Université de Montréal

Parametric Scattering Networks

par

Shanel Gauthier

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

May 12, 2022

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Parametric Scattering Networks

présenté par

Shanel Gauthier

a été évalué par un jury composé des personnes suivantes :

Gauthier Gidel

(président-rapporteur)

Irina Rish et Guy Wolf

(directeur de recherche)

Aaron Courville

(membre du jury)

Résumé

La plupart des percées dans l'apprentissage profond et en particulier dans les réseaux de neurones convolutifs ont impliqué des efforts importants pour collecter et annoter des quantités massives de données. Alors que les mégadonnées deviennent de plus en plus répandues, il existe de nombreuses applications où la tâche d'annoter plus d'un petit nombre d'échantillons est irréalisable, ce qui a suscité un intérêt pour les tâches d'apprentissage sur petits échantillons.

Il a été montré que les transformées de diffusion d'ondelettes sont efficaces dans le cadre de données annotées limitées. La transformée de diffusion en ondelettes crée des invariants géométriques et une stabilité de déformation. Les filtres d'ondelettes utilisés dans la transformée de diffusion sont généralement sélectionnés pour créer une trame serrée via une ondelette mère paramétrée. Dans ce travail, nous étudions si cette construction standard est optimale. En nous concentrant sur les ondelettes de Morlet, nous proposons d'apprendre les échelles, les orientations et les rapports d'aspect des filtres. Nous appelons notre approche le *Parametric Scattering Network*. Nous illustrons que les filtres appris par le réseau de diffusion paramétrique peuvent être interprétés en fonction de la tâche spécifique sur laquelle ils ont été entrainés. Nous démontrons également empiriquement que notre transformée de diffusion paramétrique partage une stabilité aux déformations similaire à la transformée de diffusion traditionnelle. Enfin, nous montrons que notre version apprise de la transformée de diffusion génère des gains de performances significatifs par rapport à la transformée de diffusion standard lorsque le nombre d'échantillons d'entrainement est petit. Nos résultats empiriques suggèrent que les constructions traditionnelles des ondelettes ne sont pas toujours nécessaires.

Mots clés : Apprentissage en profondeur, Données étiquetées limitées, Apprentissage sur peu d'échantillons, Transformées de diffusion, Ondelettes

Abstract

Most breakthroughs in deep learning have required considerable effort to collect massive amounts of well-annotated data. As big data becomes more prevalent, there are many applications where annotating more than a small number of samples is impractical, leading to growing interest in small sample learning tasks and deep learning approaches towards them.

Wavelet scattering transforms have been shown to be effective in limited labeled data settings. The wavelet scattering transform creates geometric invariants and deformation stability. In multiple signal domains, it has been shown to yield more discriminative representations than other non-learned representations and to outperform learned representations in certain tasks, particularly on limited labeled data and highly structured signals. The wavelet filters used in the scattering transform are typically selected to create a tight frame via a parameterized mother wavelet. In this work, we investigate whether this standard wavelet filterbank construction is optimal. Focusing on Morlet wavelets, we propose to learn the scales, orientations, and aspect ratios of the filters to produce problem-specific parameterizations of the scattering transform. We call our approach the *Parametric Scattering Network*. We illustrate that filters learned by parametric scattering networks can be interpreted according to the specific task on which they are trained. We also empirically demonstrate that our parametric scattering transforms share similar stability to deformations as the traditional scattering transforms. We also show that our approach yields significant performance gains in small-sample classification settings over the standard scattering transform. Moreover, our empirical results suggest that traditional filterbank constructions may not always be necessary for scattering transforms to extract useful representations.

Keywords: Deep Learning, Limited Labeled Data, Few-sample learning, Scattering Transforms, Wavelets

Contents

List of tables

List of figures

List of acronyms and abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
LS	Learnable Scattering
ML	Machine Learning
MLP	Multilayer Perceptron
ReLU	Rectified Linear Activation Function
SGD	Stochastic Gradient Descent
TF	Tight Frame
WRN	Wide Residual Network

Acknowledgements

Throughout my masters, I received a great deal of support and encouragement from numerous people. First, I would like to thank my supervisor, Dr. Irina Rish, for welcoming me to her team and trusting my abilities. I also want to thank my co-supervisor, Dr. Guy Wolf, for his mentorship, encouragement and academic advice. Thank you for taking a chance on me. You gave me insightful feedback and comments that prompted me to refine my thinking.

I would also like to thank Dr. Eugene Belilovky and Dr. Michael Eickenberg, who collaborated on the research project. Thank you for supporting the project and giving thoughtful feedback. I am grateful for your guidance and insights.

Next, I would like to thank all my collaborators Benjamin Thérien, Laurent Alcène-Racicot and Muawiz Chaudary, who made this work possible. They devoted time and effort to the research project. I am impressed by their incredible work ethic. It was really fun working with all of them.

I would also like to thank my collaborators from the CHUM (Center Hospitalier de Montréal) including Dr. An Tang, Dr. Emmanuel Montagon, Pedro de Oliveira Vianna, Hongliang Li, Sara-Ivana Calce, Boyan Fan, Laurent Patry-Beaudoin and Cassandra Larocque. It was a pleasure working with all of you.

None of this would have been possible without the support and encouragement of my parents and my boyfriend, Justin Charbonneau. During the most stressful times, they were there for me. They always encouraged me to continue my studies. They have been tremendously supportive. Finally, I would like to thank my friend Jessica Bergeron for reminding me to take breaks and have fun when I was stressed.

Introduction

In the past decade, we have witnessed tremendous breakthroughs in deep learning. Most of the breakthroughs did involve massive amounts of labeled data. As big data becomes more prevalent, there are many applications where annotating more than a small number of samples is infeasible. The limitation of labeled data has sparked a growing interest in small sample learning tasks and deep learning approaches towards them [**23, 17, 18, 24**]. Scattering transforms have been shown to be useful in applications involving scarcely annotated or limited labeled data [**26, 128, 108, 44**].

The scattering transform, proposed in [**93**], is a cascade of wavelets and complex modulus nonlinearities, which can be thought as a convolutional neural network (CNN) with predetermined filters. It has been shown to give impressive results on problems involving highly structured signals [**26, 107, 9, 129, 62, 61, 44, 7, 131, 112**], outperforming several other classic signal processing techniques. Since scattering transforms are instantiations of CNNs, they have been studied as mathematical models for understanding the impressive success of CNNs in image classification [**26, 94**]. A theoretical and empirical study of information encoded in scattering networks indicates that they often promote linear separability, which leads to effective representations for downstream classification tasks [**26, 106, 6, 44**].

Recent work has shown that, in image classification, state-of-the-art results can be obtained by hybrid networks that harness the scattering transform as their first layers followed by learned layers based on a wide residual network architecture [**108**]. In this thesis, we further advance this research by proposing to use the scattering network not only as fixed preprocessing layers in a concatenated architecture but also as a parametric prior to learn filters in a CNN. It allows us to also shed light on whether the standard wavelet construction [**92**] is an optimal approach for building filterbanks from a mother wavelet for discriminative tasks.

Recall that the scattering construction is based on complex wavelets, generated from a mother wavelet via dilations and rotations, aimed to cover the frequency plane while having the capacity to encode informative variability in input signals [**26**]. Further, discrete parameterization and indexing of these operations (i.e., by dilation scaling or rotation angle) have traditionally been carefully constructed to ensure the resulting filter bank forms an efficient tight frame [**92, 93**] with well-established energy preservation properties. On the other hand, it has been observed that the first layers of convolutional networks resemble wavelets but may not necessarily form a tight frame [**80**]. The question then arises: is it necessary to use the standard wavelet filterbank construction? Here, we relax the traditional construction by considering another alternative where a small number of wavelet parameters used to create the wavelet filterbanks are optimized for the task at hand. In other words, we let the scattering network learn the optimal parameters of each wavelet in an end-to-end differentiable architecture.

Contribution

In the thesis, we propose to learn the scales, orientations, and aspect ratios of the filters to produce problem-specific parameterizations of the scattering transform. We call our approach the *Parametric Scattering Network*. To our knowledge, this is the first work that aims to learn the wavelet filters of scattering networks in 2D signals. Our main findings and contributions can be summarized as follows:

- We empirically demonstrate that our parametric scattering network shares similar stability to deformations as the traditional scattering transform.
- We illustrate that filters learned by parametric scattering networks can be interpreted for the task at hand.
- We empirically demonstrate that the parametric scattering network trained in a supervised and unsupervised manner yields significant performance gains in small-sample classification settings over the standard scattering network.
- Our empirical results suggest that traditional filterbank constructions may not always be necessary for scattering transforms to extract effective representations.

Outline

The thesis is organized as follows. Chapter 1 provides the background information needed to understand the fundamentals of deep learning (DL). We describe deep neural networks and cover different deep learning architectures. We also introduce the concepts of underfitting and overfitting and the difference between supervised and unsupervised learning. In Chapter 2, we describe the challenge of limited labeled data and describe various techniques to overcome it. In Chapter 3, we revisit the scattering transform. We start by defining the stability to deformations using the Lipschitz continuity condition. We then define the formulation of the scattering transform and discuss its properties. We also introduce different variations of the scattering transform and describe different packages that implement the transformation. In Chapter 4, we introduce the parametric scattering transform. We describe two scattering parameter initializations and how to differentiate through this parametrized scattering transform. We describe the experimental protocol used to evaluate the parametric scattering network and explore the different filter construction schemes. In Chapter 5, we evaluate the robustness of the parametric scattering networks to deformations. We then demonstrate the advantages of our approach in limited labeled data settings and study the adaptation of the wavelet parameters towards a supervised task. We also investigate the adaptation of the parametrized scattering using an unsupervised objective. Finally, in Chapter 6, we summarize our contributions and discuss future work.

Working Paper

Chapter 3, 4 and 5 are based on the working paper called *Parametric Scattering Networks* [**48**], which has been accepted for publication at the 2022 Conference on Computer Vision and Pattern Recognition (CVPR). As the first co-author of the paper, I contributed to all the stages, including literature review, implementation, and paper writing. Further, this work has also previously been presented at the 2021 Mathematical Theory of Deep Learning conference. The code accompanying the work is available on `https://github.com/sgaut023/kymatio_mod`. The content of this link contains complete instructions on how to run the code to reproduce the results presented in this thesis.

Funding Acknowledgment

During my master's degree, I received a scholarship funded by an IVADO grant (PRF-2019-6121178858, PI Tang) administered by CHUM (Centre Hospitalier de l'Université de Montréal). The Faculty of Arts and Sciences awarded me a scholarship in the field of artificial intelligence in December 2020. I was also awarded the NSERC CGS-M scholarship from May 2021 to April 2022. The CHUM and Mila (Quebec Artificial Intelligence Institute) provided me with computer equipment and resources to carry out the various experiments.

Chapter 1

Background

Over the past few years, we have seen a lot of interest and enthusiasm around Artificial Intelligence (AI) and, more specifically, around deep learning (DL). Geoff Hinton, a pioneer in AI, even claimed that deep learning would be able to do everything [**54**]. With such claims, one might wonder what deep learning is. Deep learning is a subset of Machine Learning (ML) and AI, as shown in Figure 1.1. [**51**]. Goodfellow et al. [**51**] introduce deep learning as an approach that gathers a "hierarchy of concepts that enables the computer to learn complicated concepts by building them out of simpler ones". The concepts are stacked on top of each other to form deep layers of knowledge. That is why we call the approach *deep learning*. Although we have recently seen an increased interest in AI, we can track deep learning approaches as early as 1943 [**95**]. The recent success is attributed to the enormous increase in the amount of available labeled data and to the growth in the size of computer infrastructures [**51**].

This chapter provides the information needed to understand the fundamentals of deep learning. In what follows, we first introduce the perceptron, the simplest form of a neural network, and then present the multilayer perceptron. We also introduce the convolutional

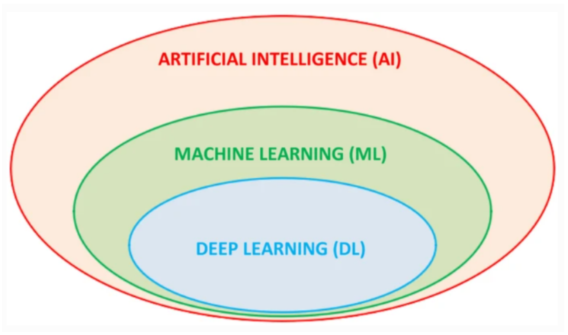

Fig. 1.1. Artificial intelligence venn diagram. Image from: [**67**].

neural network (CNN), a well-known deep learning architecture for image processing. We finally describe the different categories of ML algorithms.

1.1. Single-Layer Perceptron

In the early days of deep learning, Frank Rosenblatt introduced the perceptron [**118**], the simplest form of Artificial Neural Network (ANN), at the Cornell Aeronautical Laboratory [**119**]. The perceptron architecture includes an input layer and an output layer, as shown in Figure 1.2 (a). The perceptron maps its input $\boldsymbol{x} \in \mathbb{R}^n$ to an output value $\hat{y}$. Each input $\boldsymbol{x}$ is associated with a class label y. Historically, the class label was binary, such as $y \in [-1,1]$. The output value, also called the predicted value, is computed as:

$$\hat{y} = \phi(\sum_{i=1}^{n} w_i x_i + b) \tag{1.1.1}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is the weight vector, b is the bias constant and ϕ is the activation function [**118**]. The activation function is used to map the input value to a binary output value (1 or -1). There are different activation functions, such as the step function defined in Equation 1.1.2.

$$H(x) = \begin{cases} 1 & if x \geq 0 \\ -1 & if x < 0 \end{cases} \tag{1.1.2}$$

Other typical activation functions are the sigmoid function $\phi(x) = (1 + e^{-x})^{-1}$ and the hyperbolic tangent function $\tanh(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$. The perceptron can be seen as a binary classifier, also known as a logistic regression classifier. The weights of the network show the importance of each input node and are adjusted to draw a linear decision boundary between

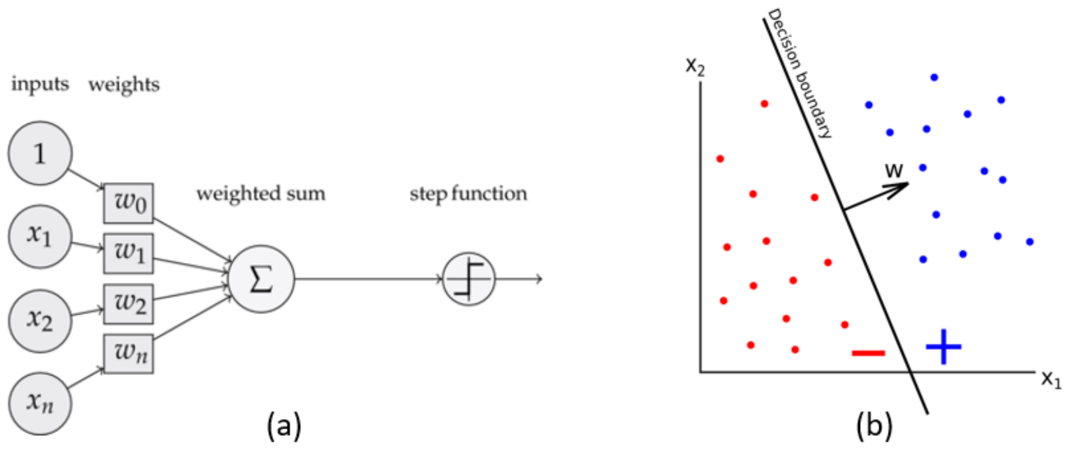

Fig. 1.2. Perceptron architecture and linear decision boundary visualization.(a) Perceptron architecture. Image from: [**16**]. (b) Example of linear decision boundary. Image from: [**82**].

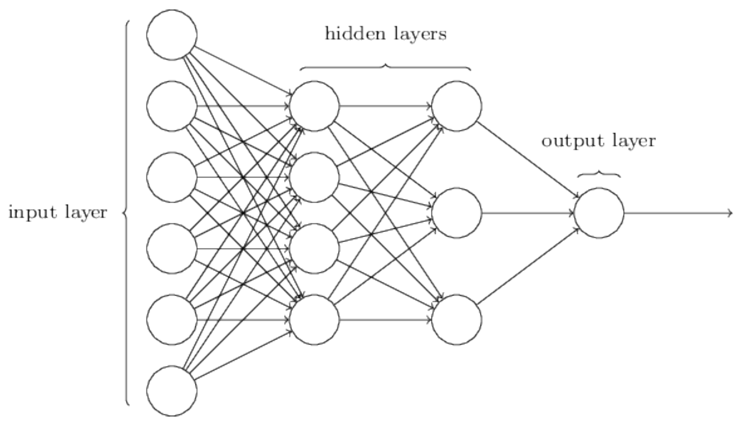

Fig. 1.3. Artificial neural network architecture with two hidden layers. Image from: [**116**].

the two classes. In other words, the perceptron divides the input space into two classes (-1 and 1) based on the value of the weights and the bias. An example of a linear decision boundary is shown in Figure 1.2 (b). The slope of the decision boundary is defined by the weight vector that is optimized using the learning algorithm described in Algorithm A.1 (see Appendix A). The intercept of the decision boundary is defined by the bias b.

1.2. Feed Forward Artificial Neural Networks

The perceptron, being the simplest form of ANN, has its limitations. Indeed, the perceptron cannot solve nonlinearly separable problems. The Multilayer Perceptron (MLP) overcomes this limitation. The first MLP was introduced in 1968 by Ivakhnenko et al. [**71**]. A MLP is a neural network composed of layers of perceptrons whose architecture is illustrated in Figure 1.3. A MLP is also called feed forward artificial neural network because information circulates in the network starting from the input nodes then passing through the hidden nodes and finally through the output nodes.

The neural network algorithms are designed to solve a particular task. The algorithms can be classified into different groups based on the task they are trying to solve: classification or regression. The objective of ANN is to approximate a target function f^* associated with the task at hand. The target function is a function that maps the input to the output. In a classification problem, the target function maps an input to a discrete output. In a regression problem, the target function maps an input to a continuous output. The network computes a function $f(\boldsymbol{x};\boldsymbol{\theta})$ to approximate f^* where $\boldsymbol{x} \in \mathbb{R}^n$ is the input. The network f is parametrized by the weights of the network $\boldsymbol{\theta}$. The input data points are linearly transformed to a set of features $\boldsymbol{a}^{(1)}(\boldsymbol{x}) \in \mathbb{R}^m$ using the weight matrix $\boldsymbol{W}^{(1)} \in \mathbb{R}^{n\times m}$ and the bias $b^{(1)}$ such as: $\boldsymbol{a}^{(1)}(\boldsymbol{x}) = b^{(1)} + \boldsymbol{W}^{(1)}\boldsymbol{x}$. Then, a nonlinear activation function ϕ is applied element-wise to the linearly transformed input data to obtain a new set of features such as: $\boldsymbol{h}^{(1)}(\boldsymbol{x}) = \phi(\boldsymbol{a}^{(1)}(\boldsymbol{x}))$ where $\boldsymbol{h}^{(1)}(\boldsymbol{x}) \in \mathbb{R}^m$. A commonly nonlinear function used is the Rectified Linear Units function (ReLU) [**103**] defined as $\text{ReLU}(x) = max(x,0)$.

We suppose that we have a feed forward ANN with L hidden layers. Thus, we compute the pre-activation hidden nodes at layer $k > 0$ as shown in Equation 1.2.1.

$$\boldsymbol{a}^{(k)}(\boldsymbol{x}) = b^{(k)} + \boldsymbol{W}^{(k)}\boldsymbol{h}^{(k-1)}(\boldsymbol{x}) \tag{1.2.1}$$

Then, the hidden layer activation nodes are computed as shown in Equation 1.2.2.

$$\boldsymbol{h}^{(k)}(\boldsymbol{x}) = \phi(\boldsymbol{a}^{(k)}(\boldsymbol{x})) \tag{1.2.2}$$

In classification, the output layer consists of two or more nodes. We estimate the probabilities that the predicted class $\hat{y}$ is equal to the different classes c by using the softmax activation function, which is defined as:

$$\text{softmax}(\boldsymbol{a}) = \left[\frac{\exp(a_1)}{\sum_c \exp(a_c)},...,\frac{\exp(a_C)}{\sum_c \exp(a_c)}\right]^T \tag{1.2.3}$$

Once the softmax function is applied, the values of the output nodes sum up to one and are strictly positive. The predicted class is the class associated with the node with the highest value. Thus, the output value of the network is computed as:

$$\boldsymbol{h}^{(L+1)}(\boldsymbol{x}) = \text{softmax}(\boldsymbol{a}^{(L+1)}(\boldsymbol{x})) = f(\boldsymbol{x};\boldsymbol{\theta}) \tag{1.2.4}$$

where $\boldsymbol{\theta} = (\boldsymbol{W}^{(1)}, b^{(1)}, \ldots, \boldsymbol{W}^{(k)}, b^{(k)}, \ldots, \boldsymbol{W}^{(L+1)}, b^{(L+1)})$ is the set of all parameters in the network. The network is trained based on a principle known as empirical risk [**141**] in which the training of the network is converted into an optimization problem. In empirical risk minimization [**141**], training a network is the task of finding a set of parameters $\boldsymbol{\theta}$ that minimize Equation 1.2.5 (i.e., minimize an objective function).

$$\frac{1}{T}\sum_t l(f(\boldsymbol{x}^{(t)};\boldsymbol{\theta}), y^{(t)}) + \lambda\Omega(\boldsymbol{\theta}) \tag{1.2.5}$$

In Equation 1.2.5, T is the number of training samples, $x^{(t)}$ is t^{th} training sample and $y^{(t)}$ is the label associated with sample $x^{(t)}$. The regularizer Ω penalizes particular values of $\boldsymbol{\theta}$ to help the model generalize better on unseen samples [**122**]. A common regularizer is weight decay also called L_2 regularization where $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2^2$ [**51**]. The loss function, denoted by l in Equation 1.2.5, compares the network's output with the true label. A loss function is a proxy for the classification error since optimizing a classification error can be

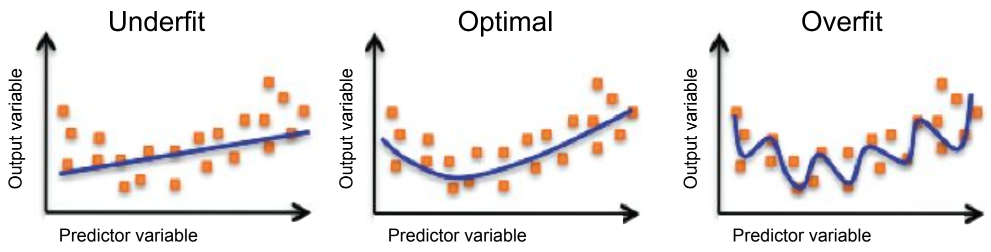

Fig. 1.4. Overfitting and underfitting illustration. (Left) The problem of underfitting when the model is not sufficiently aligned with the training data. (Middle) Ideal balance. (Right) The problem of overfitting where the model is too closely aligns with the training data. The models learned the noise of the training data. Image adapted from: [**12**]

tricky considering it is not a smooth function [**51**]. One of the most popular loss functions used in classification tasks is called cross-entropy and corresponds to the negative of the log-likelihood [**51**]. The balance between optimizing the average loss and the regularization is controlled by λ, a hyperparameter[1].

