \tag{18}$$

Scale-space allows us to use differential operators on images, due to linearity of convolution we are able to compute the exact derivatives of the scaled underlying function by convolution with derivatives of the Gaussian kernel

$$G(.;\sigma) * F(x) = \sum_{m=0}^{N} \frac{(G^m(.;\sigma) * F)(a)}{m!}(x-a)^m, \tag{19}$$

where $*$ denotes convolution, $G(.;\sigma)$ is a Gaussian kernel with scale σ and $G^m(.;\sigma)$ is the m^{th} order Gaussian derivative with respect to it's spatial variable.

Thus, a convolution with a basis of weighted Gaussian derivatives receptive fields is the functional equivalent to pixel values in a standard CNN operating on a scaled infinitely differentiable version of the image.

To construct the full basis set in practice, one can show that the Hermite polynomials emerge from a sequence of Gaussian derivatives up to order M [41]. A Gaussian derivative of arbitrary order can be obtained from the orthogonal Hermite polynomials H_m through pointwise multiplication with a Gaussian envelope

$$G^m(.;\sigma) = (-1)^m \frac{1}{\sqrt{\sigma^m}} H_m(\frac{x}{\sigma\sqrt{2}}) \circ G(x;\sigma). \tag{20}$$

The resulting operators allow computation of an image's local geometry at scale σ and location x up to any order of precision M. This basis is thus a complete set. Each derivative corresponds to an independent degree of freedom, making it also a minimal set.
Thus, an RFNN is a general CNN when a complete polynomial up to infinite order is considered. We restrict the basis based on the requirement that one can construct quadrature pair filters as suggested by Scattering and by evidence from Scale-space theory [27] that considers all orders up to a maximum of 4, as it has been suggested that orders beyond that do not carry any information meaningful to visual perception.

3.3.2 Transformation properties of the basis

The isotropic Gaussian derivatives exhibit multiple desirable properties. It is possible to create complex multi-orientation pyramids that constitute wavelet representations similar to the Morlet Wavelet pyramids used in Scattering Networks [30]. A complex multiresolution filterbank can be constructed from a dilated and rotated Gaussian derivative quadrature. The exact dilated versions of an arbitrary Gaussian derivative G^m can be obtained through convolution with a Gaussian kernel of scale $\sigma = n$ according to

$$G^m(.;\sqrt{j^2+n^2}) = G^m(.;j) * G(.;n). \tag{21}$$

Arbitrary rotations of Gaussian derivative kernels can be obtained from a minimal set of basis filters without the need to rotate the basis itself. This property is referred to as steerability [10]. Steerability is a property of all functions that can be expressed in a polynomial in x and y times an isotropic Gaussian. This certainly holds for the Gaussian derivatives according to equation 20. For example a quadrature pair of 2nd and 3rd order Gaussian derivatives G^{xx} and G^{xxx} rotated by an angle θ can be obtained from a minimal 3 and 4 x-y separable basis set.

Algorithm 1 RFNN Learning - updating the parameters α_{ij}^l between input map indexed by i and output map indexed by j of layer l in the Mini-batch Gradient Decent framework.

1: **Input:** input feature maps o_i^{l-1} for each training sample (computed for the previous layer, o^{l-1} is the input image when $l = 1$), corresponding ground-truth labels $\{y_1, y_2, \ldots, y_K\}$, the basic kernels $\{\phi_1, \phi_2, \ldots, \phi_M\}$, previous parameter α_{ij}^l.
2: compute the convolution $\{\zeta_1, \zeta_2, \ldots, \zeta_m\}$ of $\{{o^{l-1}}_i\}$ respect to the basic kernels $\{\phi_1, \phi_2, \ldots, \phi_M\}$
3: obtain the output map $o_j^l = \alpha_{ij1}^l \cdot \zeta_1 + \alpha_{ij2}^l \cdot \zeta_2 + ... + \alpha_{ijM}^l \cdot \zeta_M$
4: compute the δ_{jn}^l for each output neuron n of the output map o_j^l
5: compute the derivative $\psi'(t_{jn}^l)$ of the activation function
6: compute the gradient $\frac{\partial E}{\partial \alpha_{ij}^l}$ respect to the weights α_{ij}^l
7: update parameter $\alpha_{ij}^l = \alpha_{ij}^l - r \cdot \frac{1}{K} \cdot \sum_{k=1}^K [\frac{\partial E}{\partial \alpha_{ij}^l}]_k$, r is the learning rate
8: **Output:** α_{ij}^l, the output feature maps o_j^l

Given by:

$$\begin{aligned} G_\theta^{xx} &= \cos^2(\theta) G^{xx} - 2\cos(\theta)\sin(\theta) G^{xy} + \sin^2(\theta) G^{yy} \\ G_\theta^{xxx} &= \cos^3(\theta) G^{xxx} - 3\cos^2(\theta)\sin(\theta) G^{xxy} \\ &\quad + 3\cos(\theta)\sin^2(\theta) G^{xyy} - \sin^3(\theta) G^{yyy} \end{aligned} \tag{22}$$

A general derivation of the minimal basis set necessary for steering arbitrary orders can be found in [10]. Note that the anisotropic case can be constructed in analogous manner according to [42]. This renders Scattering as a special case of the RFNN for fixed angles and scales, given a proper choice of pooling operations and possibly skip connections to closely resemble the architecture described in [30]. In practice this allows for seamless integration of the Scattering concept into CNNs to achieve a variety of hybrid architectures.

3.3.3 Learning basis filter parameters

Learning a feature representation boils down to convolution kernel learning. Where a classical CNN learns pixel values of the convolutional kernel, a RFNN learns Gaussian derivative basis function weights that combine to a convolution kernel function. A 2D filter kernel function $F(x, y)$ in all layers, is a linear combination of i unique (non-symmetric) Gaussian derivative basis functions ϕ

$$F(x, y) = \alpha_1 \phi_1 + \cdots + \alpha_n \phi_i, \tag{23}$$

where $\alpha_1, ..., \alpha_i$ are the parameters being learned.

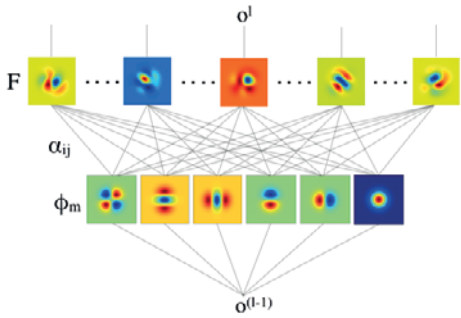


Figure 4: An illustration of the basic building block in an RFNN network. A linear comibination of a limited basis filter set ϕ_m yields an arbitrary number of effective filters. The weights α_{ij} are learned by the network.

We learn the filter's weights α by mini-batch stochastic gradient descent and compute the derivatives of the loss function E with respect to the parameters α through backpropagation. It is straightforward to show the independence between the basis weights α and the actual basis (see Appendix for derivation). Thus, we formulate the basis learning as a combination of a fixed basis layer with a 1x1 convolution layer that has a kernel depth equal to the basis order. Propagation through the 1x1 layer is done as in any CNN while propagation through the basis layer is achieved by a convolution with flipped versions of the Gaussian filters. This makes it straightforward to include into any existing deep learning framework. The basic structured receptive field building block is illustrated in figure 4, showing how each effective filter is composed out of multiple basis filters. Note that the linearity of convolution allows us to never actually compute the effective filters. Convolving with effective filters is the same as convolving with the basis and then recombining the feature maps, allowing for efficient implementation. Algorithm 1 shows how the parameters are updated.

3.3.4 The network

In this work, we choose the Network in Network (NiN) architecture [39] as the basis into which we integrate the structured receptive fields. It is particularly suited for an analysis of the RFNN approach, as the absence of a fully connected layer ensures all parameters to be fully concerned with re-combining basis filter outcomes of the current layer. At the same time, it is powerful, similar in spirit to the state of the art Googlenet [29], while being comparably small and fast to train.

NiN alternates one spatial convolution layer with 1x1 convolutions and pooling. The 1x1 layers form non-linear combinations of the spatial convolution layers outputs. This procedure is repeated four times in 16 layers, with different number of filters and kernel sizes for the spatial convolution layer. The final pooling layer is a global average pooling layer. Each convolution layer is followed by a rectifier nonlinearity. Details on the different NiNs for Cifar and Imagenet can be found in the Caffe model zoo [43].

In the RFNN version of the Network in Network model, the basis layer including the Gaussian derivatives set is replacing the spatial convolution layer and corresponds to ϕ_m in equation 23. Thus, each basis convolution layer has a number of filters depending on order and scale of the chosen basis set. The basis set is fixed: no parameters are learned in this layer. The linear recombination of the filter basis is done by the subsequent 1x1 convolution layer, corresponding to α_{ij} in equation 23. Note that there is no non-linearity between ϕ_m and α_{ij} layer in the RFNN case, as the combinations of the filters are linear. Thus the RFNN model is almost identical to the standard Network in Network. We evaluate the model with and without multiple scales σ_s. When including scale, we extract 4 scales, as the original model includes 3 pooling steps and thus operates on 4 scales at least. In the first layer we directly compute 4 scales, sampled continuously with $\sigma_s = 2^s$ where $s = scale$ as done in [30]. In each subsequent layer we discard the lowest scale. The dimensionality reduction by max pooling renders it meaningless to insert the lowest scale of the previous layer into the filter basis set as it is already covered by the pyramidal structure of the network. This enables us to save on basis filters in the higher layers of the network. In conclusion we reduce the total number of 2D filters in the network from 520,000 in the standard Network in Network to between 12 and 144 in the RF Network in Network (RFNiN), while retaining the models expressiveness as shown in the experimental section.

3.4 EXPERIMENTS

The experiments are partitioned into four parts. i) We show insight in the proposed model to investigate design choices; ii) we show that our model combines the strengths of Scattering and CNNs; iii) we show structured receptive fields improve classification performance when limiting training data; iv) we show a 3D version of our model that outperforms the state-of-the-art, including a 3D-CNN, on two brain MRI classification datasets where large pre-training datasets are not available. We use the Caffe library [43] and Theano [44] where we added RFNN as a separate module. Code is available on github[1].

1 https://github.com/jhjacobsen/RFNN

3.4.1 Experiment 1: Model insight

The RFNN used in this section is the structured receptive field version of the Network in Network (RFNiN) introduced in section 3.3. We gain insight into the model by evaluating the scale and order of the basis filters. In addition, we analyze the performance compared to the standard Network in Network (NiN) [39] and Alexnet [34] and show that our proposed model is not merely a change in architecture. To allow overnight experiments we use the 100 largest classes of the ILSVRC2012 ImageNet classification challenge [21]. Selection is done by folder size, as more than 100 classes have 1,300 images in them, yielding a dataset size of 130,000 images. This is a real-world medium sized dataset in a domain where CNNs excel.

Experimental setup. The Network in Network (NiN) model and our Structured Receptive Field Network in Network (RFNiN) model are based on the training definitions provided by the Caffe model zoo [43]. Training is done with the standard procedure on Imagenet. We use stochastic gradient descent, a momentum of 0.9, a weight decay of 0.0005. The images are re-sized to 256x256, mirrored during training and the dataset mean is subtracted. The base learning rate was decreased by a factor of 10, according to the reduction from 1,000 to 100 classes, to ensure proper scaling of the weight updates, NiN didn't converge with the original learning rate. We decreased it by a factor of 10 after 50,000 iterations and again by the same factor after 75,000 iterations. The networks were trained for 100,000 iterations. Results are computed as the mean Top-1 classification accuracy on the validation set.

Filter basis order. In table 1, the first four rows show the result of RFNiN architectures with 1st to 4th order Gaussian derivative basis filter set comprised of 12 to 60 individual Gaussian derivative filters in all layers of the network. In these experiments the value of σ=1, fixed for all filters and all layers. Comparing first to fourth order filter basis in table 1, we conclude that third order is sufficient, outperforming first and second order as predicted by Scale-space theory [27]. The fourth order does not add any more gain.

Filter scale. The RFNiN-Scale entries of table 1 show the classification result up to fourth order now with 4 different scales, σ=1, 2, 4, 8 for the lowest layer, σ=1, 2, 4 for the second layer, σ=1, 2 for the third, and σ=1 for the fourth. This implies that the basis filter set expands from 24 up to 144 filters in total in the network. Comparing the use of single scale filters in the network to dilated copies of the filters with varying scale indicates that a considerable gain can be achieved by including filters with different scales. This observation is supported by Scattering [30], showing that the multiple scales can directly be extracted from the first layer on. In fact, normal CNNs are also capable of similar behavior, as positive valued low-pass filter feature maps are not affected by rectifier nonlinearities [32].

ILSVRC2012-100 Subset			
Method	Top-1	2DFilters	#Params
RFNiN 1^{st}-order	44.83%	12	1.8M
RFNiN 2^{nd}-order	61.24%	24	3.4M
RFNiN 3^{rd}-order	63.64%	40	5.5M
RFNiN 4^{th}-order	62.92%	60	8.1M
RFNiN-Scale 1^{st}-order	57.21%	24	2.2M
RFNiN-Scale 2^{nd}-order	67.56%	54	4.2M
RFNiN-Scale 3^{rd}-order	**69.65%**	94	6.8M
RFNiN-Scale 4^{th}-order	68.95%	144	10.1M
Network in Network	67.30%	520k	8.2M
Alexnet	54.86%	370k	60.0M

Table 1: Results on 100 Biggest ILSVRC2012 classes: The table shows RFNiN with 1st, 2nd, 3rd and 4th order filters in the whole network. Row 1-4 are applying basis filters in all layers on a scale of σ=1. RFNiN-scale in row 5-8 applies basis filters on 4 scales, where σ=1,2,4,8. The results show that a 3^{rd} order basis is sufficient while incorporating scale into the network gives a big gain in performance. The RFNiN is able to outperform the same Network in Network architecture.

Thus, scale can directly be computed from the first layer onwards, which yields a much smaller set of basis filters and fewer convolutions needed in the higher layers. Note that number of parameters is not directly correlated with performance.

Analysis of network layers. For the network RFNiN 4th-order Figure 5 provides an overview of the range of basis weights per effective filters in all layers, where the x-axis indexes the spatial derivative index and y-axis the mean value plus standard deviation of weights per layer over all effective filter kernels. The figure indicates that weights decrease towards higher orders as expected. Furthermore zero order filters have relatively high weights in higher layers, which hints to passing on scaled incoming features.

Comparison to Network in Network. The champion RFNiN in table 1 slightly outperforms the Network in Network with the same setting and training circumstances while only having 94 instead 520,000 spatial filters in the network in total. Note that the number of parameters is relatively similar though, as the scale component increases the number of basis functions per filter significantly. The result shows that our basis representation is sufficient for complex tasks like Imagenet.

Refactorize Network in Network. To illustrate that our proposed model is not merely a change in architecture we compare to a third architecture.

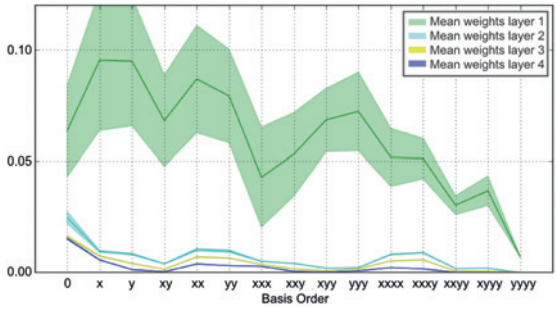


Figure 5: Mean of filter weights and variances per layer for 15 basis filters with no scale, as trained on ILSVRC2012-100 subset. Note that the lower order filters have the highest weights while zero-order filters are most effective in higher layers for combinations of lower responses.

