SCATTERING CONVOLUTIONAL HIDDEN MARKOV TREES

IMAGE REPRESENTATION AND SCATTERING TRANSFORM MODELLING

Jean-Baptiste REGLI

2014-2015

RESEARCH REPORT

Academic supervisor : James Nelson
Sponsor : Dstl/UCL Impact studentship

UCL
Department of Statistical Science
London

Contents :

List of figures :

1 Introduction :

Nowadays statistical signal processing and machine learning methods are applied to a wide range of problems from prediction to optimal control. While some of those applications involves low-dimensional data, full branches of those fields are dedicated to studying high-dimensional inputs. Such problems have been made very popular lately firstly because, over the couple last decades data got cheaper to collect. This is mainly due to the progresses made on digital cameras, microphones and other sensors for signal acquisition. Second the cost of data storage massively decrease during that time. And finally the computational power available today allows to try more computationally intensive methods on always bigger datasets. Hence methods to leverage those data have been developed across the range of applications in machine learning and statistical signal processing.

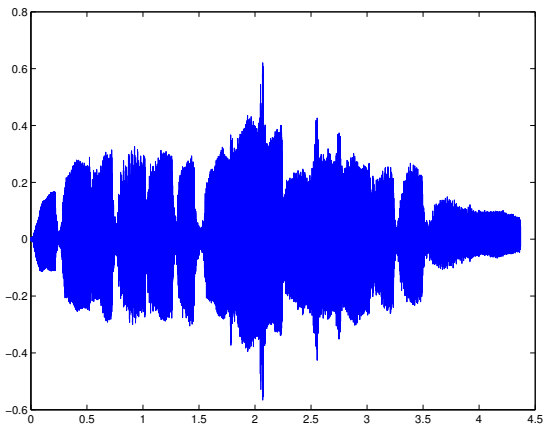
The standard approach when working with high dimensional signals can be expressed as a two step procedure. First the data are projected in a feature space where the task at hand (classification, regression...) is simplified. Then prediction is done using a simple predictor in this new representational space. Predictors such as logistic regression or linear Support vector Machine are common choices. The projection can either be hand-build —e.g. Fourier transform, wavelet transform— or learned. In the last decade methods for learning the projection have drastically improved under the impulsion of the so called deep learning methods. Using deep neural network (sometime enriched by convolutional architecture), systems have been able to learn very effective representations for a given dataset and a given task. Such method have achieved state of the art on many standard problems as well as real world applications. And this despite using a very simple prediction mechanism —on top of a very clever projection method.

This document proposes a method combining a new deterministic analytically tractable transformation able to compete with deep architecture in terms of representational power to a probabilistic graphical model in order to create a powerful probabilistic tool to handle high dimensional prediction problems. In a similar fashion to the work done by Crouse on wavelet trees [Crouse et al., 1998], we propose to describe Mallat's scattering convolutional scattering transform [Bruna and Mallat, 2010] using a hidden Markov tree. Doing so we develop a new framework to model high-dimensional inputs. As opposed to the commonly used simple classification method, our model can tackle prediction problems as well as any inference task.

1.1 Signal representation :

Consider the problem of learning the labelling function f, say, given N sampled training values $\{\mathbf{x}_i, y_i = f(\mathbf{x}_i)\}_{i \leq N}$ where for all $i \in [\![1, N]\!]$, $\mathbf{x}_i = \{x[1] \dots x[d]\}$ with $d \sim 10^6$, $x[.] \in \mathbb{R}$ and $y_i \in \mathbb{N}$. Such signals are, for example, speech waveforms, digital photographies but also electrocardiograms, sonar imagery and many others.

A naive solution to this problem would be to infer the class of a new realization $\mathbf{x}^{new}$ by



(a) Sound of a flute

(b) Picture of a mandrill

FIGURE 1.1 – High dimensional signals.

looking at its neighbours in a similar fashion to K-Nearest Neighbours (KNN), for example. This approach is sound for low-dimensional problems [Cover and Hart, 1967]. However it shows limitations in high dimensional cases [Beyer et al., 1999] because the number of samples required to find a neighbour of a new realization $\mathbf{x}^{new}$ grows exponentially with the number of dimensions. This issue is known in the statistical learning community as the curse of dimensionality.

One can make the assumption that the signal $\mathbf{x}$ belongs to a manifold Ω of $\mathbb{R}^d$ and this leads to two cases. The subset Ω can be low dimensional and the curse of dimensionality is avoided once the manifold has been isolated. The task at hand is thus a manifold learning problem [Lin and Zha, 2008] [Zhang et al., 2012] or a sparse dictionary representation problem [Kreutz-Delgado et al., 2003]. However for complex signals the manifold Ω is also high dimensional. And in this case one has to reduce the volume of the signal space without loosing crucial information required to characterize signals. Hence one has to reduce the volume of Ω according to the properties of the classification function f.

A mapping Φ is sought which represents the signal in a new space such that the classification task is simpler. The space should not only capture the main information and discriminatory content in the data but it should also remain stable with respect to appropriate transformations and deformations. Before providing a formal mathematical description of this mapping, it is instructive to consider the following intuitive example.

1.1.1 Intuition of a "good" signal representation :

Properties of a "good" representation for classification can be intuited by looking how humans handle the classification/labelling of visual stimulus and what are the elements ensuring good generalization capacities. Following this approach, one can intuit the following properties :

— The projection has to be informative enough to permit classification. This means ensuring that Φ preserves separability between the different classes.

— The mapping also has to be invariant to translations. Indeed to a human eye there is no difference in the information carried by a signal if it is shifted. This means that the transformation Φ has to provide close, if not equal, outputs for shifted versions of the same signal.

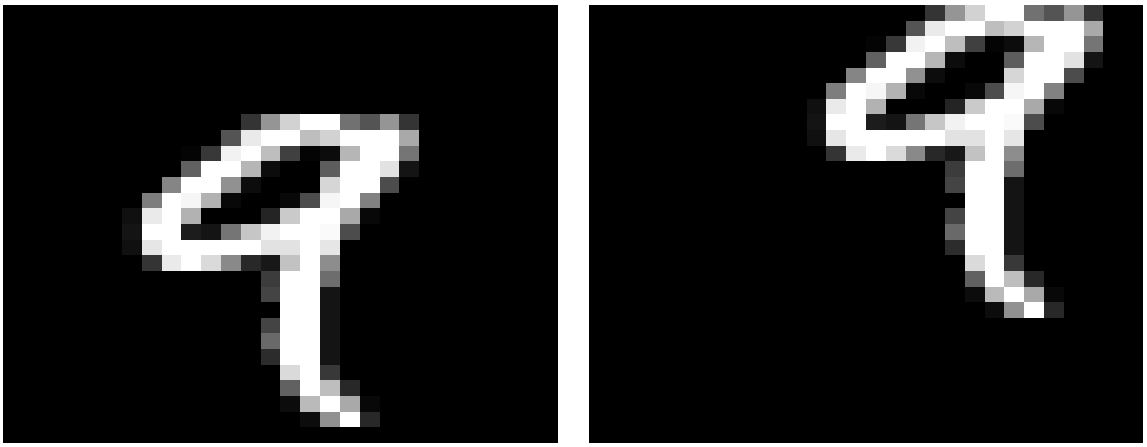


FIGURE 1.2 – Translation invariance : A human can easily tell that those two images are from the same class.

— To some extend the mapping has to be stable under deformations. Once again to a human eye, it is still possible to recognize a signal if it has undergone —small— deformations. Yet if the deformations are too important the information content of the signal can be lost (see Figure 1.3). This means that to a certain degree the projections of morphed realizations of the same signal should be mapped to a same region of the representational space. However one need to define a limit to this invariance to ensure that the representation created is still informative enough.

— Again to a certain degree the projection has to be invariant to rotations. Only local invariant to rotation is wanted because excessive rotation applied to the original signal can be destructive for the information carried (see Figure 1.4). Solutions based on the method described in this document exist [Sifre and Mallat, 2013] [Oyallon and Mallat, 2014] but they are still work in progress and thus they will not be addressed nor used in this review.

1.1.2 Formalization of a "good" signal representation :

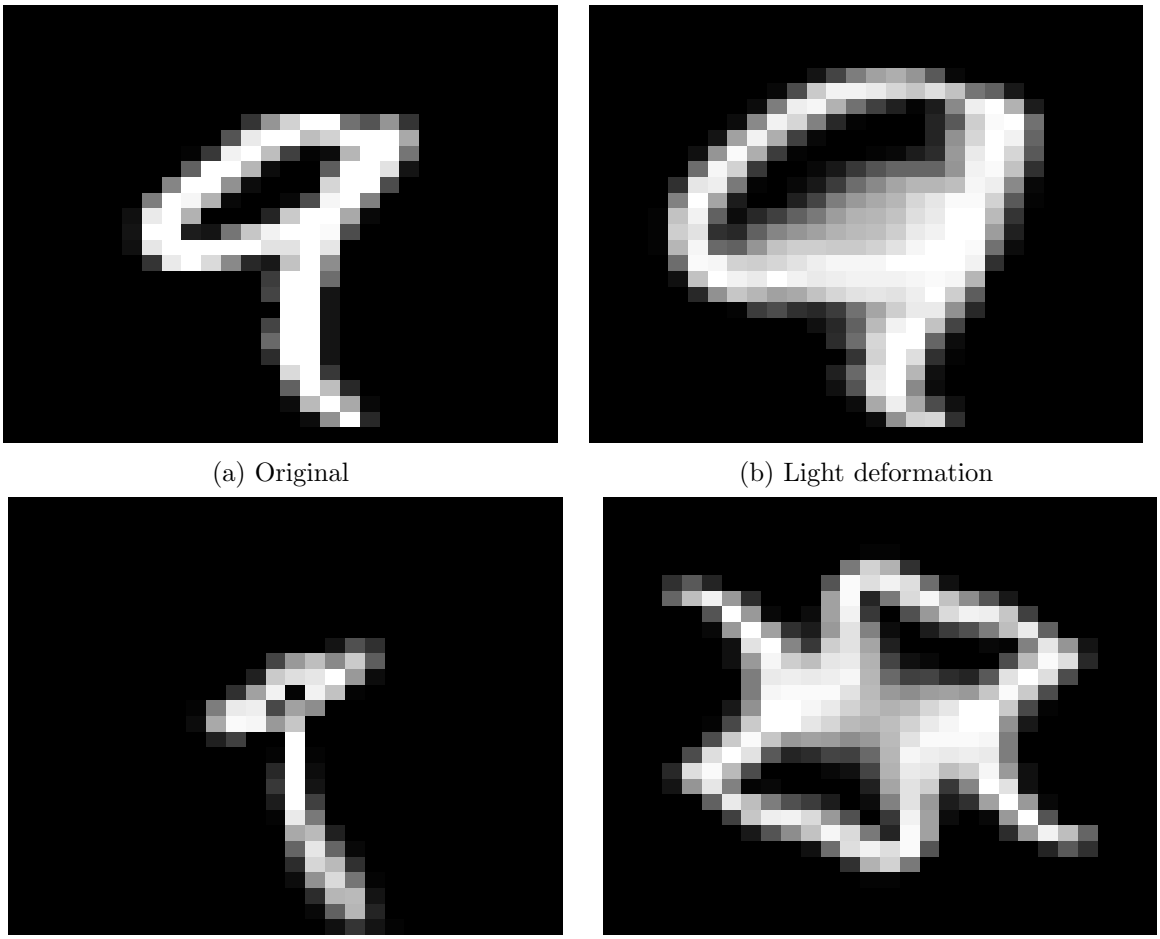
Throughout, attention is restricted to signal represented by square-integrable d-dimensional functions over the real, namely $\mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d)$. For more accuracy, the classification function is split in two stages. f now denotes the soft classification function, i.e. $f(\mathbf{x}) \in \mathbb{R}^K$ where K is the dimension of the mapping space, and represents the distance to the separating surface. h is the labelling function such that $y = h \circ f(\mathbf{x})$ is now the label associated to a signal $\mathbf{x}$. To be informative enough, a representation must preserve separability between elements of different classes. This is encapsulated by the following definition.