Now that the problem of training a neural network is seen as an optimization problem, we can use different algorithms from the optimization literature, such as Stochastic Gradient Descent (SGD). The weights and the biases are optimized using backpropagation which is described in detail in [**51**].

1.3. Underfitting and Overfitting

The desired objective of training a model (i.e., optimization of weights and biases) is to be able to generalize from the training data to any data from the problem domain [**72**]. Generalization refers to the ability of a model to predict unseen data points correctly [**40, 73**]. However, generalization is sometimes not possible due to the problem known as underfitting. Underfitting occurs when a model is not able to capture the relationship between the input and output data [**68**]. Usually, this is characterized by poor performance on the training and testing data. Figure 1.4 (left) illustrates the phenomenon of underfitting where we observe that the model is not sufficiently aligned with the training data. Underfitting is unlikely since, according to the Universal Approximation Theorem [**64**], a neural network with only one hidden layer can approximate any continuous function well with enough hidden units. ANNs are known as Universal Function Approximators. Underfitting is also unlikely, considering that models are now deeper and wider (i.e., high-capacity models). We can increase the capacity of a neural network by increasing the number of hidden nodes and layers. Zhang et al. [**152**] show that high-capacity models can easily fit a random labeling of the training data.

[1]A hyperparameter is not learned during the training but is set before the learning process.

On the other hand, high-capacity models can easily overfit on the training data. Overfitting is the opposite of underfitting. This phenomenon can happen when the model is trained for too many epochs or with limited labeled data. To explain the problem of overfitting, Hawkins et al. [**56**] first define the principle of parsimony as the use of models or procedures that contain only the necessary information, but nothing more. Overfitting is defined as using models or procedures that violate the principle of parsimony. Figure 1.4 (right) illustrates the phenomenon where the model fits exactly the data. The model learned the noise and artifacts of the training data. It is characterized by a good performance on the training set and poor performance on the testing set.

1.4. Convolutional Neural Networks

One way to deal with overfitting is to decrease the complexity of the model [**121**]. In the context of image classification, feed-forward ANNs are highly complex since they require an unmanageable number of parameters in the fully connected layers. Ideally, we would want a neural network capable of processing high-dimensional images without requiring a considerable number of parameters.

LeCun et al. [**84**] introduced, in 1989, the Convolutional Neural Network (CNN), also called ConvNet. CNN is a deep learning algorithm with fewer trainable parameters that takes the spatial information of images into account. In a CNN, hidden units are connected to sub-regions of the image and connected to all channels, thus reducing the number of parameters. CNN utilizes the concept of parameter sharing with kernels (also called filters) that slide along the input image, further reducing the number of parameters. As shown in Figure 1.5, typical CNNs are made up of three different layers: the convolutional layer, the subsampling/pooling layer, and the fully-connected layer. The three types of layers are described below.

Convolutional layers extract the features using kernel matrices, also called filters. A kernel matrix is the hidden weight matrix $\boldsymbol{W}$ where the rows and columns are flipped. The outputs of a convolutional layer are called the feature maps and are computed by performing a discrete convolution ($\star$) of an image $\boldsymbol{X}$ with a kernel matrix $\boldsymbol{K}$. The width and height of the kernel are denoted by V and H, respectively. The output of the 2D discrete convolution $\boldsymbol{Y}$ is computed as shown in Equation 1.4.1.

$$\boldsymbol{Y}(x,y) = \sum_{u=0}^{V} \sum_{v=0}^{H} \boldsymbol{X}(x-u, y-u)\boldsymbol{K}(u,v) \tag{1.4.1}$$

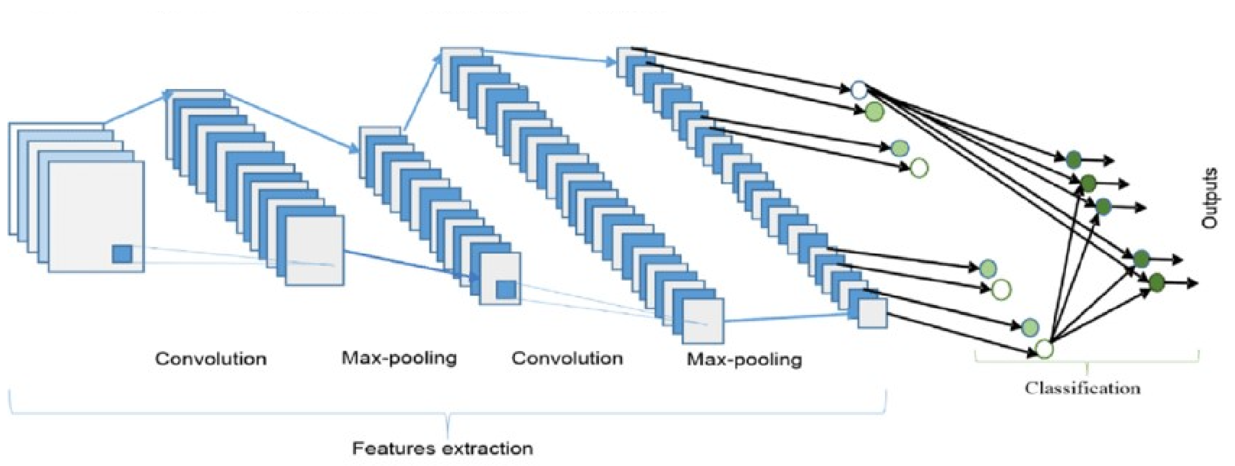

Fig. 1.5. Convolutional neural network architecture. Typical CNNs are built of convolutional layers, pooling layers and fully-connected layers. Image from:[**4**].

We slide the kernel across the input matrix as demonstrated in Figure 1.6. The step size used while sliding the kernel on the matrix is called "stride". In Figure 1.6, we use a stride of 1. We can increase the size of the input matrix by adding zero pixels to the borders. We call this operation "zero-padding". Padding is needed in some cases because a discrete signal is not defined outside its borders. Thus, a convolution is also not defined outside the borders of an image [**28**]. The "zero-padding" approach is useful when the signal at the boundaries are supposed to be zeros. If it is not the case, this method creates dark borders around the image. Thus, non-zero pixels can also be used in the padding operation. Once the discrete convolution is computed, the non-linear operation can be applied at every position of $\boldsymbol{Y}$, such as ReLU.

Equation 1.4.2 demonstrates how to compute the vertical output size O_v and horizontal output size O_h of a convolutional layer where N is the height of the input matrix, M the width of the input matrix, P the padding value and S the stride value [**2**].

$$O_v = 1 + \frac{N + 2P - H}{S} \qquad O_h = 1 + \frac{M + 2P - V}{S} \tag{1.4.2}$$

Convolutional layers have learnable kernels with one trainable bias per feature map and follow the standard procedure for the forward pass and backpropagation [**124**].

Subsampling/Pooling layers reduce the dimensionality of feature maps and make the features invariant to translations [**124**]. Like the convolutional layer, pooling operations are computed on subregions (patches) of the feature maps. The patches can be overlapping or not [**124**]. Two standard pooling methods are known as average pooling and max pooling. Average pooling computes the average value of the subregions in the feature maps. Max pooling returns the maximum value of each subregion. In Figure 1.6, the pooling operation

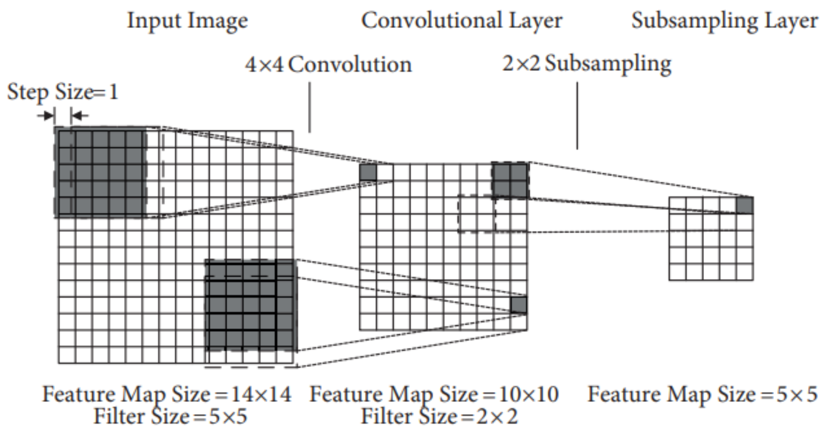

Fig. 1.6. Example of a convolutional and a subsampling layer. The dimension of the input matrix is 14 × 14. A kernel of size 5 × 5 is convolved on the input image using no padding and stride of value 1. The output of the convolutional layer has a dimension of 10 × 10. We can also apply at each pixel a non-linear operation such as ReLU. Then, we use a kernel of size 2 × 2 in the sampling layer to reduce the dimension of the output matrix to 5 × 5. Image from: [**89**]

is computed on non-overlapping patches of dimension 2 × 2. Usually, the pooling layer comes after the convolutional layer.

Fully-Connected layers are the layers used in a feed-forward ANN. The objective is to convert the extracted features to a vector with the size equal to the number of classes. It is used for end-of-network classification.

Over time, many convolutional neural network architectures have emerged. In 1998, LeCun et al. [**85**] proposed LeNet-5, a pioneering CNN used to classify handwritten digits. In 2012, the AlexNet [**80**] architecture marked a paradigm shift. The deep model was trained to classify 1.2 million images of the ImageNet dataset, which has 1000 classes [**80**]. AlexNet consists of 5 convolutional layers followed by ReLU activation functions and max-pooling layers. The dropout regularization method is used in the three fully connected layers and is described in more detail in Section 2.2.1.

In 2014, Simonyan et al. [**130**] introduced VGG Net in which they increased significantly the depth of the network to 16-19 layers. Later in 2014, Szegedy et al. [**137**] proposed the Inception architecture where they increased both the depth and the width of the network by using 1 × 1 convolutions. In 2015, He et al. [**58**] proposed a residual learning framework called ResNet to ease the training of deep networks by combining residual blocks using shortcut connections. As shown in Figure 1.7, using residual blocks, the desired mapping

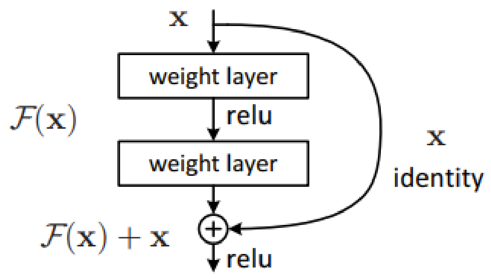
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
x
identity

Fig. 1.7. Residual learning block. Image from: [**58**]

is equal to $F(x) + x$. The authors designed deeper architectures ranging from 18 layers (ResNet-18) to 150 layers (ResNet-150). The residual learning framework is among the first architectures to use batch normalization, which is described in Section 2.2.1.

1.5. Supervised and Unsupervised Learning

So far, all of the algorithms described and most of the deep learning algorithms are fully supervised learning algorithms. In supervised learning, the algorithms learn using labels from the training data. Kotsiantis [**79**] defines supervised learning as algorithms that reason from labels to produce general hypotheses that are then used to make predictions on unseen samples. Supervised learning algorithms can be divided into two tasks: classification and regression. In a classification problem, we train a model to approximate a function that maps an input to a discrete output. In a regression problem, we train a model to approximate a function that maps an input to a continuous output.

On the other hand, some algorithms don't need labeled data, which we call unsupervised learning algorithms. No target attributes are used in training. The input data is grouped and interpreted based on the relationship between the input data [**41**]. In [**63**], Hofmann even suggests using unsupervised learning algorithms to create labels associated with the input data and then using the generated labels for a classification or regression task.

Supervised learning algorithms tend to be more accurate than unsupervised ones [**37**] but can easily overfit if the number of labeled data is not sufficient. A limited amount of labeled data is often accompanied by a much larger collection of unlabeled samples. Some algorithms mix of unsupervised and supervised learning by performing model pre-training on unlabeled data followed by fine-tuning on limited labeled samples. This type of learning, called self-supervised learning, is covered in Section 2.2.7.

Chapter 2

Limited Labeled Data

Most breakthroughs in deep learning in general, and CNNs in particular, involve significant effort in collecting massive amounts of well-annotated data. While big data is becoming increasingly prevalent, there are numerous applications where the task of annotating more than a small number of samples is infeasible, giving rise to increasing interest in small-sample learning tasks and deep learning approaches towards them [**23, 17, 18, 24**]. This chapter focuses on the limited amount of labeled data challenge by describing the challenge itself and summarizing various techniques for overcoming it.

2.1. Data Scarcity

Over the past decade, a lot of effort was dedicated to building, creating, and annotating massive datasets. Table 2.1 lists some popular datasets. We observe that the number of samples per dataset is enormous. These datasets are now used as common benchmarks to evaluate different models. For example, the ImageNet1K dataset contains 1.2 million images. To annotate this number of images, an online platform called Amazon Mechanical Turk (AMT) was used for outsourcing the images and allowing humans to annotate them and, in return, get paid [**38**]. Big datasets, such as ImageNet1K, have been a key component in the success and revolution of deep learning. Figure 2.1 shows the performance of a model based on the amount of labeled data available and is used for illustrative purposes only. Models trained on big datasets live in the top right region of the figure. In this region, deep

Tableau 2.1. Number of samples per dataset

Dataset	Number of Samples
IWSLT15	133k
COCO 2014	164k
LibriSpeech	280k
ImageNet1K	1.2M
WMT-14	4.5M

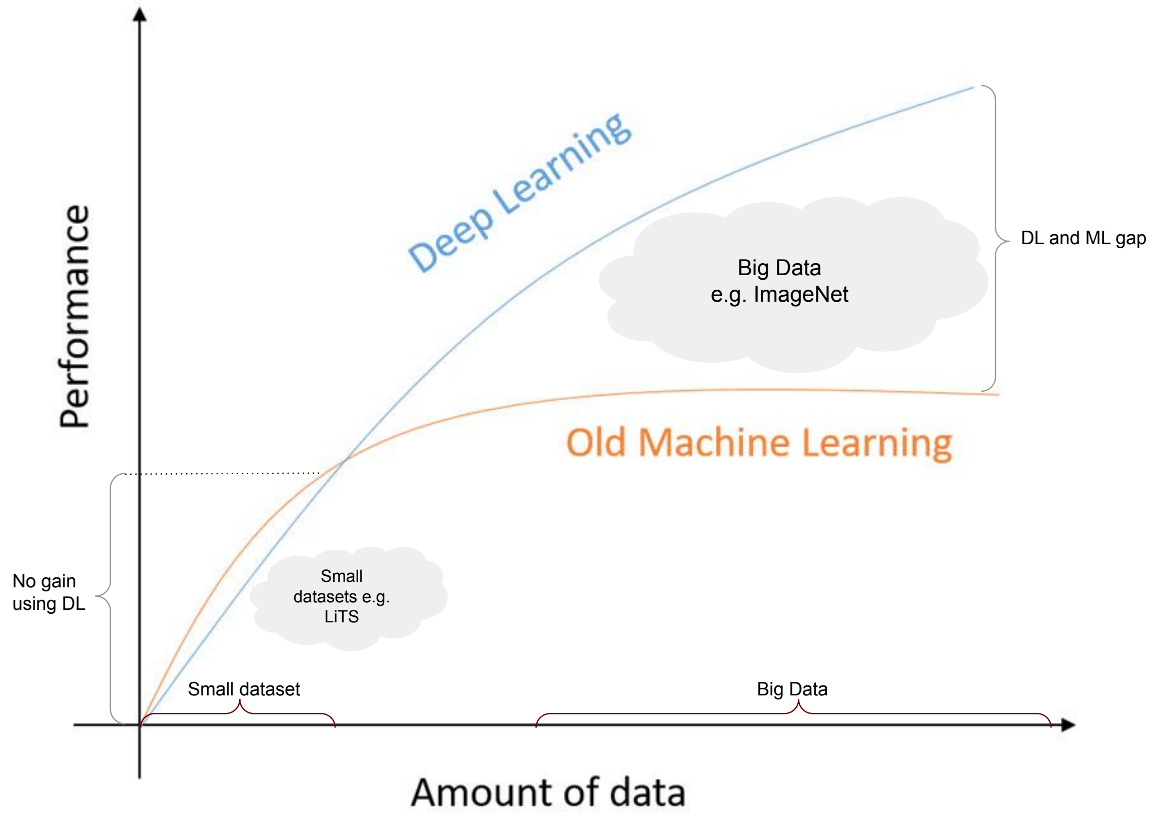

Fig. 2.1. Performance of a model based on the amount of labeled data available. Models trained on big datasets live in the top right region of the figure. In this region, deep learning models outperform old machine learning models. Small datasets live in the bottom left region of the figure. There is no gain in using deep learning over old machine learning models in this region. Image adapted from: [**5**]

learning models outperform old machine learning models.

It is not always possible to have an enormous amount of labeled data. For example, in biomedicine and healthcare, it requires significant efforts from highly qualified and busy medical experts to manually annotate the data [**55**]. Table 2.2 lists examples of medical datasets. We observe that the numbers of samples are much lower than those shown in Table 2.1. Privacy laws make it even harder to obtain medical data. All the medical data must be de-identified, or anonymized [**102**]. If someone successfully gets medical data and complies with privacy laws, then the challenge is to annotate it. In medical applications, it is not possible to outsource data to ask people to annotate it for reasons of confidentiality. The medical field is not the only field that suffers from limited labeled data. For example, financial institutions have a large volume of contracts and agreements used to build risk models. However, few contracts and agreements are processed to extract the information [**22**]. Also, in enterprise IT, there is a large amount of customer service chat

Tableau 2.2. Number of samples per medical dataset

Dataset	Image Modality	Number of Samples
DRIVE [**135**]	retinal	40
LiTS [**21**]	CT scan	200
BUSY [**1**]	ultrasound	780
ISIC2017 [**19**]	skin lesion	2,750
Pediatric Chest X-ray [**74**]	chest X-ray	5,863

logs available, but few are annotated [**22**].

Learning useful representations from little training data [**18**] is arduous. Small datasets live in the bottom left region of Figure 2.1. In this region, there is not much gain in using deep learning over old machine learning approaches. As a result, interest in the small-sample learning tasks and deep learning approaches has increased in recent years [**18, 29**].

2.2. Strategies

Different strategies have been designed to overcome the limited labeled data challenge. In this section, we summarize common ones.

2.2.1. Regularization

Regularization techniques were designed to reduce generalization error but at the cost of increasing the training error [**51**]. Here, we describe two common regularization techniques: dropout and batch normalization.

Dropout regularization technique, which was proposed by Srivastava et al. [**134**], was developed to reduce overfitting and increase generalization. As the name suggests, the model drops certain units randomly during training [**60**]. At each iteration of the training, some layer outputs are ignored, preventing units from co-adapting [**60**]. Co-adaptating occurs when hidden units have highly correlated behavior [**50**]. We can think of dropout as an ensemble of subnetworks where each subnetwork is constructed by dropping out units [**51**]. Tompson et al. [**83**] extended the idea of dropout to CNN where entire features maps are ignored instead of only dropping out neurons.

Batch Normalization, proposed by Ioffe et Szegedy [**70**], is a method that normalizes the training mini-batch to enable faster and more stable training [**123**]. Batch normalization helps reduce internal covariate shift between different layers of a neural network [**70**]. Ioffe et Szegedy [**70**] define covariate shift as the change in the input distribution of each layer during training due to the change of parameters in the previous layers. We apply batch normalization over a mini-batch $B = \{x_1, \dots x_m\}$ by computing the mini-batch mean μ_B and mini-batch variance σ_B^2. We normalize the input by subtracting the mini-batch mean and dividing it by the variance. We then scale and shift it by using learnable parameters (β, γ) to compute the output y_i of the batch normalization layer. Equation 2.2.1 shows how to compute the output value of a batch normalization layer where ϵ is a constant used for numerical stability.

$$y_i = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{2.2.1}$$

2.2.2. Data Augmentation

Data augmentation is a technique used to overcome the problem of data scarcity. It was developed to reduce overfitting by directly addressing the root of the problem [**127**]. This technique is based on the assumption that more information can be extracted from the initial input data [**127**]. Data augmentation consists of increasing the size and quality of the training set via augmentations. Two types of augmentations can be used: data warping or oversampling [**127**].

Data warping. Shorten et al. [**127**] define data warping as inflating the size of a dataset by transforming existing data samples using geometric transformations (e.g., flipping, cropping, rotation, translation and noise injection), color transformations and adversarial training. The augmentations are usually manually designed. Instead of manually designing the augmentations, Cubuk et al. [**36**] proposed the use of the autoaugment procedure that automatically searches for the best data augmentation policies.

Oversampling. Oversampling creates synthetic training instances. This can be done by mixing images together [**69**], by augmenting features directly in the feature space [**78, 39**] or by training generative adversarial networks (GANs) [**52**].

2.2.3. Transfer Learning

A common strategy used when the number of labeled data is limited is called *Transfer Learning*. In fact, it has been observed that the first layers of convolutional networks resemble Gabor filters [**80**], as shown in Figure 2.2. Gabor filters are described in more detail in Chapter 4. The first-layer features do not appear to be specific to one particular dataset [**147**]. On the contrary, the first-layer features seem general and reusable across different tasks. Transfer learning was built on this idea of transferring features across tasks. In traditional machine learning, we assume that the testing data and training

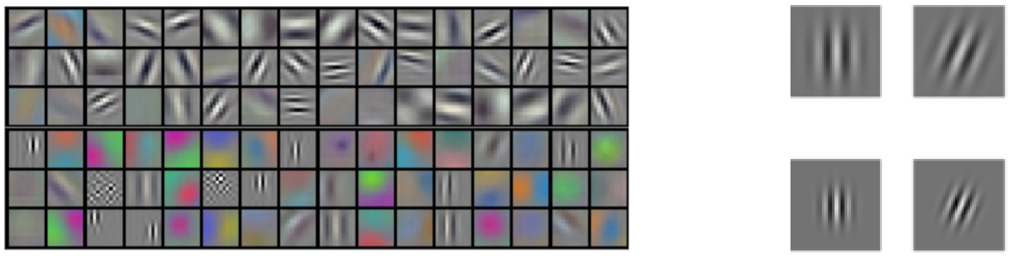

Fig. 2.2. Filters learned by the first convolutional layer in AlexNet resemble Gabor filters. (Left) Filters learned by the first convolutional layer in AlexNet. Image from: [**80**] (Right) Examples of Gabor filters from [**98**].

data come from the same domain. With transfer learning, this assumption is no longer valid [**110, 126**]. Transfer learning consists of training a network on a different dataset. Then, the learned weights are used as the initial weights on another classification task. With CNNs, we can train the models on the new classification task in two different ways. We can freeze the learned weights in the convolutional layers and only learn the weights in the fully connected layers, or we can unfreeze all the weights in the network. In the former option, we reuse the features learned from another task. In the latter option, the pre-trained weights are used as initial weights but optimized for the new task.