Model	Basis	#Params	Top-1
NiN-refactor Layer 1	Free	7.47M	64.10%
RFNiN-refactor Layer 1	Gauss	7.47M	68.63%
NiN-refactor All Layers	Free	6.87M	38.02%
RFNiN-Scale 3rd-order	Gauss	6.83M	69.65%

Table 2: Classification on ILSVRC2012-100 to illustrate influence of factorization on performance. The results show that the advantage of the Gaussian basis is substantial and our results are not merely due to a change in architecture.

We remove the Gaussian basis and we re-factorize the NiN such that it becomes identical to RFNiN. Both have almost the same number of parameters, but the NiN-factorize has a freely learnable basis. Re-factorizing only the first layer and leaving the rest of the network as in the original NiN, in table 2 we show that a Gaussian basis is superior to a learned basis. When re-factorizing all layers, RFNiN-Scale 3rd-order results are superior by far to the identical NiN-factorize All Layers.

3.4.2 Experiment 2: Scattering and RFNNs

Small simple domain. We compare an RFNN to Scattering in classification on reduced training sizes of the MNIST dataset. This is the domain where Scattering outperforms standard CNNs [30]. We reduce the number of training samples when training on MNIST as done in [30]. The network architecture and training parameters used in this section are the same as in [45].

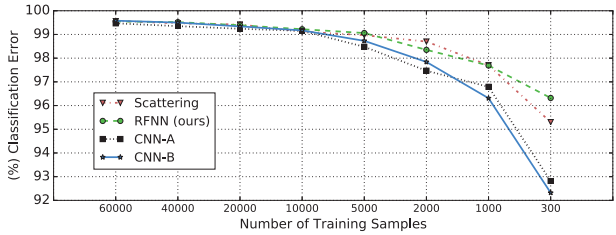


Figure 6: Classification performance of the Scattering Network on various subsets of the MNIST dataset. In comparison the state of the art CNN-A from [46]. RFNN denotes our receptive field network, with the same architecture as CNN-B. Both are shown, to illustrate that good performance of the RFNN is not due to the CNN architecture, but due to RFNN decomposition. Our RFNN performs on par with Scattering, substantially outperforming both CNNs.

The RFNN contains 3 layers with a third order basis on one scale as a multiscale basis didn't provide any gain. Scale and order are determined on a validation set. Each basis layer is followed by a layer of $\alpha_N = 64$ 1x1 units that linearly re-combine the basis filters outcomes. As comparison we re-implement the same model as a plain CNN. The CNN and Scattering results on the task are taken from [30], [46].

Results are shown in Figure 6, each number is averaged over 3 runs. For the experiment on MNIST the gap between the CNNs and networks with predefined filters increases when training data is reduced, while RFNN and Scattering perform on par even at the smallest sample size. **Large complex domain**. We compare against Scattering on the Cifar-10 and Cifar-100 datasets, as reported by the recently introduced Deep Roto-Translation Scattering approach [47], a powerful variant of Scattering networks explicitly modeling invariance under the action of small rotations. This is a domain where CNNs excel and learning of complex image variabilities is key.

The RFNiN is again a variant of the standard NiN for Cifar-10. It is similar to the model in experiment 1, just that it has one basis layer, two 1x1 convolution layers and one pooling layer less and the units in the 1x1 convolution layers are 192 in the whole network. Furthermore, we show performance of the state-of-the-art recurrent convolutional networks (RCNNs) [48] for comparison.

The results in Table 3 show a considerable improvement on Cifar-10 and Cifar-100 when comparing RFNiN to Roto-Translation Scattering [47], which was designed specifically for this dataset. RCNNs performance is considerably higher as they follow a different approach to which structured receptive fields can also be applied if desired.

Model	Cifar-10	Cifar-100
Roto-Trans Scattering	82.30%	56.80%
RFNiN	86.31%	63.81%
RCNN	91.31%	68.25%

Table 3: Comparison against Scattering on a large complex domain. State-of-the-art comparison is given by RCNN. RFNiN outperforms Scattering by large margins.

RFNNs are robust to dataset size. From these experiments, we conclude that RFNNs combine the best of both worlds. We outperform CNNs and compete with Scattering when training data is limited as exemplified on subsets of MNIST. We capture complex image variabilities beyond the capabilities of Scattering representations as exemplified on the datasets Cifar-10 and Cifar-100 despite operating in a similarly smooth parameter space on a receptive field level.

3.4.3 Experiment 3: Limiting datasize

To demonstrate the effectiveness of the RF variant compared to the Network in Network, we reduce the number of classes in the ILSVRC2012-dataset from 1000 to 100 to 10, resulting in a reduction of the total number of images on which the network was trained from 1.2M to 130k to 13k and subsequent decrease in visual variety to learn from. To demonstrate performance is not only due to smaller number of learnable parameters, we evaluate two RFNiN versions. RFNiN-v1 is RFNiN-Scale 3rd-order from table1. RFNiN-v2 is one layer deeper and wider [128/128/384/512/1000] version of the RFNiN-v1, resulting in 3 million additional parameters, which is 2,5 million more than NiN.

The results in table 4 show that compared to CNNs the RFNiN performance is better relatively speaking when the number of samples and thus the visual variety decreases. For the 13k ILSVRC2012-10 image dataset the gap between RFNiN and NiN increased to 8.0% from 2.4% for the 130k images in ILSVRC2012-100 while the best RFNiN is inferior to NiN by 2.98% for the full ILSVRC2012-1000. This supports our aim that RFNiN is effectively incorporating natural image priors, yielding a better performance compared to the standard NiN when training data and variety is limited, even when having more learnable parameters. Truly large datasets seem to contain information not yet captured by our model.

Model	#Params	1000-class	100-class	10-class
NiN	7.5M	56.78%	67.30%	76.97%
RFNiN-v1	6.8M	50.08%	69.65%	85.00%
RFNiN-v2	10M	54.04%	70.78%	83.36%

Table 4: Three classification experiments on ILSVRC2012 subsets. Results show that the bigger model (RFNiN-v2) performs better than RFNiN-Scale 3rd-order (RFNiN-v1) on the 1000-classes while on 100-class and 10-class, v1 and v2 perform similar. The gap between RFNiN and NiN increases for fewer classes.

3.4.4 Experiment 4: Small realistic data

We apply an RFNiN to 3D brain MRI classification for Alzheimer's disease [49] on two popular datasets. Neuroimaging is a domain where training data is notoriously small and high dimensional and no truly large open access databases in a similar domain exist for pre-training.

We use a 3-layer RFNiN with filters sizes [128,96,96] with a third order basis in 3 scales $\sigma \in \{1, 4, 16\}$. This time wider spaced, as the brains are very big objects and are centered due to normalization to MNI space with the FSL library [50]. Each basis layer is followed by one 1x1 convolution layer. Global average pooling is applied to the final feature maps. The network is implemented in Theano [44] and trained with Adam [51].

3D MRI classification	Accuracy	TPR	SPC
3D-RFNiN (ours)	**97.79%**	**97.14%**	**98.78%**
ICA [52]	80.70%	81.90%	79.50%
Voxel-Direct-D-gm [49]	-	81.00%	95.00%
3D-CNN [53]	95.70%	-	-
NIB [54]	94.74%	95.24%	94.26%

Table 5: Alzheimer's classification with 150 train and test 3D MRI images from the widely used ADNI benchmark. RFNiN, ICA and Voxel-Direct-D-gm are trained on the subset introduced in [49], 3D-CNN and NIB were trained on their own subset of ADNI, using an order of magnitude more training data. RFNiN outperforms all published results. Reported is accuracy, true positive rate and specificity.

The results are shown in table 5. Note that [53], [54] train on their own subset and use an order of magnitude more training data. We follow standard practice [49] and train on a smaller subset. Nevertheless we outperform all published methods on the ADNI dataset.

The same 3 layer NiN as our RFNiN model has 84.21% accuracy, more than 10% worse while being hard to train due to unstable convergence. On the OASIS AD-126 Alzheimer's dataset [55], we achieve an accuracy of 80.26%, compared to 74.10% with a SIFT-based approach [56]. Thus, we show our RFNiN can effectively learn comparably deep representations even when data is scarce and exhibits stable convergence properties.

3.5 DISCUSSION

The experiments show that structuring convolutional layers with a filter basis grounded on Scale-space principles improves performance when data is limited. The filter basis provides regularization especially suited for image data by restricting the parameter space to smooth features up to fourth order. The Gaussian derivative basis opens up a new perspective for reasoning in CNNs, connecting them with a rich body of prior multiscale image analysis research that can now be readily incorporated into the models. This is especially interesting for applications where model insight and control is key.

We illustrated the effectiveness of RFNNs on multiple subsets of Imagenet, Cifar-10, Cifar-100 and MNIST. The choice of a third order Gaussian basis is sufficient to tackle all datasets which is in accordance with prior research [27], [30]. While it remains an open problem to match the performance of CNNs on very large datasets like the 1000-class ILSVRC2012, our results show that the RFNN method outperforms CNNs by large margins when data are scarce. It can also outperform CNNs on challenging medium sized datasets while being superior to Scattering on large datasets despite having more parameters as the pre-defined basis restriction allows the network to devote its full capacity to a sensible feature spaces. As a small data real world example, we verify our claims with 3D MRI Alzheimer's disease classification on two datasets where we consistently achieve competitive performance including the best results on the widely used ADNI dataset.

4 DYNAMIC STEERABLE BLOCKS IN DEEP RESIDUAL NETWORKS

4.1 INTRODUCTION

Deep Convolutional Neural Networks (CNNs) are the state-of-the-art solution to many vision tasks [20]. However, they are known to be data-inefficient, as they require up to millions of training samples to achieve their powerful performance [58], [59]. In this work, we propose a formulation of CNNs that more efficiently learns to exploit generic regularities known to be present in the data a priori.

For images, as well as any other sensory data, CNNs typically learn filters from individual pixel values. In this paper, we show that alternatives to the pixel basis are more natural formulations for learning models on locally well-understood data like images. We show increased classification performance in state-of-the-art ResNets [60] and Densenets [61] on the highly competitive Cifar-10 classification task by replacing the pixel basis with a basis more suitable for natural images. Further, we show that such a replacement naturally leads to powerful extensions of residual blocks as dynamic steerable interpolators that can steer their filters conditioned on the input with respect to predefined continuous geometric transformations like rotations or scalings. The proposed block allows the network to adaptively learn the degree of local invariance required for each filter, by decoupling filter learning and local geometrical pose adaption. We show the effectiveness of this approach on the BSD500 boundary detection task, where precise and adaptive local invariances are key. We outperform all competing non-pretrained methods with our approach.

Our contributions:

- We introduce the notion of frame bases to CNNs and show that classically used frames from Computer Vision aid optimization, when compared to the commonly used pixel basis. We illustrate this by improving classification performance on Cifar-10+ for multiple ResNet and Densenet architectures, by mere substitution of the pixel basis with a frame only.

- Exploiting the steerability properties of frames further, we derive Dynamic Steerable Blocks that are able to continuously transform features in a locally adaptive manner and illustrate the approach on a synthetic tasks to highlight the advantages over competing approaches.

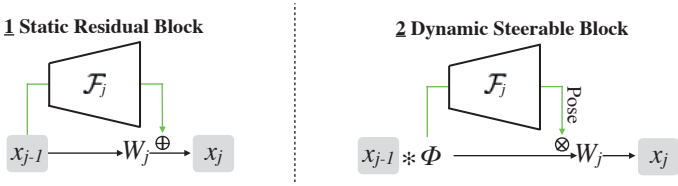


Figure 7: Left is a classical residual block as used in vanilla ResNets [57]. Outputs x_{j-1} of the previous block are combined additively with the output of a small stack of convolutional layers $\mathcal{F}_j$ to form the final output x_j. We augment this formulation in two ways to achieve our dynamic block formulation on the right. First, we apply a change to a steerable frame Φ on the input x_{j-1} and second we replace the addition operation with a multiplication. This permits us to interpret $\mathcal{F}_j$ as a pose estimating network, that directly outputs the linear projection coefficients, transforming the basis and subsequently the effective filters adaptively in a dynamic and, if desired, location dependent manner.

- To evaluate the practicality of our proposed approach, we apply Dynamic Steerable Block networks on the BSDS-500 contour detection task [62] where we achieve increased performance among competing methods that do not utilize pre-training.

The paper is organized as follows. First, we review related literature of alternative parametrizations and incorporation of prior geometrical knowledge into CNNs. Secondly, we introduce the theoretical framework of frame-based CNNs and steerable two-factor blocks, which our work rests upon. Third, we show that careful reparametrization of CNNs can increase performance on natural image classification. Lastly, we show how to extend frame-based Resnet blocks to dynamic steerable blocks that have the ability to dynamically adapt filters, with several advantages and promising results on the Berkeley Contour Detection dataset.

4.2 RELATED WORK

Convolutional Networks with alternative bases have been proposed with various degrees of flexibility. A number of works utilizes change of basis to stabilize training and increase convergence behavior [63], [64]. Another line of research is concerned with complex-valued CNNs, either learned [65], or fully designed like the Scattering networks [47], [66].

Scattering, as well as the complex-valued networks, rest upon a direct connection between the signal processing literature and CNNs.Inspired by the former, Structured Receptive Field Networks are learned from an overcomplete multi-scale frame, effectively improving performance for small datasets due to restricted feature spaces [67]. Closely related is another line of recent promising work on group-equivariant [68], [69] and steerable CNNs [7], [70]. The latter build steerable representations via appropriately chosen basis functions, illustrating that CNNs with well-chosen geometrical inductive biases consistently outperform state-of-the-art approaches in multiple domains. However, all of the former approaches rely on hand-engineered types of representations and none considers locally adaptive filtering with learned degree of invariance, we aim to bridge this gap. Inspired by CNNs learned from alternative bases, we introduce the general principle of Frame-based convolutional networks that allow for non-orthogonal, overcomplete and steerable feature spaces.

Steerable frames are a concept established early for signal processing. Initially introduced by [71], the concept was extended to the Steerable Pyramid by [72] and to a Lie-group formulation by [11], [73]. Further, steerability has recently been related to tight frames, presenting Simoncelli's Steerable Pyramid and multiple other Wavelets arising as a special case of the non-orthogonal Riesz transform [74]. Steerable pyramids have been applied to CNNs as a preprocessing step [75], but have not yet been learnable. We incorporate steerable frames in CNNs to increase their de facto expressiveness and to allow them to learn their configurations, rather than picking them a priori.

Another way to impose structure onto CNN representations and subsequently increase their data-efficiency is by pre-defining the possible transformations, as done in Transforming Autoencoders [76], which map their inputs from the image to pose space through a neural network. The Spatial Transformer Networks [77] learn global, and deformable convolutional networks [78] local transformation parameters in a similar way while applying them to a nonlinear co-registration of the feature stack to some estimated pose. Dynamic Filter Networks [79] move one step further and estimate filters for each location, conditioned on their input. These approaches are all dynamic in a sense that they condition their parameters on the input appearance. Our proposed dynamic residual block can be interpreted as a middle ground that combines the idea of Dynamic Filter Networks with explicit pose prediction into blocks that can locally estimate filter poses from a continuous input space. As such, we overcome the difficulty of estimating local filter pose, while being able to separate pose and feature learning globally without the need for differentiable samplers or locally connected layers.

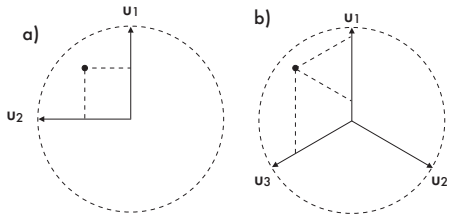


Figure 8: a) Is an orthonormal basis in $\mathbb{R}^2$, u_1 and u_2 are linearly independent and span the space of $\mathbb{R}^2$. A dot in this example represents a filter in a convolutional network with coefficients $\{u_1, u_2\}$. b) A tight frame in $\mathbb{R}^2$. u_1, u_2 and u_3 are linearly dependent. A dot in this example represents a convolutional filter with coefficients $\{u_1, u_2, u_3\}$. The frame is an overcomplete representation, again spanning $\mathbb{R}^2$ and again preserving the norm. Note that the set of filter coefficients as represented by the dot is not unique. Thus even if one u is obstructed by noisy updates or measurements, the filter may still be robust.