Definition 1.1.1. ***(Separability preservation)***
A representation Φ preserves separability if all elements of two different classes are distant of at least a margin C in the representation space,

$$\forall x, x' \in \mathbb{R}^d \quad \exists C \in \mathbb{R}^K \quad s.t. \quad h \circ f(x) \neq h \circ f(x') \quad \Rightarrow \quad \left\|\Phi(x) - \Phi(x')\right\| \geq C^{-1}$$

where K is the dimension of the mapping space.

Translations in the input space should not affect the representation. In this document let $L_{(.)}$ denote the translation operator for the function in $\mathcal{L}^2(\mathbb{R}^d)$, i.e. for $\mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d)$ and $u, c \in \mathbb{R}^d \times \mathbb{R}^d \quad L_c\mathbf{x}(u) = \mathbf{x}(u - c)$. A mapping Φ is translation invariant —respectively canonical translation invariant —if it maps a translated signal to the same point as its original version.



(a) Original (b) Light deformation

(c) Important deformation (d) Loss of information

FIGURE 1.3 – Stability to deformations : A human can easily tell that (a) and (b) are from the same class. (c) can still be recognized even though it is slightly more challenging. For (d) the informative content is lost.

Definition 1.1.2. ***(Translation invariance)***
An operator $\Phi : \mathcal{L}^2(\mathbb{R}^d) \to \mathcal{H}$ *where* $\mathcal{H}$ *is a Hilbert space is translation invariant if :*

$$\forall c \in \mathbb{R}^d \quad \textit{and} \quad \forall \mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d) \quad \Phi(L_c \mathbf{x}) = \Phi(\mathbf{x}).$$

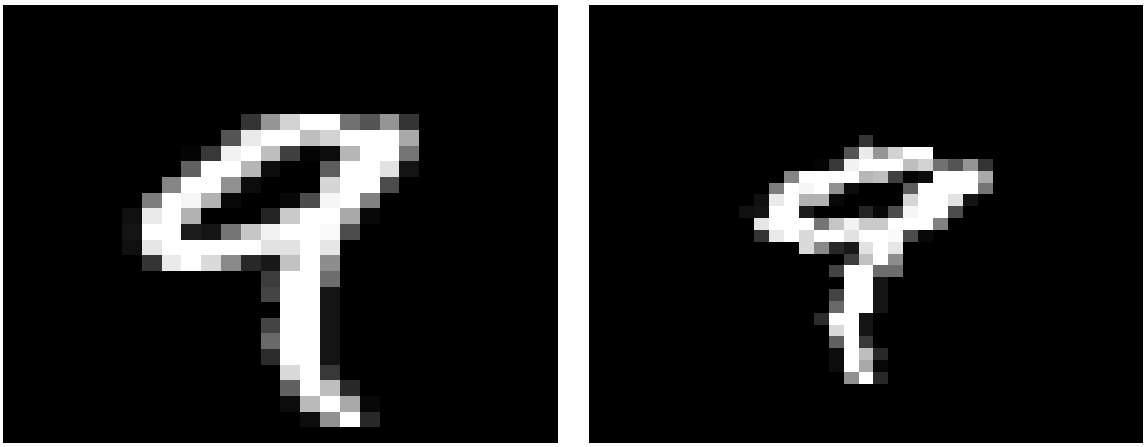
Definition 1.1.3. ***(Canonical translation invariant)***
An operator $\Phi : \mathcal{L}^2(\mathbb{R}^d) \to \mathcal{H}$ *where* $\mathcal{H}$ *is a Hilbert space is canonical translation invariant if :*

$$\forall \mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d) \quad \Phi(L_a \mathbf{x}) = \Phi(\mathbf{x}) \quad \textit{where } a \in \mathbb{R}^d \textit{ is function of } \mathbf{x}.$$

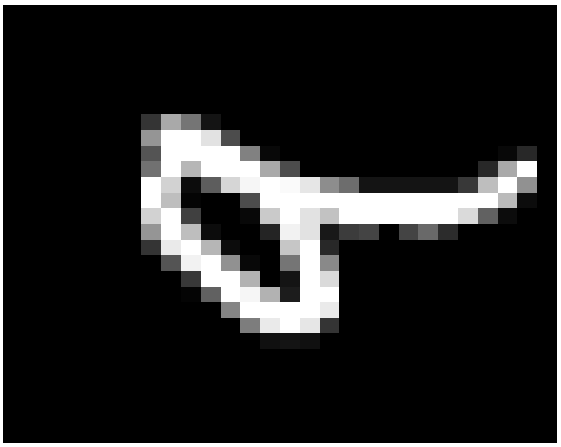
For the common representation operators, instabilities to deformations are known to appear —especially at high frequencies. To prevent this, one would like the representation to be non-expansive.

Definition 1.1.4. ***(Non-expensive representation)***
A representation Φ *is non-expensive if,*

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{L}^2(\mathbb{R}^d) \quad \left\| \Phi(\mathbf{x}) - \Phi(\mathbf{x}') \right\| \leq \left\| \mathbf{x} - \mathbf{x}' \right\|. \tag{1.1}$$



(a) Original (b) Small rotation



(c) Important rotation

FIGURE 1.4 – Rotation invariance : A human can easily tell that (a) and (b) are from the same class. (c) could be either a '6' or a '9'.

The local stability to deformations of a non-expansive operator can be expressed as its Lipschitz continuity to the action of deformations close to translations [Mallat, 2012]. Such a diffeormorphism can be expressed as,

$$\begin{aligned} L_\tau : \mathcal{L}^2(\mathbb{R}^d) &\to \mathcal{L}^2(\mathbb{R}^d) \\ \mathbf{x} &\to \mathbf{x}((.) - \tau(.)) \end{aligned}$$

where $\tau(u) \in \mathbb{R}^d$ is a displacement field.

Definition 1.1.5. ***(Lipschitz continuous)***
A non expansive operator Φ is said to be Lipschitz continuous to the action of $\mathcal{C}^2$ diffeomorphisms if for any compact $\Omega \subset \mathbb{R}^d$ there exists C such that for all $f \in \mathcal{L}^2(\mathbb{R}^d)$ supported in Ω and all $\tau \in \mathcal{C}^2(\mathbb{R}^d)$,

$$\|\Phi(\mathbf{x}) - \Phi(L_\tau \mathbf{x})\|_H \leq C \|\mathbf{x}\| \left(\sup_{u \in \mathbb{R}^d} |\nabla \tau(u)| + \sup_{u \in \mathbb{R}^d} |H\tau(u)| \right) \tag{1.2}$$

where $\nabla\tau(u)$ is a matrix whose norm $|\nabla\tau(u)|$ measure the deformation amplitude at point u, $H\tau(u)$ is the Hessian matrix of the deformation and its sup-norm $|H\tau(u)|$ measures the smoothness of the deformation.

Hence a Lipschitz continuous operator Φ is almost invariant to deformations by $\tau(.)$, up to the first and second order deformation terms. The Equation 1.2 also implies that Φ is invariant to global translations but this is already enforced by the translation invariance requirement.

1.1.3 State of the art in signal representation :

A common representational method is the modulus of the Fourier transform. This operator is informative enough to allow —to a certain extent— discrimination between different types of signals [Baker et al., 2014]. It is also translation invariant [Bracewell, 1965]. However it is well known that those operators present instabilities to deformation at high frequencies [Hörmander, 1971] and thus are not Lipschitz continuous to the action of diffeomorphisms.

Wavelet transform is another popular representation method. Again it provides a representation suitable for classification of different signals [De Chazal et al., 2000]. Plus by grouping high frequencies into dyadic packet in $\mathbb{R}^d$, wavelet operators are stable to —small— deformations [Bruna and Mallat, 2013].

$$W\mathbf{x} = \begin{pmatrix} \mathbf{x} * \phi & \rightarrow \text{averaging part} \\ \mathbf{x} * \psi_\lambda & \rightarrow \text{high frequency part} \end{pmatrix} \tag{1.3}$$

However only the averaging part of a wavelet is invariant to translation ans thus wavelets themselves are known to be non-invariant to translations.

Another popular signal representation method are the convolutional neural networks [LeCun and Bengio, 1995]. As opposed to the two previously mentioned methods, those operators are not fixed but learned from the data [Simard et al., 2003]. Over the past few years they have provided state of the art results on many standard classification tasks, on images datasets such as *MNIST*, *CIFAR* [Hinton et al., 2012] and *ImageNet* [Krizhevsky et al., 2012] as well as on speech processing problems such as *TIMIT* [Abdel-Hamid et al., 2012]. Those good results are used to advocate that those networks are learning "good" representations. However there is no mathematical formalisation of this intuition and it seems that in certain cases they learn representation of the data that are, for example, not invariant to deformations [Szegedy et al., 2013].

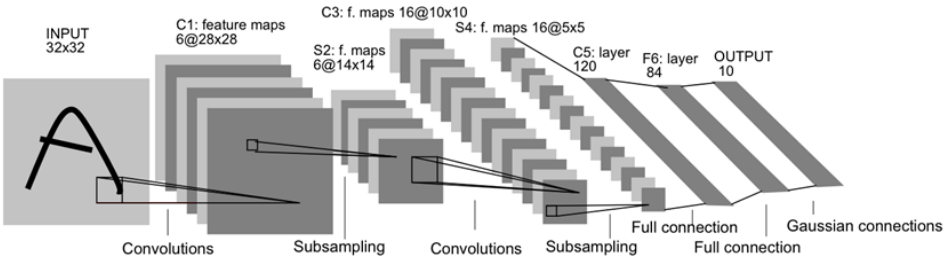


FIGURE 1.5 – Convolutional neural network with 3 convolution/sub-sampling layers and 3 fully connected layers.

1.2 Discriminative versus generative models :

When facing a prediction task for an unobserved random variable y — or random vector $\mathbf{y}$— given a new observation $\mathbf{x}^{new}$, one can choose between two different types of model.

A discriminative model directly expresses the dependencies between the unobserved variable and the observed ones. Which can be seen in a probabilistic framework as model the conditional probability distribution $P(\mathbf{y}|\mathbf{x})$. The prediction can then be done directly from the learned function. Examples of such models are the Support Vector Machines (SVMs), the Neural Networks (NNs) or the Conditional Random Fields (CRFs).

On the other hand one can use a generative model and specify the joint probability distribution $P(\mathbf{x}, \mathbf{y})$. This means that the prediction task —i.e. the conditional probability distribution— is not straight forward using the model but requires computation via Bayes' rule and specification of a prior $P(\mathbf{x})$. However this also means that those models can be used for more than simply prediction and can for example be use to generate the "most likely input" given an output. Examples of such models are the Naive Bayes classifier, the Restricted Boltzmann Machines (RBMs) or the Probabilistic Graphical Models.

Because they require extra computation generative models are usually considered as less efficient for classification than the discriminative ones [Ulusoy and Bishop, 2005] [Jordan, 2002]. However because they model the full joint probability distribution they can be use on a much broader range of tasks. They also have a readily available method for handling missing data, and often better performance when trained on small size training sets.