Multiple works have investigated the transferability of features between tasks [**151, 147, 114**]. Yosinski and al. [**147**] found that the higher layers in a CNN are more specialized to the task for which the model was trained. A lot of attention in the research community has been put on using natural images to study transfer learning onto the domain of medical imaging [**115, 148, 27, 97, 32, 99**]. However, there are fundamental differences in data sizes, features and task specifications between natural image classification and medical tasks [**114**]. Raghu et al. [**114**] explored properties of transfer learning from ImageNet to Chest X-Ray images. The authors claim that in a setting where very few images are available and where time is of the essence, transfer learning from ImageNet is beneficial. They also point out that transfer learning may seem to work in certain cases since the models used are over-parameterized.

2.2.4. Loss Function

The choice of the loss function can help improve the performance in some cases when the number of labeled data is limited. The categorical cross-entropy loss followed by softmax is commonly used in classification tasks. Recent works have investigated replacing it with other loss functions in the small data regime. Lezama et al. [**88**] replaced the categorical cross-entropy loss with a geometric loss called Orthogonal Low-rank Embedding (OLÉ) to reduce the intra-class variance and enforce inter-class margins. Barz et al. [**17**] also proposed

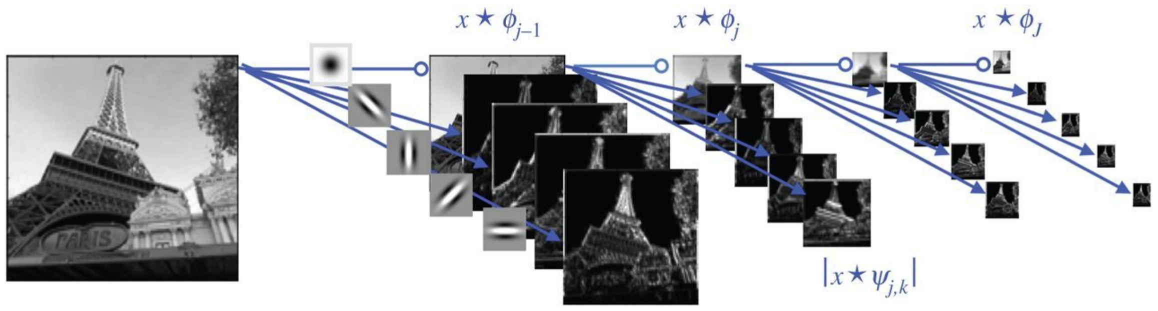

Fig. 2.3. Scattering transform of an image $\mathbf{x}(\mathbf{u})$. The scattering transform consists of a cascade of convolutions with fixed wavelet filters. In this example, the spatial scale J is set to 4 and the number of orientation L is also set to 4. Image from: [**94**]

.

to replace the cross-entropy loss, but this time with the cosine loss function to decrease overfitting in the small-sample classification settings. The cosine loss function, as opposed to the softmax function used with cross-entropy, does not push the logits of the true class to infinity as explained in [**138**]. Barz et al. [**17**] define the cosine loss function as:

$$L_{cos}(x,y) = 1 - \sigma_{cos}(f_\theta, \phi(y))$$

where $x \in X$ is a sample (in our case an image), $y \in C$ is the class label of x, σ_{cos} is the cosine similarity between two vectors, f_θ is the model parametrized by θ and $\phi(y)$ is the one-hot vector representing the class label.

2.2.5. Wavelet Scattering Transform

Scattering-based models have been proven useful in several applications involving scarcely annotated data [**26, 128, 108, 44**]. The scattering transform, proposed in [**93**], is a cascade of wavelets and complex modulus nonlinearities, which can be thought as a convolutional neural network (CNN) with fixed filters. The scattering transform is described in more details in Chapter 3. Scattering transform, shown in Figure 2.3, can be used to build representations with geometric invariants and is shown to be stable to deformations. Since scattering transforms are instantiations of CNNs, they have been studied as mathematical models for understanding the impressive success of CNNs in image classification [**26, 94**]. As discussed in [**26**], first-order scattering coefficients are similar to SIFT descriptors [**90**], and higher-order scattering can provide insight into the information added with depth [**94**]. Moreover, theoretical and empirical study of information encoded in scattering networks indicates that they often promote linear separability, which in turn leads to effective representations for downstream classification tasks [**26, 106, 6, 44**].

2.2.6. Incorporating Prior Knowledge

As discussed in Section 2.2.3, features extracted by the first layers of a CNN are generic and not specialized to a particular task, and resemble Gabor filters. Multiple works have tried to incorporate this prior knowledge into models to tackle the limited labeled data challenge. In image classification, Oyallon et al. [**108**] introduced hybrid networks where the scattering transform was shown to be an adequate replacement for early layers of learned convolutional networks on wide residual network architecture. Similarly, Raghu et al. [**114**] proposed to initialize the filters of the first layer of a CNN as synthetic Gabor filters. The filters are not constrained to remain Gabor-like during the training. Cotter et al. [**35**] also proposed a hybrid network called the learnable ScatterNet, where learned layers are intermixed between the scattering orders. Ulicny et al. [**139**] proposed the Harmonic Network (HN), a hybrid network consisting of fixed discrete cosine transform filters combined with learnable weights in CNNs. Finally, Alekseev et al. [**3**] proposed the GaborNet where the filters in the first layer of the network are constrained to fit the Gabor function.

2.2.7. Self-Supervised Learning

Limited labeled data is often accompanied by a much larger collection of unlabeled samples. In such a setting, a self-supervised learning approach can be used to utilize the information contained in the unlabeled data. Self-supervised learning approaches learn useful semantic representations and have shown promising results in the past years. In self-supervised representation learning, artificial labels are generated cheaply from the data and are used in a pretext task [**100**]. Models that have been pretrained on a pretext task can be transferred to another task where it is harder to obtain labeled data.

Finding the right pretext task for a specific downstream objective can be challenging. One of the first solutions proposed by Doersch et al. [**42**] is to train a model to predict the position of a random patch relative to another one. Pathak et al. [**111**] proposed to train a convolutional neural network to generate the contents of a random image region based on its surroundings. Noroozi and Favaro [**153**] proposed [**104**] to train a CNN to solve jigsaw puzzles as the pretext task. Zang and al. [**153**] proposed, as the pretext task, to train a model to produce realistic colorization of images. Gidaris et al. [**49**] proposed to learn image features by training a CNN to recognize 2D rotations that have been applied to the images. All these methods have in common the fact that the pretext tasks are handcrafted, which can limit the generality of the learned features [**30**].

In contrast to handcrafted pretext tasks, contrastive visual representations are learned by contrasting positive pairs against negative pairs [**30**]. These frameworks are

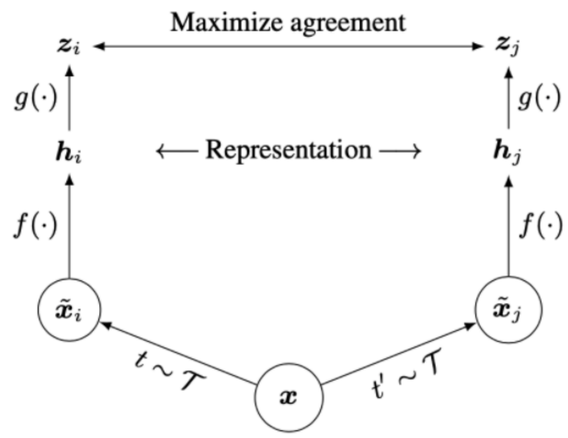

Fig. 2.4. SimCLR framework: contrastive learning of visual representations. Image from: [**30**]

usually simpler and more generic than handcrafted pretext tasks. An example of such a framework is SimCLR [**30**] where representations are learned by maximizing agreement between differently augmented views of the same data example, as shown in Figure 2.4. The augmented views of an image are created via data augmentations. Data augmentations are applied sequentially to each image to form pairs of augmented images $\{x_i, x_j\}$. The augmentations consist of random cropping and resizing it back to the original size, random colour distortions, and random Gaussian blur [**30**]. Each correlated image is fed to the base encoder $f(\cdot)$ to extract representation vectors h_i and h_j. The encoder is a ResNet-50. Then a small neural network called projection head $g(\cdot)$ maps the representations h_i and h_j to the space where the contrastive loss is applied. Given a positive pair, the other augmented examples in the mini-batch are considered negative examples. Given $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v}/||\mathbf{u}||||\mathbf{v}||$, the contrastive loss function used in the framework is called the normalized temperature-scaled cross-entropy loss and is defined for a positive pair (i,j) as:

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]}(\text{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)} \tag{2.2.2}$$

Kinakh et al. [**75**] proposed ScatSimCLR, which is built on the SimCLR framework. The difference is that in ScatSimCLR the encoder $f(\cdot)$ is not a ResNet-50, but a scattering network [**93**]. The authors demonstrate that by using a scattering network instead, the number of learnable parameters and the number of views can be considerably reduced without decreasing the classification accuracy.

Other contrastive frameworks have been proposed. He et al. [**57**] proposed Momentum Contrast (MoCo), a contrastive learning framework that uses a dynamic dictionary and a moving-averaged encoder. Grill et al. [**53**] proposed a contrastive framework called Bootstrap Your Own Latent (BYOL) that trains an online network to predict the representation of a target network based on an augmented image. The target network is updated using a slow-moving average of the online network [**53**].

Chapter 3

Wavelet Scattering Transform

In this chapter, we describe the wavelet scattering transform in detail. First, we define the stability to deformations using the Lipschitz continuity condition in Section 3.1. Next, we define the formulation of the scattering network and describe Morlet wavelets in Section 3.2 and Section 3.3 respectively. In Section 3.4, we describe an analogy between wavelets and the visual system. In Sections 3.5 and 3.6, we discuss the properties of the scattering transform. We introduce different variations of the scattering transform in Section 3.7. Finally, in Section 3.8, we list and describe various packages that implement the transformation. For simplicity, we focus here on 2D scattering networks.

3.1. Stability to Deformations

An important challenge in extracting informative features from high-dimensional data is disentangling the different sources of variation in the data. Intra-class variability is the variation between data points of the same class. Inter-class variability is the variation between data points of different classes. The challenge is to construct features from data points that maximize inter-class variability and eliminate intra-class variability. This challenge is a reality in various tasks, such as the classification of handwritten numbers. The exact number can be written differently depending on the writing style, as shown in Figure 3.1. The objective is to construct features that are invariant to rigid transformations such as rotation, translation and scaling.

The kernel method [**77**], which transforms a data point x into a new representation $\Phi(x)$, is a general approach used to tackle this challenge. Using the kernel metric, the distance between two input signals f and g is defined as: $d(f,g) = ||\Phi(f) - \Phi(g)||$. The goal is to have small distances $d(f,g)$ within a class and large distances $d(f,g)$ across classes. The new representation $\Phi(x)$ needs be stable to small deformations and preserve the information. In other words, small deformations of the input signal x should result in a

Fig. 3.1. Samples of handwritten digits. There are many variations within a single class. For example, there are 16 "0's" in the figure. Each individual "0" is different because of the writer's style. Image from: [**146**].

small modification of $\Phi(x)$. We can express the stability to deformations using the Lipschitz continuity condition. A deformation can be expressed as $L_\tau x(u) = x(u - \tau(u))$ where $\tau(u)$ deforms the image x at position u. In two-dimensional signals, u represents the spatial location of the pixel in the image. The amplitude of deformation at u is measured using the norm of the deformation gradient matrix $|\nabla\tau(u)|$. The Lipschitz continuity condition is defined in Equation 3.1.1.

$$||\Phi(L_\tau x) - \Phi(x)|| \leq C||x|| \quad \sup_u |\nabla\tau(u)|, \tag{3.1.1}$$

where $||x||^2 = \int |x(u)|^2 du$ and C is a constant. A representation $\Phi(x)$ is stable to deformations if it is Lipschitz continuous with respect to the deformation.

3.2. Scattering Convolutional Network

The scattering coefficients are constructed by cascading wavelet transform with modulus and averaging operators. For simplicity, we focus on 2D scattering networks up to their 2nd order. Subsequent orders can be computed by following the same iterative scheme but have been shown to yield negligible energy [**26**]. Given a signal $x(u)$, where u is the spatial position index, we compute the scattering coefficients S^0x, S^1x, S^2x, of order 0, 1, and 2, respectively. For integers J and L, corresponding to the spatial scale of the scattering transform and the number of orientations respectively, and assuming an $N \times N$ signal input with one channel, the resulting feature maps are of size $\frac{N}{2^J} \times \frac{N}{2^J}$, with channel sizes varying with the scattering coefficient order. Table 3.1 summarizes the number of channels per

Tableau 3.1. Number of channels per scattering coefficient order

Order	Number Channels
0	1
1	JL
2	$L^2J(J-1)/2$

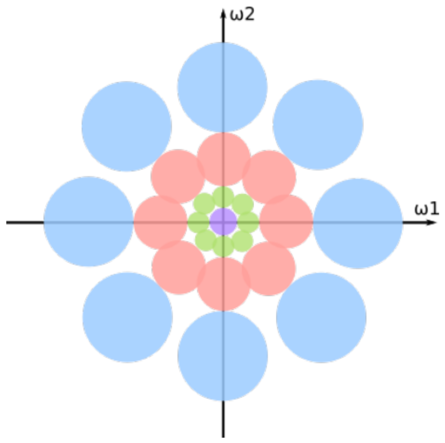

Fig. 3.2. Wavelets covering the frequency plane. Wavelets generated from a mother wavelet via dilations and rotations to cover the frequency plane. Image from: [**28**].

scattering order.

To calculate 0th-order coefficients, we consider a low pass filter ϕ_J with a spatial window of scale 2^J, such as a Gaussian smoothing function. We then convolve this filter with the signal and downsample by a factor of 2^J to obtain $S^0x(u) = x * \phi_J(2^J u)$. Two observations can be made at this point. Due to the low-pass filtering, high-frequency information is discarded here and is recovered in higher-order coefficients via wavelets introduced as in a filter bank.

The scattering coefficients of higher orders are constructed by filtering images with complex wavelets. A complex wavelet filter ψ is defined in Equation 3.2.1.

$$\psi(u) = \psi_a(u) + i\psi_b(u); u = [u_1,u_2] \tag{3.2.1}$$

The term $\psi_\lambda(u) = 2^{-2j}\psi(2^{-j}r_\theta u)$ represents all the rotated and dilated versions of the wavelet with $\lambda = (2^j, \theta)$ and $0 \leq j < J$ where J is the spatial scale and θ the orientation. The number of orientations L is used to discretize θ. The orientation of the L filters per scale are set to be $[\Theta,\ \Theta + \frac{\pi}{L},\ \Theta + \frac{2\pi}{L},\ \ldots,\ \Theta + \frac{(L-1)\pi}{L}]$. First-order scattering coefficients are calculated by first convolving the input signal with one of the generated complex wavelets. A wavelet transform is the operation of convolving the input image with each wavelet. Given a family of generated complex wavelets $\{\psi_{\lambda(u)}\}_\lambda$, a wavelet transform is defined as $\{x * \psi_\lambda\}_\lambda$. Then, the resulting filtered signal is downsampled by the scale factor 2^{j_1} of the wavelet chosen. Note that a wavelet transform is not translation *invariant*, but is translation *covariant*. Nonlinearity is needed to build translation invariant representations. Thus, a pointwise complex modulus

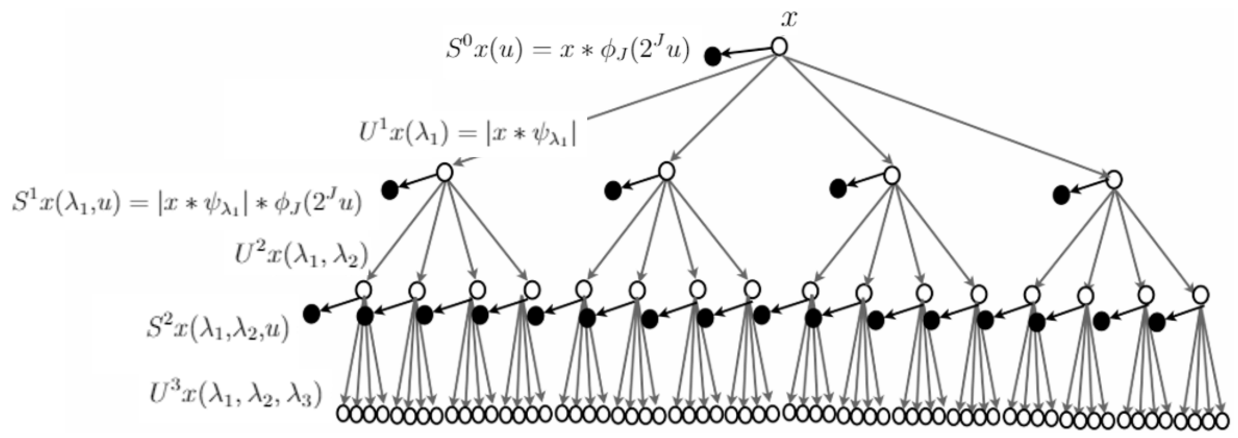
x
$S^0x(u) = x * \phi_J(2^J u)$
$U^1x(\lambda_1) = |x * \psi_{\lambda_1}|$
$S^1x(\lambda_1,u) = |x * \psi_{\lambda_1}| * \phi_J(2^J u)$
$U^2x(\lambda_1, \lambda_2)$
$S^2x(\lambda_1,\lambda_2,u)$
$U^3x(\lambda_1, \lambda_2, \lambda_3)$

Fig. 3.3. Scattering convolutional network. We compute the 0th-order coefficients $S^0x(u)$ by convolving the input signal x with a low pass filter ϕ_J with a spatial window of scale 2^J. First-order scattering coefficients $S^1x(\lambda_1,u)$ are calculated by convolving the input signal with one of the generated complex wavelets followed by a pointwise complex modulus operation. Then, the resulting real signal is smoothed via a low-pass filter. Higher orders can be computed by following the same iterative scheme. Image adapted from: [**91**].

is used to add nonlinearity, as shown in Equation 3.2.2:

$$U^1x(\lambda_1) = |x * \psi_{\lambda_1}| \tag{3.2.2}$$

Locally translation invariant representation is obtained by smoothing the resulting signal via a low pass filter ϕ_{2^J} on a spatial window of size 2^J. Finally, another downsampling step is applied, this time by a factor of 2^{J-j_1}, to obtain an optimally compressed output size. Mathematically, we have :

$$S^1x(\lambda_1,u) = |x * \psi_{\lambda_1}| * \phi_J(2^J u) \tag{3.2.3}$$

The resulting feature map has $J \cdot L$ channels. Second-order coefficients are generated similarly, with the addition of another cascade of wavelet transform and modulus operator before the low-pass smoothing, i.e.,

$$S^2x(\lambda_1,\lambda_2,u) = ||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| * \phi_J(2^J u) \tag{3.2.4}$$

Due to the interaction between the bandwidths and frequency supports of first and second order, only coefficients with $j_1 < j_2$ have significant energy. Hence, the second-order output yields a feature map with $\frac{1}{2}J(J-1)L^2$ channels.

To extract scattering coefficients of higher order moments, this procedure is applied iteratively for a path $p = (\lambda_1, \lambda_2, \dots, \lambda_m)$ of length m:

$$S^m x(p,u) = |||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| \dots | * \psi_{\lambda_m}| * \phi_J(2^J u) \tag{3.2.5}$$

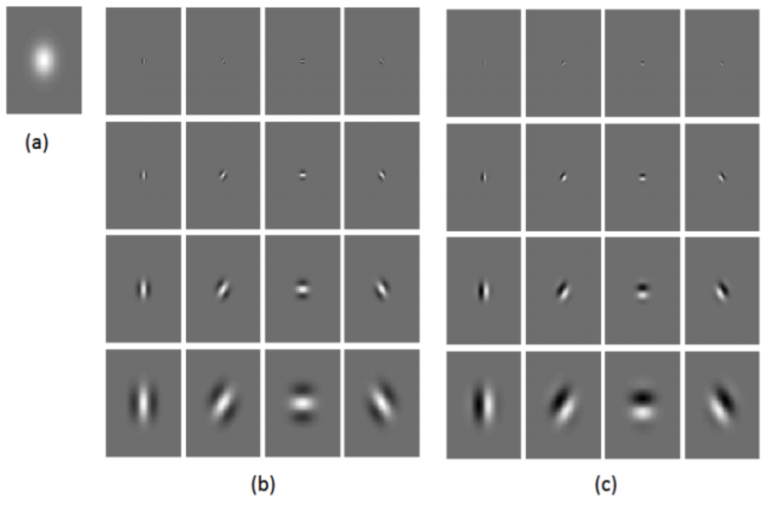

Fig. 3.4. Morlet wavelet filters with 4 scales ($J = 4$) and 4 orientations ($L = 4$). (a) Low pass filter ϕ. (b) real part of parts of Morlet wavelet filters ψ_λ and (c) imaginary parts of parts of Morlet wavelet filters ψ_λ. Filters are arranged according to orientations (columns) and scales (rows). Image from: [**20**]:

This cascade of wavelets creates the scattering convolutional network shown in Figure 3.3. The scattering coefficients are the concatenation of the coefficients $S^m x(p,u)$ for all paths p with $m \leq m_{\max}$. In Figure 3.3, these coefficients are represented by black dots.

3.3. Morlet Wavelets

Morlet wavelets are a typical example of filters used in conjunction with the scattering transform, and are defined as

$$\psi_{\sigma,\theta,\xi,\gamma}(u) = e^{-\|D_\gamma R_\theta(u)\|^2/(2\sigma^2)}(e^{i\xi u'} - \beta), \tag{3.3.1}$$

where β is a normalization constant to ensure wavelets integrate to 0 over the spatial domain, $u' = u_1 \cos\theta + u_2 \sin\theta$, R_θ is the rotation matrix of angle θ and $D_\gamma = \begin{pmatrix} 1 & 0 \\ 0 & \gamma \end{pmatrix}$. The four parameters can be adjusted and are presented in Table 3.2. Wavelets have traditionally been carefully constructed to ensure the resulting filter bank forms an efficient *tight frame* [**92, 93**] with well-established energy preservation properties. From one Morlet wavelet $\psi_{\sigma',\theta',\xi',\gamma'}(u)$,

Tableau 3.2. Parameters of Morlet wavelet

Param	Role
σ	Gaussian window scale
θ	Global orientation
ξ	Frequency scale
γ	Aspect Ratio

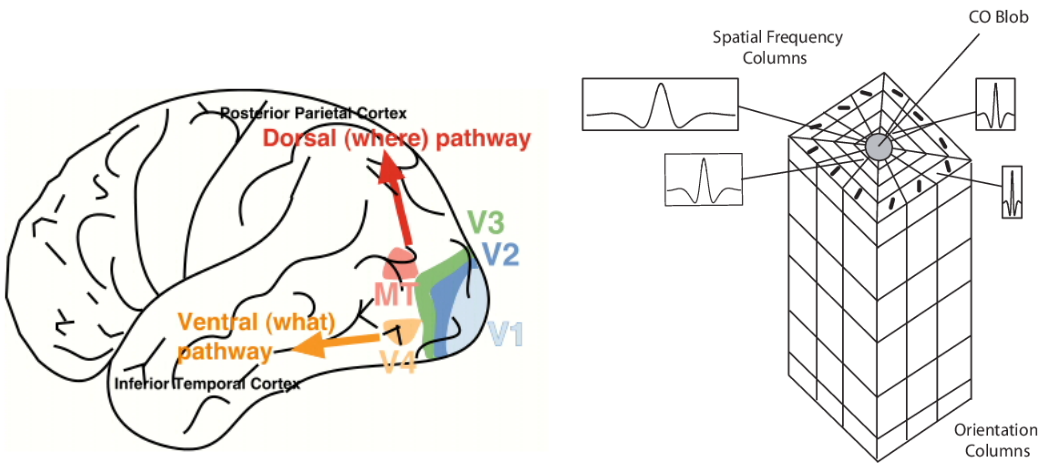

Fig. 3.5. Visual system and primary visual cortex (V1). (Left) There are two pathways of visual information. The ventral pathway is believed to carry information related to the recognition and discrimination of visual shapes. The dorsal pathway is responsible for the localization of objects. (Right) The simple cells in the primary virtual cortex are organized in hypercolumns. Each position defines an orientation and scale.

a tight frame is obtained by dilating it by factors 2^j, $0 \leq j < J$, and rotating by L angles θ equally spaced over the circle, to get $\{2^{-2j}\psi_{\sigma',\theta',\xi',\gamma'}(2^{-j}R_\theta(u))\}$, which is then completed with the low pass ϕ_J. This can be written in terms of the parameters in Table 3.2 as:

$$\psi_{2^j\sigma',\theta'-\theta,2^{-j}\xi',\gamma'}(u) = \psi(2^{-j}R_\theta(u)) \tag{3.3.2}$$

In the *tight frame* construction, the wavelets are generated from a mother wavelet via dilations and rotations, aimed to cover the frequency plane [**26**]. In Figure 3.2, we observe that the wavelets cover the entire frequency plane. By slight abuse of notations, we use ψ_λ here, $\lambda = (\sigma_j, \theta, \xi_j, \gamma_j)$, to denote such wavelets indexed by θ and j. Figure 3.4 illustrates some Morlet wavelets where J is set to 4 and L is set to 4. Morlet wavelet is not the only kind of wavelet that can be used with scattering transform. There are more families such as Haar wavelets [**31**], Daubechies wavelets [**87**] and Ricker wavelets [**120**].