4.3 CNN BASES BEYOND PIXELS

The most general set of viable bases to learn filters from are called *Frames* [80]. Frames are a natural generalization of orthogonal bases and are spanning sets that span the same space of functions an orthogonal basis does, while allowing for overcomplete representations and hence more densely sampled parameter spaces.

Frames can be seen as a superset of orthogonal bases in the sense that every basis is a frame, but not the reverse, see figure 8. Frames have three main advantages: 1) Frames can spell out signal properties more explicitly, facilitating optimization when good frames for the type of data are known, as is the case for many types of signals like images or video [80]. 2) Frames allow for overcomplete representations of the signal adding robustness and regularization to the optimization procedure, as they are more stable when measurements or updates are noisy. 3) Many frames are steerable and thus provide us with a signal representation that can be dynamically steered by simple linear projections, i.e. they have the ability to linearize group actions. Properties 1) and 2) are illustrated in experiment 3.1, where we merely replace standard pixel bases with alternative frames and evaluate the effect. Property 3) is illustrated in experiments 3.2, where we leverage the steerability property of some frames explicitly.

In a standard convolutional network, a filter kernel is a linear combination over the standard (pixel) basis for $l^2(\mathbb{N})$. This pixel basis is composed of delta functions for every dimension and W_i is the i_{th} filter of the network with parameters w_n^i.

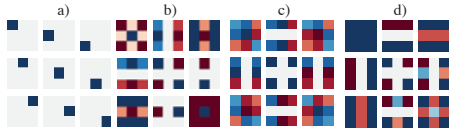


Figure 9: An illustrative plot of multiple 3x3 spanning sets Φ: a) Pixel-basis, b) Gaussian Derivatives (first 9 atoms), c) Non-orthogonal Framelet, d) Naive Frame. Note the increased symmetries in the three latter.

Without loss of generalization the orthonormal standard basis can be replaced by a frame to include steerability, non-orthogonality, overcompleteness and increased symmetries into the representation. Changing from the pixel to an arbitrary frame is as simple as replacing the pixel basis e_n with a frame of choice with elements ϕ_m as follows:

$$W_j\Phi(x_{j-1}) = \sum_{n=1}^{N} w_n^j e_n \star x_{j-1} \equiv \sum_{m=1}^{M} w_m^j \phi_m \star x_{j-1}, \tag{24}$$

where $w_1^i, ..., w_m^i$ are again the filter coefficients being learned. Note that after optimization with an overcomplete frame is done, the resulting network can be rewritten in terms of the standard pixel basis. Therefore frames only act as a regularizer during training, they do not increase the effective parameter cost of the network. In practice for CNNs working on images we investigate four typical choices of frames: i) the vanilla orthogonal pixel basis, ii) Gaussian derivatives, one of the most widely used overcomplete frames from the computer vision literature (also used in SIFT) [5], [6], [27], [67], iii) Framelets, a non-orthogonal but not overcomplete basis, designed for images [81] and iv) A "naive" frame of the form $x^p y^q$ derived from steerability requirements [73] but with no image properties in mind. See figure 9 for plots of these frames and experiments in section 4.4.1 for their performance.

Many frames are steerable, which means equivariant with respect to some group of transformation, they linearize the action of these transformation groups. More formally, if W are the weights of a filter, $g(\tau) \in G$ is some transformation of the input x and the basis Φ is steerable under this transformation, one can find a linear mapping $\mathcal{F}(\tau)$, such that:

$$W\mathcal{F}(\tau)\Phi(x_{j-1}) = W\Phi(g(\tau)x_{j-1}). \tag{25}$$

This means instead of steering the effective filters represented by $W\Phi(x_{j-1})$, it is sufficient to steer the basis Φ and the effective filters will be transformed accordingly. Thus, a change to a steerable basis Φ, makes it possible to dynamically adapt filters via linear projections, while also opening new ways to learn local and global invariants efficiently.

We will show below, that this property permits to dynamically transform filters and to directly learn the degree of invariance to variabilities like rotation, scaling and others, depending on the type of basis used.

Steerability allows to separate a feature's pose from its canonical appearance. From equation 24 follows that a steerable version of an arbitrary filter $W_i(x, y)$ under a k-parameter Lie group can be expressed as:

$$g(\tau)W_i(x, y) = \sum_{n=1}^{N} w_{ng}^{i} g(\tau)\phi_n^i. \tag{26}$$

And by substituting according to equation 25 it follows:

$$g(\tau)W_i(x, y) = \sum_{n=1}^{N} w_n^i \sum_{m=1}^{M} \mathcal{F}_m(\tau)\phi_m(x). \tag{27}$$

Thus it is sufficient to determine the group action on the fixed frame by steering it to separate the canonical feature itself from its k-parameter variants, i.e. ϕ_n^i are the frame coefficients underlying each feature $W_i(x, y)$ and $\mathcal{F}(\tau)_m$ are the steering functions governing the transformation of $g(\tau)$ acting on $W_i(x, y)$ as a whole. In the following section, we will connect this insight to frame-based static residual blocks and turn them into dynamic two-factor blocks, that perform geometrically regularized locally adaptive filtering.

To achieve precise geometrical regularization, one can further derive the steering equations for the particular steerable frame at hand and use the resulting trigonometric functions as activation functions, which is suitable for learning in a CNN. While these activation functions can be omitted in more general tasks like classification where the demand on local transformations is not precisely defined, we will show below that they serve as important regularizers in tasks where precise geometric adaption is needed, as examplified in the boundary detection experiments. For brevity, we moved steering equation derivation and steerability proofs to the supplementary material.

4.3.1 Dynamic Steerable Two-Factor Blocks

Deep Residual Networks (ResNets) are among the best performing current approaches to convolutional networks. Instead of optimizing single layers, they consist of convolutional blocks with skip connections, where the output of block j is defined as $x_j = \mathcal{H}(x_{j-1}) = \mathcal{F}(x_{j-1}) + x_{j-1}$. $\mathcal{F}$ is typically a stack of convolution layers, batch normalizations [82] and nonlinearities [57], [60]. Such residual blocks overcome vanishing gradients and facilitate optimization [83], leading to very simple, but very deep networks with no need for pooling and fully connected layers.

In the following sections, we introduce extensions of residual blocks as two-factor models, that overcome the static nature of typical CNNs by leveraging previously introduced steerable frames. A simple extension transforms static residual blocks into dynamic modules, able to change the geometrical pose of their filters conditioned on the input during training *and* inference time in a locally adaptive fashion. The key perspective the steerable two-factor block relies upon is to change the residual block from being an additive model of the form:

$$\mathcal{H}(x) = \mathcal{F}(x) + Wx, \tag{28}$$

where W is the shortcut projection introduced in a recent improvement [60], to a multiplicative two-factor model of the form:

$$\mathcal{H}(x) = \mathcal{F}(\Phi(x))\Phi(x)W. \tag{29}$$

Here, the factor $\Phi(x)W$ represents a canonical feature, while $\mathcal{F}(\Phi(x))$ represents its pose. The function $\mathcal{F}$ estimates the local pose. The space of possible poses can be pre-conditioned by choosing a suitable frame Φ, that are steerable under pre-defined sets of deformations and thus span an invariant subspace of all transformed versions of the basis itself.

The advantage of our proposed dynamic steerable block over common methods achieving local invariance is that our method goes beyond maximum search in local pose estimation and thus overcomes the inherent ambiguity of local maximum search in many natural images. This is achieved by allowing $\mathcal{F}$ to include context around the current filter location into the estimation, enabling stable and task-dependent pose estimation. Figure 10 illustrates an instance of this problem where the maximum response can not solve the task and a learned pose estimation finding poses that are neither maxima nor minima, is necessary. Note however, that if invariance is harmful for the task at hand, our model can also learn to fall back to standard CNN representations as well.

The blocks used in the boundary detection experiments are based on Gaussian derivatives, steerable with respect to continuous rotations and small ranges of isotropic scalings (σ =0.8-1.5). The Dynamic Two-Factor Block consists of four processing steps. 1) Change from input to frame space by convolving with frame, 2) The interpolator network $\mathcal{F}(\Phi(x))$ estimates local pose from this invariant subspace, outputting a set of pose variables for each location in the image and for each input/output channel (this can be changed depending on application). For the interpolator network $\mathcal{F}$ we used a small network with 8-16 units and three layers with tanh nonlinearities as they seemed suitable to approximate trigonometric steering equation solutions. For scalings, we found softplus and relu nonlinearities to work well. Optional: 3) the steering functions derived in section 2.4 are applied to the pose variable maps and effectively act as nonlinear pose-parametrized activation functions that regularize the interpolation network to output an explicitly interpretable pose space.

Method	ResNet110	ResNet164	DenseK12L40	DenseK12L100
Pixel Basis	6.70± 0.16%	5.88± 0.22%	5.26± 0.19%	4.16 ± 0.18%
Image Frame	**6.11**± 0.19%	**5.33**± 0.15%	**4.97**± 0.18%	**3.78** ± 0.17%
Naive Frame	7.29± 0.31%	6.93± 0.29%	6.40± 0.21%	5.25 ± 0.20%

Table 6: Results of 3 frames on Cifar-10+, reported as average over 5 runs with standard deviation. We compare models with a standard pixel-basis, a steerable frame basis designed for natural images and a naive steerable frame as an example of a frame that does not take natural image statistics into account. The natural image statistics based frame outperforms the pixel-basis consistently, while the naive frame consistently performs about 1% worse than the baseline, highlighting the benefit of a frame suitable for the type of input data.

4) A 1x1 convolution layer is applied to the already transformed frame outputs, this convolution represents the weights w_j^i of each feature governing the canonical appearance of the i_{th} feature map in the j_{th} layer.

4.4 EXPERIMENTS

The experimental section is organized in two parts. Experiment 4.4.1 illustrates that mere replacement of the pixel basis with frames outperforms highly optimized and flexible approaches like ResNets and Densenets, in domains where they excel, given a good frame for the data at hand is known. Note that this part does not explicitly make use of steerability, but focuses on illustrating the effect of various frame choices according to equation 24. Experiment 4.4.2 makes explicit use of steerability, according to equation 27 & 29, illustrating our dynamic steerable block mechanism and applying it on a difficult real-world task of boundary detection, where we outperform all other non pre-trained methods, among which one is rotation invariant.

4.4.1 Generalized Bases on Cifar-10+

To show the effect of replacing the commonly used pixel-basis with frames, we compare different frames in multiple state-of-the-art deep Residual Network [60] and Densenet [61] architectures on the Cifar-10 [84] dataset with moderate data augmentation of crops and flips. We evaluated our approach on two different network sizes that still comfortably train on one GPU over night each. The setup used for the ResNet is as described in [60].

The batch size is chosen to be 64 and we train for 164 epochs with the described learning rate decrease. The ResNet architectures used are without bottlenecks having 56 and 110 layers. For the Densenets we follow [61] and evaluate on the K=12 and L=40, and the K=12 and L=100 models. We run our experiments in Keras [85] and Tensorflow [86]. In the first experiment, we run the models on the standard pixel basis to get a viable baseline. Results are in line or better with the numbers reported by the authors. Secondly, we replace the pixel-basis with widely-used frames that take natural image statistics into account, namely non-orthogonal, overcomplete Gaussian derivatives [5] and non-orthogonal framelets [81] in an alternating fashion, yielding superior performance compared to the pixel-basis by replacement only. We also show that the naively derived $x^p y^q$ frame performs consistently worse than the other two choices, as it does not take natural image properties into account.

The frames used, are shown in figure 9. The results are reported in table 6. The fact that the pixel-basis can be replaced by steerable frames with well-understood properties while performance improves is remarkable. Frame-based CNNs run at approximately the same runtime as vanilla CNNs.

4.4.2 Dynamic Steerable Blocks for Boundary Detection

We evaluate our proposed dynamic steerable two-factor blocks on boundary detection, a natural task for locally adaptive filtering. In the first part, we illustrate properties of the proposed mechanism and in the second part we apply it to a challenging real-world problem, where we outperform competing approaches.

Evaluating Boundary Detection Properties on Textured Blobs

In this experiment we empirically validate the effectiveness of our approach with a single dynamic steerable block by showing that it indeed learns adaptive and non-trivial invariants, that are conditioned on local neighborhoods in the image. To show this, we create an artificial dataset of random blobs, whose boundaries have to be detected by a dynamic steerable block as pixel-wise classification task. The dataset is infinite and created on the fly. In the first case, where blob and background are binary, this task can be solved with a simple gradient magnitude invariant and presents no challenge to our algorithm.

A fully convolutional network baseline of the same size achieves almost the same performance. In the second and third example we sample textures from the KTH TIPS dataset [87] and fill blobs as well as background with different textures each. Here, gradient magnitude either only gives weak clues, or in many cases is unable to find any outline given by the target. In both cases, the fully convolutional baseline fails to converge.

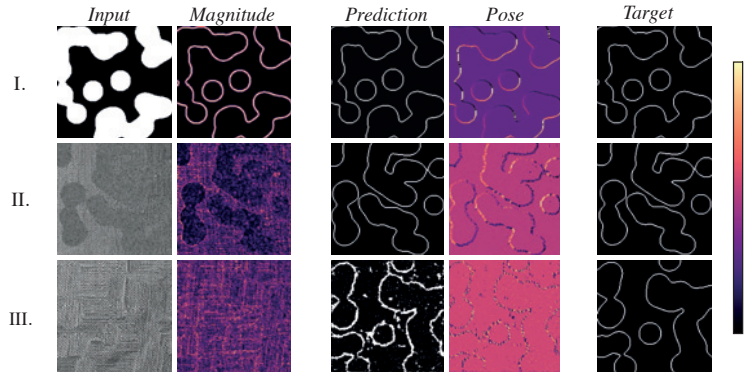


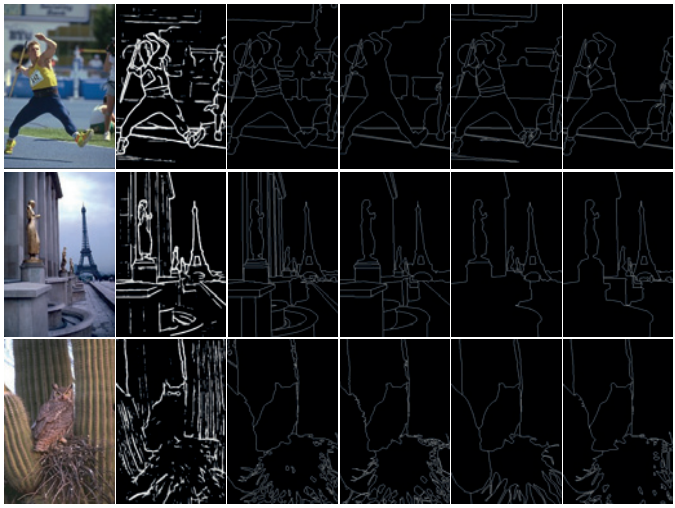
Figure 10: Results of boundary detection experiment to illustrate the workings of our proposed mechanism and its ability to find solutions that go well beyond simple maximum-guided pose invariance, that overcomes the inherent ambiguity present in natural textures. From left to right: Input x to the network; Gradient magnitude of x, as equivalent to max pooling over all rotations of the filter at each location; Pixel-wise predictions of the network $\mathcal{H}(x)$; Pose variables discovered by pose estimation network $\mathcal{F}(\Phi(x))$; Pixel-wise targets. I) Perfectly discovers the simple maximum-guided invariance rule to recover the target. II) More complex local relationships that can not be solved with simple invariants are recovered. Note, the two small blobs on the right, barely visible in input and gradient magnitude map, but still correctly segmented. III) Most challenging scenario, blobs and backgrounds can hardly be distinguished and the gradient magnitude does not contain much useful visible information for the task either. Note, that $\mathcal{F}(\Phi(x))$ learns an alternating pattern of rotation, that overcomes the irregular local gradients and recovers most of the targets successfully.