1.3 Outline of the report :

The remainder of this document is organized as follow. Section 2 summarizes and explains recent work from Stéphane Mallat and his group on the Scattering Transform (ST), a wavelet-based operator fulfilling all the properties of what has been defined as a "good" representation for signal classification. Section 3 introduces the concept of Probabilistic Graphical Models (PGMs) as generative models that can be used —among other tasks— for classification. Section 4 describes how the representation produced by the scattering transform can be modelled by a hidden Markov tree, using what we have named Scattering Convolutional Hidden Markov Trees (SCHMTs). Finally Section 5 provides some examples of application.

1.4 Contribution :

The first two sections of this documents are mostly dedicated at introducing pre-existing notions and concepts that are useful to later construct the scattering convolutional hidden Markov trees. Readers already familiar with Scattering Transforms or Probabilistic Graphical Models can respectively skip Section 2 or 3 and focus on the novelties introduced in Section 4 and onward where a framework to model a scattering convolutional network as a hidden Markov tree is presented. This new tool is then tested on classification and segmentation tasks.

Note also that even if this document is mainly written using images —i.e. 2-D signals— as examples, the framework presented is valid in any number of dimensions with minor adjustments.

2 The Scattering transform :

This section describes the construction of a mathematical operator —the scattering transform (ST)— designed to generate what has been defined earlier as an interesting representation of signal (see Section 1.1). This operator delocalizes signal's informative content into scattering decomposition paths, computed by cascading wavelet/modulus operators through an architecture similar to a Convolutional Neural Network (CNN) where the synaptic weights would be given by a wavelet operator instead of learned.

The remainder of the chapter is organized as follow. First, Section 2.1 defines the scattering operators. Second, Section 2.2 describes how those operators can be stacked to create a Scattering Convolutional Network (SCN), an architecture comparable to CNNs. Then Section 2.3 reviews some of the SCNs' important properties. And finally, Section 2.4 presents how the scattering transform is usually used in classification tasks.

2.1 Scattering wavelets :

A two-dimensional directional wavelet is obtained by scaling and rotating a single band-pass filter ψ. If one let G be a discrete, finite rotation group of $\mathbb{R}^2$, multi-scale directional wavelet filters are defined for any scale $j \in \mathbb{Z}$ and rotation $r \in G$ by,

$$\psi_{2^j r}(u) = 2^{2j}\psi(2^j r^{-1}u). \tag{2.1}$$

To simplify the notations, let now $\lambda = \lambda(j, r) \overset{d}{=} 2^j r \in G \times \mathbb{Z}$.

A wavelet transform filters the signal $\mathbf{x}$ using a family of wavelets $\{\mathbf{x} * \psi_\lambda(u)\}_\lambda$ computed from a filter bank of dilated and rotated wavelets having no orthogonality property. This generates a multi-scales and multi-orientations representation of the input signal.

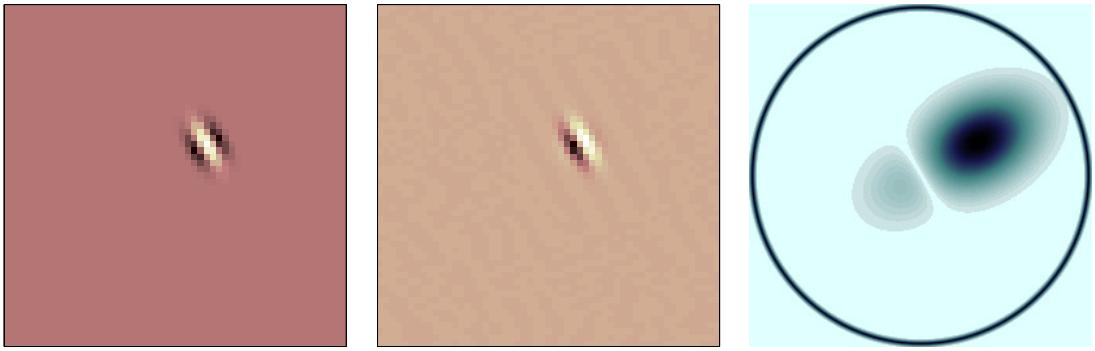
If $u.u'$ and $\|u\|$ define respectively the inner product and the norm in $\mathbb{R}^2$, the Morlet wavelet ψ is an example of wavelet given by,

$$\psi(u) = C_1(e^{iu.\xi} - C_2)e^{\|u\|^2/(2\sigma^2)},$$

where C_1, ξ and σ are meta-parameters of the wavelet and C_2 is adjusted so that $\int \psi(u)du = 0$. Figure 2.1 shows a Morlet wavelet for $\xi = 3\pi/4$, $\sigma = 0.85$ and $C_1 = 1$.

As opposed to the Fourier sinusoidal waves, wavelets are operators stable to local $\mathcal{L}^2$ deformations as they can be expressed as localized waveforms [Mallat, 1999]. However, as wavelet transform computes a convolutions with a wavelet basis, the resulting transform is a translation covariant operator [Bruna and Mallat, 2013].

To ensure a translation invariant behaviour for an operator initially commuting with translations, one has to introduce a non-linearity in the processing pipeline. Integration is an



(a) Real part of ψ. (b) Imaginary part of ψ. (c) Fourier modulus $|\hat{\psi}|i$.

FIGURE 2.1 – Complex Morlet wavelet.

example of such a non-linearity. For example, let R be an operator linear or not commuting with translations L_c, i.e. $R(L_c\mathbf{x}) = L_cR(\mathbf{x}))$, then the integral $\int R(\mathbf{x}(u))du$ is translation invariant. Applying such a non-linearity to a wavelet transform is equivalent to setting $R(\mathbf{x}) = \mathbf{x} * \psi_\lambda$ and one gets the trivial invariant,

$$\int \mathbf{x} * \psi_\lambda(u)du = 0,$$

for all signal $\mathbf{x}$ as $\int \psi_\lambda(u)du = 0$.

However to preserve the informative character of the scattering operator, one has to ensure a non-vanishing integral. To do so a second operator M has to be introduced such that $M \circ R(\mathbf{x}) = M(\mathbf{x} * \psi_\lambda)$. If M was a linear transformation commuting with translation then the integral would still vanish. Hence one has to choose M among the non-linear operator family.

Keeping in mind that the scattering transform has to be stable to deformations, one also imposes on M to commute with deformations,

$$\forall \tau(u)\ ,\ ML_\tau = L_\tau M.$$

If a weak differentiability condition is added, one can prove [Bruna, 2012] that M must necessarily be a point-wise operator, i.e. $M \circ R(\mathbf{x}(u))$ only depends on the value of $\mathbf{x}(u)$.

Finally, by adding the $\mathcal{L}^2(\mathbb{R}^2)$ stability constraint,

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{L}^2(\mathbb{R}^2), \quad \|M \circ R(\mathbf{x})\| = \|\mathbf{x}\| \quad \text{and} \quad \left\|M \circ R(\mathbf{x}) - M \circ R(\mathbf{x}')\right\| \leq \left\|\mathbf{x} - \mathbf{x}'\right\|,$$

one can also show [Bruna, 2012] that necessarily,

$$M(R(\mathbf{x})) = e^{i\alpha} \left|R(\mathbf{x})\right|.$$

To define the scattering transform, the simplest solution $\alpha = 0$ is chosen and therefore the resulting coefficients are the $\mathcal{L}^1(\mathbb{R}^2)$ norms,

$$\|\mathbf{x} * \psi_\lambda\|_1 = \int |\mathbf{x} * \psi_\lambda| \, du$$

The family of the $\mathcal{L}^1(\mathbb{R}^2)$ normed wavelet $\{\|\mathbf{x} * \psi_\lambda\|_1\}_\lambda$ generates a crude signal representation which measures the sparsity of the wavelet coefficients. It can be proven that the signal $\mathbf{x}$ can be reconstructed from $\{|\mathbf{x} * \psi_\lambda(u)|\}_\lambda$ up to a multiplicative constant [Waldspurger et al.,

2015]. This means that the information loss in $\{\|\mathbf{x} * \psi_\lambda\|_1\}_\lambda$ occurs during the integration of the absolute value $|\mathbf{x} * \psi_\lambda(u)|$ which removes all non-zero frequencies. However those components can be recovered by calculating the wavelet coefficients $|\mathbf{x} * \psi_{\lambda_1}| * \psi_{\lambda_2}(u)$ of the new signal $|\mathbf{x} * \psi_{\lambda_1}|$. By doing so their $\mathcal{L}^1(\mathbb{R}^2)$ norms define a much larger family of invariants,

$$\forall(\lambda_1, \lambda_2) \quad \||\mathbf{x} * \psi_{\lambda_1}| * \lambda_2\|_1 = \int ||\mathbf{x} * \psi_{\lambda_1}(u)| * \psi_{\lambda_2}| \, du.$$

By further iterating on the wavelet/modulus operators, more translation invariant coefficients can be generated. The building bloc of such a model —the scattering propagator— is thus the absolute value of the convolution between a wavelet and the input signal.

Definition 2.1.1. ***(Scattering propagator)***
The scattering operator U for a scale and an orientation $\lambda \in G \times \mathbb{Z}$ is defined as the absolute value of the input convolved with the wavelet operator at this scale and orientation.

$$U[\lambda](\mathbf{x}) \overset{d}{=} |\mathbf{x} * \psi_\lambda| \,. \tag{2.2}$$

Definition 2.1.2. ***(Path ordered scattering propagators)***
Any sequence $p = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ where $\forall i \in [\![1, m]\!] \;\; \lambda_i \in G \times \mathbb{Z}$ defines a path of length m, i.e. the ordered product of non-linear and non-commuting operators,

$$\begin{aligned} U[p]\mathbf{x} &\overset{d}{=} U[\lambda_m] \ldots U[\lambda_2] U[\lambda_1](\mathbf{x}) \\ &= ||||\mathbf{x} * \psi_{\lambda_1}| * \psi_{\lambda_2}| \ldots | * \psi_{\lambda_m}| \,. \end{aligned} \tag{2.3}$$

With the convention : $U[\emptyset]\mathbf{x} = \mathbf{x}$.

From there one can provide a first formal definition of the scattering coefficients.

Definition 2.1.3. ***(Scattering coefficient)***
A scattering coefficient along the path p is defined as an integral of the p ordered scattering propagators, normalized by the response of a Dirac :

$$\bar{S}[p](\mathbf{x}) \overset{d}{=} \mu_p^{-1} \int U[p]\mathbf{x}(u) du, \tag{2.4}$$

with,

$$\mu_p \overset{d}{=} \int U[p]\delta(u) du.$$

Section 2.3 shows that each scattering coefficient $\bar{S}[p](\mathbf{x})$ is —as desired— invariant to translation of the input $\mathbf{x}$ and Lipschitz continuous to deformations. But for prediction tasks, one might want to compute only localized descriptors only invariant to translations smaller than a predefined scale 2^J, while keeping the spatial variability at scales larger than 2^J. This can be achieved by localizing the scattering integral with a scaled spatial window $\phi_{2^J}(u) = 2^{-2J}\phi(2^{-2J}u)$. This yields the definition of the windowed scattering transform.