3.4. Analogy to Visual System

Hubel and Wiesel [**66**] proposed an analogy between wavelets and the visual system of a mammal. The visual system is shown in Figure 3.5 (left). There are two different pathways of visual information in the visual system: ventral and dorsal. The ventral pathway, which is believed to carry information related to recognition and discrimination of visual shapes [**59**], starts at the primary cortex (V1) and terminates at the inferior temporal cortex. The dorsal pathway begins at the primary visual cortex (V1) and ends in the superior parietal lobule, which is responsible for the localization of objects [**117**]. Hubel and Wiesel [**66**] discovered that in V1 there is a family of cells called simple cells that behave similarly to linear filters.

The simple cells behave like filtering with Morlet wavelets. They also discovered that the simple cells are organized in hypercolumns as shown in Figure 3.5 (right). The position of each cell in the column defines the orientation and scale. This is similar to the wavelets generated via dilations and rotations in the tight-frame construction.

3.5. Properties

Scattering coefficients have several desired properties which are listed and described below.

No Learning. An important property of scattering coefficients is that they are constructed without learning. In contrast to a convolutional neural network, the filters are predetermined and fixed. The scattering coefficients are constructed in an unsupervised manner without the need for data.

Contractive Operation. The scattering coefficients are contractive: $||Sx - Sy|| \leq ||x - y||$. Representations that are not contractive can lead to misclassification. For example, Su et al. [**136**] demonstrate that only changing one pixel in images can lead to misclassification. Zajac et al. [**150**] show that changes in the border of images can also lead to misclassification. It is crucial to have contractive representations.

Energy Conservation. The scattering operator Ux preserves the signal: $||Ux(p)|| = ||x||$. In the scattering network, the modulus pushes the energy towards low frequencies. In other words, when $m_{\max}$ increases, the energy of the last layers converges toward zero. The low pass filters of the last layers capture the remaining energy. Thus, the scattering coefficients captures all the energy of the signal: $||Sx|| = ||x||$.

Translation Invariant. Another important property is that the scattering coefficients are translation invariant up to the spatial window size 2^J. This property arises from the final convolution with the low pass filter. It is crucial to be invariant to translation since, in image classification, the objects we are trying to classify can be in different locations, as shown in Figure 3.6.

Stable to Deformations. One of the most important property is that the scattering transform is stable to small deformations $L_\tau x(u) = x(u - \tau(u))$. Given $||\tau||_\infty = \sup_u |\tau(u)|$, $||\nabla\tau||_\infty = \sup_u |\nabla\tau(u)| < 1$ and S computed on paths of length $m \leq m_{\max}$, it is proven in [**93**] that:

$$||S(L_\tau x) - Sx|| \leq C m_{\max} ||x|| (2^{-J} ||\tau||_\infty + ||\nabla\tau||_\infty) \tag{3.5.1}$$

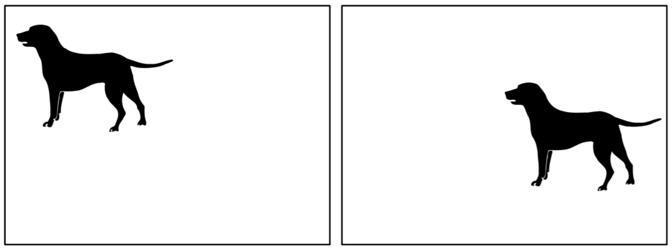

Fig. 3.6. Invariance to translation. Same object in different locations.

for a signal x of compact support with a second order Hessian term, which is negligible if $\tau(u)$ is regular. The translation term can be neglected if $2^J \geq ||\tau||_\infty$. Thus, the scattering transform is Lipschitz continuous to deformations:

$$||S(L_\tau x) - Sx|| \leq C m_{\max} ||x|| \quad ||\nabla \tau||_\infty. \tag{3.5.2}$$

The stability of small deformations is crucial in object classification. There can be a lot of intra-class variations as shown in Figure 3.1 with handwritten digits. The classification of handwritten digits should be stable to the different writing styles.

3.6. Texture Discrimination

All the properties can easily be satisfied with a trivial model, but it has been observed that scattering coefficients are useful for edge detection and texture discrimination. First-order scattering coefficients resemble classical image descriptors such SIFT descriptors [**26, 90**]. Higher-order scattering have been shown to be useful for texture discrimination [**26**]. A texture can be seen as a signal that is a realization of a stationary

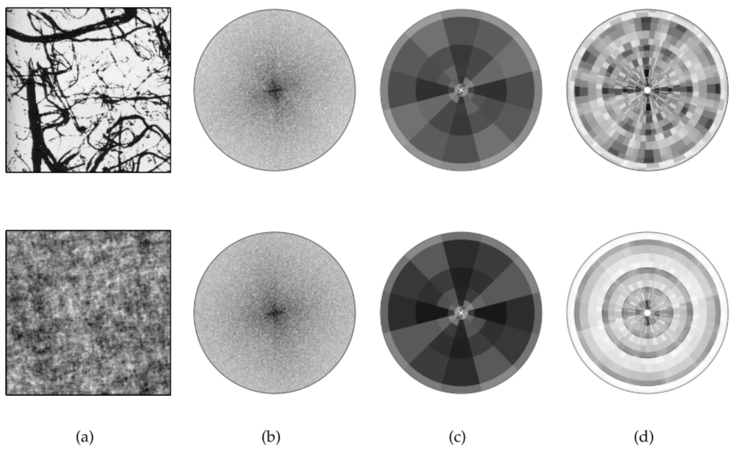

Fig. 3.7. Example of two different textures. (a) Image of two different textures. (b) Same Fourier spectrum. (c) Similar first-order scattering coefficients. (d) Different second-order scattering coefficients. Image from: [**26**]

process $X(u)$ that is not characterized by second-order moments [**26**].

A texture discrimination problem is a subproblem of image classification where the difficulty is to construct appropriate representations to classify texture. The Fourier spectrum often does not discriminate between different textures since it does not take into consideration high-order moments [**26**]. Figure 3.7 (a) shows an example of two different textures. In Figure 3.7 (b), we observe that both textures are identical from the Fourier transform point of view. Figure 3.7 (c) shows that the first-order scattering coefficients of the two textures are quite similar. On the other end, Bruna and Mallat [**26**] show that the scattering coefficients of a stationary process include second-order and higher-order moments that can discriminate between different textures [**26**]. Figure 3.7 (d) shows that the second-order coefficients are very different for each texture since the interference structures are completely different, which is why they are useful for the texture discrimination problem.

3.7. Scattering Variants

The scattering transform have been adapted to different domains, such as audio processing ([**8, 9, 7, 143, 144**]), medical signal processing ([**33**]) and quantum chemistry ([**61, 43, 44, 25**]) and still provides the same properties. In this section, we provide a non-exhaustive list of various variations of the scattering transform.

Roto-Translation Scattering Networks

As we now know, the scattering coefficients are translation invariant up to the spatial window and stable to small deformation. Sifre and Mallat [**128**] introduced the roto-translation scattering operation to construct scattering representations that are also rotation invariant. Then, Oyallon and Mallat [**105**] proposed the deep roto-translation scattering transform, where the scattering representations are not invariant to rotations but stable to rotations. To obtain representations that linearize variabilities along with the rotation angles, Oyallon and Mallat [**105**] compute a wavelet transform along with the angular variables.

3D Scattering Transform

Eickenberg et al. [**43**] introduced solid harmonic wavelets computed by multiplying solid harmonic functions with Gaussian windows dilated at different scales. They developed a 3D scattering transform and evaluated it on the estimation of quantum molecular energies. They achieved state-of-the-art results on small and large datasets of organic molecules.

Hybrid Scattering Networks

Recent work has shown that, in image classification, state-of-the-art results can be achieved by hybrid networks that harness the scattering transform as their early layers followed by learned layers based on a wide residual network architecture [**108**]. Oyallon et al. [**108**] proposed to use the scattering network as a generic and fixed initialization of the first layers of deep neural networks.

Generative Scattering Networks

Generative Adversarial Nets (GANs) [**52**] and Variational Auto-Encoders (VAEs) [**76**] have made incredible breakthroughs in the context of image generation. However, the underlying mathematics of these models is not fully understood. Angles and Mallat [**11**] study the statistical properties of these generators by introducing the generative scattering networks. The generative scattering networks consist of convolutional neural networks as the decoder to generate images and scattering networks as the encoder to obtain embedding. Since no learning is involved in the scattering transform, there is no need to train a discriminator. Jiasong et al. [**145**] pushed the idea further by proposing the generative fractional scattering networks. Instead of using the traditional scattering network as the encoder, they proposed to use more expressive fractional wavelet scattering networks to improve the quality of the generated images.

Spherical Scattering Networks

McEwen et al. [**96**] proposed the spherical scattering networks, which are essentially scattering networks constructed on the sphere. The network consists of a cascade of spherical scale-discretized wavelet transforms followed by absolute value activation functions. The scattering coefficients are obtained by projecting the resulting signal onto the spherical wavelet scaling function. This construction yields representations that are rotationally equivariant, invariant to isometries up to a particular scale and stable to diffeomorphisms [**96**]. Similar to the hybrid networks [**108**], they use the spherical scattering network as the early layers of a spherical CNN.

Learnability in the Scattering Transform

Adding learnable components to existing wavelet-based representations has been considered in several recent works in the context of time-series [**14, 125, 34, 15**]. Balestriero et al. [**14, 125**] learn a spline parametrized mother wavelet for 1D problems. Similarly, Cosentino and Aazhang [**34**] parameterized the group transform in the context of time-series data.

Graph and Geometric Scattering Networks

Various datasets are better modeled by graphs or manifolds since they have intrinsically non-Euclidean structure [**112**]. Thus, recent works have extended the scattering transform to graph domains ([**45, 46, 154, 109**]). The scattering transform has also been generalized to manifolds ([**113, 47, 112**]).

3.8. Scattering Implementations

The scattering transform is mathematically elegant, but to implement it, some careful considerations on memory and runtime are needed. There are several packages that implement the wavelet transform. Three common ones (e.g. Matlab's wavelet toolbox, PyWavelets, and Kymatio) are listed and described below.

Matlab's wavelet toolbox [**101**] offers a variety of families of wavelets that are listed in Table 3.3. The package implements 1D and 2D continuous wavelet transform. It also implements 1D, 2D, and 3D discrete wavelet transform. Some functions can perform operations on a GPU in order to accelerate the workflow.

PyWavelets [**86**] is an open-source software for Python. It offers a variety of predefined wavelets shown in Table 3.3. It also gives the possibility to the user to create custom filters banks. The package implements n-dimensional discrete wavelet transforms and 1D continuous wavelet transform. The application programming interface (API) is similar to Matlab's wavelet toolbox. The package offers additional functionality such as support for dimension $n > 3$ and support for real and complex-values data. The PyWavelets package is now used by other software packages. Scikit-image [**140**] and the Operator Discretization Library (ODL) [**65**] use the package to enable wavelet-based image denoising and to enable wavelet-based regularization in iterative inverse problems, respectively.

Tableau 3.3. Available wavelet families in Matlab's wavelet toolbox and PyWavelets

Haar	Daubechies
Biorthogonal	Reverse Biorthogonal
Gaussian	Complex Gaussian
Morlet	Complex Morlet
Symlets	Shannon
Discrete FIR approximation of Meyer	Frequency B-Spline
Mexican Hat	Coiflets

Kymatio [**10**] is an open-source package for Python that implements the wavelet scattering transform in 1D, 2D and 3D. The package is compatible with deep learning frameworks such as Pytorch and TensorFlow, which allow GPU accelerated computations. In contrast with Matlab's wavelet toolbox and PyWavelets, the Kymatio implementation of the scattering transform is differentiable. However, it only offers Morlet wavelets. To reduce memory requirements and save GPU memory, the algorithm traverses the scattering network depth-first, as shown in Figure 3.8, instead of computing the scattering coefficients layer by layer.

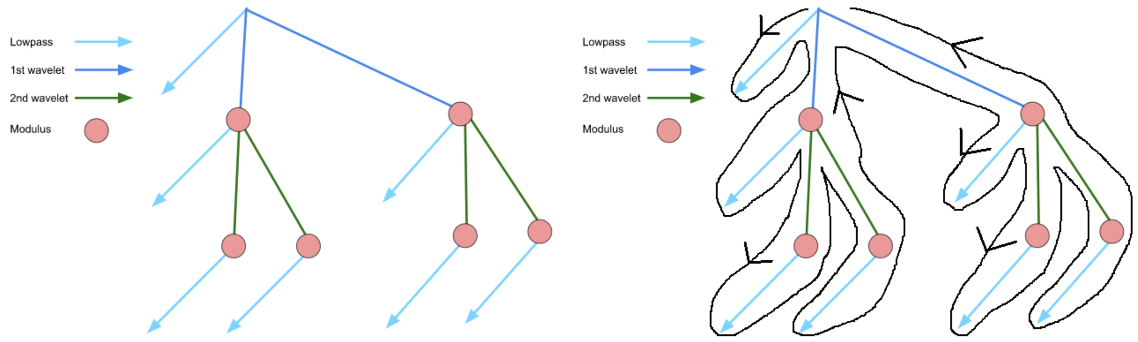

Fig. 3.8. Kymatio traversal algorithm. (Left) Scattering tree (Right) Scattering traversal strategy. The tree is traverses in a breadth-first fashion. Image adapted from: [**10**]

A comparison of the three packages is shown in Table 3.4.

Tableau 3.4. Comparison of Matlab's wavelet toolbox, PyWavelets and Kymatio.

Package	Wavelet Families	GPU	Differentiable	License	Language
Matlab's Wavelet toolbox	Variety	✓		Proprietary	MATLAB
PyWavelets	Variety			MIT	Python
Kymatio	Morlet Only	✓	✓	BSD-3	Python

Chapter 4

Parametric Scattering Networks

In this chapter, we relax the standard tight frame construction by considering another alternative where a small number of wavelet parameters used to create the wavelet filterbanks are optimized for the task at hand. Focusing on Morlet wavelets, we propose to learn the scales, orientations, and aspect ratios of the filters to produce problem-specific parameterizations of the scattering transform. We first discuss scattering parameter initialization in Section 4.1. We then introduce our parametric scattering transform in Section 4.2. In Section 4.3, we describe how to differentiate through the parametric scattering network. We describe the experimental protocol used to evaluate our approach in Section 4.4. Finally, in Section 4.5, we explore the different filter construction schemes by comparing the wavelet filter parameterizations they produce when optimized over different datasets.

4.1. Initialization

We consider two different ways of initializing the Morlet wavelet filters. First, a tight-frame initialization follows common implementations of the scattering transform by setting $\sigma_{j,\ell} = 0.8 \times 2^j$, $\xi_{j,\ell} = \frac{3\pi}{4}2^{-j}$, and $\gamma_{j,\ell} = \frac{4}{L}$ for $j = 1, \ldots, J$, $\ell = 1, \ldots, L$, while for each j, we set $\theta_{j,\ell}$ to be equally spaced on $[0,\pi)$. Second, as an alternative, we consider a random initialization where these parameters are sampled as $\sigma_{j,\ell} \sim \log(U[\exp 1, \exp 5])$, $\xi_{j,\ell} \sim U[0.5,1]$, $\gamma_{j,\ell} \sim U[0.5,1.5]$, and $\theta_{j,\ell} \sim U[0,2\pi]$. That is, orientations are selected uniformly at random on the circle, the filter width σ is selected using an exponential distribution across available scales and the spatial frequency ξ is chosen to be in the interval $[0.5, 1]$, which lies in the center of the feasible range between aliasing ($> \pi$) and the fundamental frequency of the signal size ($2\pi/N$ where N is the number of pixels). Finally, we select the *aspect ratio* variable to vary around the spherical setting of 1.0, with a bias towards stronger orientation selectivity (0.5) compared to lesser orientation selectivity (1.5). The two initialization approaches are summarized in Table 4.1.

Tableau 4.1. Parameters initialization of Morlet filters.

Parameter	Tight Frame	Random
σ	$0.8 * 2^j$	$log(U[e,e^5])$
θ	Equally spaced	$U[0,2\pi]$
ξ	$\frac{3\pi}{4}2^{-j}$	$U[0.5,1]$
γ	$\frac{4}{L}$	$U[0.5,1.5]$

4.2. Morlet Wavelet Parameterization

While the wavelet filters in the scattering transform are traditionally fixed to approximate a tight frame, we let the network learn the optimal parameters of each wavelet. In other words, we constrain our filters to always be Morlet wavelets by only optimizing the parameters in Table 3.2. We adapted the Kymatio software package [**10**] to create the parametric scattering network. We consider two parameterization approaches.

Morlet Canonical Parameterization. The first approach considered is the Morlet canonical parameterization of the wavelet. In this approach, the parameters of each wavelet filter are optimized separately.

Morlet Equivariant Parameterization. In the Morlet canonical parameterization approach, the canonical parameters of each filter are learned. As an alternative method, we consider the Morlet equivariant parameterization in which the number of learnable parameters is reduced by a factor L. Each filter per scale is constructed using the same four parameters in Table 3.2: σ, ξ, γ and Θ. However, the global orientation of the L filters for each scale are set to be $[\Theta, \Theta+\frac{\pi}{L}, \Theta+\frac{2\pi}{L}, \ldots, \Theta+\frac{(L-1)\pi}{L}]$. By construction, the tight-frame filters are equivariant.

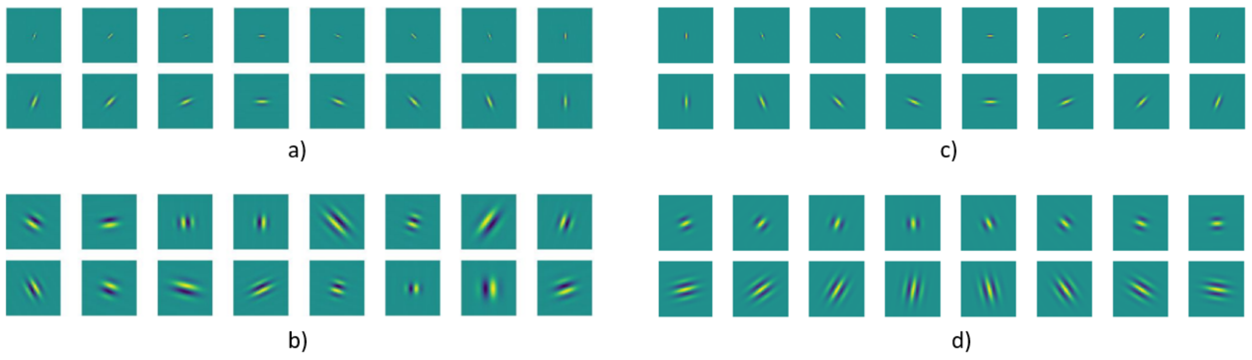

Fig. 4.1. Visualization of real part of Morlet wavelet filters with spatial scale of $J = 2$ and $L = 8$. (**a**) Canonical wavelets initialized with *tight frame*. (**b**) Canonical wavelets initialized *randomly*. (**c**) Equivariant wavelets initialized with *tight frame*. (**d**) Equivariant wavelets initialized *randomly*.

Figure 4.1 shows Morlet canonical and equivariant wavelet filters initialized with tight frame and random initialization. We observe that the canonical and equivariant wavelet filters initialized with random initialization are different. With the Morlet canonical parameterization, each filter is different since they are individually initialized with different parameters. With the Morlet equivariant approach, we observe that the filters in each row are identical, except for the global orientation. The row corresponds to a different scale. Since J is set to 2, we have two rows.