Remarkably, even though the dynamic steerable block does only receive two Gaussian derivative gradient filters as an input, it still manages to find highly non-linear steering patterns to recover the boundaries. The results are shown in figure 10. We show some unseen inputs alongside the manually calculated gradient magnitude, predictions and associated pose maps (only rotation variable shown) as estimated by the dynamic steerable block. Even in very hard cases, the dynamic block manages to largely recover the labeled boundaries. Results are evidence of the ability of our proposed method to learn adaptive invariants conditioned on the local context in the image in ways that go way beyond simple maximum response guided steering.

Boundary Detection on BSD500

In this experiment we apply the dynamic steerable blocks on a real task in which adaptive invariance is desirable. Recently, a fully equivariant CNN engineered to output locally rotation invariant predictions [7] performed very well on this task. We aim to show, that learned adaptive invariance is even more powerful, as even though final edge prediction is rotation invariant, intermediate processing benefits from a context dependent degree of invariance. We evaluate our method on the contour detection task of the Berkeley Segmentation Dataset. The dataset consists of 500 images divided in a train/val/test split. We tune hyperparameters on the validation set and report results on the test set, following the protocol in [62]. Each image is labelled by 5-7 human annotators, resulting in multiple ground truth maps per image. We follow [7] and merge the different labels by majority vote into one. We minimize the pixelwise binary cross-entropy loss between ground truth and prediction and use class balancing because the classes are heavily imbalanced (many more background pixels than contour pixels).

We use a ResNet model designed for segmentation [91] and reduce its size drastically, due to the limited data-scenario we face. Eventually we use a network with the following structure:
Conv2d[64]$\rightarrow$ 2$\times$(DynResBlock[128]$\rightarrow$StatResBlock[128]) $\rightarrow$ Conv2D[256]
Thus, we alternate static and dynamic blocks as we found that this setup stabilizes training considerably on the validation set. The dynamic blocks are based on Gaussian derivatives and their exact setup is described in the appendix. We report OIS and ODS metrics as in [62], based on F-scores. Due to the subjective aspect of the contour segmentation, the task is ambiguous and the network implicitly has to estimate the level of detail at which to segment the boundaries. While in most cases the network's prediction agrees with at least one of the human annotators (first two rows of figure 5), it sometimes segments boundaries at a too high level of detail (last row of figure 5).



Method	ODS	OIS
DynResNet (Ours)	**0.732**	**0.751**
H-Net [7]	0.726	0.742
ResNet (Ours-Static)	0.720	0.733
Kivininen [7]	0.702	0.715
HED [7]	0.697	0.709
DCNN Pre-trained [88]	0.813	0.831

Figure 11: Results of non pre-trained state-of-the-art models alongside the currently best performing approach with pre-training on BSD500. We show results from our dynResNet, from left to right: Input image; Prediction and 4 different ground truth labels. Our dynamic steerable ResNet outperforms all other methods, including the same model with static blocks, when no pre-training is performed. We also outperform the fully rotation invariant H-Net in accordance to our findings from the texture experiments, where we found that stable, non-maximum guided invariance is likely superior in ambiguous cases. Note, that Kivinen and HED are non-pretrained re-implementations of [89, 90] reported in [7]. When pre-trained, HED and DCNN outperform non-pretrained approaches. For brevity we only report the state-of-the-art pre-trained DCNN.

Our results show that we outperform state-of-the-art approaches when they are not pre-trained on Imagenet, according to re-implementations from [7], including our static ResNet baseline and the explicitly rotation invariant Harmonic Networks approach [7]. However, when pre-trained on Imagenet, DCNN (ODS: 0.813, OIS: 0.831) [88] and Holistically Nested Edge Detection (ODS: 0.782, OIS: 0.804) [90] clearly outperform the non-pre-trained approaches.

In conclusion, we show that a learned piece-wise invariance is superior compared to a hand-crafted maximum response invariance as applied in the Harmonic Networks, supporting our findings from the texture experiments. However, semantic high-level knowledge other models acquire through Imagenet pre-training still seems to include forms of knowledge our geometrical regularization can not overcome, making our results especially interesting in domains where no large datasets for pre-training are available (e.g. medical imaging).

4.5 CONCLUSION

In summary, we have introduced a framework that opens up a range of novel and efficient ways to incorporate geometrical priors and regularization into deep networks, without restricting their theoretical expressiveness.

We introduced the notion of frame-based CNNs and derived dynamic steerable blocks from frame-based residual networks. In summary, we have shown that CNNs based on frames specifically designed for the data at hand facilitate optimization consistently even when data is abundant. On the other hand, our proposed Dynamic Steerable Blocks achieve superior performance when data is limited and are superior when filters that precisely and dynamically adapt to local patterns in non-trivial ways are needed.

5 HIERARCHICAL ATTRIBUTE CNNS

5.1 INTRODUCTION

Deep convolutional neural networks have demonstrated impressive performance for classification and regression tasks over a wide range of generic problems including images, audio signals, but also for game strategy, biological, medical, chemical and physics data [1]. However, their mathematical properties remain mysterious and we are currently not able to relate their performance to the properties of the classification problem.

Classifying signals in high dimension requires to eliminate non-informative variables, and hence contract and reduce space dimensionality in appropriate directions. Convolutional Neural Networks (CNN) discover these directions via backpropagation algorithms [92]. Several studies show numerically that linearization increases with depth [58], but we do not know what type of information is preserved or eliminated. The variabilities which can be eliminated are mathematically defined as the group of symmetries of the classification function [3]. It is the group of transformations which preserves the labels of the classification problem. Translations usually belong to the symmetry group, and invariants to translations are computed with spatial convolutions, followed by a final averaging. Understanding a deep neural network classifier requires specifying its symmetry group and invariants beyond translations, especially of non-geometrical nature.

To achieve this goal, we study highly structured Hierarchical Attribute Convolution Networks (HCNNs) based on mathematical concepts introduced in [3]. Hierarchical Attribute CNNs give explicit information on invariants by disentangling progressively more signal attributes as the depth increases. The deep network operators are multidimensional convolutions along attribute indices. Invariants are obtained by averaging network layers along these attributes. Such a deep network can thus be interpreted as a non-linear mapping of the classification symmetry group into a multidimensional translation group along attributes. Signals are mapped into manifolds which are progressively more flat as depth increases, eventually allowing a simple linear classifier to estimate the class.

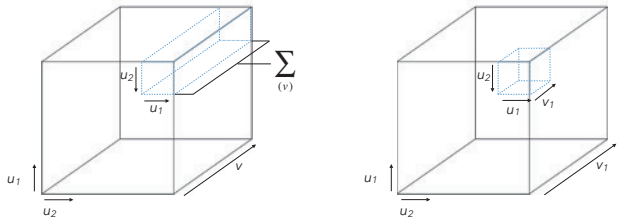


Figure 12: An illustration of the difference between a vanilla CNN (left) and a HCNN layer (right). A CNN computes convolution along u and mixes feature maps via a linear combination along the channel index v. A HCNN performs joint convolution along (u, v_j). This gives rise to an ordering of the feature maps along v_j, leading to semantically meaningful neighborhood relationships as contrasted to arbitrary ordering in a vanilla CNN layer.

Section 5.2 reviews important properties of generic CNN architectures [1]. Section 5.3 describes Hierarchical Attribute CNNs, which are particular CNNs in which linear operators are multidimensional convolutions along progressively more attributes that give rise to an ordering of the channel dimension, see figure 12 for an illustration. Section 5.4 describes an efficient implementation, which reduces inner layers dimensions by computing invariants with an averaging along attributes. Numerical experiments on the CIFAR database show that this hierarchical network obtains comparable performances to state of the art CNN architectures, with a reduced number of parameters. Hierarchical Attribute CNNs are the first type of CNN that generically orders the channel domain and explicitly disentangles attributes of the data. Section 5.5 studies the organization obtained by this deep network. Our proposed architecture provides a mathematical and experimental framework to understand deep neural network classification properties. The numerical results are reproducible and code is available online [1].

5.2 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 [93]. We briefly review their properties.

1 https://github.com/jhjacobsen/HierarchicalCNN

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 index of x, and v is a 1-dimensional channel parameter. A deep convolutional network iteratively computes $x_{j+1} = \rho\, W_{j+1}\, x_j$ with $x_0 = x$. Each W_{j+1} computes sums over v of convolutions along u, with filters of small support. It usually also incorporates a batch normalization [82]. 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 pointwise non-linearity In this work, we shall use exponential linear units ELU [94]. It transforms each coefficient $z(t)$ plus a bias $c = z(t) + b$ into c if $c < 0$ and $e^c - 1$ if $c < 0$.

As the depth increases, the discriminative variations of x along u are progressively transferred to the channel index v. 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 attributes with an arbitrary ordering. Hierarchical Attribute CNNs address 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

$$\Phi_j = \rho\, W_j \ldots \rho\, W_1,$$

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

$$\forall (x, x') \in \mathbb{R}^{2N} \quad \Phi_j(x) = \Phi_j(x') \quad \Rightarrow \quad f(x) = f(x') \,. \tag{30}$$

This is necessary and sufficient to guarantee that there exists a classification function f_j such that $f = f_j \Phi_j$ and hence

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

The function $f(x)$ can be characterized by its groups of symmetries. A group of symmetries of f is a group of operators g which transforms any x into $x' = g.x$ which belong to the same class: $f(x) = f(g.x)$. The discriminative property (30) implies that if $\Phi_j(x) = \Phi_j(g.x)$ then $f(x) = f(g.x)$. The discrimination property (30) is thus equivalent to impose that groups of symmetries of Φ_j are groups of symmetries of f. Learning appropriate change of variables can thus be interpreted as learning progressively more symmetries of f [3].

The network must be sufficiently flexible to compute change of variables Φ_j whose symmetries approximate the symmetries of f.

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.

5.3 HIERACHICAL ATTRIBUTE CONVOLUTION NETWORKS

Hierarchical attribute convolution networks are highly structured convolutional networks. The one-dimensional index v is replaced by a 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 $y = f(x)$. The operators W_j are defined as convolutions along a group which is a parallel transport in the index space (u, v). With no loss of generality, in this implementation, the transport is a multidimensional translation along (u, v). The operators W_j are therefore multidimensional convolutions, which are covariant to translations along (u, v). As previously explained, this covariance to translations implies that the sum $\sum_{v_k} x_j(u, v_0, ..., v_j)$ 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

$$z \star w(u, v) = \sum_{(u', v') \in S} z(u - u', v - v')\, w(u', v') \; . \tag{31}$$

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 accurcies.

The network takes as 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 $w_{1,v_0,v_1}(u)$

$$x_1(u, v_1) = \rho\Big(\sum_{v_0} x(\cdot, v_0) \star w_{1,v_0,v_1}(u) \Big) \; . \tag{32}$$

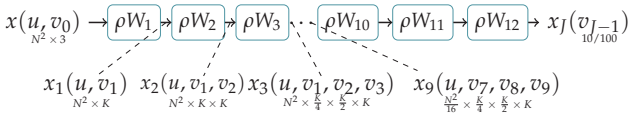


Figure 13: Implementation of a Hierarchical Attribute convolutional network as a cascade of $5D$ convolutions W_j. The figure gives the size of the intermediate layers stored in $5D$ arrays. Dashed dotted lines indicate the parametrization of a layer x_j and its dimension. We only represent dimensions when the output has a different size from the input.

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:

$$x_j(u, v, v_j) = \rho\Big(x_{j-1} \star w_{v_j}(u, v)\Big) . \tag{33}$$

As explained in [3], W_j has two roles. First, these convolutions indexed by v_j prepares the discriminability (30) 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 (33), 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. 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:

$$x_J(v_{J-1}) = \sum_{u, v_1, ..., v_{J-1}} x_{J-1}(u, v_1, ..., v_{J-1}) . \tag{34}$$

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 5.3.1. *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 pointwise operator, it is also covariant to translations. Translating x_j along $(u, v_1, ..., v_j)$ thus translates x_{J-1}. Since (36) 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 $f_j(x_j) = f(x)$ by an operator which is invariant to translations along the high-dimensional index $(u, v) = (u, v_1, ..., v_j)$. The change of variable x_j thus aims at mapping the symmetry group of f into a high-dimensional translation group, which is a flat symmetry group with no curvature. It means that classes of x_j where $f_j(x_j)$ is constant define surfaces which are progressively more flat as j increases. 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 $(v_1, ..., v_j)$ is 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 (32) computes convolutions along u: $x_1(u, v_1) = \rho(x \star w_{v_1}(u))$. 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 either well defined transformations on x 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

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

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 translation $x_j(u, v_1, .., v_j)$ along the attributes $(v_1, ..., v_j)$ usually do not correspond to a transformation of x.

5.4 FAST LOW-DIMENSIONAL ARCHITECTURE

5.4.1 Dimensionality Reduction

Hierarchical Attribute CNN 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 (32) and (33) 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}:

$$w_{v_j}(u, v_{j-3}, v_{j-2}, v_{j-1}) = w_{v_j}(u, v_{j-2}, v_{j-1}) \, . \tag{35}$$

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

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

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 subsmapling factors s_j are adjusted with a trade-off between computations, memory and classification accuracy. The final layer is computed with a sum (36) over all parameters but the last one, which corresponds to the class index:

$$x_J(v_{J-1}) = \sum_{u, v_{J-3}, v_{J-2}} x_{J-1}(u, v_{J-3}, v_{J-2}, v_{J-1}) \, . \tag{36}$$

This architecture is illustrated in Figure 13.

5.4.2 Filter Factorization for Training

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 momentum 0.9 for 240 epochs. An initial learning rate of 0.25 is chosen while being reduced by a factor 10 every 40 epochs. Each minibatch is of size 50. The learning is regularized by a weight decay of $2\,10^{-4}$ [95]. We incorporate a data augmentation with random translations of 6 pixels and flips [96].

Just as in any other CNNs, the gradient descent is badly conditioned because of the large number of parameters [97]. 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:

$$w_{v_j}(u, v_{j-3}, v_{j-2}, v_{j-1}) = \sum_{q=1}^{Q} h_{j,q}(u)\, g_{v_j,q}(v_{j-2}, v_{j-1})\,, \tag{37}$$

and introduce an intermediate normalization before the sum. Let us write $h_{j,q}(u, v) = \delta(u)\, h_{j,q}(u)$ and $g_{v_j,q}(u, v) = \delta(u)\, g_{v_j,q}(v)$. The batch normalization is applied to $x_{j-1} \star h_{j,q}$ and substracts a mean array $m_{j,q}$ while normalizing the standard deviations of all coefficients $\sigma_{j,q}$:

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

This normalized output is retransformed according to (37) by a sum over q and a subsampling:

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

The convolution operator W_j is thus subdivided into a first operator W_j^h which computes standardized convolutions along u cascaded with W_j^g which sums Q 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 [98], 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 relative increase of the classification accuracy of 4.2% and then of 1.1%.

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

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

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.

For classification of CIFAR images, the total depth is $J = 12$ and a downsampling by 2 along u is applied at depth $j = 5, 9$. Figure 13 describes our model architecture as a cascade of W_j and ρ, and gives the size of each layer. Each attribute can take at most $K = 16$ values.