Definition 2.1.4. ***(-Windowed- scattering coefficient of order m)***
If p is a path of length $m \in \mathbb{N}$, the —windowed— scattering coefficient of order m at scale 2^J ($J \in \mathbb{N}$) is defined as :

$$
\begin{aligned}
S_J[p](\mathbf{x}) &\stackrel{d}{=} U[p]\mathbf{x} * \phi_{2^J}(u) \\
&= \int U[p]\mathbf{x}(v)\phi_{2^J}(u-v)dv \\
&= ||||\mathbf{x} * \psi_{\lambda_1}| * \psi_{\lambda_2}| \ldots | * \psi_{\lambda_m}| * \phi_{2^J}(u),
\end{aligned} \tag{2.5}
$$

With the convention : $S_J[\emptyset]\mathbf{x} = \mathbf{x} * \phi_{2^J}$.

2.2 Scattering Convolution Network :

This section introduces the scattering transform as an iterative process over a one-step operator and creates a parallel with convolutional neural networks [LeCun et al., 2010].

For $J \in \mathbb{N}^*$, let $U_J[\Omega] \stackrel{d}{=} \{U_J[p]\}_{p\in\Omega}$ and $S_J[\Omega] \stackrel{d}{=} \{S_J[p]\}_{p\in\Omega}$. They defines families of operators indexed by a set of paths Ω. One can compute a windowed scattering transform by iterating over the one-step —windowed— propagator U_J.

Definition 2.2.1. ***(Scattering one-step propagator)***
The one-step propagator U_J *can be defined as,*

$$
\forall \mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d) \quad \textit{and} \quad \forall \textit{ path } p, \quad U_J\mathbf{x} = \{A_J\mathbf{x},\ (U[\lambda]\mathbf{x})_{\lambda\in\Lambda_J}\}, \tag{2.6}
$$

where $A_J\mathbf{x} = \mathbf{x} * \phi_{2J}$, $U[\lambda]\mathbf{x} = |\mathbf{x} * \psi_\lambda|$ *and* $\Lambda_J = G \times [\![0, J]\!]$.

Indeed after calculating $U_J\mathbf{x}$, applying U_J again to each $U[\lambda]$ generates a larger infinite family of functions. And since $U[\lambda]U[p] = U[p+\lambda]$ and $A_JU[p] = S_J[p]$ it holds that,

$$
\forall \mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d) \quad U_JU[p]\mathbf{x} = \{S_J[p]\mathbf{x},\ (U[p+\lambda]\mathbf{x})_{\lambda\in\Lambda_J}\}, \tag{2.7}
$$

Let now Λ_J^m denotes the set of all paths of length m with the convention $\Lambda_J^0 = \{\emptyset\}$, its propagation is,

$$
\forall \mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d) \quad U_JU[\Lambda_J^m]\mathbf{x} = \{S_J[\Lambda_J^m]\mathbf{x},\ (U[\Lambda_J^{m+1}]\mathbf{x})_{\lambda\in\Lambda_J}\}. \tag{2.8}
$$

Definition 2.2.2. ***(Scattering transform)***
The scattering transform of order m *at scale* J *can be define as the set of scattering coefficients obtained for all paths of length* 0 *to* m,

$$
S_J[\mathcal{P}_J^m]\mathbf{x} = \left[S_J[\Lambda_J^0]\mathbf{x}, \ldots, S_J[\Lambda_J^m]\mathbf{x}\right]. \tag{2.9}
$$

Hence the scattering transform of infinite depth $S_J[\mathcal{P}_J]\mathbf{x}$ can be computed from $\mathbf{x} = U[\emptyset]\mathbf{x}$ by iteratively computing $U_JU[\Lambda_J^m]\mathbf{x}$ for m going from 1 to ∞. This iterative process is illustrated in Figure 2.2.

One can notice that the scattering calculation as the same general architecture as the convolutional neural networks introduced by [LeCun and Bengio, 1995]. Both CNN and Scattering Convolutional Network (SCN) cascade convolutions and a "pooling" non linearity. However while convolutional neural networks use kernel filters learned from the data with back-propagation algorithm, SCNs use a fixed wavelet filter bank.

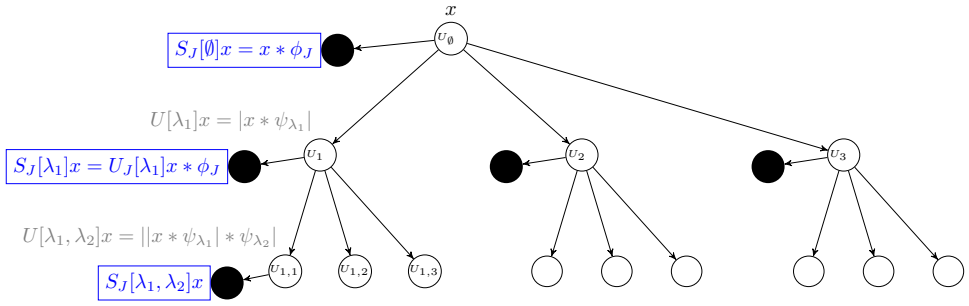


FIGURE 2.2 – A scattering propagator U_J applied to a signal $\mathbf{x}$ computes each $U[\lambda_i]\mathbf{x} = |\mathbf{x} * \psi_{\lambda_i}|$ and outputs $S[\emptyset]\mathbf{x} = \mathbf{x} * \phi_{2^J}$. Applying U_J to each $U[\lambda_i]\mathbf{x}$ computes all $U[\lambda_i, \lambda_j]\mathbf{x}$ and outputs $S_j[\lambda_i] = U[\lambda_i] * \phi_{2^J}$. Applying iteratively U_J to each $U[p]\mathbf{x}$ outputs $S_J[p]\mathbf{x} = U[p]\mathbf{x} * \phi_{2^J}$ and computes the next path layer.

2.3 Properties of the scattering transform :

The scattering coefficient having been defined, one can be interested in the characteristics of such a data representation. This section provides an overview of some of the properties of the scattering transform. It also introduces an approximation to the scattering convolution network defined in the previous section, leading to computationally tractable networks.

Note. Formal proofs of most of those properties can be found in [Mallat, 2012].

2.3.1 Non-expansivity :

The scattering one-step propagator,

$$U_J\mathbf{x} = \{A_J\mathbf{x},\ (U[\lambda]\mathbf{x})_{\lambda\in\Lambda_J}\} = \{A_J\mathbf{x},\ (|W_J\mathbf{x}|)_{\lambda\in\Lambda_J}\},$$

results of the composition of a wavelet transform W_J that is unitary and of a modulus operator that is non-expansive —as $\forall(a,b) \in \mathbb{C}^2\ ||a| - |b|| \leq |a - b|$— and is thus also non-expansive. Since $S_J[\mathcal{P}_J]$ iterates on U_J, which is non-expansive, the proposition (proved by [Lohmiller and Slotine, 1998]) proves that $S_J[\mathcal{P}_J]$ is also non-expansive.

Proposition 2.3.1. ***(Non-expansive)***
The scattering transform is non expansive.

$$\forall\mathbf{x},\mathbf{x}' \in \mathcal{L}^2(\mathbb{R}^d) \quad \left\|S_J[\mathcal{P}_J]\mathbf{x} - S_J[\mathcal{P}_J]\mathbf{x}'\right\| \leq \left\|\mathbf{x} - \mathbf{x}'\right\| \tag{2.10}$$

2.3.2 Energy preservation :

Each propagator $U[\lambda]\mathbf{x} = |\mathbf{x} * \psi_\lambda|$ captures the frequency energy contained in the signal $\mathbf{x}$ over a frequency band covered by the Fourier transform $\hat{\psi}_\lambda$ and propagates this energy

towards lower frequencies. It can thus be proved that under some assumptions on the wavelet —admissible wavelet, the whole scattering energy ultimately reaches the minimum frequency 2^{-J} and is trapped by the low-pass filter ϕ_{2^J}. Thus the energy propagated by a —windowed— scattering transform goes to 0 as the path length increases, implying that $\|S_J[\mathcal{P}_J]\| = \|x\|$

But prior to showing this, one must states the necessary assumptions to be made on the wavelet used.

Definition 2.3.2. ***(Admissible scattering wavelet)***
A scattering wavelet ψ is admissible if there exist $\eta \in \mathbb{R}^d$ and $\rho \in \mathcal{L}^2(\mathbb{R}^d)$ positive, with $|\hat{\rho}(\omega)| \leq \left|\hat{\phi}(2\omega)\right|$ and $\hat{\rho}(\omega) = 0$, such that the function,

$$\hat{\Psi}(\omega) = |\hat{\rho}(\omega - \eta)|^2 - \sum_{k=1}^{+\infty} k(1 - \left|\hat{\rho}(2^{-k}(\omega - \eta)\right|^2), \tag{2.11}$$

satisfies,

$$\alpha = \inf_{1 \leq |\omega| \leq 2} \sum_{j=-\infty}^{+\infty} \sum_{r \in G} \hat{\Psi}(2^{-j} r^{-1} \omega) \left|\hat{\psi}(2^{-j} r^{-1} \omega)\right|^2 > 0. \tag{2.12}$$

For an admissible wavelet one can prove that the scattering transform conserves the energy of the signal.

Theorem 2.3.3. ***(Energy conservation)***
If the scattering wavelet ψ is admissible, then for all signal $\mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d)$,

$$\lim_{m \to +\infty} \|U[\Lambda_J^m]\mathbf{x}\|^2 = \lim_{m \to +\infty} \sum_{n=m}^{+\infty} \|S_J[\Lambda_J^n]\mathbf{x}\|^2 = 0, \tag{2.13}$$

and

$$\|S_J[P_J]\mathbf{x}\|^2 = \|\mathbf{x}\|. \tag{2.14}$$

The proof of the Theorem 2.3.3 also shows that the scattering energy propagates progressively towards lower frequencies and that the energy of $U[p]\mathbf{x}$ is mainly concentrated along frequency decreasing paths $p = (\lambda_k)_{k \leq m}$, i.e. for which $|\lambda_{k+1}| \leq |\lambda_k|$. The energy contained in the other paths is negligible and thus for the applications in this document only frequency decreasing paths are considered.

Moreover, the decay of $\sum_{n=m}^{+\infty} \|S_J[\Lambda_J^n]x\|^2$ implies that there exist a path length $M > 0$ after which all longer path can be neglected. For signal processing applications, this decay appears to be exponential. And for classification applications path of length $M = 3$ provides the most interesting results [Andén and Mallat, 2011], [Bruna and Mallat, 2010].

The two restrictions stated above yield an easier parametrization of a scattering network. Indeed when only the frequency decreasing paths up until a given order a scattering network is completely defined by :

— ψ : The admissible wavelet used. Unless stated otherwise the Morlet wavelet is used.
— M : The maximum path length considered.
— J : The coarsest scale level considered.

— L : The number of orientation considered, which can be define as the cardinality of the previously define ensemble G.

Hence for a given set of parameter (ψ, M, J, L), one can generate one and only one frequency decreasing paths scattering network. Let $ST_{(\psi,M,J,L)}(\mathbf{x})$ now denotes the frequency decreasing windowed scattering convolutional network of parameter (ψ, M, J, L) evaluated for signal $\mathbf{x}$. Each node i of this network generates a -possibly empty- set of of nodes of size $(j_i - 1) \times L$ where j_i is the scale of node i and L is the number of orientations considered. Finally the number of nodes O of this network can be expressed as,

$$O = \sum_{m=0}^{M-1} \binom{J}{m} . L^m \tag{2.15}$$

and it has the architecture displayed by Figure 2.3.