4.3. Backpropagation through the Parametric Scattering Network

We show that it is possible to backpropagate through the parametric scattering network. Namely, we verify the differentiability of our approach by explicitly computing the partial derivatives with respect to parameters of Table 3.2. First, the $\mathbb{R}$-linear derivative of the complex modulus $f(z) = |z|$ is $f'(z) = \frac{z}{|z|}$. Next, we show the differentiation of convolution with wavelets with respect to their parameters. For simplicity, we focus here on differentiation of the Gabor portion of the filter construction from Equation 3.3.1, written as:

$$\begin{aligned}\varphi(u) = &\exp(-\frac{1}{2\sigma^2}(u_1^2(\cos^2(\theta) + \sin^2(\theta)\gamma^2) + u_2^2(\cos^2(\theta)\gamma^2 + \sin^2(\theta)) \\ &+ 2\cos(\theta)\sin(\theta)u_1u_2(1-\gamma^2)) + i\xi(\cos(\theta)u_1 + \sin(\theta)u_2)).\end{aligned}$$

It is not difficult to extend this derivation to Morlet wavelets, but the resulting expressions are rather cumbersome and left out for brevity. Its derivatives with respect to the parameters are

$$\begin{aligned}
\frac{\partial\varphi}{\partial\theta}(u) &= \frac{1}{\sigma^2}(u_2\cos\theta - u_1\sin\theta)(i\xi\sigma^2 + u_1(\gamma^2-1)\cos\theta + u_2(\gamma^2-1)\sin\theta)\varphi(u);\\
\frac{\partial\varphi}{\partial\sigma}(u) &= \frac{1}{\sigma^3}(u_1^2(\cos^2\theta + \gamma^2\sin^2\theta) + u_2^2(\gamma^2\cos^2\theta + \sin^2\theta) + 2u_1u_2\cos\theta\sin\theta(1-\gamma^2))\varphi(u);\\
\frac{\partial\varphi}{\partial\xi}(u) &= i(u_1\cos\theta + u_2\sin\theta)\varphi(u);\ \text{and}\\
\frac{\partial\varphi}{\partial\gamma}(u) &= -\frac{1}{\sigma^2}(u_1^2\gamma\sin^2\theta + u_2^2\gamma\cos^2\theta - 2u_1u_2\gamma\cos\theta\sin\theta)\varphi(u).
\end{aligned}$$

Finally, the derivative of the convolution with such filters is given by

$$\frac{\partial}{\partial\zeta}(f*\varphi)(t) = \int f(t-u)\frac{\partial\varphi}{\partial\zeta}(u)du$$

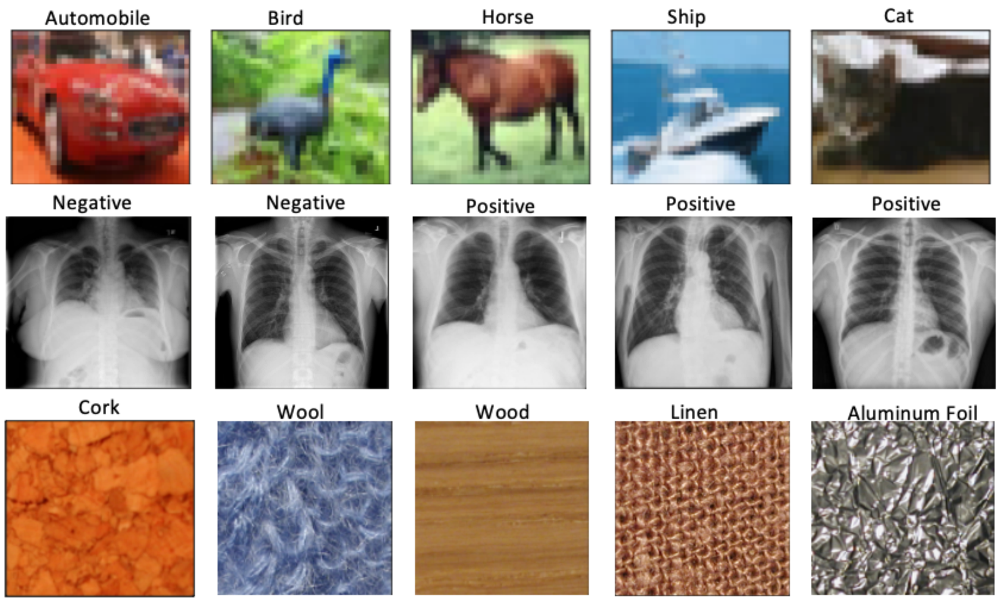

Fig. 4.2. Samples from the three datasets: CIFAR-10 (Top), COVIDx CRX-2 (Middle), and KTH-TIPS2 (Bottom).

where ζ is any of the filter parameter from Table 3.2. It is easy to verify that these derivations can be chained together to propagate through the scattering cascades. We can now learn the parameters jointly with other parameters in an end-to-end differentiable architecture.

4.4. Experimental Protocol

Our empirical evaluations are based on three image datasets: CIFAR-10, COVIDx CRX-2, and KTH-TIPS2, as illustrated in Figure 4.2. CIFAR-10 and KTH-TIPS2 are natural image and texture recognition datasets, correspondingly. They are often used as general-purpose benchmarks in similar image analysis settings [**13, 128**]. COVIDx CRX-2 is a dataset of X-ray scans for COVID-19 diagnosis; its use here demonstrates the viability of our parametric scattering approach in practice, e.g., in medical imaging applications.

We evaluate the use of the parametrized scattering network with two standard models. In the first case, we consider the scattering as feeding into a simple linear model (denoted LL). The LL configurations are used to evaluate the linear separability of the obtained scattering representations and have the added benefit of providing a more interpretable model. In the second case, we take the approach of [**108**] and consider the scattering as the first stage of a deeper CNN, specifically a Wide Residual Network (WRN) [**149**]. The description of the WRN architecture used for the experiments is given in Table 4.2.

For both models (LL and WRN), we compare learned parametric scattering networks

(LS) to fixed ones (S). For learned scattering (LS), we consider two scattering parameterization approaches: *Morlet canonical*, described in Section 4.2 and *Morlet equivariant*, described in Section 4.2. To show the importance of the parametric approach, we also ablate the naive parameterization where all pixels of the wavelets are adapted, which we refer to as a pixel-wise parameterization. Parametric scattering networks constrain the filters always to be Morlet wavelets. In the pixel-wise parameterization approach, we relax the constraints and instead optimize the pixels of the Fourier transform of the wavelets. This alternative is similar to initializing the kernels of a CNN with Morlet wavelets and letting the model optimize the pixels of the kernels without any constraint using backpropagation.

We consider both random and tight-frame (TF) initialization for each scattering architecture. The fixed scattering models determined by the TF construction are equivalent to traditional scattering transforms. Finally, we also compare our approach to a fully learned WRN (with no scattering priors), and ResNet-50 [**58**] applied directly to input data. We note that the latter is unmodified form its ImageNet architecture and that we do not initialize it with pre-trained weights.

Across all scattering configurations, a batch-normalization layer with learnable affine parameters is added after all scattering layers. Classification is performed via a softmax layer yielding the final output. All models are trained using cross-entropy loss, minimized by stochastic gradient descent with a momentum of 0.9. Weight decay is applied to the linear model and to the WRN. The learning rate is scheduled according to the one cycle scheduler, which improves convergence during optimization, especially in the small data regime, due to its so-called super convergence policy [**132**]. The scheduler's div factor is always set to 25.

4.5. Exploring Dataset-Specific Parameterizations

We first compare dataset-specific Morlet wavelet parameterizations and evaluate their similarities to a tight frame. Specifically, we train our parametric scattering networks using the canonical Morlet wavelet formulation with a linear classification layer and quantitatively and qualitatively compare the similarities of the learned filter bank to the tight-frame initialization. To facilitate quantitative comparison, we use a distance metric for comparing the sets of Morlet wavelet filters and Morlet wavelet filterbanks (i.e., scattering network instantiations), allowing us to measure deviations from the tight-frame initialization.

We evaluate distances between two individual Morlet wavelets as:

$$\Upsilon(M_1,M_2) = \left\|(\sigma_1,\xi_1,\gamma_1)^T - (\sigma_2,\xi_2,\gamma_2)^T\right\|_2 + \text{arcdist}(\theta_1,\theta_2) \tag{4.5.1}$$

Tableau 4.2. Description of WRN hybrid architecture used for the experiments. Each convolutional layer represents a 2D convolution followed by a batch normalization and a ReLU non-linearity.

Stage	**Description**
scattering	Learned or Tight Frame
conv1	3×3, Conv Layer $128 \rightarrow 256$
conv2	$\begin{bmatrix} 3 \times 3, \text{Conv Layer } 256 \\ 3 \times 3, \text{Conv Layer } 256 \end{bmatrix} \times 4$
conv3	$\begin{bmatrix} 3 \times 3, \text{Conv Layer } 256 \\ 3 \times 3, \text{Conv Layer } 256 \end{bmatrix} \times 4$
avg-pool	Avg pooling to a size 1x1

where $M_i = (\sigma_i,\xi_i,\gamma_i,\theta_i)^T$ denotes the parameterization of the Morlet wavelet. We use the arc distance on the unit circle to compare values of theta. Since the set of learned scattering filters does not have a canonical order, to compare a learned scattering network to the tight frame scattering network, we use a matching algorithm to match one set of filters to another. Specifically, we first compute Υ between all combinations of filter pairs from both networks, then use a minimum cost bipartite matching algorithm [**81**] to find the minimal distance match between the two sets of filters. The final distance we use as a notion of similarity between two scattering networks is the sum of Υ for all matched pairs in the bipartite graph. Henceforth, we will refer to this distance as the *filterbank distance*.

The graph in Figure 4.3 leverages the *filterbank distance* to show the evolution of scattering networks initialized from a tight frame and trained on different datasets. Each network is trained on 1188 samples of its respective dataset (the standard size for KTH-TIPS2). All filters deviate quickly from a tight frame, but KTH-TIPS2's keep changing the longest and ultimately deviate the most. We also observe that filters initialized with the random initialization become more similar to our tight-frame initialization during training (see Appendix B).

On the left-hand side of Figure 4.3, we visualize the dataset-specific scattering network parameterizations in Fourier space. White contours are drawn around each Morlet wavelet for clarity. The top black border corresponds to tight-frame initialization at J=2, shown for comparison to CIFAR-10 in blue (also J=2). The bottom black frame corresponds to tight-frame initialization at J=4, shown for comparison to COVIDX-CRX2 red and KTH-TIPS2 yellow (both J=4).

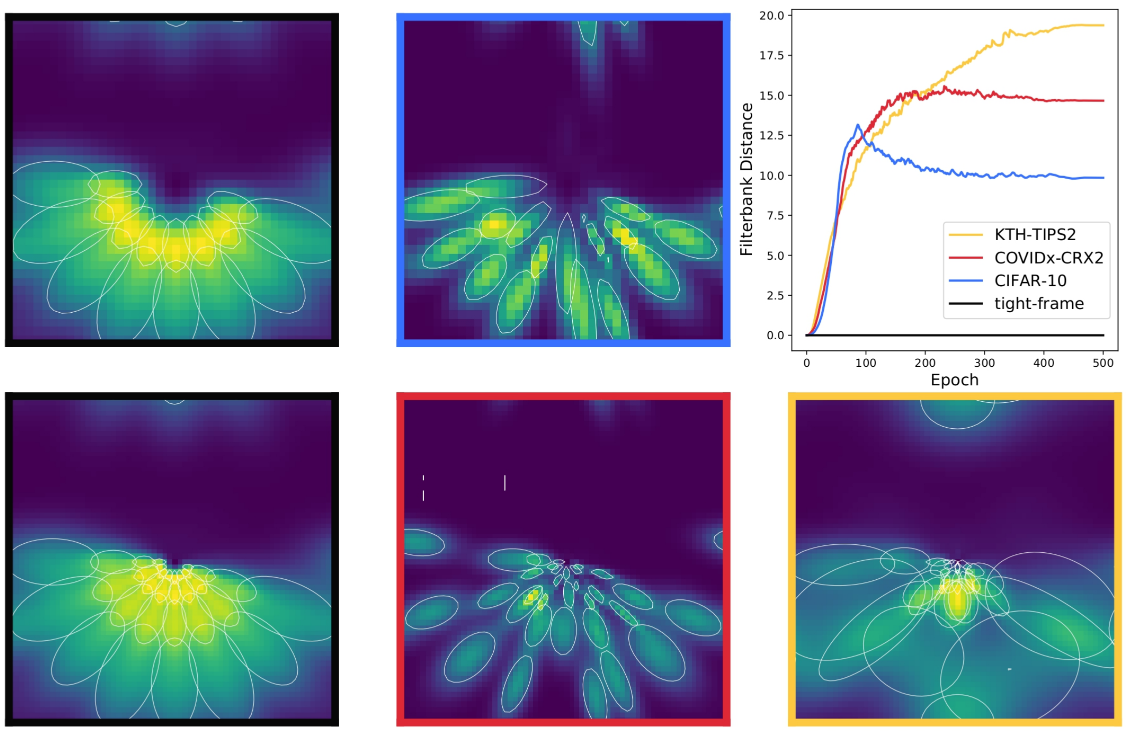

Fig. 4.3. Parametric scattering network learns dataset specific filters initialized with tight frame. The graph (top right) shows the *filterbank distance* over epochs as the filters are trained on different datasets. We visualize dataset specific parameterizations of scattering filterbanks (border colors from the legend) in Fourier space. Scattering filters optimized for natural (CIFAR-10) and medical image (COVIDx CRX2) become more orientation-selective, i.e., thinner in the Fourier domain. On the other hand, filters optimized for texture discrimination (KTH-TIPS2) become less orientation-selective and deviate most from a tight-frame setup.

The filters optimized on the KTH-TIPS2 texture dataset (yellow) become less orientation-selective (wider in Fourier space) than the tight-frame initialization, with filters at J=0 becoming the least orientation-selective of the whole filter bank. We note that the filters at spatial scales J= 2 and 3 seem to change the most from a tight frame. In contrast, the filters optimized on COVIDx-CRX2 become more orientation-selective in general, i.e., thinner in Fourier space. The filters optimized on CIFAR-10 mirror those optimized on COVIDx-CRX2, also becoming more orientation-selective than their tight-frame counterparts. We suspect that this is due to a reliance on edges for object classification datasets, which seem to require more orientation-selective filters. On the other hand, the Morlet wavelets optimized for texture classification seem to discard some edge information favouring less orientation-specific filters. Each dataset-specific parameterization seems to discard unneeded information from the tight-frame initialization to accentuate problem-specific attributes. In Section 5.2, we demonstrate these learned filters are not only interpretable but improve

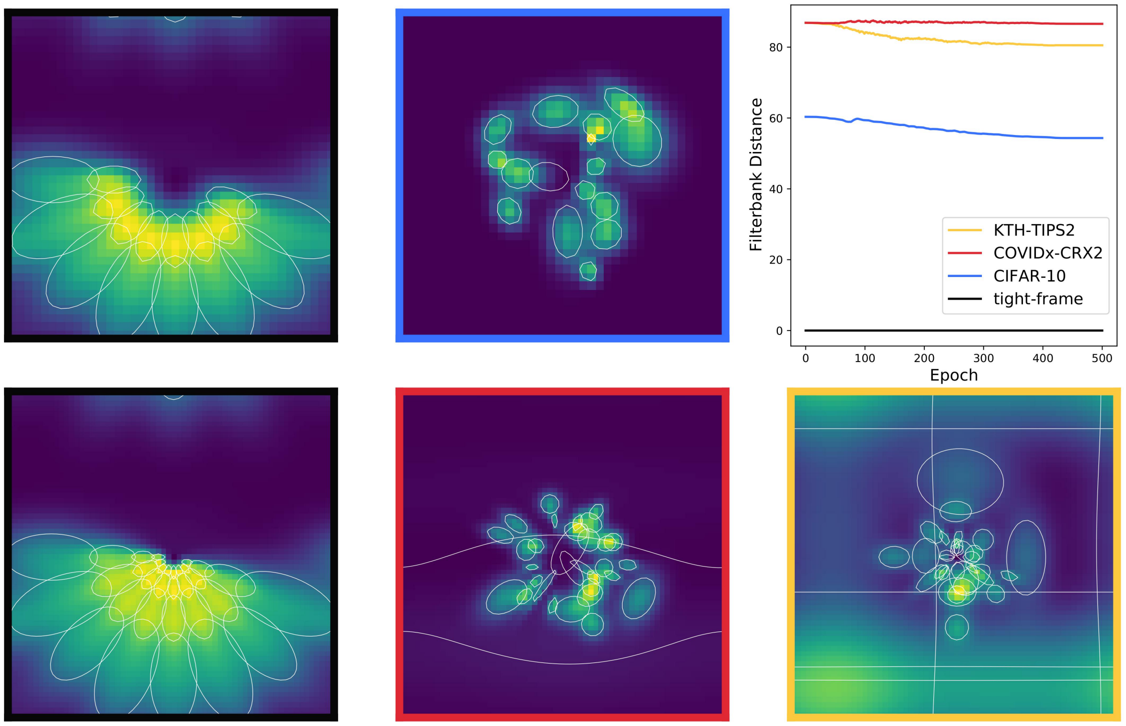

Fig. 4.4. Parametric scattering network learns dataset specific filters initialized randomly. The graph shows the *filterbank distance* over epochs as the filters, initialized from random init, are trained on different datasets. To the left, we visualize dataset specific parameterizations of scattering filterbanks in Fourier space. The graph on the right shows that the randomly initialized filterbanks become more similar to a tight frame during training.

task performance, suggesting the tight frame is not optimal for many problems of interest. Nonetheless, a tight frame does constitute a good starting point for learning. Indeed, the dataset-specific parameterizations for COVIDX-CRX2 and KTH-TIPS2 are, visually, very different. Yet, they move similar filterbank distances from the tight-frame initialization (see Figure 4.3), which are small relative to the distances observed for randomly initialized and trained models. In Figure 4.4, we show how the filters adapt when initialization begins from a random setting. We note the deviation to a tight frame is much greater than when we initialize in a tight frame. However, as per our filterbank distance, we observe the filters do move closer to the tight frame than their initialization.

We have introduced the parametric scattering network, presented two initialization and parameterization approaches, described the experimental protocol and explored the different filter construction schemes. In the next chapter, we evaluate the robustness of the parametric scattering network to deformation and its performance in the small-sample classification settings.

Chapter 5

Experiments

In this chapter, we present the results obtained from various experiments. We first evaluate, in Section 5.1, the robustness of our parametric scattering network to deformation. In Section 5.2, we demonstrate the advantages of our approach in limited labeled data settings and study the adaptation of the wavelet parameters toward a supervised task. Finally, in Section 5.3, we investigate the adaptation of the parametrized scattering using an unsupervised objective.

5.1. Robustness to Deformation

In [**93**], it is shown that the scattering transform is stable to small deformations of the form $x(u - \tau(u))$ where $x(u)$ is a signal and τ a diffeomorphism. Given the substantial changes to the filter composition in the learning process, we ask now whether these seem to significantly deviate from the stability result obtained from the carefully handcrafted construction proposed in [**93**], and extensively used in previous work, e.g., [**26, 44**]. To evaluate the robustness of our parametric scattering network to different geometric distortions, we apply several tractable deformations to a chest X-ray image x with varying deformation strength. The transformed image is denoted by $\tilde{x}$. For each of the different deformation strengths, we plot the Euclidean distance between the scattering feature constructed from the original image $S(x)$ and the scattering feature constructed from the transformed image $S(\tilde{x})$. We then normalize the obtained distance by $S(x)$ to measure the relative deviation in scattering coefficients (handcrafted or learned). The learned scattering networks use the Morlet canonical parameterization and are combined with a linear layer during training.

Tableau 5.1. Deformations and their maximum value

Deformation	Maximum Value
Rotation	10
Translation	22
Custom1	1
Custom2	1

Deformations used are rotation, translation, and several diffeomorphisms (denoted Custom 1 and Custom 2), and strengths for the deformations range from 0 to the maximum value for the deformation given in Table 5.1.

Custom 1, $\tau_\epsilon^1(u)$, and Custom 2, $\tau_\epsilon^2(u)$, are defined as such:

$$\tau_\epsilon^1(u) = \epsilon \begin{bmatrix} 0.3u_1^2 + 0.2u_2^2 \\ 0.2(0.2u_1) \end{bmatrix}, \tau_\epsilon^2(u) = \epsilon \begin{bmatrix} 0.3(u_1^2 + u_2^2) \\ -0.3(2u_1u_2) \end{bmatrix}.$$

Figure 5.1 demonstrates representative results for a small rotation, translation and custom transformations on images from the COVIDx datasets. We observe that the substantial change in the filter construction retains the scattering robustness properties for these simple deformations, thus indicating that the use of learned filters (instead of designed ones) does not necessarily detract from the stability of the resulting transform.

5.2. Small Data Regime

We evaluate the parametric scattering network in limited labeled data settings. Following the evaluation protocol from [**108**], we subsample each dataset at various sample sizes to showcase the performance of scattering-based architectures in the small data regime. In our experiments, we train on a small random subset of the training data but always test on the entire test set as done in [**108**]. To obtain comparable and reproducible results, we control for deterministic GPU behavior and assure that each model is initialized the same way for the same seed. Furthermore, we use the same set of seeds for models evaluated on the same number of samples. For instance, the TF learnable hybrid with a linear model would be evaluated on the same ten seeds as the fixed tight-frame hybrid with a linear model when trained on 100 samples of CIFAR-10. Some fluctuation is inevitable when subsampling datasets. Hence all our figures include averages and standard error calculated over different seeds.

5.2.1. CIFAR-10

CIFAR-10 consists of 60,000 images of size $32 \times 32 \times 3$ from ten classes. The train set contains 50,000 class-balanced samples, while the test set contains the remaining images. The linear models were trained using a max learning rate of 0.06 for all parameters on 5K, 1K, 500, and 500 epochs for 100, 500, 1K, 50K samples, respectively. The hybrid WRN models were trained using a max learning rate of 0.1 on 3K, 2K, 1K, and 200 epochs for 100, 500, 1K, and 50K samples, respectively. We use batch gradient descent except when the models are trained with 50K samples where we use mini-batch gradient descent of size 1024. On the entire training set, we also train the models on ten seeds and, in all cases, the standard errors are always inferior to 0.3. All scattering networks use a

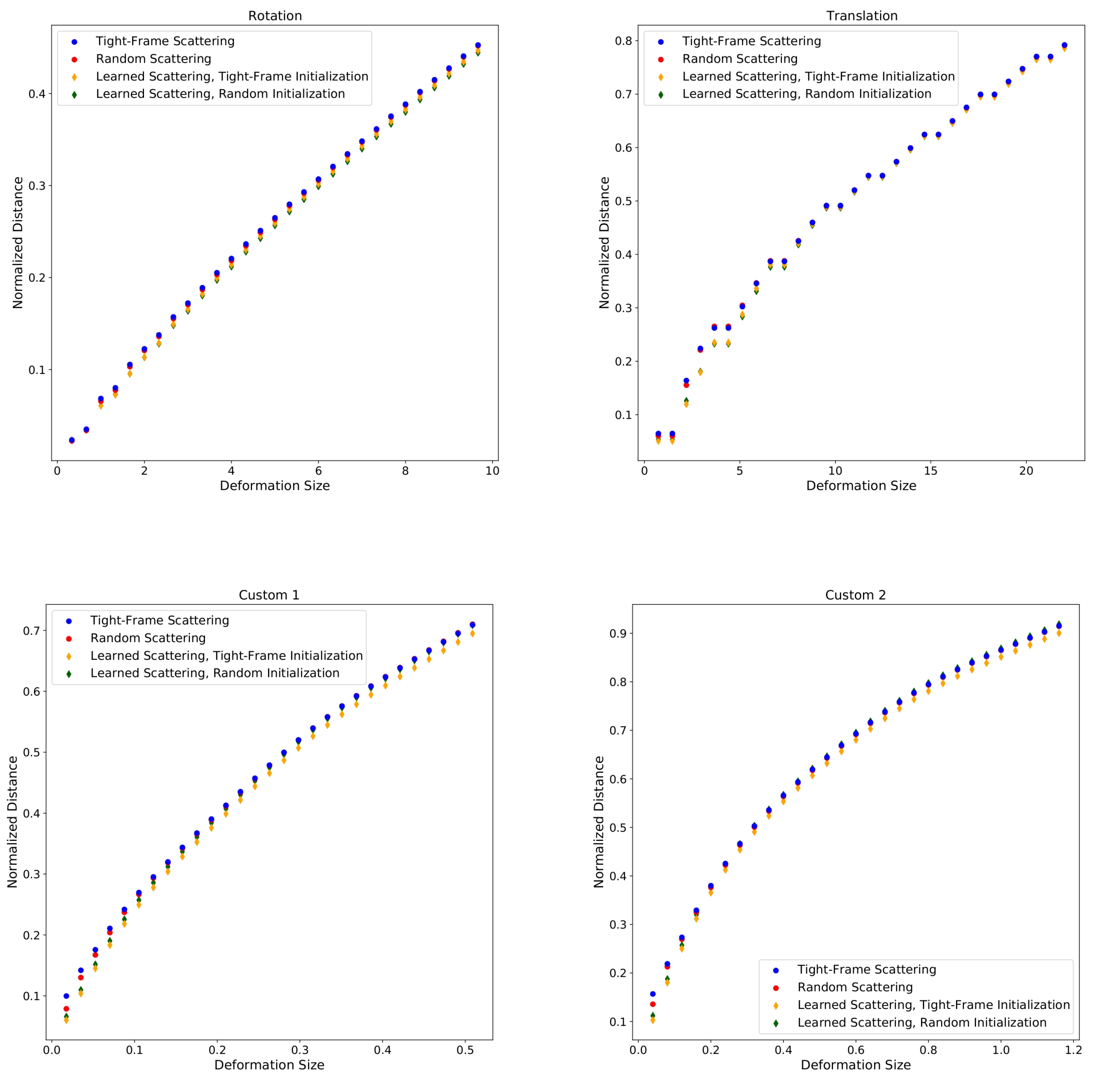

Fig. 5.1. Normalized distances between scattering representations of an image and its deformation. (Top Left) Rotation Transformation. (Top Right) Translation Transformation. (Bottom Left) Custom 1. (Bottom Right) Custom 2 Transformation.

spatial scale $J = 2$. Table 5.2 reports the evaluation of our learnable scattering approach on CIFAR-10 with training sample sizes of 100, 500, 1K, and 50K. The training set is augmented with horizontal flipping, random cropping, and pre-specified autoaugment [**36**] for CIFAR-10. We used autoaugment [**36**] to showcase the best possible small-sample results.