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. 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 number of parameters with architectures that have similar numerical performances.

5.5 AN EXPLICIT STRUCTURATION

This section shows that Hierarchical Attribute CNNs have comparable classification accuracies on the CIFAR image dataset than state-of-the-art architectures, with much fewer parameters. We also investigate the properties of translations along the attributes v_j learned on CIFAR.

5.5.1 Classification Performance

We evaluate our Hierarchical CNN on CIFAR-10 (table 7) and CIFAR-100 (table 8) in the setting explained above. Our network achieves an error of 8.6% on CIFAR-10, which is comparable to recent state-of-the-art 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.

Model	# Parameters	% Accuracy
Hiearchical CNN	0.098M	91.43
Hiearchical CNN (+)	0.34M	92.50
All-CNN	1.3M	92.75
ResNet 20	0.27M	91.25
Network in Network	0.98M	91.20
WRN-student	0.17M	91.23
FitNet	2.5M	91.61

Table 7: Classification accuracy on CIFAR10 dataset.

Classification algorithms using a priori defined representations or representations computed with unsupervised algorithms have an accuracy which barely goes above 80% on CIFAR-10 [47]. On the contrary, supervised CNN have an accuracy above 90% as shown by Table 7. 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 [99]. Our proposed architecture is based on "plain vanilla" CNN architectures to which we compare our results in Table 7. Applying residual connections [57], densely connected layers [61], or similar improvements, might overcome the 4% accuracy gap with the best existing architectures. In the following, we study the properties resulting from the hierarchical structuration of our network, compared with classical CNN.

5.5.2 Reducing the number of parameters

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.

This section compares Hierarchical Attribute CNNs to other structured architectures and algorithms which reduce the number of parameters of a CNN during, and after training. We show that Hierarchical Attribute CNNs involves 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 [100], [101] *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.

Model	# Parameters	% Accuracy
Hiearchical CNN	0.25M	62.01
Hiearchical CNN (+)	0.89M	63.19
All-CNN	1.3M	66.29
Network in Network	0.98M	64.32
FitNet	2.5M	64.96

Table 8: Classification accuracy on CIFAR100 dataset.

Another strategy to reduce the number of network weights is to use teacher and student networks [102], [103], 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 [104], [105]. 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 [98] represent the filters with few parameters in different bases, instead of imposing tha they have a small spatial support. These filters are represented as linear combinations of a given family of filters, for example, computed with derivatives Gaussians. This approach is structuring jointly the channel and spatial dimensions. Finally, HyperNetworks [106] permits to drastically reducing the number of parameters used during the training step, to $0.097M$ and obtaining 91.98% accuracy. However, we do not report them as $0.97M$ corresponds to a non-linear number of parameters for the network.

Table 7 and 8 give the performance of different CNN architectures with their number of parameters, for the CIFAR10 and CIFAR100 datasets. For hierarchical 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 [107] 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 [57] has more parameters and performs similarly whereas it is a more complex architecture, due to the shortcut connexions. WRN-student is a student resnet [102] with 0.2M parameters trained via a teacher using 0.6M parameters and which gets an accuracy of 93.42% on CIFAR10. FitNet networks [103] 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.

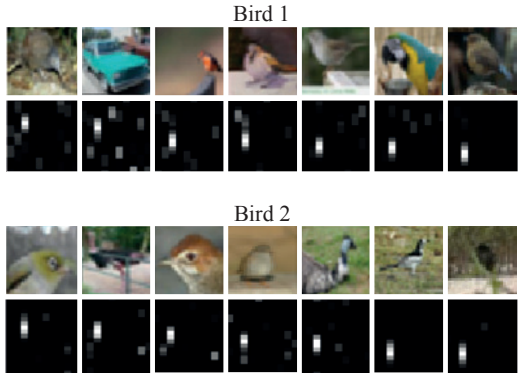


Figure 14: The first images of the first and third rows are the two input image x. Their invariant attribute array $\bar{x}_j(v_{j-1}, v_j)$ is shown below for $j = J - 1$, with high amplitude coefficients appearing as white points. Vertical and horizontal axes correspond respectively to v_{j-1} and 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 databasis.

5.5.3 Interpreting the translation

The structure of Hierarchical Attribute CNNs 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 these attributes translations to modifications of image properties. As explained in Section 5.3, 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 variability along v_{j-2} and define an invariant attribute array by choosing the central spatial position u_0:

$$\bar{x}_j(v_{j-1}, v_j) = \sum_{v_{j-2}} x_j(u_0, v_{j-2}, v_{j-1}, v_j). \tag{38}$$

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 an averaging filter, before computing their translation.

Figure 15: 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 databasis. The value of τ is the column index minus 1.

The network used in this experiment is implemented with circular convolutions to avoid border effects, which have nearly the same classification performance. Figure 14 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 ouptut, we expect that translated images 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 15 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 databasis are obtained with a distance over coefficients which are invariant relatively to all other attributes.

These images are thus very different and involve variabilities relative 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 attribute 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.

5.6 CONCLUSION

Hierarchical Attribute CNNs give a mathematical framework to study invariants computed by deep neural networks. Layers are parameterized in progressively higher dimensional spaces of hierarchical attributes, which are learned from training data. All network operators are multidimensional convolutions along attribute indices, so that invariants can be computed by summations along these attributes.

This paper gives image classification results with an efficient implementation computed with a cascade of 5D convolutions and intermediate non-linearities. Invariant are progressively calculated as depth increases. Good classification accuracies are obtained with a reduced number of parameters compared to other CNN.

Translations along attributes at shallow depth correspond to low-level image properties at fine scales like color, whereas attributes at deep layers correspond to modifications of large scale pattern structures like faces. Understanding better the multiscale properties of these attributes and their relations to the symmetry group of f is an important issue, which can lead to a better mathematical understanding of CNN learning algorithms.

As we did not consistently observe localized coefficients along the channeld domain for all layers at once, the results have to be interpreted with care. We leave further analysis in this direction for future work.

6 i-REVNET: DEEP INVERTIBLE NETWORKS

6.1 INTRODUCTION

A CNN may be very effective in classifying images of all sorts [57, 95], but the cascade of linear and nonlinear operators reveals little about the contribution of the internal representation to the classification. The learning process is characterized by a steady reduction of large amounts of uninformative variability in the images while simultaneously revealing the essence of the visual class. It is widely believed that this process is based on progressively discarding uninformative variability about the input with respect to the problem at hand [12–15]. However, the extent to which information is discarded is lost somewhere in the intermediate non-linear processing steps. In this paper, we aim to provide insight into the variability reduction process by proposing an invertible convolutional network, that does not discard any information about the input.

The difficulty to recover images from their hidden representations is found in many commonly used network architectures [14, 15]. This poses the question if a substantial loss of information is necessary for successful classification. We show information does not have to be discarded. By using homeomorphic layers, the invariance can be built only at the very last layer via a projection.

In [12], minimal sufficient statistics are proposed as a candidate to explain the reduction of variability. [108] introduces the information bottleneck principle which states that an optimal representation must reduce the mutual information between an input and its representation to reduce as much uninformative variability as possible. At the same time, the network should maximize the mutual information between the desired output and its representation to effectively preserve each class from collapsing onto other classes. The effect of the information bottleneck was demonstrated on small datasets in [12], [13].

However, in this work, we show it is not a necessary condition and we build a cascade of homeomorphic layers, which preserves the mutual information between input and hidden representation and shows that the loss of information can only occur at the final layer. This way we demonstrate that a loss of information can be avoided while maintaining discriminability, even for large-scale problems like ImageNet. One way to reduce variability is progressive contraction with respect to a meaningful ℓ^2 metric in the intermediate representations.

Several works [58, 109] observed a phenomenon of progressive separation and contraction in non-invertible networks on limited datasets. Those progressive improvements can be interpreted as the creation of progressively stronger invariants for classification. Ideally, the contraction should not be too brutal to avoid removing important information from the intermediate signal. This shows that a good trade-off between discriminability and invariance has to be progressively built. In this paper, we extend some findings of [58], [109] to ImageNet [110] and, most importantly, show that a loss of information is not necessary for observing a progressive contraction.

The duality between invariance and separation of the classes is discussed in [3]. Here, intra-class variabilities are modeled as Lie groups that are processed by performing a parallel transport along those symmetries. Filters are adapted through learning to the specific bias of the dataset and avoid to contract along discriminative directions. However, using groups beyond the Euclidean case for image classification is hard. Mainly because groups associated with abstract variabilities are difficult to estimate due to their high-dimensional nature, as well as the appropriate degree of invariance required. An illustration of this framework on the Euclidean group is given by the scattering transform [111], which builds invariance to small translations while being recoverable to a certain extent. In this work, we introduce a network that cannot discard any information except at the final classification stage, while we demonstrate numerically progressive contraction and separation of the signal classes.

We introduce the *i*-RevNet, an invertible deep network.[1] *i*-RevNets retain all information about the input signal in any of their intermediate representations up until the last layer. Our architecture builds upon the recently introduced RevNet [17], where we replace the non-invertible components of the original RevNets by invertible ones. *i*-RevNets achieve the same performance on Imagenet compared to similar non-invertible RevNet and ResNet architectures [17, 57].

To shed light on the mechanism underlying the generalization-ability of the learned representation, we show that *i*-RevNets progressively separate and contract signals with depth. Our results are evidence for an effective reduction of variability through a contraction with a recoverable input obtained from a series of one-to-one mappings.

1 Code is available at: https://github.com/jhjacobsen/pytorch-i-revnet

6.2 RELATED WORK

Several recent works show that significant information about the input images is lost with depth in successful Imagenet classification CNNs [14, 15]. To understand the loss of information, the references propose to invert the representations by means of learned or hand-engineered priors. The approximate inversions indicate increased geometric and photometric invariance with depth. Multiple other works report progressive properties of deep networks that may be linked to discarded information in the representations as well, such as linearization [112], linear separability [58], contraction [109] and low-dimensional embeddings [113]. However, it is not clear from above observations if the loss of information is a necessity for the observed progressive phenomena. In this work, we show that progressive separation and contraction can be obtained while at the same time allowing an exact reconstruction of the signal.

Multiple frameworks have been introduced that permit to learn invertible representations under certain conditions. Parseval networks [114] have been introduced to increase the robustness of learned representations with respect to adversarial attacks. In this framework, the spectrum of convolutional operators is constrained to norm 1 during learning. The linear operator is thus injective.

As a consequence, the input of Parseval networks can be recovered if but only if the built-in non-linearities are invertible as well, which is typically not the case. [16] derive conditions under which pooling representations are, but our method directly overcomes this issue. The Scattering transform [111] is an example of predefined deep representation, approximately invariant to translations, that can be reconstructed when the degree of invariance specified is small. Yet, it requires a gradient descent optimization and no guarantee of convergences are known. In summary, the references make clear that invertibility requires special care in designing the architecture or special care in designing the optimization procedure. In this paper, we introduce a network, that overcomes these issues and has an exact inverse by construction.

Our main inspiration for this work is the recent reversible residual network (RevNet), introduced in [17]. RevNets are in turn closely related to NICE and Real-NVP architectures [18, 19], which make use of constrained Jacobian determinants for generative modeling. All these architectures are similar to the lifting scheme [115] and Feistel cipher diagrams [116], as we will show. RevNets illustrate how to build invertible ResNet-type blocks that avoid storing intermediate activations necessary for the backward pass. However, RevNets still employ multiple non-invertible operators like max-pooling and downsampling operators as part of the network. As such, RevNets are not invertible by construction. In this paper, we show how to build an invertible type of RevNet architecture that performs competitively with RevNets on Imagenet, which we call *i*-RevNet for invertible RevNet.

6.3 THE i-REVNET

This section introduces the general framework of the i-RevNet architecture and explains how to explicitly build an inverse or a left-inverse to an i-RevNet. Its practical implementation is discussed, and we demonstrate competitive numerical results.

6.3.1 An invertible architecture

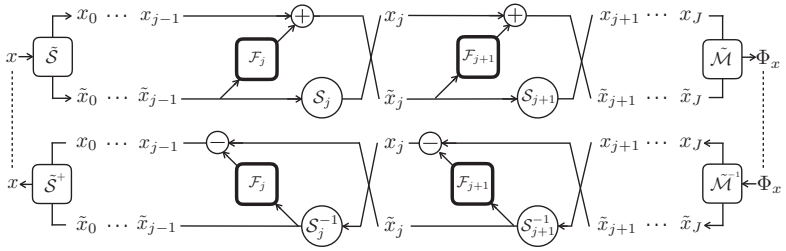


Figure 16: The main component of the i-RevNet and its inverse. RevNet blocks are interleaved with convolutional bottlenecks $\mathcal{F}_j$ and reshuffling operations $\mathcal{S}_j$ to ensure invertibility of the architecture and computational efficiency. The input is processed through a splitting operator $\tilde{\mathcal{S}}$, and output is merged through $\tilde{\mathcal{M}}$. Observe that the inverse network is obtained with minimal adaptations.

We describe i-RevNets in their general setting. Their foundations are largely grounded in the recent RevNet architecture [17]. In an i-RevNet, an initial input is split into two sublayers $(x_0, \tilde{x}_0)$ of equal size, thanks to a splitting operator $\tilde{\mathcal{S}}x \triangleq (x_0, \tilde{x}_0)$, in this paper we choose to split the channel dimension as is done in RevNets. The operator $\tilde{\mathcal{S}}$ is linear, injective, reduces the spatial resolution of the coefficients and can potentially increase the layer size, as wider layers usually improve the classification performance [117]. We can thus build a pseudo inverse $\tilde{\mathcal{S}}^+$ that will be used for the inversion. Recall that if $\tilde{\mathcal{S}}$ is invertible, then $\tilde{\mathcal{S}}^+ = \tilde{\mathcal{S}}^{-1}$.

The number of coefficients of the next block is maintained, and at each depth j, the representation $\Phi_j x$ is again decoupled into two variables $\Phi_j x \triangleq (x_j, \tilde{x}_j)$ that play interlaced roles.

The strategy implemented by an i-RevNet consists in an alternation between additions, and non-linear operators $\mathcal{F}_j$, while progressively down-sampling the signal thanks to the operators $\mathcal{S}_j$. Here, $\mathcal{F}_j$ consists of convolutions and non-linearity on $\tilde{x}_j$. The pair of the final layer is concatenated through a merging operator $\tilde{\mathcal{M}}$. We will omit $\tilde{\mathcal{M}}, \tilde{\mathcal{M}}^{-1}, \tilde{\mathcal{S}}^+$ and $\tilde{\mathcal{S}}$ for the sake of simplicity, when not necessary.

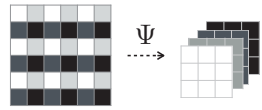


Figure 17: Illustration of the invertible down-sampling

Figure 16 describes the blocks of an i-RevNet. The design is similar to the Feistel cipher diagrams [116] or a lifting scheme [115], which are invertible and efficient implementations of complex transforms like second generation wavelets.

In this way, we avoid the non-invertible modules of a RevNet (e.g. max-pooling or strides) which are necessary to train them in a reasonable time and are designed to build invariance w.r.t. translation variability. Our method shows we can replace them by linear and invertible modules $\mathcal{S}_j$, that can reduce the spatial resolution (we refer to it as a spatial down-sampling for the sake of simplicity) while maintaining the layer's size by increasing the number of channels.