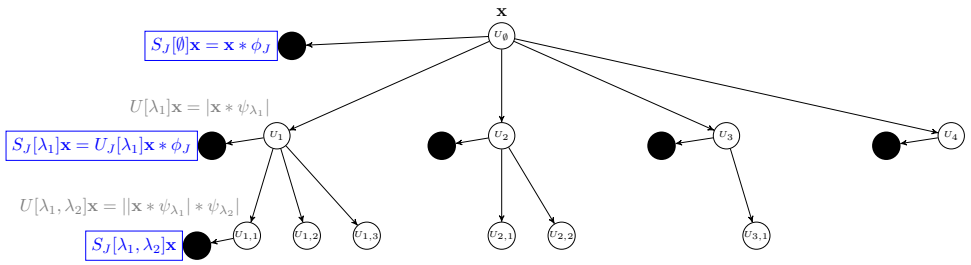


FIGURE 2.3 – Frequency decreasing scattering convolution network with $J = 4$, $L = 1$ and $M = 2$. A node i at scale j_i generates $(j_i - 1) \times L$ nodes.

2.3.3 Translation invariance :

The translation invariance of the scattering transform $S_J[\mathcal{P}_J]$ can be proved for a limit metric when J goes to infinity. To do so one can first prove that the scattering distance $\|S_J[\mathcal{P}_J]\mathbf{x} - S_J[\mathcal{P}_J]\mathbf{x}'\|$ converges when J goes to infinity — as it is non-increasing when J increases (see section 2.3.1). From there one can bound the distance between the scattering transform of the signal and the one of its translated version $\|S_J[\mathcal{P}_J]\mathcal{L}_c\mathbf{x} - S_J[\mathcal{P}_J]\mathbf{x}\|$ and prove that this bound tends to 0 when J goes to infinity. This proves the translation invariance.

Theorem 2.3.4. ***(Translation invariance)***
For admissible scattering wavelets,

$$\forall \mathbf{x} \in \mathcal{L}^2(\mathbb{R}^d),\ \forall c \in \mathbb{R}^d \quad \lim_{J\to\infty} \|S_J[\mathcal{P}_J]\mathcal{L}_c\mathbf{x} - S_J[\mathcal{P}_J]\mathbf{x}\| = 0 \tag{2.16}$$

2.3.4 Lipschitz continuity to the action of diffeomorphisms :

The Lipschitz continuity to the action of diffeomorphisms of $\mathbb{R}^d$ can be proved for deformations sufficiently close to a translation. Such diffeomorphisms map u to $u - \tau(u)$ where $\tau(u)$ is a displacement filed such that $\|\nabla\tau\|_\infty < 1$ —i.e. invertible transformations [Bruna and Mallat, 2013]. Let $L_\tau\mathbf{x}(u) = \mathbf{x}(u - \tau(u))$ denotes the action of such diffeomorphisms on the signal $\mathbf{x}$. Once again one can find an upper bound to the distance between the scattering transform of the signal and the one of its deformed version $\|S_J[\mathcal{P}_J]\mathcal{L}_\tau\mathbf{x} - S_J[\mathcal{P}_J]\mathbf{x}\|$. With a bit of work on this bound one can then proved that the consequences of the action of L_τ is bounded by a translation term proportional to $2^{-J}\|\tau\|_\infty$ and a deformation error proportional to $\|\nabla\tau\|_\infty$. Finally some more work on the bounding term provides the Lipschitz continuity.

Theorem 2.3.5. ***(Lipschitz continuity to the action of diffeomorphisms)*** *There exists C such that all $\mathbf{x} \in \mathcal{L}(\mathbb{R}^d)$ with $\|U[\mathcal{P}_J]\mathbf{x}\|_1 < \infty$ and all $\tau \in \mathcal{C}^2(\mathbb{R}^d)$ with $\|\nabla\tau\|_\infty < \frac{1}{2}$ satisfy,*

$$\|S_J[\mathcal{P}_J]\mathcal{L}_\tau\mathbf{x} - S_J[\mathcal{P}_J]\mathbf{x} + \tau.\nabla S_J[\mathcal{P}_J]\mathbf{x}\| \leq C\,\|U[\mathcal{P}_J]\mathbf{x}\|_1\,K(\tau), \tag{2.17}$$

with

$$K(\tau) = 2^{-2J}\,\|\tau\|_\infty^2 + \|\nabla\tau\|_\infty \left(\max\left(\log\frac{\|\Delta\tau\|_\infty}{\|\nabla\tau\|_\infty}, 1\right)\right) + \|H\tau\|_\infty . \tag{2.18}$$

Remark. If the case where $2^J \gg \|\tau\|_\infty$ and $\|\nabla\tau\|_\infty + \|H\tau\|_\infty \ll 1$, then $K(\tau)$ becomes negligible and the displacement field $\tau(u)$ can be estimated at each $u \in \mathbb{R}^d$. This can be done by solving the linear equation resulting from Equation 2.17 under the assumptions mentioned above,

$$\forall p \in \mathcal{P}_J\ \|S_J[p]\mathcal{L}_\tau\mathbf{x} - S_J[p]\mathbf{x} + \tau.\nabla S_J[p]\mathbf{x}\| \approx 0. \tag{2.19}$$

This estimate of the displacement field can be used for many applications such as object tracking in video sequences or image sequence restoration [Brailean and Katsaggelos, 1996].

2.4 Application to classification :

The scattering transform maps a given realisation of a high-dimensional signal into an even higher-dimensional space where the classification task is simplified due to the inherent properties described in the previous section. The scattering transform has been applied to classify a wild variety of signals such as audio signals [Andén and Mallat, 2011], images [Oyallon and Mallat, 2014] or electrocardiograms [Chudacek et al., 2014] and in the vast majority —if not all— the classification task has been done using the features generated by the transform of the dataset as inputs for a discriminative classifier, e.g. Support Vector Machine classifier. The new input vector is obtained by concatenating the scattering coefficients of all orders, scale and orientations into a unique 1-D vector -for 2-D signal the scattering coefficients are also flattened. Leveraging the richness of the representation generated the scattering transform combined to an SVM classifier provides performance comparable to those of a -small- deep convolutional neural network [Oyallon et al., 2013]. This section proposes to test this framework on the handwritten digit dataset *MNIST*.

MNIST is composed of 28×28 binary and centred images of handwritten digits. The dataset is split into a training set of 50000 images and a testing set of 10000 images and the task at hand is a 10 classes classification problem.

For this task the frequency decreasing scattering convolutional network has $M = 3$ layers, breaking down the images into $J = 3$ scales and $L = 6$ orientations. For each input image this networks generates 127 scattering coefficients (see Equation 2.15) and thus yields a 99568 dimensional feature vector. The discriminative classifier used is a set of binary SVM classifiers with a Gaussian radial basis function kernel [Schölkopf et al., 1997]. This classifier have two meta-parameters. γ defines how influential a single training example is and C the trade off between misclassification of training examples and simplicity of the decision surface. Those meta-parameters are fine-tuned by cross-validation to $C = 3$ and $\gamma = 0.0018$.

Using this set-up, the trained model scores 99.47% accuracy on the test set, i.e. 9947 true positive out of 10000 realisations. This accuracy is of the same order of what can be obtained

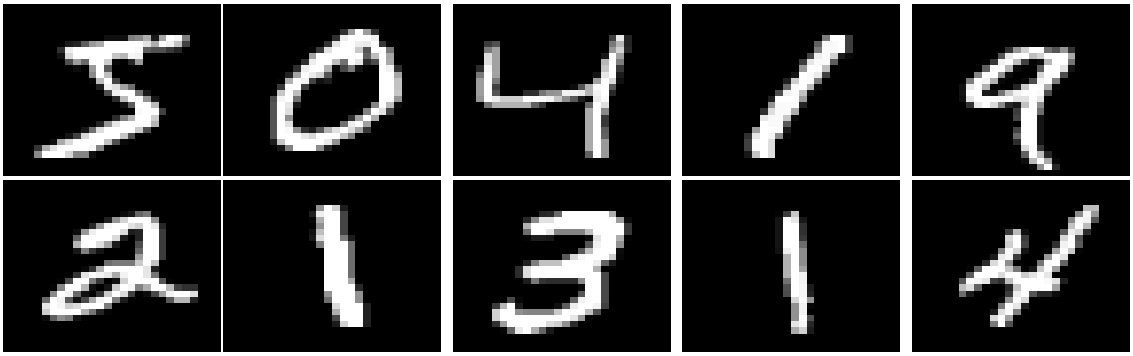


FIGURE 2.4 – Samples from MNIST.

using a large convolutional neural network [Jarrett et al., 2009] [LeCun, 2015].

This approach of classification have been used successfully for many more applications but it unfortunately does not directly leverage the structure created by the scattering transform and the possible information contained into it. Nor that it provides a generative models of the data, with all the advantages encompassed (see Section 1.2). The remainder of this document focuses on building a generative model describing a scattering convolutional network.

3 Probabilistic graphical models :

Probabilistic Graphical Models (PGMs) offer an efficient framework to express joint distributions and conditional dependencies. They rely on the usage of a graph based representation of conditional dependence between a set of random variables to encode a complete distribution over a multi-dimensional space in a compact —or factorized— manner. The PGMs can be split into two main families, the Bayesian Networks (BNs) and the Markov models (MMs). Both families encompass the properties of factorization and independences defined by the graph, but differ when it comes to the specificities of the set of independences they can encode as well as the factorization of the distribution that they can induce [Bishop, 2006].

Before using a graphical model to describe the scattering transform (see Section 4) this report provide an introduction to PGMs. Note however that the aim of this section is not to provide a complete overview of the probabilistic graphical models but rather to introduce some interesting concepts that have been used in the remainder of the document either in Section 4 to define the scattering convolutional hidden Markov tree or in the Section 6 as possible leads for future work. A reader with more interest in PGMs could refer to [Heckerman, 1998] or [Bishop, 2006] for a more complete introduction to those models.

This chapter introduces those two main classes of probabilistic graphical models. Section 3.1 focuses on the Bayesian networks, while Section 3.2 provides more details about Markov models.

3.1 Bayesian Network :

A BN is subclass of probabilistic graphical model where the set of random variables and their conditional dependencies are expressed via a Directed Acyclic Graph (DAG). Those model can be used to describe either continuous or discrete random variables as well as system governed by a mix of those. The architecture of the Bayesian Networks is further explained in section 3.1.1. Then section 3.1.3 presents a brief overview of the state of the art in terms of learning algorithm for BNs and section 3.1.2 describes the inference mechanism for those networks.

3.1.1 Architecture :

A Bayesian network is a graphical model encoding a joint probability distribution via a DAG.

Definition 3.1.1. ***Bayesian Network***
For a set of random variables $\mathbf{R} = \{R_i\}_{i \in [\![1,N]\!]}$*, a Bayesian network consists of a direct acyclic graph* $\mathcal{G}$ *encoding a set of conditional independence assertions about the random variables in* $\mathbf{R}$ *and a set* P *of local probability distribution associated with each variable.*
Each node of $\mathcal{G}$ *represents one of the random variable* R_i *and each edge* $E_{i \to j}$ *represents the conditional dependence between the nodes* R_i *and* R_j*.*

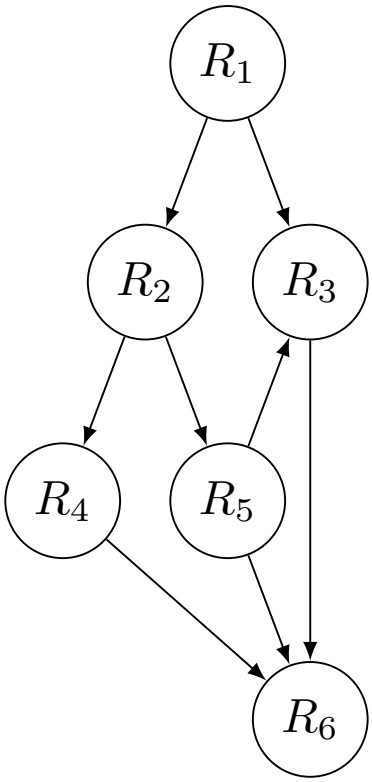


FIGURE 3.1 – Simple Bayesian network.