As shown in Table 5.2, the scattering networks with wavelets optimized pixel-wise perform the worst in the small-data regime. It shows that with limited labeled samples, there is not enough data and too many learnable parameters to effectively learn the wavelets' pixels. Adding more constraints (i.e., constraining the wavelets to be Morlet) is beneficial

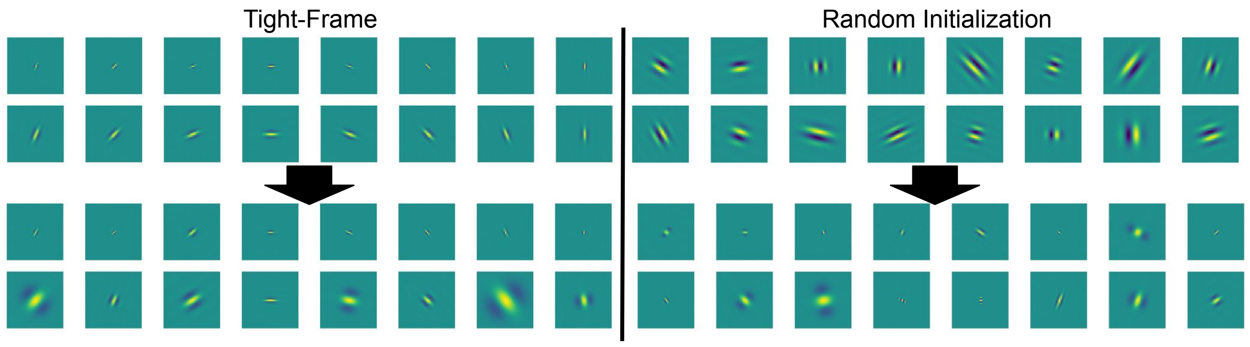

Fig. 5.2. Initialized canonical wavelet filters pre and post-training. Real part of Morlet canonical wavelet filters with $J = 2$ initialized with *tight-frame* (Left) and *random* (Right) schemes before (Top) and after (Bottom) training. The filters were optimized on the entire CIFAR-10 training set with linear model. For the tight-frame filters, we observe substantial changes in both scale and aspect ratio. On the other hand, all random filters undergo major changes in orientation and scale.

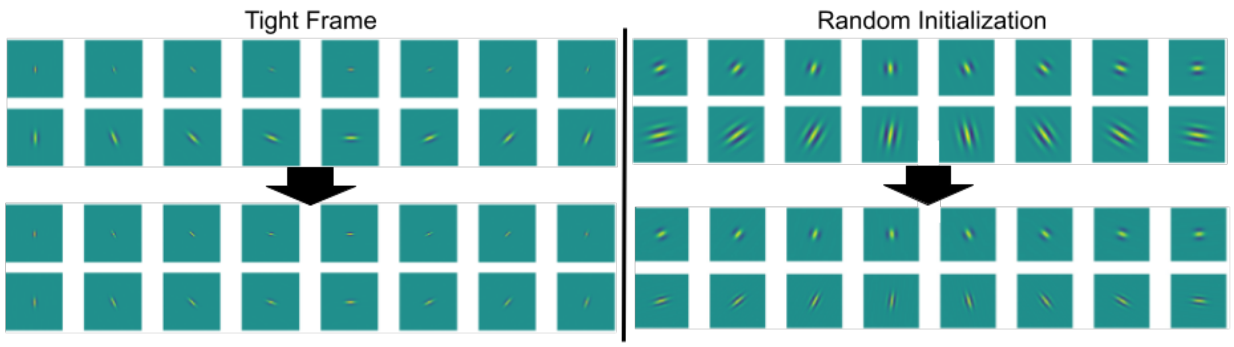

Fig. 5.3. Initialized equivariant wavelet filters pre and post-training. Real part of Morlet equivariant wavelet filters with $J = 2$ initialized with *tight-frame* (Left) and *random* (Right) schemes before (Top) and after (Bottom) training. The filters were optimized on the entire CIFAR-10 training set with linear model. In the two figures, each row corresponds to a different scale. Since J is set to 2, we have two rows. We observe that the equivariant filters in each scale/row are the same, except for the global orientation.

in this setting. We also observe that the Morlet canonical parameterization yields similar performance to the Morlet equivariant parameterization (i.e., most standard errors overlap). Thus, adding even more constraints, by reducing the number of learnable parameters in the parametric scattering transform, does not degrade the performance in the small-data regime.

We observe that randomly initialized learnable with canonical parameterization only achieves similar performance to TF learnable canonical when trained on the whole dataset. These results suggest the TF initialization, derived from rigorous signal processing principles, is empirically beneficial as a starting point in the very few sample regime but can be improved upon by learning.

Tableau 5.2. CIFAR-10 mean accuracy and std. error over 10 seeds, with $J = 2$ and multiple training sample sizes. Learnable scattering with TF initialization improves performance for all architectures, while randomly initialized scattering requires more training data to reach similar performance.

Arch.	Init.	Parametrization	100 samples	500 samples	1000 samples	All
LS+LL†	TF	Canonical	37.84 ± 0.57	$\mathbf{52.68} \pm 0.31$	$\mathbf{57.43} \pm 0.17$	**69.57**
LS+LL†	TF	Equivariant	$\mathbf{39.69} \pm 0.56$	51.98 ± 0.25	57.01 ± 0.16	66.65
LS+LL	TF	Pixel-Wise	32.30 ± 0.69	47.14 ± 0.91	51.87 ± 0.34	64.53
S +LL	TF	-	36.01 ± 0.55	48.12 ± 0.25	53.25 ± 0.24	65.58
LS+LL†	Rand	Canonical	34.81 ± 0.60	49.6 ± 0.39	55.72 ± 0.39	69.39
LS+LL†	Rand	Equivariant	34.67 ± 0.73	46.59 ± 0.60	52.95 ± 0.36	65.64
LS+LL	Rand	Pixel-Wise	29.44 ± 0.41	42.14 ± 0.27	47.44 ± 0.43	62.72
S +LL	Rand	-	29.77 ± 0.47	41.85 ± 0.41	46.3 ± 0.37	57.72
LS+WRN†	TF	Canonical	$\mathbf{43.60} \pm 0.87$	$\mathbf{63.13} \pm 0.29$	70.14 ± 0.26	93.61
LS+WRN†	TF	Equivariant	39.86 ± 1.59	62.85 ± 0.32	69.52 ± 0.23	92.57
LS+WRN	TF	Pixel-Wise	39.20 ± 0.80	54.14 ± 0.68	57.59 ± 0.48	92.97
S +WRN	TF	-	43.16 ± 0.78	61.66 ± 0.32	68.16 ± 0.27	92.27
LS+WRN†	Rand	Canonical	41.42 ± 0.65	59.84 ± 0.40	67.40 ± 0.28	93.36
LS+WRN†	Rand	Equivariant	40.84 ± 1.02	60.81 ± 0.40	68.62 ± 0.31	92.53
LS+WRN	Rand	Pixel-Wise	31.49 ± 0.63	45.85 ± 0.43	50.72 ± 0.28	91.86
S +WRN	Rand	-	32.08 ± 0.46	46.84 ± 0.21	52.76 ± 0.33	85.35
WRN-16-8	-	-	38.78 ± 0.72	62.97 ± 0.41	$\mathbf{71.37} \pm 0.31$	**96.84**
ResNet-50	-	-	33.17 ± 0.92	52.13 ± 0.74	64.42 ± 0.40	91.23

†: ours; TF: Tight-Frame; LS: Learnable Scattering; S: Scattering; Rand: Random
# params : 156k for S+LL; 155k for LS+LL; 22.6M for S+WRN; 22.6M for LS+WRN; 22.3M for WRN; and 22.5M for ResNet

Among the linear models, our TF-initialized learnable scattering networks (i.e., Morlet canonical and equivariant) significantly outperform all others in few sample settings. This demonstrates that learnable scattering networks obtain a more linearly separable representation than their fixed counterparts, perhaps by building greater dataset-specific intra-class invariance.

Figure 5.3 and shows the real part of the canonical wavelet filters before and after optimization on the entire training set. Among the WRN hybrid models, the TF-initialized canonical learnable scattering performs best. Canonical TF learnable still improves over TF fixed when paired with a WRN, indicating some loss of information in the fixed scattering representation is mitigated by data-driven tuning or optimization. Finally, our approach outperforms the fully trained ResNet-50 and outperforms the WRN-16-8 on 100 and 500 training samples, demonstrating the effectiveness of the scattering prior in the small data regime. However, the WRN-16-8 outperforms our model on 1,000 samples and 50,000 samples.

The training set of CIFAR-10 is augmented with pre-specified autoaugment in Table 5.2 to demonstrate the best possible results. To understand the effect of autoaugment, we replicate the same experiments except for not augmenting the training set with autoaugment. Table 5.3 reports the performance of the different architectures on CIFAR-10. We observe

Tableau 5.3. CIFAR-10 mean accuracy and std. error over 10 seeds with $J = 2$ and multiple training sample sizes. The table compares the effect of augmenting the training set with pre-specified autoaugment. When the scattering network is followed by a WRN, using autoaugment is necessary to obtain better performance.

Init.	Arch.	AA	100 samples	500 samples	1000 samples	All
TF	LS+LL†	Yes	37.84 ± 0.57	$\mathbf{52.68} \pm 0.31$	$\mathbf{57.43} \pm 0.17$	69.57 ± 0.1
TF	LS+LL†	No	$\mathbf{39.70} \pm 0.62$	50.74 ± 0.30	54.76 ± 0.22	$\mathbf{74.94} \pm 0.06$
TF	S +LL	Yes	36.01 ± 0.55	48.12 ± 0.25	53.25 ± 0.24	65.58 ± 0.04
TF	S +LL	No	$\mathbf{37.55} \pm 0.62$	$\mathbf{49.67} \pm 0.33$	$\mathbf{53.96} \pm 0.48$	$\mathbf{70.71} \pm 0.03$
Rand	LS+LL†	Yes	$\mathbf{34.81} \pm 0.60$	$\mathbf{49.6} \pm 0.39$	$\mathbf{55.72} \pm 0.39$	69.39 ± 0.41
Rand	LS+LL†	No	32.64 ± 0.38	42.88 ± 0.23	47.40 ± 0.32	$\mathbf{74.71} \pm 0.08$
Rand	S +LL	Yes	29.77 ± 0.47	$\mathbf{41.85} \pm 0.41$	$\mathbf{46.3} \pm 0.37$	57.72 ± 0.1
Rand	S +LL	No	$\mathbf{31.71} \pm 0.34$	40.57 ± 0.32	44.42 ± 0.51	$\mathbf{61.79} \pm 0.31$
TF	LS+WRN†	Yes	$\mathbf{43.60} \pm 0.87$	$\mathbf{63.13} \pm 0.29$	$\mathbf{70.14} \pm 0.26$	$\mathbf{93.61} \pm 0.12$
TF	LS+WRN†	No	34.95 ± 0.96	54.21 ± 0.39	62.17 ± 0.28	90.17 ± 0.34
TF	S +WRN	Yes	$\mathbf{43.16} \pm 0.78$	$\mathbf{61.66} \pm 0.32$	$\mathbf{68.16} \pm 0.27$	$\mathbf{92.27} \pm 0.05$
TF	S +WRN	No	35.15 ± 0.43	52.77 ± 0.35	60.72 ± 0.21	89.05 ± 0.38
Rand	LS+WRN†	Yes	$\mathbf{41.42} \pm 0.65$	$\mathbf{59.84} \pm 0.40$	$\mathbf{67.4} \pm 0.28$	$\mathbf{93.36} \pm 0.19$
Rand	LS+WRN†	No	31.08 ± 1.00	48.37 ± 0.76	55.41 ± 0.49	88.80 ± 0.47
Rand	S +WRN	Yes	$\mathbf{32.08} \pm 0.46$	$\mathbf{46.84} \pm 0.21$	$\mathbf{52.76} \pm 0.33$	$\mathbf{85.35} \pm 1.06$
Rand	S +WRN	No	27.73 ± 0.43	41.05 ± 0.32	47.19 ± 0.37	79.67 ± 0.59

†: ours TF: tight-frame LS: Learnable Scattering AA: Autoaugment
# params : 156k for S+LL; 155k for LS+LL; 11M for S+WRN; 22.6M LS+WRN; and 22.3M for WRN

that the scattering networks followed by WRN underperform when no autoaugment is used. The difference in performance between using autoaugment and not using it is smaller when the scattering network is followed with a linear layer. Surprisingly, the performance of the scattering networks followed with a linear layer trained on all data increased without autoaugment. It seems that in the case of a scattering network followed by a linear model, autoaugment is not as useful as with a deep model on top and can also be harmful in some cases.

5.2.2. COVIDx CRX-2

COVIDx CRX-2 is a two-class (positive and negative) dataset of $1024 \times 1024 \times 1$ chest X-Ray images of COVID-19 patients [**142**]. The train set contains 15,951 unbalanced images, while the test set contains 200 positive and 200 negative images. The spatial scale of the scattering transform is set to $J = 4$. We always train on a class-balanced subset of the training set in our experiments. We resize the images to 260×260 and train our network with random crops of 224×224 pixels. The only data augmentation we use is random horizontal flipping. All models were trained on 400 epochs using a max learning rate of 0.01. All hybrid models are trained with a mini-batch size of 128. Table 5.4 reports our evaluation on sample sizes of 100, 500, and 1K images. We use the same protocol as for CIFAR-10. Morlet canonical parameterization yields similar performance to the Morlet

Tableau 5.4. COVIDx CRX-2 mean accuracy & std. error with $J = 4$ over 10 seeds. TF-initialized learnable scattering network performs better than models that do not incorporate scattering priors.

Arch.	Init.	Parameterization	100 samples	500 samples	1000 samples
LS+LL†	TF	Canonical	82.30 ± 1.78	$\mathbf{88.50} \pm 0.71$	$\mathbf{89.90} \pm 0.40$
LS+LL†	TF	Equivariant	$\mathbf{83.06} \pm 1.53$	87.56 ± 0.94	89.15 ± 0.60
S +LL	TF	-	81.08 ± 1.88	87.20 ± 0.77	89.23 ± 0.69
LS+LL†	Rand	Canonical	76.85 ± 1.50	86.45 ± 0.95	89.70 ± 0.65
LS+LL†	Rand	Equivariant	76.73 ± 1.57	85.64 ± 1.38	87.98 ± 0.55
S +LL	Rand	-	76.08 ± 1.56	84.13 ± 0.91	86.80 ± 0.41
LS+WRN†	TF	Canonical	81.20 ± 1.73	90.50 ± 0.70	93.68 ± 0.35
LS+WRN†	TF	Equivariant	$\mathbf{81.86} \pm 2.07$	$\mathbf{91.56} \pm 0.52$	$\mathbf{93.97} \pm 0.34$
LS+WRN	TF	-	80.85 ± 1.85	89.05 ± 0.59	91.90 ± 0.54
LS+WRN†	Rand	Canonical	80.95 ± 1.54	88.08 ± 0.70	91.65 ± 0.55
LS+WRN†	Rand	Equivariant	80.12 ± 1.76	87.44 ± 1.17	91.40 ± 0.67
S +WRN	Rand	-	80.63 ± 1.73	86.68 ± 0.59	90.60 ± 0.50
WRN-16-8	-	-	80.50 ± 1.15	85.95 ± 2.04	88.82 ± 1.64
ResNet-50	-	-	74.04 ± 1.35	86.45 ± 0.51	90.86 ± 0.57

# params : 493K for LS/S+LL; 23.05M for LS/S+WRN; 22.3M for WRN;23.5M for ResNet
†: ours; TF: Tight-Frame; LS: Learnable Scattering; S: Scattering; Rand: Random

equivariant parameterization (i.e., most standard errors overlap), as also observed with CIFAR-10.

When the scattering networks are postpended with a linear layer, TF-initialized learnable (i.e., Morlet canonical and equivariant) performs better than TF fixed, showing the viability of our approach on real-world data. We observe that randomly initialized learnable yields lower performance than TF learnable on 100 and 500 samples. On 1K, it achieves similar performance, demonstrating that random initialization can achieve comparable performance to TF with enough data. WRN-16-8 performs worse than TF-initialized learnable followed with a linear layer. When combined with a CNN, TF-initialized learnable performs better than TF fixed and outperforms WRN-16-8 and ResNet-50.

5.2.3. KTH-TIPS2

KTH-TIPS2 contains 4,752 images from 11 material classes. The images captured the material at scales. Each class is divided into four *samples* (108 images each) of different scales. Using the standard protocol, we train the model on one *sample* ($11 * 108$ images), while the rest are used for testing [**133**]. In total, each training set contains 1,188 images. We resize the images to 200×200 and train our network with random crops of 128×128 pixels. The training data is augmented with random horizontal flips and random rotations. All scattering networks use a spatial scale of 4. We set the maximum learning rate of the scattering parameters to 0.1 while it is set to 0.001 for all other parameters. All hybrid models are trained with a mini-batch size of 128. The hybrid linear models are trained for 250 epochs, while the hybrid WRN models are trained for 150 epochs. We evaluate each

Tableau 5.5. KTH-TIPS2 mean accuracy & std. error with $J = 4$ over 16 seeds. The WRN-16-8 and ResNet-50 perform extremely poorly relative to hybrid models.

Arch.	Init.	Parameterization	1188 samples
LS+LL†	TF	Canonical	66.09 ± 1.05
LS+LL†	TF	Equivariant	$\mathbf{66.41} \pm 1.24$
S +LL	TF	-	66.17 ± 1.10
LS+LL†	Rand	Canonical	65.79 ± 0.85
LS+LL†	Rand	Equivariant	65.31 ± 1.42
S +LL	Rand	-	61.37 ± 0.82
LS+WRN†	TF	Canonical	$\mathbf{69.23} \pm 0.67$
LS+WRN†	TF	Equivariant	68.55 ± 0.80
S +WRN	TF	-	68.84 ± 0.71
LS+WRN†	Rand	Canonical	68.30 ± 0.47
LS+WRN†	Rand	Equivariant	67.50 ± 0.72
S +WRN	Rand	-	66.29 ± 0.36
WRN-16-8	-	-	51.24 ± 1.37
ResNet-50	-	-	44.95 ± 0.65

†: ours; TF: Tight-Frame; LS: Learnable Scattering; S: Scattering;
# params : 883K for LS/S+LL; 23.7M for LS/S+WRN; 22.3M for WRN; 23.5M for ResNet

model, training it with four different seeds on each *sample* of material, amounting to 16 total runs.

Table 5.5 reports the classification accuracies. With TF initialization and a linear layer, we observe that the performance is similar for the different architectures. The performance of randomly initialized learnable is also similar to TF. The fixed and randomly initialized model perform the worst, showing that even poorly initialized filters can effectively be optimized. Altogether, these results further corroborate our previous findings, notably that TF initialization acts as a good prior for scattering networks. Out of all the WRN hybrid models, the TF-initialized learnable model using canonical parameterization achieves the highest average accuracy. While WRN increases the performance compared to the linear layer, it also significantly increases the total number of parameters, therefore exhibiting a tradeoff between performance and model complexity. The WRN-16-8 and ResNet-50 perform extremely poorly relative to hybrid models, showing the effectiveness of the scattering priors for texture discrimination.

5.2.4. Cosine Loss

In the context of limited amount of labeled data, Lezama et al. [**17**] propose to replace the categorical cross-entropy loss with the cosine loss function to decrease overfitting in the small-sample classification settings. The cosine loss function, as opposed to the softmax function used with cross-entropy, does not push the logits of the true class to infinity as explained in [**138**]. We replicate the experiments with learnable scattering networks followed by a WRN on CIFAR-10, COVIDx-CRX2, and KTH-TIPS2. We use the same parameters

Tableau 5.6. CIFAR-10, COVIDx-CRX2 and KTH-TIPS2 mean accuracy and std. error using cosine loss function.

Init.	Arch.	Dataset	Loss	100 samples	500 samples	1000 samples	1188 samples
TF	LS+WRN	CIFAR-10	CE	**43.6** $\pm$ 0.87	**63.13** $\pm$ 0.29	**70.14** $\pm$ 0.26	-
TF	LS+WRN	CIFAR-10	Cosine	42.94 $\pm$ 0.77	61.42 $\pm$ 0.26	68.29 $\pm$ 0.18	-
TF	LS+WRN	COVIDx	CE	**81.20** $\pm$ 1.73	**90.50** $\pm$ 0.70	**93.68** $\pm$ 0.35	-
TF	LS+WRN	COVIDx	Cosine	80.03 $\pm$ 2.16	89.53 $\pm$ 0.89	92.75 $\pm$ 0.65	-
TF	LS+WRN	KTH-TIPS2	CE	-	-	-	69.23 $\pm$ 0.67
TF	LS+WRN	KTH-TIPS2	Cosine	-	-	-	**70.86** $\pm$ 0.67

TF: tight-frame LS: Learnable Scattering S: Scattering CE: Cross-Entropy Loss

except for using the cosine loss function [**17**] instead of cross-entropy. The cosine loss is described in Section 2.2.4. Wavelet filters are initialized using the tight frame construction. Table 5.6 demonstrates the average accuracy on the three datasets. For CIFAR-10 and COVIDx-CRX2, the performance is lower when models are trained using cosine loss. The same behavior is not observed when the models are trained on KTH-TIPS2. The performance increases slightly by using the cosine loss function. Thus, cosine loss can improve performance over small data regimes for some datasets.