We keep the computational cost manageable by tightly coupling downsampling and increase in width of the network. Reducing the spatial resolution can be undesirable, so $\mathcal{S}_j$ can potentially be the identity. We refer to such networks as i-RevNets. This leads to the following equations:

$$\begin{cases} x_{j+1} = \mathcal{S}_{j+1}\tilde{x}_j \\ \tilde{x}_{j+1} = x_j + \mathcal{F}_{j+1}\tilde{x}_j \end{cases} \iff \begin{cases} \tilde{x}_j = \mathcal{S}_{j+1}^{-1} x_{j+1} \\ x_j = \tilde{x}_{j+1} - \mathcal{F}_{j+1}\tilde{x}_j \end{cases} \tag{39}$$

Our downsampling layer can be written for u the spatial variable and λ the channel index:

$$\mathcal{S}_j x(u, \lambda) = x(\Psi(u, \lambda))$$

where Ψ is some invertible mapping. In principle, any invertible downsampling operation like e.g. dilated convolutions [118] can be considered here. We use the inverse of the operation described in [119] as illustrated in Figure 17, since it preserves roughly the spatial ordering, and thus permits to avoid mixing different neighborhoods via the next convolution. $\tilde{\mathcal{S}}$ is similar, but also linearly increases the channel dimensionality, for example by concatenating 0.

The final layer $\Phi x \triangleq \Phi_J x = (x_J, \tilde{x}_J)$ is then averaged along the spatial dimension, followed by a ReLU non-linearity and finally a linear projection on the class probes, which are fed to a supervised training algorithm. From a given i-RevNet, it is possible to define a left-inverse Φ^+, i.e. $\Phi^+\Phi x = x$ or even an inverse Φ^{-1}, i.e. $\Phi^{-1}\Phi x = \Phi^{-1}\Phi x = x$ if $\tilde{\mathcal{S}}$ is invertible. In these cases, the convolutional sections are as well some i-RevNets.

Architecture	Injective	Bijective	Top-1 error	Parameters
ResNet	-	-	24.7	26M
RevNet	-	-	25.2	28M
i-RevNet (a)	yes	-	24.7	181M
i-RevNet (b)	yes	yes	26.7	29M

Table 9: Comparison of different architectures trained on ILSVRC-2012, in terms of classification accuracy and number of parameters

An i-RevNet is the dual of its inverse, in the sense that it requires to replace $(\mathcal{S}_j, \mathcal{F}_j)$ by $(\mathcal{S}_j^{-1}, -\mathcal{F}_j)$ at each depth j, and to apply $\tilde{\mathcal{S}}^+$ on the output. In consequence, its implementation is simple and specified by Equation (39). In Subsection 6.4.2, we discuss that the inverse of Φ does not suffer from significant round-off errors, while however being very sensitive to small variations of an input on a large subspace, as shown in Subsection 6.4.1.

6.3.2 Architecture, training and performances

In this subsection, we describe two models that we trained: an injective i-RevNet (a) and a bijective i-RevNet (b), with fewer parameters. The hyperparameters were selected to be either close to the ResNet and RevNet baselines in terms of the number of layers (a) or parameters (b) while keeping performance competitive. For the same reasons as in [17], our scheme also allows avoiding storing any intermediate activations at training time, making memory consumption for very deep i-RevNets not an issue in practice. We compare our implementation with a RevNet with 56 layers corresponding to $28M$ parameters, as provided in the open source release of [17], and with a standard ResNet of 50 layers, with $26M$ parameters [57].

Each block $\mathcal{F}_j$ is a bottleneck block, which consists of a succession of 3 convolutional operators, each preceded by Batchnormalization [82] and ReLU non-linearity. The second layer has four times fewer channels than the other two, while their corresponding kernel sizes are respectively $1 \times 1, 3 \times 3, 1 \times 1$.

The final representation is spatially averaged and projected onto the 1000 classes after a ReLU non-linearity. We now discuss how we progressively decrease the spatial resolution, while increasing the number of channels per layer by use of the operators $\mathcal{S}_j$.

We first describe the model (a), that consists of 56 layers which have been optimized to match the performances of a RevNet or a ResNet with approximatively the same number of layers. In particular, we explain how we progressively decrease the spatial resolution, while increasing the number of channels per block by use of the operators $\mathcal{S}_j$.

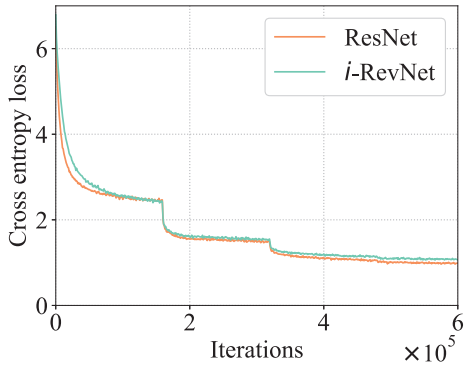


Figure 18: Training loss of the i-RevNet (b), compared to the ResNet, on ImageNet.

The splitting operator $\tilde{\mathcal{S}}$ consists in a linear and injective embedding that downsamples by a factor 4^2 the spatial resolution by increasing the number of output channels from 48 to 96 by simply adding 0. The latter permits to increase the initial layer size, and consequently, the size of the next layers as performed in [17]; it is thus not a bijective yet an injective i-RevNet. At depth j, $\mathcal{S}_j$ allows us to reduce the number of computations while maintaining good classification performance. It will correspond to a downsampling operator respectively at the depth $3j = 15, 27, 45$ (3j as one block corresponds to three layers), similar to a normal RevNet. The spatial resolution of these layers is reduced by a factor 2^2 while increasing the number of channels by a factor of 4 respectively to 48, 192, 768 and 3072. Furthermore, it means that the corresponding spatial resolutions for an input of size 224^2 are respectively $112^2, 56^2, 28^2, 14^2, 7^2$. The total number of coefficients at each layer is then about $0.3M$. All the remaining blocks $\mathcal{S}_j$ are kept fix to the identity as explained in the section above.

Architecture (b) is bijective, it consists of 300 layers (100 blocks), whose total numbers of parameters have been optimized to match those of a RevNet with 56 layers. Initially, the input is split via $\tilde{\mathcal{S}}$, which corresponds to an invertible spatial downsampling of 2^2 that increases the number of channels from 3 to 12. It thus keeps the dimension constant and permits building a bijective i-RevNet. Then, at depth $3j = 3, 21, 69, 285$, the spatial resolution is reduced by 2^2 via $\mathcal{S}_j$. Contrary to the architecture (a), the dimensionality of each layer is constantly equal to 3×224^2, until the final layer, with channel sizes of 24, 96, 384, 1536.

For both networks, the training on Imagenet follows the same setup as [17]. We train with SGD and momentum of 0.9. We regularized the model with a ℓ^2 weight decay of 10^{-4} and batch normalization. The dataset is processed for 600k iterations on a batch size of 256, distributed on 4GPUs.

The initial learning rate is 0.1, dropped by a factor of ten every 160k iterations. The dataset was augmented according to [17]. The images values are mapped to $[0, 1]$ while following geometric transformations were applied: random scaling, random horizontal flipping, random cropping of size 224^2, and finally color distortions. No other regularizations were incorporated into the classification pipeline. At test time, we rescale the image size to 256^2 and perform a center crop of size 224^2.

We report the training loss (i.e. Cross entropy) curves in Figure 18 of our i-RevNet (b) and the ResNet baseline, displayed is a moving average over 100 iterations. Observe that the decrease of both training-losses are very similar which indicates that the constraint of invertibility does not interfere negatively with the learning process. However, we observed one third longer wall-clock times for i-RevNets compared to plain RevNets because the channel size becomes larger. The Table **??** reports the performances of our i-RevNets, with comparable RevNet and ResNet. First, we compare the i-RevNet (a) with the RevNet and ResNet. Indeed, those CNNs have the same number of layers, and the i-RevNet (a) increases the channel width of the initial layer as done in [17]. The drawback of this technique is that the kernel sizes will be larger for all subsequent layers.

The i-RevNet (a) has about 6 times more parameters than a RevNet and a ResNet but leads to a similar accuracy on the validation set of ImageNet. On the contrary, the i-RevNet (b) is designed to have roughly the same number of parameters as the RevNet and ResNet, while being bijective. Its accuracy decreases by 1.5% absolute percent on ImageNet compared to the RevNet baseline, which is not surprising because the number of channels was not drastically increased in the earlier layers as done in the baselines [17, 57, 95]; we did not explore wide ranges of hyper-parameters, thus the gap between (a) and (b) can likely be reduced with additional engineering.

6.4 ANALYSIS OF THE INVERSE

We now analyze the representation Φ built by our bijective neural network i-RevNet (b) and its inverse Φ^{-1}, as trained on ILSVRC-2012. We first explain why obtaining Φ^{-1} is challenging, even locally. We then discuss the reconstruction, while displaying in the image space linear interpolations between representations.

6.4.1 An ill-conditioned inversion

In the previous section, we have described the i-RevNet architecture, that permits defining a deep network with an explicit inverse.

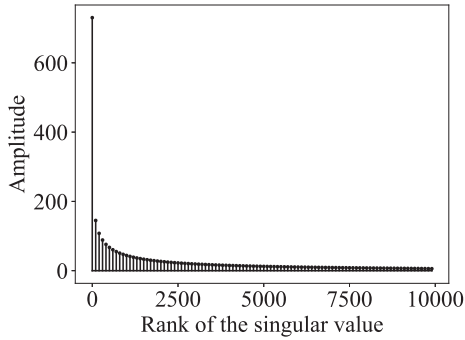


Figure 19: Normalized sorted singular values of $\partial\Phi_x$.

We explain now why this is normally difficult, by studying its local inversion. We study the local stability of a network Φ and its inverse Φ^{-1} w.r.t. to its input, which means that we will quantify locally the variations of the network and its inverse w.r.t. to small variations of an input. As Φ is differentiable (and its inverse as well), an equivalent way to perform this study is to analyze the singular values of the differential $\partial\Phi$ at some point, as for (a, b) close the following holds:

$$\Phi a \approx \Phi b + \partial\Phi_b(a - b).$$

Ideally, a well-conditioned operator has all its singular values constant equal to 1, for instance as achieved by the isometric operators of [114].

In our numerical application to an image x, $\partial\Phi_x$ corresponds to a very large matrix (square of the number of coefficients of the image at least) whose computations are expensive. Figure 19 corresponds to the singular values of the differential (i.e. the square roots of the eigen values of $\partial\Phi^*\partial\Phi$), in decreasing order, for a given natural image from ImageNet. The example we plot is typical of the behavior of $\partial\Phi$. Observe there is a fast decay: numerically, the first 10^3 and 10^4 singular values are responsible respectively for 80% and 97% of the cumulated energy (i.e. sum of squared singular values). This indicates Φ linearizes the space locally in a considerably smaller space in comparison to the original input dimension. However, the dimensionality is still quite large (i.e. > 10) and thus we can not infer that Φ lays locally in a low-dimensional manifold. It also proves that inversing Φ is difficult and is an ill-conditioned problem. Thus obtaining implicitly this inverse would be a challenging task that we avoided, thanks to the formal reconstruction algorithm provided by Subsection 6.3.1.

6.4.2 Linear interpolation and reconstruction

Visualizing or understanding the important directions in the representation of inner layers of a CNN, and in particular, the final layer is complex because typically the cascade is either not invertible or unstable. One approach to reconstruct from an output layer consists in finding the input image that matches the activation through via gradient descent. However, this technique leads only to a partial or informal reconstruction [120].

Another method consists in embedding the representation in a lower dimensional space and comparing the common attributes of nearest neighbors [121]. It is also possible to train a CNN to reconstruct the representation [14]. Yet these methods require a priori knowledge in order to find the appropriate embeddings or training sets. We now discuss the improvements achieved by the i-RevNet.

Our main claim is that while the local inversion is ill-conditioned, the inverse Φ^{-1} computations do not involve significant round-off errors. The forward pass of the network does not seem to suffer from significant instabilities, thus it seems coherent to assume that this will hold for Φ^{-1} as well. For example, adding constraints beyond vanishing moments in the case of a Lifting scheme is difficult [9, 115], and this is a weakness of this method. We validate our claim by computing the empirical relative error on several subsets $\mathcal{X}$ of data:

$$\epsilon(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \frac{\|x - \Phi^{-1}\Phi x\|}{\|x\|}$$

We evaluate this measure on a subset $\mathcal{X}_1$ of $|\mathcal{X}_1| = 10^4$ independent uniform noises and on the validation set $\mathcal{X}_2$ of ImageNet. We report $\epsilon(\mathcal{X}_1) = 5 \times 10^{-6}$ and $\epsilon(\mathcal{X}_2) = 3 \times 10^{-6}$ respectively, which are close to the machine error and indicates that the inversion does not suffer from significant round-off errors.

Given a pair of images $\{x^0, x^1\}$, we propose to study linear interpolations between the pair of representations $\{\Phi x^0, \Phi x^1\}$, in the feature domain. Those interpolations correspond to existing images as Φ^{-1} is an exact inverse. We reconstruct a convex path between two input points; it means that if:

$$\phi^t = t\Phi x^0 + (1-t)\Phi x^1,$$

then:

$$x^t = \Phi^{-1}\phi^t$$

is a signal that corresponds to an image.

We discretized $[0, 1]$ into $\{t_1, ..., t_k\}$, adapt the step size manually and reconstruct the sequence of $\{x^{t_1}, ..., x^{t_k}\}$. Results are displayed in the Figure 20. We selected images from the basel face dataset [122], describable texture dataset [123] and imagenet.

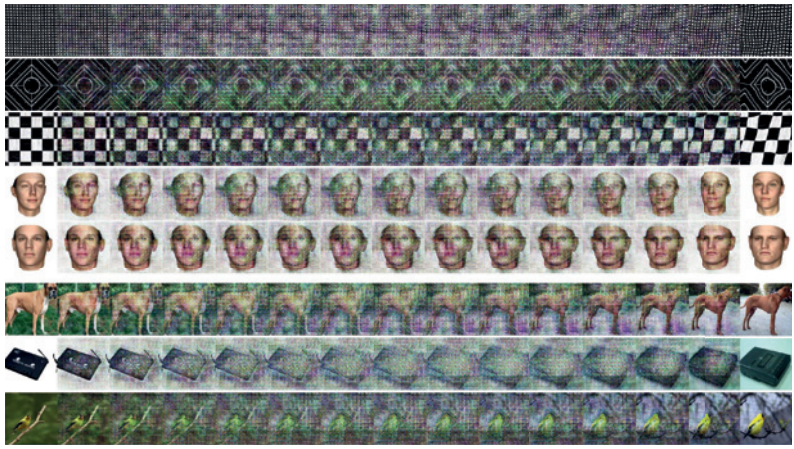


Figure 20: This graphic displays several reconstructed sequences $\{x^t\}_t$. The left image corresponds to x^0 and the right image to x^1.

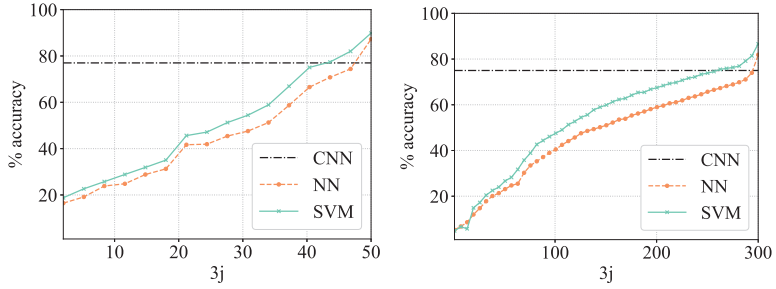
We now interpret the results. First, observe that a linear interpolation in the feature space is not a linear interpolation in the image space and that intermediary images are noisy, even for small deformations, yet they mostly remain recognizable. However, some geometric transformations such as a 3D-rotation seem to have been linearized, as suggested in [113]. In the next section, we thus investigate how the linear separation progresses with depth.

6.5 A CONTRACTION

In this section, we study again the bijective *i*-RevNet. We first show that a localized or linear classifier progressively improves with depth. Then, we describe the linear subspace spanned by Φ, namely the *feature space*, showing that the classification can be performed on a much smaller subspace, which can be built via a PCA.