Such networks encodes the conditional independence properties of the distribution [Nilsson, 1998].

Proposition 3.1.2. ***(Conditional independence for Bayesian networks)*** *In a Bayesian network each node of the graph are conditionally independent of any subset of the nodes that are not descendants of itself given its parent.*

$$P(\{R_i\}_{i\in[\![1,N]\!]}) = \prod_{i=1}^{N} P(R_i|\rho(R_i)) \tag{3.1}$$

where $\rho(R_i)$ are the parents of the node R_i.

Thus in such a network, a node with no parents is not conditioned on any other random variable considered. Such a node defines a prior probability.

Using Property 3.1.2 allows to simplify the computation of the joint probability distribution. For example for the network defined by figure 3.1, one can obtain $P(R_1, R_2, R_3, R_4, R_5, R_6)$ by the use of chain rules and theory on conditional independent,

$$\begin{aligned} P(R_1, R_2, R_3, R_4, R_5, R_6) &= P(R_6|R_3, R_4, R_5)P(R_1, R_2, R_3, R_4, R_5) \\ &= P(R_6|R_3, R_4, R_5)P(R_3|R_1, R_5)P(R_4|R_2)P(R_5|R_2)P(R_1, R_2) \\ &= P(R_6|R_3, R_4, R_5)P(R_3|R_1, R_5)P(R_4|R_2)P(R_5|R_2)P(R_2|R_1)P(R_1). \end{aligned}$$

3.1.2 Inference :

A Bayesian network is an example of generative model and thus encodes the full joint distribution of the studied random variables. This knowledge can be used to perform several

interesting inference taks among which are :

— Belief updating (i.e. prediction) : Given some evidences, i.e. : values for some nodes of the network $\{R_j\}_{j\in J}$ where J is a subset of the graph, what is the probability associated with an unobserved variable,

$$R_i^* = P(R_i|\{R_j\}_{j\in J}). \tag{3.2}$$

This is the inference mechanism used for classification tasks. But the belief updating is more than just a classification tool as it can provide probabilistic prediction even with incomplete observations. Note that the variable to be predict can also be a set of random variables.

— Optimal decision : A probabilistic graphical model can be used to express actions taken by an agent to modify the state of an uncertain world. In this case given some evidences $\{R_j\}_{j\in J}$ where J is a subset of the graph $\mathcal{G}$ and a utility function $U(x)$ expressing an agent's preferences between world states, one is interested in finding the set of action $\{A_i\}_{i\in\mathcal{A}}$ where $\mathcal{A}$ is the set of all possible actions, maximizing the probability of the outcome ,

$$\{A_i^*\}_{i\in\mathcal{A}} = \underset{\mathcal{A}}{\operatorname{argmax}}\, P(O|\{A_i\}_{i\in\mathcal{A}}, \{R_j\}_{j\in J}). \tag{3.3}$$

This type of inference is useful in Reinforcement learning framework for example where one is interested in learning the optimal set of action to complete a task.

— Sensitivity analysis : Given some evidences, i.e. : values for some nodes of the network $\{R_j\}_{j\in J}$ where J is a subset of the graph used for belief updating, one can be interested in assessing which among those random variable as the most influence on the prediction quality. This means find,

$$\Delta_k^* = \underset{k\in J}{\operatorname{argmax}}\, \Delta_k \tag{3.4}$$

where Δ_k defines the difference between the probabilities given the full set of evidences and given the set minus the k-th evidence,

$$\Delta_k = P(R_i|\{R_j\}_{j\in J}) - P(R_i|\{R_j\}_{j\in J\setminus\{k\}}). \tag{3.5}$$

This type of inference can be useful in the case where the evidences are expensive to collect, or where there is a time limit to provide an answer. The sensitivity analysis allow to focus the effort into collecting the most important information.

3.1.3 Learning :

In most applications, the full characterization of the BN is not provided but has to be learned from a set of observations $\mathbf{X} = \{\mathbf{x}_n\}_{n\in[\![1,N]\!]}$. One can split the learning problem into two main categories :

— Learning the local probability distributions : In this case the structure of the graph $\mathcal{G}$ is known and fixed before hand. It can be provided by an expert (e.g. IBM trouble shooting system [Rish et al., 2002]) or be imposed by some construction rules (e.g. Boltzmann Machine [Ackley et al., 1985], Restricted Boltzmann Machine [Smolensky, 1986] ...). The task at end is then to learn the parameters Θ governing the local probability distributions of the Bayesian network.

— Learning the architecture and local probability distributions : In this case the architecture of the network $\mathcal{G}$ has to be learned along side with the local probability distributions' parameters Θ. This problem is not developed in the rest of this paper, but one could refer to [Margaritis, 2003] for an introduction to the existing methods.

Leaving aside the case where the network architecture has to be learned, the problem of learning the parameter of a Bayesian network can again be split into two main categories.

Complete data :

In this case each training example $\mathbf{x}$ contains the value of every random variable R of the graph. In such a case one can use methods such as the Maximum Likelihood estimates where the parameters of the network are selected to maximize the log-likelihood of the data given the model,

$$\Theta_{ML} = \underset{\Theta}{\operatorname{argmax}} \log P(\mathbf{X}|\Theta). \tag{3.6}$$

Another common estimator used is the Maximum A Posteriori (MAP) estimate where one maximize the posterior distribution of the network's parameters given the data,

$$\Theta_{MAP} = \underset{\Theta}{\operatorname{argmax}} \log P(\Theta|\mathbf{X}) = \underset{\Theta}{\operatorname{argmax}} \log(P(\mathbf{X}|\Theta)P(\Theta)). \tag{3.7}$$

Incomplete data :

In this case each training example $\mathbf{x}$ only contains the value of some random variable $\{R_i\}$ of the graph. In such a case one has to use a two step iterative algorithm named the Expectation Maximisation (EM) algorithm. The first step (Expectation) aims at estimating the values of the unobserved random variables given the current estimate of the parameters Θ, while the Maximisation step aims at providing a ML estimate of Θ. More details on this procedure are available in both Subsection 3.2.3 and Section 4.3.

3.2 Markov Models :

Markov models are a subclass of graphical models useful when it comes to describe a system whose observations are randomly changing over an event. The key assumption in those models is that the upcoming observations only depends on a finite number of past observations.

$$\forall t \in \mathbb{N} \quad P(\text{obs}_{t+1} \mid \{\text{obs}_k\}_{k\in\mathbb{N}}) \quad = \quad P(\text{obs}_{t+1} \mid \{\text{obs}_k\}_{k\in[\![t-l,t]\!]}) \tag{3.8}$$

where $l \in \mathbb{N}$ characterized the number of past steps used to condition the next observation. However most of the time, for sake of computational tractability as well as because this constraint has proved to be strong enough, the upcoming observation is assumed to be only dependent on the current one.

$$\forall t \in \mathbb{N} \quad P(\text{obs}_{t+1} \mid \{\text{obs}_k\}_{k\in\mathbb{N}}) \quad = \quad P(\text{obs}_{t+1} \mid \text{obs}_t) \tag{3.9}$$

Maybe the most famous class of Markov model is the Markov chain. It has applications in finance (Brownian motion [Duncan et al., 2000]), in Internet page ranking (Google page rank [Haveliwala and Kamvar, 2003]) or as a sampling procedure (Markov Chain Monte-Carlo (MCMC) [Gilks, 2005]). However this document does not cover those models. If one is interested in learning more about Markov chains [Kemeny and Snell, 1960] provides a good entry point to the field.

Another flavour of Markov models are the Hidden Markov Models (HMMs). The remainder of this section is dedicated to providing a general introduction to those models. Subsection 3.2.1 formalizes the HMM modelling. Then Subsection 3.2.2 and 3.2.3 respectively describes the inference mechanism and the learning method for HMMs.

3.2.1 Architecture :

An HMM is a stochastic finite automaton, where each hidden state generates, i.e. emits, an observation. Let O_t be the observation at step t and H_t denotes the hidden state at this step. Let also be K the number of possible states such that $H_t \in [\![1, K]\!]$. HMMs can handle discrete, continuous or mix type of random variables.

The model's parameters are :

— The initial state distribution $\pi(i) = P(H_0 = i)$ where π is a multinomial distribution.

— The transition model at step t, $\epsilon_t^{(ij)} = P(H_{t+1} = j | H_t = i)$ where ϵ_t is a stochastic matrix.

— The observation model $P(O_t|H_t)$. Usually the emission model is defined by a parametric distribution governed by $\theta_{k,t}$. In the case of discrete observations it is defined by a matrix a multinomial distributions such that,

$$\forall (l, k) \in |O| \times [\![1, K]\!] \quad P(O_t = l | H_t = k) = P_{\theta_{k,t}}(l).$$

In the continuous case, the observation model is a continuous parametric distribution such that,

$$\forall (l, k) \in \mathbb{R}^d \times [\![1, K]\!] \quad P(O_t = l | H_t = k) = P_{\theta_{k,t}}(l).$$

Those models can be simplified by assuming stationarity. Meaning that the transition matrix and observation models are shared, i.e. tied, across observations.

3.2.2 Inference :

Inference about the nodes in HMMs —what is called belief update for BNs— can be performed using the Maximum A Posteriori (MAP) algorithm. The aims of this procedure is to estimate the most likely hidden states given an observation and the model parameters. The initial MAP algorithm is due toViterbi [1967] and was originally designed to analyse Markov processes observed in memoryless noise. Forney Jr [1973] expressed this algorithm as being equivalent to finding the shortest path in a graph with weighted edges.

Note. This procedure is described at length in the special case of hidden Markov trees in Section 4.4 but this section aims at providing an informal explanations on the methodology.

Beside that, as HMMs are generative models they can perform all the inference tasks described for Bayesian networks, with the difference that for hidden Markov models the states of the hidden nodes always have to be inferred prior to any other tasks.

3.2.3 Learning :

Learning the parameters of an HMM from data is somehow similar to learning the parameters of a Bayesian network in the case of incomplete data, as only parts of the nodes of an HMM are observed. Hence the parameters of an HMM model can be learned using an offline maximum likelihood (ML) estimation methods known as the EM — or Baum-Welch— algorithm.

Let $\{O^n_{[1:T]}\}_{n\in[\![1,N]\!]}\}$ be a set of observe HMMs used as the training set. The learning procedure would be straight forward if one had access to the sequences of hidden state $H^n_{[0:T]}$ for all $n \in [\![1, N]\!]$. In this case, the ML estimate of the transition matrix could be computed by normalizing the matrix of co-occurrences,

$$\epsilon^{(ij)}_{t,ML} = \frac{C_t(i,j)}{\sum_{k=1}^{K} C_t(i,k)} \tag{3.10}$$

where

$$C_t(i,j) = \sum_{n=1}^{N} \mathbb{1}(H_{t+1} = j, H_t = i) \tag{3.11}$$

and $\mathbb{1}(\text{event})$ is the binary indicator of occurrence of a event. The initial distribution as well as the observation model could be estimated in a similar fashion.