5.2.5. Number of Filters per Spatial Scale

Next, we investigate the effect of modifying the number of filters (L) per spatial scale on CIFAR-10. So far, in all experiments, we have set the number of filters per scale at 8. For this experiment, we train a parametric scattering network followed by a linear layer where the wavelets are initialized using the tight frame construction and the canonical parameterization. The spatial scale is set to 2. We do not use autoaugment on the training set since, as shown in Table 5.3, autoaugment can be harmful when the scattering network is followed by a linear layer. Table 5.7 shows the accuracy of the entire training set for different values of

Tableau 5.7. CIFAR-10 accuracy of learnable scattering followed by a linear layer (LS + LL) and multiple numbers of filters per scale (L) trained on all the training set. The wavelet filters are initialized using the tight frame construction and the canonical parameterization. The spatial scale is set to 2. No autoaugment is used for this experiment. We observe that the performance increases when the number of filters per scale (L) also increases. Around 14 filters per spatial scale, the performance seems to have stopped increasing.

L	All Data
2	63.59
4	70.94
6	74.03
8	74.94
10	76.40
12	77.01
14	**77.36**
16	77.33

L. We observe that the performance increases when the number of filters per scale increases. Around 14-16 filters per spatial scale, the performance seems to have stopped increasing. Table 5.8 demonstrates the mean accuracy over different sizes of training samples where the number of filters per scale is set to 8 or 16. We also consider the fixed version of the scattering transform. Over all training sample sizes, we observe that the highest performance is obtained with learnable scattering using 16 filters per spatial scale. When the scattering network is fixed, performances are lower with 16 filters instead of 8. It seems that increasing the number of filters per scale is only beneficial in the learned version of the scattering network.

Tableau 5.8. CIFAR-10 mean accuracy and std. error over 10 seeds with multiple training sample sizes and different values of L. The wavelet filters are initialized using the tight frame construction. The spatial scale is set to 2. No autoaugment is used for this experiment. Over all training sample sizes, we observe that the highest performance is obtained with learnable scattering using 16 filters per spatial scale.

Arch.	Parameterization	L	100 samples	500 samples	1000 samples	All
LS+LL†	Canonical	8	$\mathbf{39.70} \pm 0.62$	50.74 ± 0.30	54.76 ± 0.22	74.94
LS+LL†	Canonical	16	$\mathbf{39.73} \pm 0.39$	$\mathbf{54.17} \pm 0.36$	$\mathbf{58.36} \pm 0.29$	**77.33**
S +LL	-	8	37.55 ± 0.62	49.67 ± 0.33	53.96 ± 0.48	70.71
S +LL	-	16	35.85 ± 0.48	48.2 ± 0.27	52.74 ± 0.25	70.64

TF: tight-frame LS: Learnable Scattering S: Scattering

5.3. Unsupervised Learning of Scattering Parameters

We have studied the adaptation of the wavelet parameters towards a supervised task. We now perform a preliminary investigation to determine if the scattering representation can be improved in a purely unsupervised manner. We consider the recently popularized SimCLR framework [**30**] described in Section 2.2.7, which encourages representations from two data augmentations of the same input to lie close together. We use the same framework except the encoder $f(\cdot)$ is not a ResNet-50, but a parametric scattering network. We learn the scattering network parameters on CIFAR-10 using this unsupervised objective function and subsequently evaluate the discriminativeness of the features under a standard linear evaluation experiment on the full CIFAR-10 dataset and in the small data regimes comparing them to the standard scattering transform (experiments illstrated in Figure 5.4). The results are shown in Table 5.9. We observe the unsupervised learning of filter parameters can improve the scattering representation under standard unsupervised learning evaluation protocols.

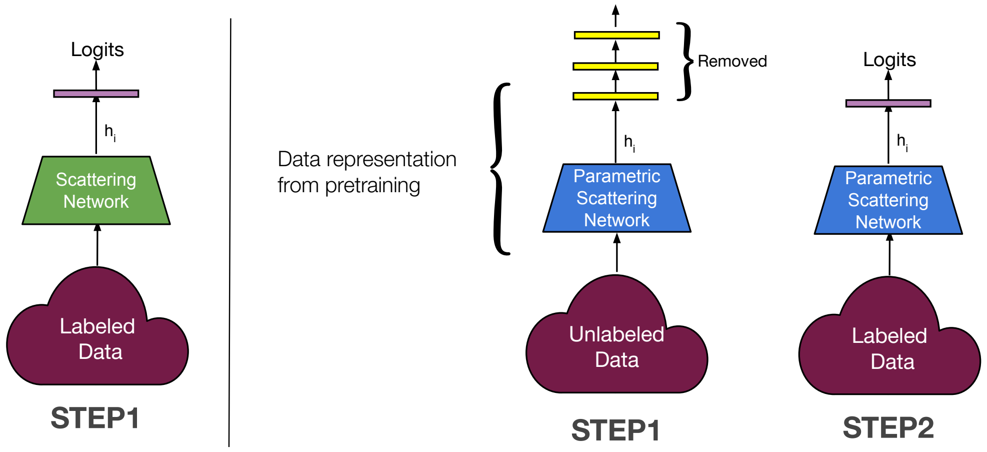

Fig. 5.4. Unsupervised learning of scattering networks visualization. (Left) Visualization of fixed scattering method where the representations are constructed from fixed scattering network and evaluated using linear layer colored in purple. (Right) Visualization of learned scattering method. In step 1, the parametric scattering network followed by a MLP (colored in yellow) is trained using the unsupervised objective function on the entire training set. In step 2, the representations are constructed from the trained parametric scattering network and evaluated using linear probing colored in purple. Images adapted from: [**30**].

Tableau 5.9. Scattering and learned unsupervised scattering features evaluated by training a linear classifier on CIFAR-10. We observe the unsupervised learned scattering improves the representation.

Method	100 samples	500 samples	1000 samples	All
Scattering (Fixed)	36.01 ± 0.55	48.12 ± 0.25	53.25 ± 0.24	65.58 ± 0.04
Unsupervised Learnt Scattering	$\mathbf{38.05} \pm 0.45$	$\mathbf{52.92} \pm 0.28$	$\mathbf{57.76} \pm 0.25$	$\mathbf{68.47} \pm 0.04$

Chapter 6

Conclusion and Future Work

In this work, we relax the scattering constructions to allow data-driven learning. We propose to adapt the filters' scales, orientations, and aspect ratios to produce problem-specific parametrizations of the scattering transforms. We call our approach the *Parametric Scattering Network*, in which the Morlet wavelet parameters are optimized for the task at hand in an end-to-end differentiable architecture. To assess the importance of the traditional construction of wavelet filter banks, we considered two different ways of initializing wavelets: tight-frame initialization and random initialization. We also considered two different Morlet wavelet parametrizations: canonical and equivariant.

The experiments demonstrate the competitive results of adapting a small number of Morlet wavelet filter parameters in the scattering network. We show that it is possible to backpropagate through the parametric scattering transform and we illustrate that filters learned by parametric scattering can be interpreted in relation to the specific task (e.g., becoming thinner in object recognition tasks that require sensitivity to edges). We also empirically demonstrate that our parametric scattering transform shares similar stability to deformations as the traditional scattering transform. We show that with limited labeled samples, there is not enough data and too many learnable parameters to learn the pixels of the wavelets effectively. Adding more constraints (i.e., constraining the wavelets to be Morlet) is beneficial in this setting. The results suggest that standard filterbank initialization is empirically beneficial as a starting point in the few sample regimes but can be improved upon by learning. In other words, our empirical results suggest that traditional filterbank constructions may not always be necessary for scattering transforms to extract effective representations.

We also observe that Morlet's canonical parametrization gives similar performance to Morlet's equivariant parametrization, showing that adding even more constraints does not degrade performance in the small data regime. We demonstrate, on CIFAR-10, that increasing the number of filters per spatial scale allows a performance gain. We also empirically demonstrate that increasing the number of filters per scale is only beneficial in the learned version of the scattering network. We show that in the case of a scattering network followed by a linear model trained on CIFAR-10, autoaugment is not as useful as with a deep model on top. Overall, we find that the parametric scattering network provides state-of-the-art results for classification in the low-data regime when combined with a linear layer and as well in a hybrid CNN. These results go towards bridging the gap between the handcrafted filter design in scattering transforms, which provides tractable properties and supports low-parameter models, and the fully (unparametrized) learned ones commonly used in CNN work, especially in computer vision and generally on 2D structured data.

There are some limitations to this study that could be addressed in future research. First, the current implementation is limited to two-dimensional data. The implementation could naturally be extended to one-dimensional and three-dimensional data in future work. Second, there are pre-trained models available for popular datasets, such as CIFAR-10. In the study, to compare performance with our approach, we considered a fully learned WRN-16-8 and ResNet-50, but we did not consider pre-trained models.

Our results may also lead to future work. We are working on a study affiliated with the Center Hospitalier de Montréal (CHUM), where the task is to classify chronic liver disease from B-mode ultrasound images. The number of images is limited, and the quality of the images can be influenced by different factors, such as the motion of the machinery and equipment. Global translations and rotations can be observed within the same class, depending on the positioning of the ultrasound collection device. Local scaling can be associated with variation between the relative size of organs. We obtain promising results on this task using the traditional scattering network. The next step is to evaluate our parametric scattering network on this dataset and see if we can increase the current performance.

Several avenues for improving our approach can also be explored, such as investigating the impact of downsampling on the representations learned by the parametric scattering network, as well as application to uncertainty estimation by leveraging the low parameter CNN in a Bayesian framework. We could vary the number of scales and create a scale constraint equivalent to the equivariant parameterization. In the equivariant parameterization, we could investigate whether it makes a difference to initialize the filters on the Cartesian axis or oblique to it. More exploration would be helpful to glean a deeper understanding of the parametric scattering network.

References

[1] Walid Al-Dhabyani, Mohammed Gomaa, Hussien Khaled et Aly Fahmy : Dataset of breast ultrasound images. *Data in brief*, 28:104863, 2020.

[2] Saad Albawi, Tareq Abed Mohammed et Saad Al-Zawi : Understanding of a convolutional neural network. *In 2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[3] Andrey Alekseev et Anatoly Bobe : Gabornet: Gabor filters with learnable parameters in deep convolutional neural network. *In 2019 International Conference on Engineering and Telecommunication (EnT)*, pages 1–4. IEEE, 2019.

[4] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal et Vijayan K Asari : A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.

[5] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal et Vijayan K Asari : A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.

[6] Joakim Andén, Vincent Lostanlen et Stéphane Mallat : Joint time-frequency scattering for audio classification. *In 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2015.

[7] Joakim Andén, Vincent Lostanlen et Stéphane Mallat : Joint time–frequency scattering. *IEEE Transactions on Signal Processing*, 67(14):3704–3718, 2019.

[8] Joakim Andén et Stéphane Mallat : Multiscale scattering for audio classification. *In ISMIR*, pages 657–662. Miami, FL, 2011.

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

[10] Mathieu Andreux, Tomás Angles, Georgios Exarchakis, Roberto Leonarduzzi, Gaspar Rochette, Louis Thiry, John Zarka, Stéphane Mallat, Joakim Andén, Eugene Belilovsky *et al.* : Kymatio: Scattering transforms in python. *J. Mach. Learn. Res.*, 21(60):1–6, 2020.

[11] Tomás Angles et Stéphane Mallat : Generative networks as inverse problems with scattering transforms. *arXiv preprint arXiv:1805.06621*, 2018.

[12] AWS : Model fit: Underfitting vs. overfitting. `https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html`.

[13] Idan Azuri et Daphna Weinshall : Generative latent implicit conditional optimization when learning from small sample. *In 2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8584–8591, 2021.

[14] Randall Balestriero, Romain Cosentino, Hervé Glotin et Richard Baraniuk : Spline filters for end-to-end deep learning. *In International conference on machine learning*, pages 364–373. PMLR, 2018.

[15] Randall Balestriero, Hervé Glotin et Richard G Baraniuk : Interpretable super-resolution via a learned time-series representation. arXiv:2006.07713, 2020.

[16] Danilo Bargen : Tikz: Diagram of a perceptron. `https://tex.stackexchange.com/questions/104334/tikz-diagram-of-a-perceptron`, Aug 1961.

[17] Bjorn Barz et Joachim Denzler : Deep learning on small datasets without pre-training using cosine loss. *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1371–1380, 2020.

[18] Nihar Bendre, Hugo Terashima Marín et Peyman Najafirad : Learning from few samples: A survey. arXiv:2007.15484, 2020.

[19] Matt Berseth : Isic 2017-skin lesion analysis towards melanoma detection. *arXiv preprint arXiv:1703.00523*, 2017.

[20] R Bharath et Pachamuthu Rajalakshmi : Deep scattering convolution network based features for ultrasonic fatty liver tissue characterization. *In 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1982–1985. IEEE, 2017.

[21] Patrick Bilic, Patrick Ferdinand Christ, Eugene Vorontsov, Grzegorz Chlebus, Hao Chen, Qi Dou, Chi-Wing Fu, Xiao Han, Pheng-Ann Heng, Jürgen Hesser *et al.* : The liver tumor segmentation benchmark (lits). *arXiv preprint arXiv:1901.04056*, 2019.

[22] Cloudera Blog : Learning with limited labeled data. `https://blog.fastforwardlabs.com/2019/03/20/learning-with-limited-labeled-data.html`, 2019.

[23] Lorenzo Brigato, Björn Barz, Luca Iocchi et Joachim Denzler : Tune it or don't use it: Benchmarking data-efficient image classification. *In 2nd Visual Inductive Priors for Data-Efficient Deep Learning Workshop*, 2021.

[24] Robert-Jan Bruintjes, Attila Lengyel, Marcos Baptista Rios, Osman Semih Kayhan et Jan van Gemert : Vipriors 1: Visual inductive priors for data-efficient deep learning challenges. arXiv:2103.03768, 2021.

[25] Xavier Brumwell, Paul Sinz, Kwang Jin Kim, Yue Qi et Matthew Hirn : Steerable wavelet scattering for 3d atomic systems with application to li-si energy prediction. *arXiv preprint arXiv:1812.02320*, 2018.

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

[27] Michał Byra, Grzegorz Styczynski, Cezary Szmigielski, Piotr Kalinowski, Łukasz Michałowski, Rafał Paluszkiewicz, Bogna Ziarkiewicz-Wróblewska, Krzysztof Zieniewicz, Piotr Sobieraj et Andrzej Nowicki : Transfer learning with deep convolutional neural network for liver steatosis assessment in ultrasound images. *International journal of computer assisted radiology and surgery*, 13(12):1895–1903, 2018.

[28] Renato Campanini, Dott Matteo Roffilli et Eugenio Nurrito : Scattering networks: Efficient 2d implementation and application to melanoma classification. *Universita di Bologna*, 2016.

[29] Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal et Diyi Yang : An empirical survey of data augmentation for limited data learning in nlp. *arXiv preprint arXiv:2106.07499*, 2021.

[30] Ting Chen, Simon Kornblith, Mohammad Norouzi et Geoffrey E. Hinton : A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[31] Xiuyuan Cheng, Xu Chen et Stéphane Mallat : Deep haar scattering networks. *Information and Inference: A Journal of the IMA*, 5(2):105–133, 2016.

[32] Vikash Chouhan, Sanjay Kumar Singh, Aditya Khamparia, Deepak Gupta, Prayag Tiwari, Catarina Moreira, Robertas Damaševičius et Victor Hugo C De Albuquerque : A novel transfer learning based approach for pneumonia detection in chest x-ray images. *Applied Sciences*, 10(2):559, 2020.

[33] Václav Chudácek, Ronen Talmon, Joakim Andén, Stéphane Mallat, Ronald R Coifman, Patrice Abry et Muriel Doret : Low dimensional manifold embedding for scattering coefficients of intrapartum fetale heart rate variability. *In 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6373–6376. IEEE, 2014.

[34] Romain Cosentino et Behnaam Aazhang : Learnable group transform for time-series. *In International Conference on Machine Learning*, pages 2164–2173. PMLR, 2020.

[35] Fergal Cotter et Nick Kingsbury : A learnable scatternet: Locally invariant convolutional layers. *In 2019 IEEE International Conference on Image Processing (ICIP)*, pages 350–354. IEEE, 2019.

[36] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan et Quoc V Le : Autoaugment: Learning augmentation policies from data. arXiv:1805.09501, 2018.

[37] Julianna Delua : "supervised vs. unsupervised learning: What's the difference?". v, March 2021.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li et Li Fei-Fei : Imagenet: A large-scale hierarchical image database. *In 2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[39] Terrance DeVries et Graham W Taylor : Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017.

[40] Tom Dietterich : Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[41] MATLAB Documentation : Machine learning in matlab. `https://www.mathworks.com/help/stats/machine-learning-in-matlab.html`.

[42] Carl Doersch, Abhinav Gupta et Alexei A Efros : Unsupervised visual representation learning by context prediction. *In Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[43] Michael Eickenberg, Georgios Exarchakis, Matthew Hirn et Stéphane Mallat : Solid harmonic wavelet scattering: Predicting quantum molecular energy from invariant descriptors of 3d electronic densities. *In Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6543–6552, 2017.

[44] Michael Eickenberg, Georgios Exarchakis, Matthew Hirn, Stéphane Mallat et Louis Thiry : Solid harmonic wavelet scattering for predictions of molecule properties. *The Journal of chemical physics*, 148(24):241732, 2018.

[45] Fernando Gama, Alejandro Ribeiro et Joan Bruna : Diffusion scattering transforms on graphs. *arXiv preprint arXiv:1806.08829*, 2018.

[46] Fernando Gama, Alejandro Ribeiro et Joan Bruna : Stability of graph scattering transforms. *Advances in Neural Information Processing Systems*, 32:8038–8048, 2019.

[47] Feng Gao, Guy Wolf et Matthew Hirn : Geometric scattering for graph data analysis. *In International Conference on Machine Learning*, pages 2122–2131. PMLR, 2019.

[48] Shanel Gauthier, Benjamin Thérien, Laurent Alsène-Racicot, Irina Rish, Eugene Belilovsky, Michael Eickenberg et Guy Wolf : Parametric scattering networks. *arXiv preprint arXiv:2107.09539*, 2021.

[49] Spyros Gidaris, Praveer Singh et Nikos Komodakis : Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[50] Machine Learning Glossary : Co-adaptation. `https://machinelearning.wtf/terms/co-adaptation/`, 2017.

[51] Ian Goodfellow, Yoshua Bengio et Aaron Courville : *Deep learning*. MIT press, 2016.

[52] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville et Yoshua Bengio : Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[53] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar *et al.* : Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[54] Karen Hao : Ai pioneer geoff hinton: "deep learning is going to be able to do everything". `https://www.technologyreview.com/2020/11/03/1011616/ai-godfather-geoffrey-hinton-deep-\learning-will-do-everything/`, Nov 2020.

[55] Hamed Hassanzadeh, Mahnoosh Kholghi, Anthony Nguyen et Kevin Chu : Clinical document classification using labeled and unlabeled data across hospitals. *In AMIA annual symposium proceedings*, volume 2018, page 545. American Medical Informatics Association, 2018.

[56] Douglas M Hawkins : The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[57] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie et Ross Girshick : Momentum contrast for unsupervised visual representation learning. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun : Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[59] Martin N Hebart et Guido Hesselmann : What visual information is processed in the human dorsal stream? *Journal of Neuroscience*, 32(24):8107–8109, 2012.

[60] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever et Ruslan R Salakhutdinov : Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[61] Matthew Hirn, Stéphane Mallat et Nicolas Poilvert : Wavelet scattering regression of quantum chemical energies. *Multiscale Modeling & Simulation*, 15(2):827–863, 2017.

[62] Matthew Hirn, Nicolas Poilvert et Stéphane Mallat : Quantum energy regression using scattering transforms. arXiv:1502.02077, 2015.

[63] Thomas Hofmann : Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1):177–196, 2001.

[64] Kurt Hornik : Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2): 251–257, 1991.

[65] Gao Huang, Zhuang Liu, Laurens Van Der Maaten et Kilian Q Weinberger : Densely connected convolutional networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[66] David H Hubel et Torsten N Wiesel : Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[67] Ernesto Iadanza, Rachele Fabbri, Džana Bašić-ČiČak, Amedeo Amedei et Jasminka Hasic Telalovic : Gut microbiota and artificial intelligence approaches: a scoping review. *Health and Technology*, pages 1–16, 2020.

[68] IBM : Underfitting. https://www.ibm.com/cloud/learn/underfitting, March 2021.

[69] Hiroshi Inoue : Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[70] Sergey Ioffe et Christian Szegedy : Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In International conference on machine learning*, pages 448–456. PMLR, 2015.

[71] A. Ivakhnenko, V. G. Lapa et R. Mcdonough : *Cybernetics and forecasting techniques*, volume 8. American Elsevier Publishing Company, 1967.

[72] Brownlee Jason : Overfitting and underfitting with machine learning algorithms. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-\algorithms/, March 2019.

[73] Kenji Kawaguchi, Leslie Pack Kaelbling et Yoshua Bengio : Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[74] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan *et al.* : Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[75] Vitaliy Kinakh, Olga Taran et Svyatoslav Voloshynovskiy : Scatsimclr: self-supervised contrastive learning with pretext task regularization for small-scale datasets. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1098–1106, 2021.

[76] Diederik P Kingma et Max Welling : Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[77] Soheil Kolouri, Yang Zou et Gustavo K Rohde : Sliced wasserstein kernels for probability distributions. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5258–5267, 2016.

[78] Tomohiko Konno et Michiaki Iwazume : Icing on the cake: An easy and quick post-learnig method you can try after deep learning. *arXiv preprint arXiv:1807.06540*, 2018.

[79] Sotiris B Kotsiantis, I Zaharakis, P Pintelas *et al.* : Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1): 3–24, 2007.

[80] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton : Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[81] Harold W Kuhn : The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[82] Dorian Lazar : Perceptron: Explanation, implementation and a visual example. https://towardsdatascience.com/perceptron-explanation-implementation-and-a-visual-example-\3c8e76b4e2d1, May 2021.

[83] Yann LeCun, Yoshua Bengio et Geoffrey Hinton : Deep learning. *nature*, 521(7553):436–444, 2015.

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

[85] Yann LeCun, Léon Bottou, Yoshua Bengio et Patrick Haffner : Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[86] Gregory Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt et Aaron O'Leary : Pywavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237, 2019.

[87] AS Lewis et G Knowles : Vlsi architecture for 2d daubechies wavelet transform without multipliers. *Electronics letters*, 27(2):171–173, 1991.

[88] José Lezama, Qiang Qiu, Pablo Musé et Guillermo Sapiro : Ole: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8109–8118, 2018.

[89] Shan Sung Liew, Mohamed Khalil Hani, Syafeeza Ahmad Radzi et Rabia Bakhteri : Gender classification: a convolutional neural network approach. *Turkish Journal of Electrical Engineering & Computer Sciences*, 24(3):1248–1264, 2016.

[90] David G Lowe : Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[91] S Mallat : Communications on pure and applied mathematics. *Wiley Periodicals, Inc*, 65:1331–1398, 2012.

[92] Stéphane Mallat : *A wavelet tour of signal processing*. Elsevier, 1999.

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

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

[95] Warren S McCulloch et Walter Pitts : A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[96] Jason D McEwen, Christopher GR Wallis et Augustine N Mavor-Parker : Scattering networks on the sphere for scalable and rotationally equivariant spherical cnns. *arXiv preprint arXiv:2102.02828*, 2021.