6.5.1 Progressive linear separation and contraction

We show that both a ResNet and an *i*-RevNet build a progressively more linearly separable and contracted representation as measured in [109]. Observe this property holds for the *i*-RevNet despite the fact that it can not discard any information.



(a) ResNet **(b)** i-RevNet

Figure 21: Accuracy at depth j for a linear SVM and a 1-nearest neighbor classifier applied to the spatially averaged Φ_j.

We investigate these properties in each block, with the following experimental protocol. To reduce the computational burden we used a subset of 100 randomly selected imagenet classes, that consist of $N = 120k$ images, and keep the same subset during all our following experiments. At each depth j, we extract the features $\{\Phi_j x^n\}_{n \leq N}$ of the training set, we average them along the spatial variable and standardize them in order to avoid any ill-conditioning effects. We used both a nearest neighbor classifier and a linear SVM. The former is a localized classifier that indicates that the ℓ^2 metric is progressively more important for classification, while a linear SVM measures the linear separation of the different classes. The parameters of the linear SVM are cross-validated on a small subset of the training set, prior to training on the 100 classes. We evaluate both classifiers for each model on the validation set of ImageNet and report the Top-1 accuracy in Figure 21.

We observe that both classifiers progressively improve similarly with depth for each model, the linear SVM performing slightly better than the nearest neighbor classifier because it is the more robust and discriminative classifier of the two. In the case of the i-RevNet, the classification performed by the CNN leads to 77%, and the linear SVM performs slightly better because we did not fine-tune the model to 100 classes. Observe that there is a more intense jump of performance on the 3 last layers, which seems to indicate that the former layers have prepared the representation to be more contracted and linearly separated for the final layers.

The results suggest a low-dimensional embedding of the data, but this is difficult to validate as estimating local dimensionality in high dimensions is an open problem. However, in the next section, we try to compute the dimension of the discriminative part of the representation built by an i-RevNet.

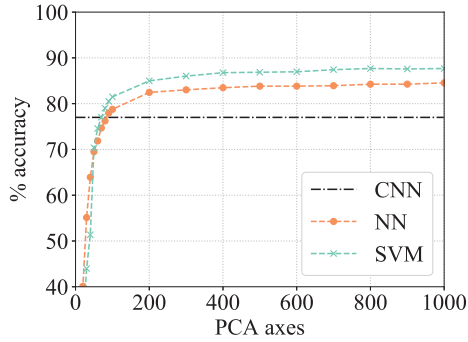


Figure 22: Accuracy of a linear SVM and nearest neighbor against the number of principal components retained.

6.5.2 Dimensionality analysis of the feature space

In this section, we investigate if we can refine the dimensionality of informative variabilities in the final layer of an *i*-RevNet. Indeed, the cascade of convolutional operators has been trained on the training set to separate the 1000 different classes while being a homeomorphism on its feature space. Thus, the dimensionality of the feature space is potentially large.

As shown in the previous subsection, the final layer is progressively prepared to be projected on the final probes corresponding to the classes. This indicates that the non-informative variabilities for classification can be removed via a linear projection on the final layer Φ, which lie in a space of dimension 1000, at most. However, this projection has been built via supervision, which can still retain directions that have been contracted and thus will not be selected by an algorithm such as PCA. We show in fact a PCA retains the necessary information for classification in a small subspace.

To do so, we build the linear projectors π_d on the subspace of the d first principal components, and we propose to measure the classification power of the projected representation with a supervised classifier, e.g. nearest neighbor or a linear SVM, on the previous 100 class task. Again, the feature representation $\{\Phi x^n\}_{n \leq N}$ are spatially averaged to remove the translation variability, and standardized on the training set. We apply both classifiers, and we report the classification accuracy of $\{\pi_d \Phi x^n\}_{n \leq N}$ w.r.t. to d on the Figure 22. A linear projection removes some information that can not be recovered by a linear classifier, therefore we observe that the classification accuracy only decreases significantly for $d \leq 200$.

This shows that the signal indeed lies in a subspace much lower dimensional than the original feature dimensions that can be extracted simply with a PCA that only considers directions of largest variances, illustrating a successful contraction of the representation.

6.6 CONCLUSION

Invertible representations and their relationship to loss of information are on the agenda of deep learning for some time. Understanding how transformations in feature space are related to the corresponding input is an important step towards interpretable deep networks, invertible deep networks may play an important role in such analysis since, for example, one could potentially back-track a property from the feature space to the input space. To the best of our knowledge, this work provides the first empirical evidence that learning invertible representations that do not discard any information about their input on large-scale supervised problems is possible.

To achieve this we introduce the *i*-RevNet class of CNN which is fully invertible and permits to exactly recover the input from its last convolutional layer. *i*-RevNets achieve the same classification accuracy in the classification of complex datasets as illustrated on ILSVRC-2012, when compared to the RevNet [17] and ResNet [57] architectures with a similar number of layers. Furthermore, the inverse network is obtained for free when training an *i*-RevNet, requiring only minimal adaption to recover inputs from the hidden representations.

The absence of loss of information is surprising, given the wide believe, that discarding information is essential for learning representations that generalize well to unseen data. We show that this is not the case and propose to explain the generalization property with empirical evidence of progressive separation and contraction with depth, on ImageNet.

7 CONCLUSION

7.1 SUMMARY OF CONTRIBUTIONS

Deep convolutional networks are a core ingredient of modern artificial intelligence systems and the workhorse of computer vision. Their shortcomings are, however, their lack of transparency and hunger for data. To overcome these issues, it is important to understand how they achieve their impressive performance and what the underlying core principles they exploit are. This work magnifies, structures and analyzes various aspects of learned representations, including low-level, mid-level and global properties and associated invariants. In the search for principles underlying good generalization, we have introduced various types of structured CNNs.

In chapter 3 we focus on the fact that learning powerful feature representations with CNNs is hard when training data are limited. Pre-training is one way to overcome this, but it requires large datasets sufficiently similar to the target domain. Another option is to design priors into the model, which can range from tuned hyperparameters to fully engineered representations like Scattering Networks. We combine these ideas into structured receptive field networks, a model which has a fixed filter basis and yet retains the flexibility of CNNs. This flexibility is achieved by expressing receptive fields in CNNs as a weighted sum over a fixed basis which is similar in spirit to Scattering Networks. The key difference is that we learn arbitrary effective filter sets from the basis rather than modeling the filters. This approach explicitly connects classical multiscale image analysis with general CNNs and answers research question 1. With structured receptive field networks, we improve considerably over unstructured CNNs for small and medium dataset scenarios as well as over Scattering for large datasets. We validate our findings on ILSVRC2012, Cifar-10, Cifar-100 and MNIST. As a realistic small dataset example, we show state-of-the-art classification results on popular 3D MRI brain-disease datasets where pre-training is difficult due to a lack of large public datasets in a similar domain.

In chapter 4 we generalize and extend the ideas of 3 and investigate the generalized notion of frames designed with image properties in mind, as alternatives to this parametrization.

We show that frame-based ResNets and Densenets can improve performance on Cifar-10+ consistently while having additional pleasant properties like steerability. By exploiting these transformation properties explicitly, we arrive at dynamic steerable blocks. They are an extension of residual blocks, that are able to seamlessly transform filters under pre-defined transformations, conditioned on the input at training and inference time. Dynamic steerable blocks learn the degree of invariance from data and locally adapt filters, allowing them to apply a different geometrical variant of the same filter to each location of the feature map. When evaluated on the Berkeley Segmentation contour detection dataset, our approach outperforms all competing approaches that do not utilize pre-training. Our results highlight the benefits of image-based regularization to deep networks and answer research question 2.

Chapter 3 and chapter 4 focused on the spatial structure of filters and associated symmetries. In chapter 5, we introduce structure into deep neural network algorithms that sheds light on the channel domain. We introduce a new class of convolutional networks, called Hierarchical Attribute CNNs, to answer research question 3. They are the first type of CNN that generically orders the channel dimension of each layer. This is done by formulating layers as a cascade of multi dimensional convolutions. Convolution not only along space but along channels as well, permits to build a hierarchical ordering of the channel axes of each layer, which we call attribute indexes. Constraining channels to be organized highly regularizes the network. First, our Hierarchical Attribute CNNs require a reduced number of parameters compared to vanilla CNNs to achieve comparable performance. Secondly, we show that one can investigate intermediate layers features and define notions of proximity that corresponds to semantic meaning of increasing complexity as depth increases. Our results present first steps towards organizing the memory of deep networks to increase our understanding of their properties and the invariants they compute. We release code to reproduce our experiments, with good performance on CIFAR10 and CIFAR100.

In the final chapter 6 we question the wide believe that the success of deep convolutional networks is based on progressively discarding uninformative variability about the input with respect to the problem at hand. This has been supported empirically by the difficulty of recovering images from their hidden representations, in most commonly used network architectures. To answer research question 4, we show via a one-to-one mapping that this loss of information is not a necessary condition to learn representations that generalize well to complicated problems, such as ImageNet. Via a cascade of homeomorphic layers, we build the *i*-RevNet, a network that can be fully inverted up to the final projection onto the classes, i.e. no information is discarded.

Building an invertible architecture is difficult, for one, because the local inversion is ill-conditioned, we overcome this by providing an explicit inverse. An analysis of *i*-RevNets learned representations suggests an alternative explanation for the success of deep networks by a progressive contraction and linear separation with depth. To shed light on the nature of the model learned by the *i*-RevNet we reconstruct linear interpolations between natural image representations.

7.2 CONCLUDING REMARKS

The current work has expanded the boundaries of the understandable structure that can be introduced into discriminative deep networks while keeping them general enough to tackle realistic and large-scale computer vision problems. Our work is one step on the way to a point where deep networks are not merely a "black art", but become principled architectures that can be tested and designed more rigorously. We have shown how incorporating well-understood mathematical structure can aid small and large data performance, as well as interpretability of deep networks.

Despite the fact that there has been much progress in understanding and structuring deep networks, it is still a long way to go before these algorithms can be truly understood. Beyond the many benefits of incorporating well-known structure into deep CNNs, our work suggests, that low-dimensional geometry and concepts like loss of information and invariance are not sufficient to explain generalization in deep networks. There is currently no theory available that can reliably explain the success of these models on complex problems. Understanding the mathematical structure of the contraction and separation that occurs throughout the succession of layers remains an unsolved problem. But it is an important challenge, as understanding how such a system can create a notion of proximity even between exorbitantly different instances of the same object class, will enable us to have algorithms that can not only learn but also reveal their powerful heuristics to us. Beyond the mere insight into how deep networks learn, such an understanding would have a wide impact on many application they struggle with. In the mid-term, this understanding will allow leveraging unlabeled data for semi-supervised learning, to have models that can learn from less annotated data. Further, it will make it possible to learn more well-behaved representations, that have quantifiable robustness and failure-modes. In the long-term, it might give science as a whole a powerful tool to understand how experience can be efficiently organized to reveal hidden regularities of the data that can be extracted from these models as a learned analog to scientific hypotheses or theories.

BIBLIOGRAPHY

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *Arxiv preprint arxiv:1611.03530*, 2016.

[3] S. Mallat, "Understanding deep convolutional networks," *Phil. trans. r. soc. a*, vol. 374, no. 2065, p. 20 150 203, 2016.

[4] J. J. Koenderink, "The structure of images," *Biological cybernetics*, vol. 50, no. 5, pp. 363–370, 1984.

[5] L. M. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Scale and the differential structure of images," *Image and vision computing*, vol. 10, no. 6, pp. 376–388, 1992.

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. the proceedings of the seventh ieee international conference on*, Ieee, vol. 2, 1999, pp. 1150–1157.

[7] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic networks: Deep translation and rotation equivariance," *Arxiv preprint arxiv:1612.04642*, 2016.

[8] M. Weiler, F. A. Hamprecht, and M. Storath, "Learning steerable filters for rotation equivariant cnns," *Arxiv preprint arxiv:1711.07289*, 2017.

[9] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.

[10] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *Tpami*, no. 9, pp. 891–906, 1991.

[11] M. Michaelis and G. Sommer, "A lie group approach to steerable filters," *Pattern recognition letters*, vol. 16, no. 11, pp. 1165–1174, 1995.

[12] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *Arxiv preprint arxiv:1703.00810*, 2017.

[13] A. Achille and S. Soatto, "On the emergence of invariance and disentangling in deep representations," *Arxiv preprint arxiv:1706.01350*, 2017.

[14] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 4829–4837.

[15] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *International journal of computer vision*, vol. 120, no. 3, pp. 233–255, 2016.

[16] J. Bruna, A. Szlam, and Y. LeCun, "Signal recovery from pooling representations," *Arxiv preprint arxiv:1311.4025*, 2013.

[17] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," *Arxiv preprint arxiv:1707.04585*, 2017.

[18] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *Arxiv preprint arxiv:1605.08803*, 2016.

[19] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *Arxiv preprint arxiv:1410.8516*, 2014.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Ijcv*, pp. 1–42, 2015.

[22] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *Cvpr*, 2014.

[23] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Nips*, 2014.

[24] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[25] T. Lindeberg, *Scale-space theory in computer vision*. Springer Science & Business Media, 2013, vol. 256.

[26] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–eccv 2014*, 2014.

[27] J. J. Koenderink and A. J. van Doorn, "Representation of local geometry in the visual system," *Biological cybernetics*, vol. 55, no. 6, pp. 367–375, 1987.

[28] A. P. Witkin, "Scale-space filtering," in *International joint conference on artificial intelligence*, 1983.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Arxiv:1409.4842*, 2014.

[30] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *Tpami*, vol. 35, no. 8, pp. 1872–1886, 2013.

[31] S. Mallat, "Group invariant scattering," *Communications on pure and applied mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[32] L. Sifre and S. Mallat, "Rotation, scaling and deformation invariant scattering for texture discrimination," in *Cvpr*, 2013.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the ieee*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Nips*, 2012.

[35] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Computer vision and pattern recognition (cvpr), 2014 ieee conference on*, IEEE, 2014, pp. 1701–1708.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[37] J Moody, S Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in neural information processing systems*, vol. 4, pp. 950–957, 1995.

[38] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Cvpr*, 2014.

[39] M. Lin, Q. Chen, and S. Yan, "Network in network," *Arxiv:1312.4400*, 2013.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Arxiv:1409.1556*, 2014.

[41] B. M. H. Romeny, *Front-end vision and multi-scale image analysis: Multi-scale computer vision theory and applications, written in mathematica*. Springer Science & Business Media, 2008, vol. 27.

[42] P. Perona, "Steerable-scalable kernels for edge detection and junction analysis," in *Eccv*, Springer, 1992, pp. 3–18.

[43] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *Arxiv preprint arxiv:1408.5093*, 2014.

[44] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: New features and speed improvements," *Arxiv preprint arxiv:1211.5590*, 2012.

[45] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *Arxiv preprint arxiv:1301.3557*, 2013.

[46] M. A. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer vision and pattern recognition, 2007. cvpr'07. ieee conference on*, IEEE, 2007, pp. 1–8.

[47] E. Oyallon and S. Mallat, "Deep roto-translation scattering for object classification," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 2865–2873.

[48] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 3367–3375.

[49] R. Cuingnet, E. Gerardin, J. Tessieras, G. Auzias, S. Lehéricy, M.-O. Habert, M. Chupin, H. Benali, O. Colliot, A. D. N. Initiative, *et al.*, "Automatic classification of patients with alzheimer's disease from structural mri: A comparison of ten methods using the adni database," *Neuroimage*, vol. 56, no. 2, pp. 766–781, 2011.

[50] M. Jenkinson, C. F. Beckmann, T. E. Behrens, M. W. Woolrich, and S. M. Smith, "Fsl," *Neuroimage*, vol. 62, no. 2, pp. 782–790, 2012.