However since $H^n_{[0:T]}$ is hidden, one first have to estimate the hidden state prior to performing the ML update. The rough idea of the EM-algorithm is to estimate the hidden states given the observations using a the Maximum A Posteriori approached described in the previous section with the current set of parameters. This is the Expectation (E) step. Those estimated sequences of hidden state are then used to update the parameters' estimate. This is the Maximisation (M) step.

One can prove [Baum et al., 1970] [Dempster et al., 1977] the convergence of this procedure toward a —local— maximum of the the likelihood.

4 Scattering convolutional hidden Markov tree :

Section 2.4 introduced the usage of scattering networks combined with an support vector machine classifier to achieve state of the art classification performance on some problems. However this method provides a boolean label for each class. Some methods to express the output of an SVM as a probability exists [Platt et al., 1999] but they are just a rescaling of the output and not a true probabilistic approach. If one is interested in a true probabilistic model to describe the scattering coefficient it is quite natural to try to express them as a probabilistic graphical model. Indeed if one ignores the propagation step from the scattering transform (see Section 2.2) the scattering network defines the tree structure displayed in Figure 4.1.

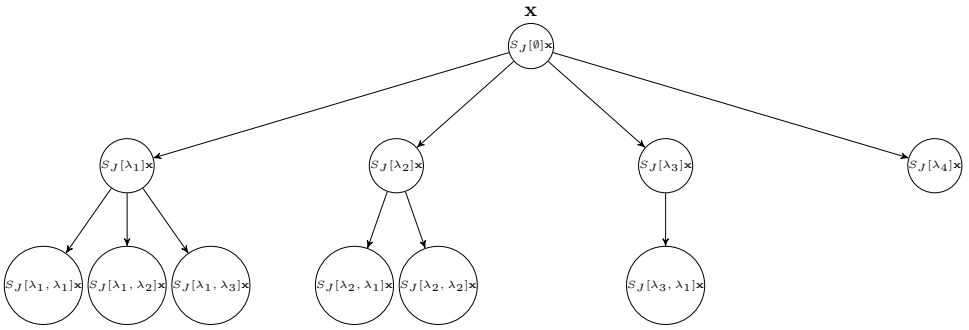


FIGURE 4.1 – Scattering transform tree.

To simplify notations in the remainder of this section, let $\mathcal{T}$ denotes the tree structure defines as stated above for a scattering convolutional network $ST_{(\psi,J,M,L)}(.)$ restricted to frequency decreasing path of length shorter than M, considering J scales and L orientations and depicted in Figure 4.1. Let I denotes the total number of nodes —i.e. scattering coefficients— and let S_i for $i \in [\![0, I-1]\!]$ denotes one of the node of $\mathcal{T}$ for a given path $p_i = [\lambda_0 ... \lambda_u]$ $(u \in \mathbb{N})$. Note that S_i represents a node and does not depends on the signal $\mathbf{x}$. For a given signal $\mathbf{x}$, a realization of the node i is denoted by $S_i = s_i = S[p_i]\mathbf{x}$. Note also that in the remainder of the paper the shorter notation $i \in \mathcal{T}$ will be used to denote the path p_i to a node. Let also use the convention $S_0 = S[\emptyset](.)$. Finally let ρ_i and $\mathcal{C}(i)$ denote respectively the parent of a node i and the set of children of the node i. Note also that a node S_i can have no children, in such a case this node is a leaf of the tree.

The remainder of this section is organized as follows : Section 4.1 introduces related work and provides a description of our proposed SCHMT model. Section 4.2 details the hypothesis needed to develop this model as well as provides some intuition on their validity. Finally Section 4.3 and 4.4 respectively describe the proposed learning algorithm for the parameters of an SCHMT and the classification method.

4.1 SCHMT model and related works :

The idea behind the SCHMT model is to say that the more detailed representation of the signal is somehow correlated to the less detailed one from which it is generated. More formally this means that for a signal $\mathbf{x}$, s_i is somehow correlated to $s_{\rho(i)}$. This assumption yields a modelling of the scattering network by a Markov tree and assumes,

$$P(S_i|\mathcal{T}) = P(S_i|S_{\rho(i)}). \tag{4.1}$$

Those independence properties are encoded in the graph displayed in Figure 4.1. However models trying to describe directly the correlation across coefficients at different scales have been studied for traditional wavelet transforms [Lee et al., 1996] but are in conflict with the compression property of the wavelet —i.e. the fact that most wavelet representations are sparse [Crouse et al., 1998]. Thus it seems that a simple one-step Markovian assumption across scale is not satisfying to describe the complex relationship between wavelet or scattering coefficients.

A common approach when a direct Markovian model does not hold is to introduce hidden states and to assume the Markovian property across those unobserved states. The observed nodes being then only dependent on their respective state. This is the architecture we have adopted for the SCHMT and its graph is represented in Figure 4.2. This model has the following independence properties,

$$P(H_i|\mathcal{T}) = P(H_i|H_{\rho(i)}), \tag{4.2}$$

$$P(S_i|\mathcal{T}) = P(S_i|H_i). \tag{4.3}$$

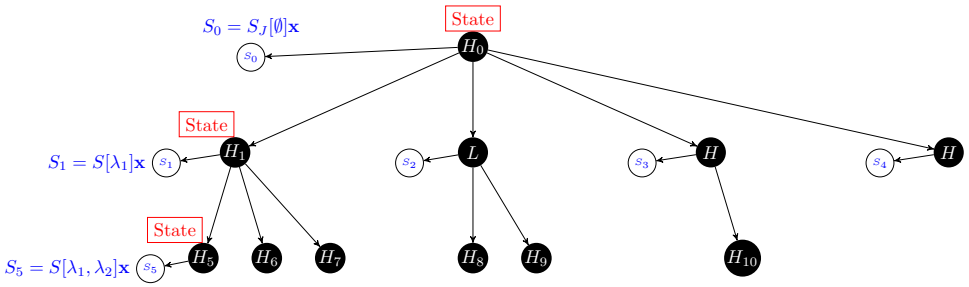


FIGURE 4.2 – Scattering convolutional hidden Markov tree.

As the scattering transform is closely related to wavelet transform it is not surprising to find similar ideas exploited for wavelet trees. Crouse has developed a model where a Hidden Markov Tree Model is used to model the wavelet coefficients [Crouse et al., 1998] of a standard wavelet trees. Later Kingsbury [2001] has adapted Crouse's model to his Dual Wavelet Complex Trees. The resulting hidden Markov tree models provides better classification performance than Crouse's Wavelet Hidden Markov Tree (WHMT) as the wavelet used generates a "better" representation of the signal in the sense defined in Section 1.1.1. Indeed this version can leverage the quasi-translation invariance property of the complex wavelets. The improvement in performance due to the quasi-invariance property provides a good motivation to try the hidden Markov tree modelling on the scattering transform as they have even "better" representational

properties (see Section 2.3). The parameters of the original WHMT are trained using a version of the Expectation-Maximization adapted to binary hidden Markov trees. However since this learning method suffers from underflowing issues [Devijver, 1985], Durand et al. [2004] proposed a smoothed version of the training algorithm preventing this from happening.

We have adapted those models to create a scattering convolutional hidden Markov tree composed of a set of visible nodes $\{\mathbf{S}_i\}_{i\in\mathcal{T}}$ and a set of hidden node $\{\mathbf{H}_i\}_{i\in\mathcal{T}}$. Both sets are organized in a tree structure with the following characteristics,

— For any index i of the tree, $S_i \in \mathbb{R}$ and $H_i \in [\![1, K]\!]$ where K is the number of possible hidden states.

— The initial hidden state is drawn from a discrete non uniform initial distribution π_0 such that :

$$\forall k \in [\![1, K]\!] \quad \pi_0(k) = P(H_0 = k). \tag{4.4}$$

— For any index i of the tree, the emission distribution describes the probability of the visible node S_i conditional to the hidden state H_i,

$$\forall i \in \mathcal{T} \ \ \forall k \in [\![1, K]\!] \ \text{ and } \ \forall s \in \mathbb{R} \qquad P(S_i = s_i | H_i = k) = P_{\theta_{k,i}}(s) \tag{4.5}$$

Where $P_{\theta_{k,i}}$ belongs to a parametric distribution family and where $\theta_{k,i}$ is the vector of emission parameters for the state k and the node i. In the remainder of the paper the emission distribution is a Gaussian so that,

$$\forall i \in \mathcal{T} \ \ \forall k \in [\![1, K]\!] \ \text{ and } \ \forall s \in \mathbb{R} \qquad P(S_i = s | H_i = k) = \mathcal{N}(\mu_{k,i}, \sigma_{k,i}). \tag{4.6}$$

where $\theta_{k,i} = (\mu_{k,i}, \sigma_{k,i})$ with $\mu_{k,i}$ and $\sigma_{k,i}$ being respectively the mean and the variance of the Gaussian for the k-th value of the mixture and the node i.

— For any index i of the tree, the probability to be in a state k given the father's state g is characterized by a transition probability,

$$\forall i \in \mathcal{T}\backslash\{0\} \quad \forall g, k \in [\![1, K]\!]^2 \qquad \epsilon_i^{(gk)} = P(H_i = k | H_{\rho(i)} = g), \tag{4.7}$$

where ϵ_i defines a transition probability matrix such that,

$$\forall i \in \mathcal{T}\backslash\{0\} \quad \forall k \in [\![1, K]\!] \qquad P(H_i = k) = \sum_{g=1}^{K} \epsilon_i^{(gk)} P(H_{\rho(i)} = g). \tag{4.8}$$

Note that using the chain rule of probability one can express $P(H_i)$ from the root node's initial distribution.

Thus for a given scattering architecture —i.e. fixed M, J and L— the SCHMT model is fully parametrized by,

$$\Theta = \big(\pi_0, \{\epsilon_i, \{\theta_{k,i}\}_{k\in[\![1,K]\!]}\}_{i\in\mathcal{T}}\big). \tag{4.9}$$

Our SCHMT model differs from the previous works by the shape of its tree structure. Previous works are based on regular binary trees where all the leaves have the same depth while the scattering tree is irregular. Indeed as seen in Section 2.2 each node has a variable number of children, which yields an architecture where the number of descendants is not constant

and where leaves can be found at any depth of the tree. However Section 4.3 describes an adaptation of Durand et al. [2004] learning algorithm to un-regular, un binary trees. Another difference between SCHMT and the previous works is the non-homogeneity of the transition matrix. Indeed by the nature of the scattering transform one can expect a non homogeneous transmission of the information across the orders and especially across the orientations. Hence to reflect we allows non-homogeneous transition matrices across nodes from a same father and across images themselves. Again Section 4.3 presents our adaptation of the learning algorithm to this case.

Even though the theoretical framework of SCHMT holds for any $K \in \mathbb{N}^*$, in the all the application of the SCHMT K is set to 2. This means that the scattering coefficients are described by a mixture of two Gaussians and can be in two states, either (L) Large or (S) Small. This model yields a sparser representation as the number of hidden states is highly constrained.