[97] Dan Meng, Libo Zhang, Guitao Cao, Wenming Cao, Guixu Zhang et Bing Hu : Liver fibrosis classification based on transfer learning and fcnet for ultrasound images. *Ieee Access*, 5:5804–5810, 2017.

[98] Oliver Meynberg et Georg Kuschk : Airborne crowd density estimation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2:49–54, 2013.

[99] Shervin Minaee, Rahele Kafieh, Milan Sonka, Shakib Yazdani et Ghazaleh Jamalipour Soufi : Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning. *Medical image analysis*, 65:101794, 2020.

[100] Matthias Minderer, Olivier Bachem, Neil Houlsby et Michael Tschannen : Automatic shortcut removal for self-supervised representation learning. *arXiv preprint arXiv:2002.08822*, 2020.

[101] Michel Misiti, Yves Misiti, Georges Oppenheim et Jean-Michel Poggi : Wavelet toolbox. *The MathWorks Inc., Natick, MA*, 15:21, 1996.

[102] Blake Murdoch : Privacy and artificial intelligence: challenges for protecting health information in a new era. *BMC Medical Ethics*, 22(1):1–5, 2021.

[103] Vinod NAIR et Geoffrey E HINTON : Rectified linear units improve restricted boltzmann machines. *In Icml*, 2010.

[104] Mehdi NOROOZI et Paolo FAVARO : Unsupervised learning of visual representations by solving jigsaw puzzles. *In European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[105] E. OYALLON et S. MALLAT : Deep roto-translation scattering for object classification. *In Proc. of CVPR*, pages 2865–2873, 2015.

[106] Edouard OYALLON, Eugene BELILOVSKY et Sergey ZAGORUYKO : Scaling the scattering transform: Deep hybrid networks. *In Proceedings of the IEEE international conference on computer vision*, pages 5618–5627, 2017.

[107] Edouard OYALLON, Stéphane MALLAT et Laurent SIFRE : Generic deep networks with wavelet scattering. arXiv:1312.5940, 2013.

[108] Edouard OYALLON, Sergey ZAGORUYKO, Gabriel HUANG, Nikos KOMODAKIS, Simon LACOSTE-JULIEN, Matthew BLASCHKO et Eugene BELILOVSKY : Scattering networks for hybrid representation learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2208–2221, 2018.

[109] Chao PAN, Siheng CHEN et Antonio ORTEGA : Spatio-temporal graph scattering transform. *arXiv preprint arXiv:2012.03363*, 2020.

[110] Sinno Jialin PAN et Qiang YANG : A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[111] Deepak PATHAK, Philipp KRAHENBUHL, Jeff DONAHUE, Trevor DARRELL et Alexei A EFROS : Context encoders: Feature learning by inpainting. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[112] Michael PERLMUTTER, Feng GAO, Guy WOLF et Matthew HIRN : Geometric wavelet scattering networks on compact riemannian manifolds. *In Mathematical and Scientific Machine Learning*, pages 570–604. PMLR, 2020.

[113] Michael PERLMUTTER, Guy WOLF et Matthew HIRN : Geometric scattering on manifolds. *arXiv preprint arXiv:1812.06968*, 2018.

[114] Maithra RAGHU, Chiyuan ZHANG, Jon KLEINBERG et Samy BENGIO : Transfusion: Understanding transfer learning for medical imaging. *arXiv preprint arXiv:1902.07208*, 2019.

[115] Hariharan RAVISHANKAR, Prasad SUDHAKAR, Rahul VENKATARAMANI, Sheshadri THIRUVENKADAM, Pavan ANNANGI, Narayanan BABU et Vivek VAIDYA : Understanding the mechanisms of deep transfer learning for medical images. *In Deep learning and data labeling for medical applications*, pages 188–196. Springer, 2016.

[116] RCASSANI : Rcassani/mlp-example: Code for a simple mlp (multi-layer perceptron). `https://github.com/rcassani/mlp-example`.

[117] Giulia RIGHI et Jean VETTEL : *Dorsal Visual Pathway*, pages 887–888. Springer New York, New York, NY, 2011.

[118] Frank ROSENBLATT : The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[119] Frank ROSENBLATT : Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

[120] Harold RYAN : Ricker, ormsby; klander, bntterwo-a choice of wavelets, 1994.

[121] Abhinav SAGAR : "5 techniques to prevent overfitting in neural networks". `https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html`, Nov 2019.

[122] Claude SAMMUT et Geoffrey I WEBB : *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[123] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas et Aleksander Mądry : How does batch normalization help optimization? *In Proceedings of the 32nd international conference on neural information processing systems*, pages 2488–2498, 2018.

[124] Dominik Scherer, Andreas Müller et Sven Behnke : Evaluation of pooling operations in convolutional architectures for object recognition. *In International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[125] Léonard Seydoux, Randall Balestriero, Piero Poli, Maarten De Hoop, Michel Campillo et Richard Baraniuk : Clustering earthquake signals and background noises in continuous seismic data with unsupervised deep learning. *Nature communications*, 11(1):1–12, 2020.

[126] Ling Shao, Fan Zhu et Xuelong Li : Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, 26(5):1019–1034, 2014.

[127] Connor Shorten et Taghi M Khoshgoftaar : A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[128] Laurent Sifre et Stéphane Mallat : Rotation, scaling and deformation invariant scattering for texture discrimination. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1233–1240, 2013.

[129] Laurent Sifre et Stéphane Mallat : Rigid-motion scattering for texture classification. arXiv:1403.1687, 2014.

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

[131] Paul Sinz, Michael W Swift, Xavier Brumwell, Jialin Liu, Kwang Jin Kim, Yue Qi et Matthew Hirn : Wavelet scattering networks for atomistic systems with extrapolation of material properties. *The Journal of Chemical Physics*, 153(8):084109, 2020.

[132] Leslie N Smith et Nicholay Topin : Super-convergence: Very fast training of neural networks using large learning rates. *In Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612, 2019.

[133] Yang Song, Fan Zhang, Qing Li, Heng Huang, Lauren J O'Donnell et Weidong Cai : Locally-transferred fisher vectors for texture classification. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 4912–4920, 2017.

[134] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever et Ruslan Salakhutdinov : Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[135] Joes Staal, Michael D Abràmoff, Meindert Niemeijer, Max A Viergever et Bram Van Ginneken : Ridge-based vessel segmentation in color images of the retina. *IEEE transactions on medical imaging*, 23(4):501–509, 2004.

[136] Jiawei Su, Danilo Vasconcellos Vargas et Kouichi Sakurai : One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[137] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke et Andrew Rabinovich : Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[138] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens et Zbigniew Wojna : Rethinking the inception architecture for computer vision. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[139] Matej ULICNY, Vladimir A. KRYLOV et Rozenn DAHYOT : Harmonic networks with limited training samples. *In 2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[140] Stefan Van der WALT, Johannes L SCHÖNBERGER, Juan NUNEZ-IGLESIAS, François BOULOGNE, Joshua D WARNER, Neil YAGER, Emmanuelle GOUILLART et Tony YU : scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[141] Vladimir VAPNIK : Principles of risk minimization for learning theory. *In Advances in neural information processing systems*, pages 831–838, 1992.

[142] Linda WANG, Zhong Qiu LIN et Alexander WONG : Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):1–12, 2020.

[143] Guy WOLF, Stéphane MALLAT et Shihab SHAMMA : Audio source separation with time-frequency velocities. *In 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2014.

[144] Guy WOLF, Stephane MALLAT et Shihab SHAMMA : Rigid motion model for audio source separation. *IEEE Transactions on Signal Processing*, 64(7):1822–1831, 2015.

[145] Jiasong WU, Xiang QIU, Jing ZHANG, Fuzhi WU, Youyong KONG, Guanyu YANG, Lotfi SENHADJI et Huazhong SHU : Fractional wavelet-based generative scattering networks. *Frontiers in neurorobotics*, 15, 2021.

[146] Christopher J.C. Burges YANN LECUN, Corinna Cortes : Mnist dataset sample. `http://yann.lecun.com/exdb/mnist/`, 1999.

[147] Jason YOSINSKI, Jeff CLUNE, Yoshua BENGIO et Hod LIPSON : How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

[148] Yuhai YU, Hongfei LIN, Jiana MENG, Xiaocong WEI, Hai GUO et Zhehuan ZHAO : Deep transfer learning for modality classification of medical images. *Information*, 8(3):91, 2017.

[149] Sergey ZAGORUYKO et Nikos KOMODAKIS : Wide residual networks. *In Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[150] Michał ZAJAC, Konrad ZOŁNA, Negar ROSTAMZADEH et Pedro O PINHEIRO : Adversarial framing for image and video classification. *In Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10077–10078, 2019.

[151] Amir R ZAMIR, Alexander SAX, William SHEN, Leonidas J GUIBAS, Jitendra MALIK et Silvio SAVARESE : Taskonomy: Disentangling task transfer learning. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

[152] Chiyuan ZHANG, Samy BENGIO, Moritz HARDT, Benjamin RECHT et Oriol VINYALS : Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[153] Richard ZHANG, Phillip ISOLA et Alexei A EFROS : Colorful image colorization. *In European conference on computer vision*, pages 649–666. Springer, 2016.

[154] Dongmian ZOU et Gilad LERMAN : Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis*, 49(3):1046–1074, 2020.

Appendix A

Perceptron Learning Algorithm

The perceptron learning algorithm is described in Algorithm A.1.

Algorithm A.1. Perceptron Learning Algorithm

```
Inputs: training samples $(\mathbf{x}^0, \mathbf{x}^1, ..., \mathbf{x}^N)$ where all $\mathbf{x}^i \in \mathbb{R}^n$,
the labels associated with the samples $(y^0, y^1, ..., y^N)$ where all $y^i \in \{-1,1\}$
and $r$ the learning rate.
$\mathbf{w} := 0 \in \mathbb{R}^n$     Initialize weight vector with 0
$b := 0$            Initialize bias with 0
while not converged do:
    for $i \in \{1,2,...N\}$ do:     for each sample
        $\hat{y} \leftarrow \phi(\mathbf{w}^T\mathbf{x}^i)$          predict
        if $\hat{y} \neq y^i$ then:          if error
            $\mathbf{w} \leftarrow \mathbf{w} + r(y^i\mathbf{x}^i)$:
return $\mathbf{w}$,$b$
```

Appendix B

Dataset Specific Parameterizations

The following figures show the configuration obtained by our minimal bipartite matching algorithm. Each wavelet titles share a common naming scheme. The first letter of the title is either F (fixed filters) or O (optimized filters). The next character is always a number and corresponds to the ID of the match. For all Oixxxxxxx titles, there will be a corresponding Fixxxxxxxx title. These filters are matched to each other. The next character is always D (distance). It is superseded by a numerical value, the Morlet wavelet distance between the filter and its match. The next character is the Gaussian window scale σ, followed by a number corresponding to the magnitude of the distances between the σ parameters of both filters. The next character is the aspect ratio γ, followed by a number corresponding to the magnitude of the distances between the γ parameters of both filters. The next character is the frequency scale ξ, followed by a number corresponding to the magnitude of the distances between the ξ parameters of both filters.

B.1. COVIDX-CRX2

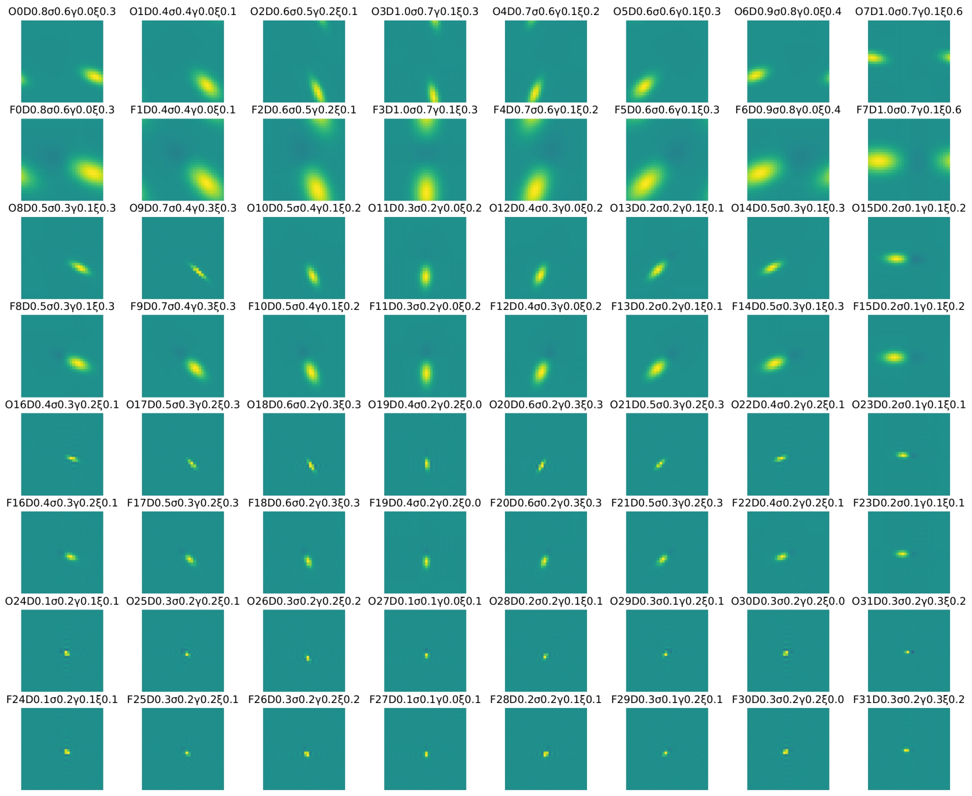

Fig. B.1. Filters trained on 1188 samples of COVIDx-CRX2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the "closest" (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row 2.

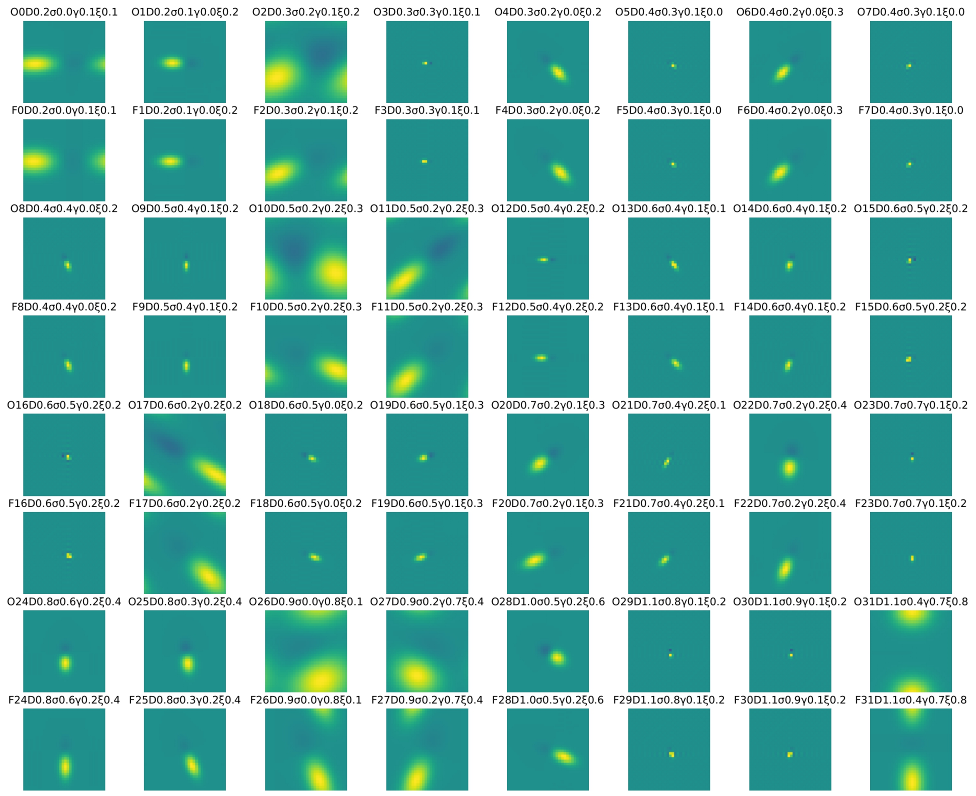

Fig. B.2. Filters trained on 1188 samples of COVIDx-CRX2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the "closest" (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row 2. The filters are displayed in increasing order of their distances. The top left corner corresponds to the filters that changed the least from their initialization, while the filters in the bottom right corner changed the most.

B.2. KTH-TIPS2

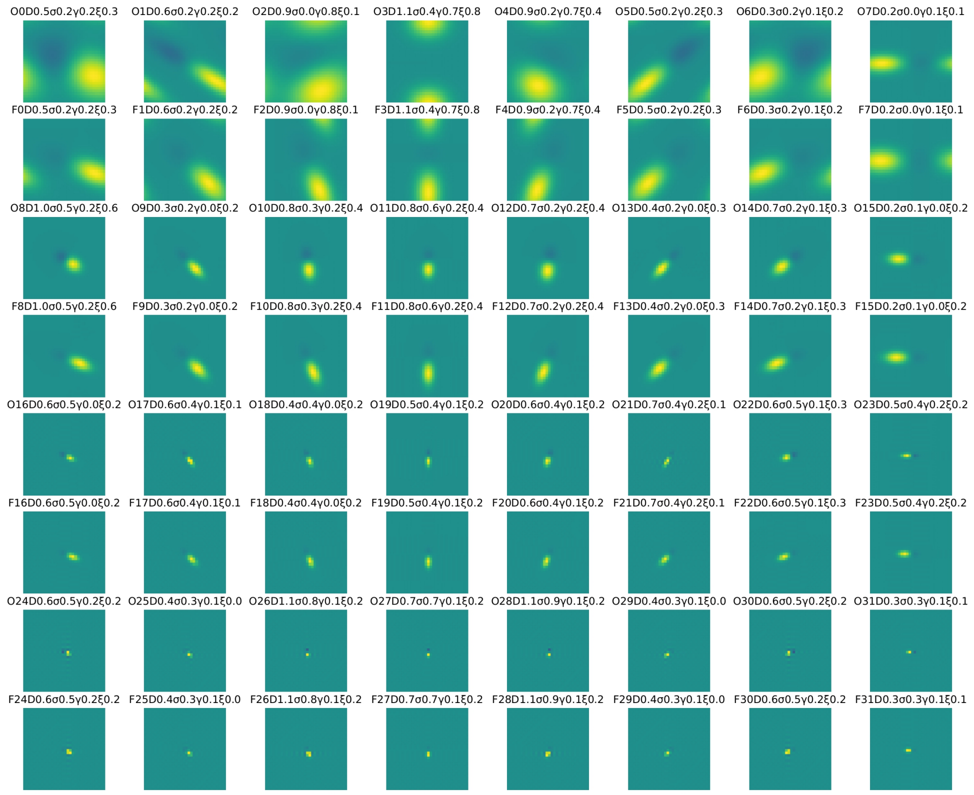

Fig. B.3. Filters trained on 1188 samples of KTH-TIPS2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the "closest" (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row 2.

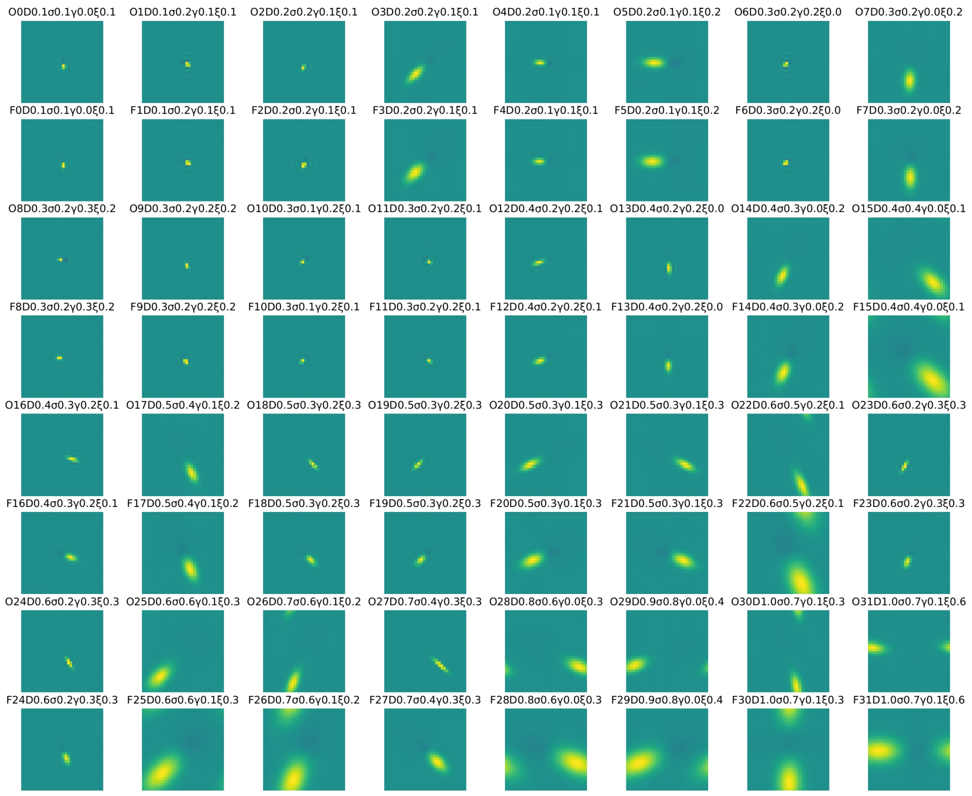

Fig. B.4. Filters trained on 1188 samples of KTH-TIPS2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the "closest" (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row 2. The filters are displayed in increasing order of their distances. The top left corner corresponds to the filters that changed the least from their initialization, while the filters in the bottom right corner changed the most.

B.3. CIFAR-10

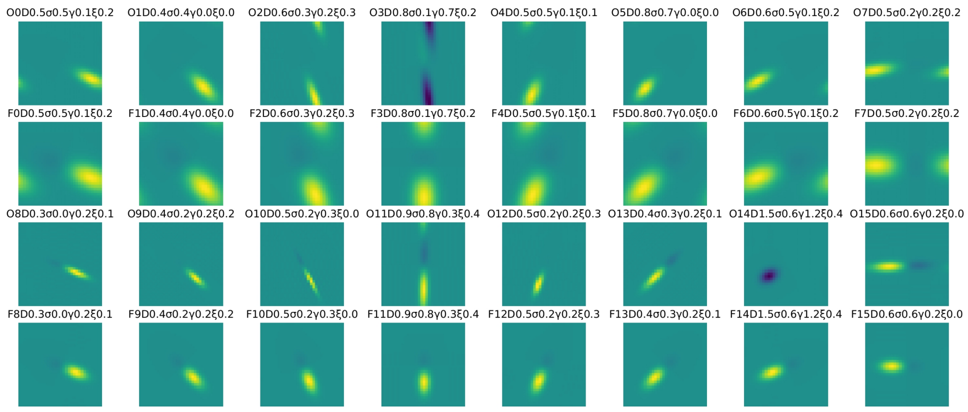

Fig. B.5. Filters trained on 1190 samples of CIFAR-10 for 500 epochs, the first and third, rows correspond to filters optimized from a tight-frame, while the second and fourth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the "closest" (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row 2.