[51] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Arxiv preprint arxiv:1412.6980*, 2014.

[52] W. Yang, R. L. Lui, J.-H. Gao, T. F. Chan, S.-T. Yau, R. A. Sperling, and X. Huang, "Independent component analysis-based classification of alzheimer's mri data," *Journal of alzheimer's disease: Jad*, vol. 24, no. 4, p. 775, 2011.

[53] A. Payan and G. Montana, "Predicting alzheimer's disease: A neuroimaging study with 3d convolutional neural networks," *Corr*, vol. abs/1502.02506, 2015. [Online]. Available: http://arxiv.org/abs/1502.02506.

[54] A. Gupta, M. Ayhan, and A. Maida, "Natural image bases to represent neuroimaging data," in *Proceedings of the 30th international conference on machine learning (icml-13)*, 2013, pp. 987–994.

[55] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open access series of imaging studies (oasis): Cross-sectional mri data in young, middle aged, nondemented, and demented older adults," *Journal of cognitive neuroscience*, vol. 19, no. 9, pp. 1498–1507, 2007.

[56] Y. Chen, J. Storrs, L. Tan, L. J. Mazlack, J.-H. Lee, and L. J. Lu, "Detecting brain structural changes as biomarker from magnetic resonance images using a local feature based svm approach," *Journal of neuroscience methods*, vol. 221, pp. 22–31, 2014.

[57] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[58] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[59] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *Arxiv preprint arxiv:1603.05027*, 2016.

[61] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *Arxiv preprint arxiv:1608.06993*, 2016.

[62] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *Ieee transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[63] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in neural information processing systems*, 2015, pp. 2449–2457.

[64] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," *Arxiv preprint arxiv:1511.06464*, 2015.

[65] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, and A. Szlam, "A mathematical motivation for complex-valued convolutional networks," *Neural computation*, 2016.

[66] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *Ieee transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[67] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. Smeulders, "Structured receptive fields in cnns," 2016.

[68] T. S. Cohen and M. Welling, "Group equivariant convolutional networks," *Arxiv preprint arxiv:1602.07576*, 2016.

[69] S. Dieleman, J. De Fauw, and K. Kavukcuoglu, "Exploiting cyclic symmetry in convolutional neural networks," *Arxiv preprint arxiv:1602.02660*, 2016.

[70] T. S. Cohen and M. Welling, "Steerable cnns," *Arxiv preprint arxiv:1612.08498*, 2016.

[71] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *Ieee transactions on pattern analysis and machine intelligence*, vol. 13, no. 9, pp. 891–906, 1991.

[72] E. P. Simoncelli and W. T. Freeman, "The steerable pyramid: A flexible architecture for multi-scale derivative computation.," in *Icip (3)*, 1995, pp. 444–447.

[73] Y. Hel-Or and P. C. Teo, "Canonical decomposition of steerable functions," *Journal of mathematical imaging and vision*, vol. 9, no. 1, pp. 83–95, 1998.

[74] M. Unser and N. Chenouard, "A unifying parametric framework for 2d steerable wavelet transforms," *Siam journal on imaging sciences*, vol. 6, no. 1, pp. 102–135, 2013.

[75] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman, "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks," *Arxiv preprint arxiv:1607.02586*, 2016.

[76] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International conference on artificial neural networks*, Springer, 2011, pp. 44–51.

[77] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[78] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," *Arxiv preprint arxiv:1703.06211*, 2017.

[79] B. De Brabandere, X. Jia, T. Tuytelaars, and L. Van Gool, "Dynamic filter networks," *Arxiv preprint arxiv:1605.09673*, 2016.

[80] O. Christensen, *An introduction to frames and riesz bases*. Springer, 2003, vol. 7.

[81] I. Daubechies, B. Han, A. Ron, and Z. Shen, "Framelets: Mra-based constructions of wavelet frames," *Applied and computational harmonic analysis*, vol. 14, no. 1, pp. 1–46, 2003.

[82] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[83] K. Greff, R. K. Srivastava, and J. Schmidhuber, "Highway and residual networks learn unrolled iterative estimation," *Arxiv preprint arxiv:1612.07771*, 2016.

[84] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[85] F. Chollet, *Keras*, https://github.com/fchollet/keras, 2015.

[86] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *Arxiv preprint arxiv:1603.04467*, 2016.

[87] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh, "On the significance of real-world conditions for material classification," *Computer vision-eccv 2004*, pp. 253–266, 2004.

[88] I. Kokkinos, "Pushing the boundaries of boundary detection using deep learning," *Arxiv preprint arxiv:1511.07386*, 2015.

[89] J. Kivinen, C. Williams, and N. Heess, "Visual boundary prediction: A deep neural prediction network and quality dissection," in *Artificial intelligence and statistics*, 2014, pp. 512–521.

[90] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 1395–1403.

[91] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Arxiv preprint arxiv:1611.10080*, 2016.

[92] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[93] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Circuits and systems (iscas), proceedings of 2010 ieee international symposium on*, IEEE, 2010, pp. 253–256.

[94] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *Arxiv preprint arxiv:1511.07289*, 2015.

[95] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[96] A. Krizhevsky and G Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, 2010.

[97] I. J. Goodfellow, O. Vinyals, and A. M. Saxe, "Qualitatively characterizing neural network optimization problems," *Arxiv preprint arxiv:1412.6544*, 2014.

[98] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. Smeulders, "Structured receptive fields in cnns," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 2610–2619.

[99] D. Budden, A. Matveev, S. Santurkar, S. R. Chaudhuri, and N. Shavit, "Deep tensor convolution on multicores," *Arxiv preprint arxiv:1611.06565*, 2016.

[100] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[101] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *Arxiv preprint arxiv:1405.3866*, 2014.

[102] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *Arxiv preprint arxiv:1612.03928*, 2016.

[103] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *Arxiv preprint arxiv:1412.6550*, 2014.

[104] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 2857–2865.

[105] Z. Lu, V. Sindhwani, and T. N. Sainath, "Learning compact recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2016 ieee international conference on*, IEEE, 2016, pp. 5960–5964.

[106] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *Arxiv preprint arxiv:1609.09106*, 2016.

[107] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *Arxiv preprint arxiv:1412.6806*, 2014.

[108] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Information theory workshop (itw), 2015 ieee*, IEEE, 2015, pp. 1–5.

[109] E. Oyallon, "Building a regular decision boundary with deep networks," *Arxiv preprint arxiv:1703.01775*, 2017.

[110] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[111] S. Mallat, "Group invariant scattering," *Communications on pure and applied mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[112] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *Arxiv preprint arxiv:1511.06434*, 2015.

[113] M. Aubry and B. C. Russell, "Understanding deep features with computer-generated imagery," in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 2875–2883.

[114] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *International conference on machine learning*, 2017, pp. 854–863.

[115] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," *Siam journal on mathematical analysis*, vol. 29, no. 2, pp. 511–546, 1998.

[116] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[117] S. Zagoruyko and N. Komodakis, "Wide residual networks," *Arxiv preprint arxiv:1605.07146*, 2016.

[118] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *Arxiv preprint arxiv:1511.07122*, 2015.

[119] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[120] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[121] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *Arxiv preprint arxiv:1312.6199*, 2013.

[122] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *Advanced video and signal based surveillance, 2009. avss'09. sixth ieee international conference on*, Ieee, 2009, pp. 296–301.

[123] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2014, pp. 3606–3613.

8 SAMENVATTING – SUMMARY IN DUTCH

Diepe Convolutionale Neurale Netwerken (CNN) zijn belangrijke ingredient gebruikt in veel moderne AI-systemen en het werkpaard van computervisie. Hun tekortkomingen zijn echter hun gebrek aan transparantie en honger naar data. Om deze problemen op te lossen, is het belangrijk om te begrijpen hoe zij hun indrukwekkende prestaties bereiken en wat de onderliggende kernprincipes zijn die zij gebruiken. Dit werk vergroot, structureert en analyseert verschillende aspecten van geleerde representaties, inclusief low-level, mid-level en global eigenschappen en geassocieerde invarianten. Bij het zoeken naar principes die ten grondslag liggen aan een goede generalisatie, hebben we verschillende soorten gestructureerde CNNs geïntroduceerd.

In hoofdstuk 3 concentreren we ons op het feit dat het leren van krachtige functie-representaties met CNNs moeilijk is wanneer de hoeveelheid training data beperkt is. Pre-training is een manier om dit te ondervangen, maar het vereist grote datasets die vergelijkbaar zijn met het doeldomein. Een andere optie is om priors in het model te verwerken. Deze kunnen variren van getunede hyperparameters tot volledig ontworpen representaties zoals Scattering Networks. We combineren deze ideen in een gestructureerde receptieve veldnetwerk. Het is een model dat een vaste filterbasis heeft en toch de flexibiliteit van CNNs behoudt. Deze flexibiliteit wordt bereikt door receptieve velden in CNNs uit te drukken als een gewogen som over een vaste basis die vergelijkbaar is met het concept van Scattering Networks. Het belangrijkste verschil is dat we arbitraire effectieve filtersets van de basis leren in plaats van de filters te modelleren. Deze benadering verbindt expliciet de klassieke 'multiscale image-analysis' met algemene CNNs. Met structured receptive field networks doen we het aanzienlijk beter ten opzichte van ongestructureerde CNNs voor scenario's van kleine en middelgrote datasets, evenals over Scattering voor grote datasets. We hebben onze resultaten op ILSVRC2012, Cifar-10, Cifar-100 en MNIST gevalideerd. Als een realistisch voorbeeld van een kleine dataset, laten we de resultaten van een state-of-the-art classificatie op een 3D MRI-hersenscan datasets zien. Hierbij is pre-training moeilijk vanwege een gebrek aan grote openbare datasets in een vergelijkbaar domein.

In hoofdstuk 4 generaliseren we de ideen van 3 en breiden ze uit. We onderzoeken de algemene notie van frames die rekening houden met de afbeeldingseigenschappen als alternatieven voor deze parametrisering. We laten zien dat frame-based ResNets en Densenets de prestaties op Cifar-10+ consequent kunnen verbeteren, terwijl ze tegelijkertijd extra aangename eigenschappen zoals 'steerability' hebben. Door deze transformatie-eigenschappen expliciet te gebruiken in ons model, komen we tot 'dynamic steerable blocks'. Ze zijn een uitbreiding van 'residual blocks' (zoals in ResNets), die naadloos in staat zijn om filters te transformeren onder vooraf gedefinieerde transformaties, geconditioneerd op de invoer tijdens de training en inferentietijd. Dynamic steerable blocks leren de mate van invariantie in data en passen de filters lokaal aan, waardoor ze een andere geometrische variant van hetzelfde filter op elke locatie van de featuremap kunnen toepassen. Bij evaluatie op de Berkeley Segmentation contour detection dataset, overtreft onze aanpak alle concurrerende benaderingen die geen gebruik maken van pre-training. Onze resultaten benadrukken de voordelen van 'image-based regularization' voor diepe netwerken.

Hoofdstuk 3 en hoofdstuk 4 richten zich op de ruimtelijke structuur van filters en bijbehorende symmetrien. In hoofdstuk 5 introduceren we structuur in deep neural network algoritmen die het licht werpen op het kanaaldomein. We introduceren een nieuwe klasse van convolutionele netwerken, genaamd Hiearchical Attribute CNNs. Dit zijn het eerste type van CNNs dat op een generieke manier de kanaaldimensie van elke laag sorteert. Dit wordt gedaan door lagen te formuleren als cascade van multi-dimensionale convoluties. Een convolutie, niet alleen in de ruimte, maar ook langs de kanalen, maakt het mogelijk om een hirarchische structuur van de kanaalassen van elke laag te maken, die we 'attribute indexes' noemen. Beperkende kanalen die georganiseerd moeten worden zorgen voor een grote regularisatie van het netwerk. Ten eerste vereisen onze hirarchische kenmerkende CNNs een kleiner aantal parameters in vergelijking met standaard CNNs om vergelijkbare prestaties te bereiken. Ten tweede laten we zien dat we intermediaire lagen kunnen onderzoeken en begrippen van nabijheid kunnen definiren die overeenkomen met de algemene semantische betekenis van toenemende complexiteit naarmate de diepte toeneemt. Onze resultaten laten zien hoe het organiseren van geheugen in diepe netwerken ons een beter inzicht kunnen geven in hun eigenschappen en de invarianten die berekend worden.

In het laatste hoofdstuk zetten we vraagtekens bij de breed-gedragen overtuiging dat het succes van diepe convolutionele netwerken gebaseerd is op het geleidelijk wegnemen van niet-informatieve variaties over de input. Dit wordt empirisch ondersteund door de moeilijkheid om afbeeldingen te herstellen van hun verborgen representaties, in de meest gebruikte netwerkarchitecturen.

In dit artikel laten we via een n-op-n-toewijzing zien dat dit informatieverlies geen noodzakelijke voorwaarde is om representaties te leren die goed zijn voor gecompliceerde problemen, zoals ImageNet. Via een cascade van homeomorfe lagen bouwen we het *i*-RevNet, een netwerk dat volledig inverteerbaar is tot de uiteindelijke projectie op de klassen, d.w.z. geen informatie wordt weggegooid. Het bouwen van een inverteerbare architectuur is moeilijk, omdat de lokale inversie slecht is geconditioneerd, en dit wordt overwonnen door een expliciete inverse te gebruiken. Een analyse van geleerde representaties van *i*-RevNet suggereert een alternatieve verklaring voor het succes van diepe netwerken, namelijk door een progressieve contractie en lineaire scheiding met diepte. Om licht te werpen op de aard van het door het *i*-RevNet geleerde model reconstrueren we lineaire interpolaties tussen natuurlijke beeldrepresentaties.

9 ACKNOWLEDGEMENTS

I would like to thank my supervisor Arnold Smeulders for giving me the chance and freedom to do a PhD on the topics that interested me, accompanied by the right amount of guidance and advice to not get lost along the way. I have learned and developed a lot in these years. I would like to thank Jan van Gemert to help me kickstart in a field I hadnt known existed a couple of months earlier. I would also like to thank Edouard Oyallon and Stéphane Mallat for giving me the chance to collaborate with my favorite research lab, it was a true honor. Our discussions were formative for my view on many problems and I wouldnt want to miss them. A big thank you also to Bert de Brabandere and Patrick Putzky for exciting collaborations.

Furthermore, I would like to thank the many nice people I have met and become friends with on the corridors of science park 904: Svetlana, Agni, Kandan, Karen, Max, Kyriacos, Peter, Berkay, Patrick, Mijung, Spencer, Pascal, Stratis, William, Devanshu, Gjiorgi, Dennis, Ran, Zhenyang, Amir, Stevan and many more from our adjacent labs. I would also like to thank all the great people that I have spent way too little time with over the last years, for still being my friends and always being a source of inspiration and enjoyment in my life. Robert, David, Robert, Hans, Pico, Matze, Arian, Mende, Jenny, Bina, Malte, Jonas, Arif, David, Eva, Magda, Freddy, Magda, Johannes, Abe, Guido, Esrah, Ludwig, Heiko, Steffen, Maxi, Florian, Jan, Berry and if I forgot you here, its not personal, you know who you are.

Most importantly, I would like to thank Mandy and my family. Mandy, thank you for your patience, support, and interest in my work throughout all the years of my PhD, I couldnt have done it without you and I wouldnt have wanted to do it without you. And a big thank you to the rest of my family for all your support, I am very grateful to have you all!

Amsterdam, thank you for being such a beautiful place and home, I will miss you and your free spirit. To go full circle, I close this thesis with the first thing Arnold taught me in my PhD:

”This is the Netherlands, here you can do whatever you want to do.”

PS: I like to think I did ;-)