4.2 Hypothesis :

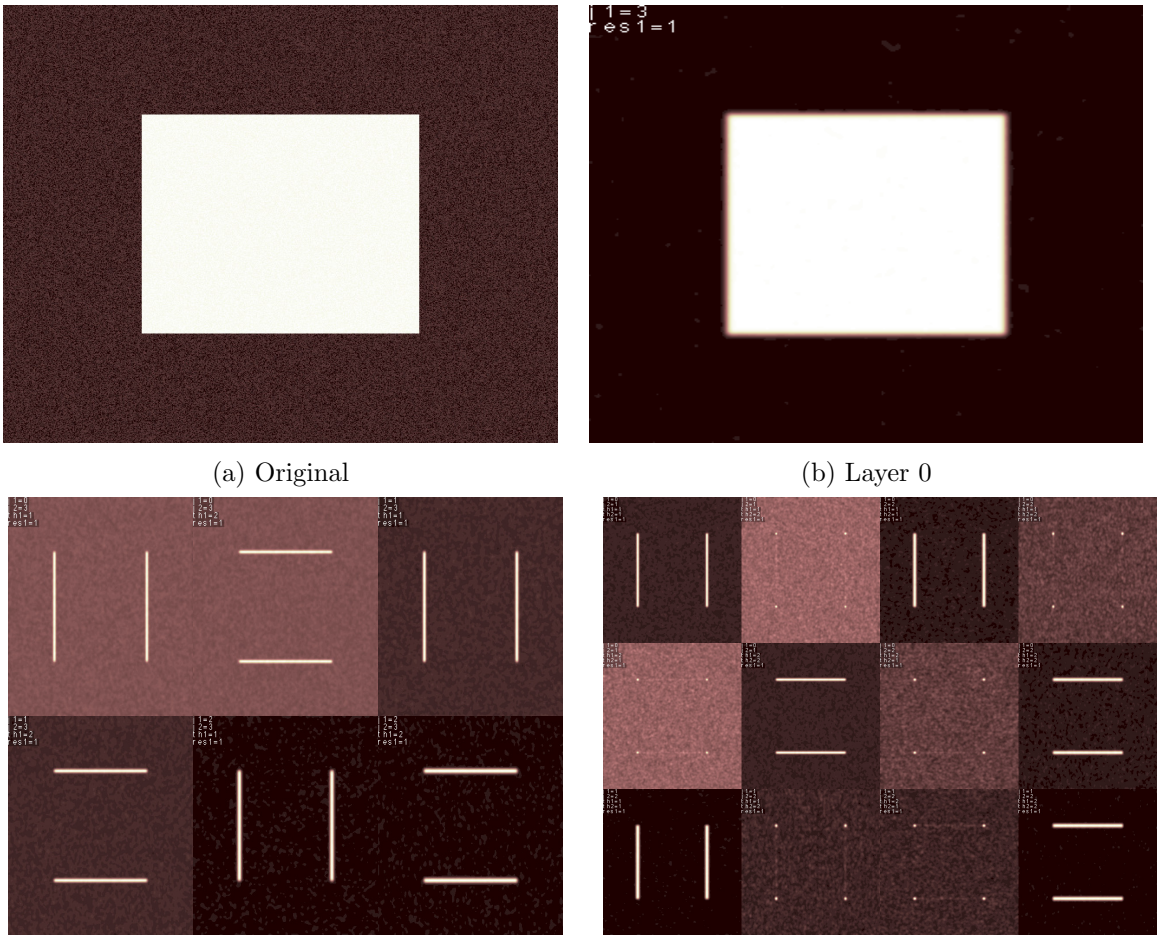
Expressing the dependencies between the scattering coefficients as a Hidden Markov Tree implies two modelling assumptions. The first one reflects the fact that the scattering coefficients can effectively be expressed by two hidden states,

Assumption 1. K **populations :**
A signal's scattering coefficients can be described by K clusters. The smooth regions are represented by small scattering coefficients, while edges, ridges, and other singularities are represented by large coefficients.

As this assumption is common for standard or complex wavelets [Kingsbury, 2001] and because a scattering coefficient of order m can simply be seen as the modulus of the wavelet transform of a "new" signal —i.e. the scattering coefficient of order $m-1$, the two-populations assumption for scattering network is reasonable. This intuition can be confirmed by Figure 4.3 and 4.4 displaying the scattering coefficients at a given node obtained for several signals.

Figure 4.3 displays the scattering coefficients of a noisy binary square. Note that for sake of clarity a "small" network has been used. This does not affect the observations that can be made and one can notice that the largest values of the scattering coefficient are obtained on highly informative pixels (edges in this case) while the less informative pixels are represented by scattering coefficients near 0. Similar observations can be made for more complex signal —such as the one displays in Figure 4.4. A statistical interpretation of the K populations assumption implies that scattering coefficients have non-Gaussian marginal statistics, that is, their marginal probability density functions have a large peak at zero due to the many small coefficients and heavy tails a few large coefficients are observed. Finally since many real-world signals (photograph-like images, for example) consist mostly of smooth regions separated by a few singularities, the K populations assumption tells us that the scattering coefficients are a sparse representation for these signals (this notion of sparsity can be made mathematically precise ; see for example [Donoho, 1993] or [DeVore et al., 1992]). Most of the scattering coefficient magnitudes are small, while a few of them encoding the singularities and the informative content are large.

The second assumptions expresses the smoothness of the states across the scattering transform.



(a) Original (b) Layer 0

(c) Layer 1 (d) Layer 2

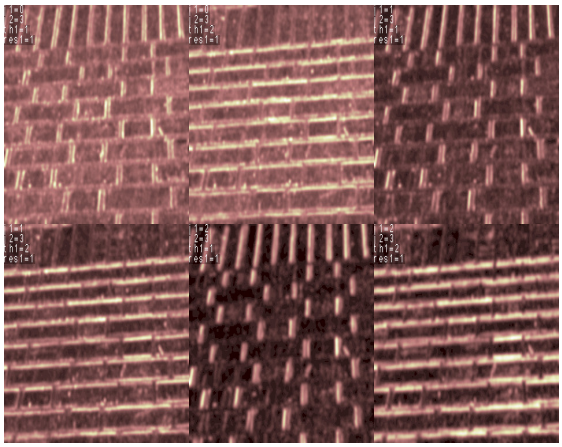
FIGURE 4.3 – K populations - Experiment 1 : The signal is a binary square (0 : background, 1 : square) with noise. The scattering network has $M = 2$ layers, $J = 3$ scales and $L = 2$ orientations.

Assumption 2. **Persistence :**
Along a scattering path, high and low scattering coefficient values cascade across the scattering orders.

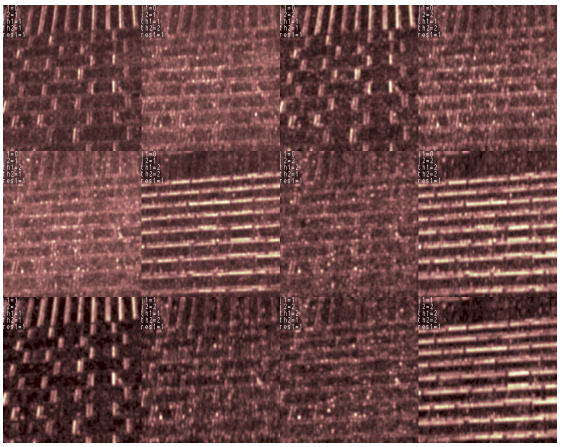
This assumption codifies how the hidden states are structured. Smooth regions/singularities are represented by low/high values at every order. Persistence leads to scattering coefficient values that are statistically dependent along the branches of the scattering tree. Figure 4.5 displays the magnitude of the scattering coefficient for a given node i of the tree against those of its father $\rho(i)$. However one could expect the difference of orientations between the father and the child to have an influence on how strong the correlation between the scattering coefficients is. One could intuit that the closer the orientations the higher the correlation. This intuition can be supported by the difference in between Figure 4.5a and Figure 4.5b. Figure 4.5a displays the correlation between a third order scattering coefficient and its second order father in the case where the child, the father and its father have the same orientation. In this case a high correlation coefficient is observed. Figure 4.5b also displays the correlation between a third order scattering coefficient and its second order father but in the case where the children, the father and its father have different orientations. Not surprisingly, a lower correlation coefficient is observed. Table 4.1 reports the average correlation across all the pairs (father, child) of a SCHMT. Those two experiments tend to confirm the existence of a correlation as well as a

(a) Original

(b) Layer 0



(c) Layer 1



(d) Layer 2

FIGURE 4.4 – K populations - Experiment 2 : The signal is a realisation of the class "brick wall" of the $CUReT$ texture dataset. The scattering network has $M = 2$ layers, $J = 3$ scales and $L = 2$ orientations.

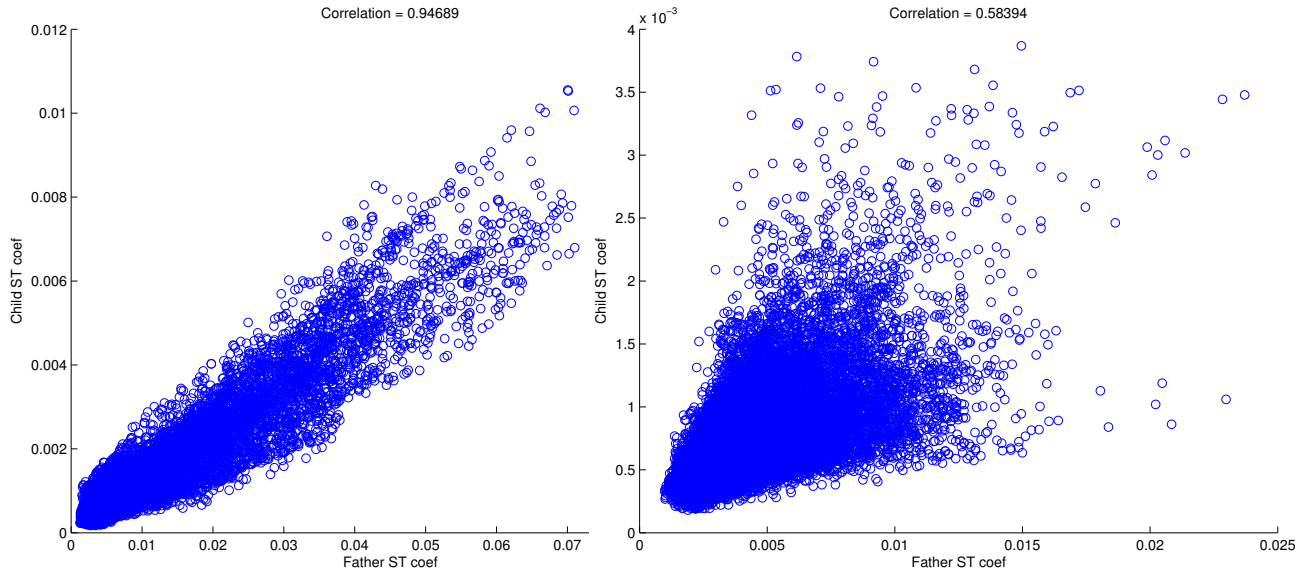
potential dependency over the orientations.

4.3 Learning the tree structure :

As seen in Section 3.2.3, training Markov models can be done using Expectation-Maximization methods. Hidden Markov chains use a version of the EM algorithm called Forward-Backward algorithm allowing the propagation of the hidden states along the chain. Crouse et al. [1998] proposed the Upward-Downward algorithm, an adaptation to the hidden Markov trees of the Forward-Backward algorithm. Both algorithms were suffering from underflowing problems [Ephraim and Merhav, 2002] and Durand et al. [2004] adapted Devijver [1985] smoothing trick to create a smoothed version of the learning algorithm for trees. This section proposes our rewritten version of the smoothed EM algorithm adapted to non-regular non-binary HMTs.

To do so one needs to introduce the following notation :

— $\forall i \in \mathcal{T}$, let n_i be the number of children of the node i.



(a) Same orientation. (b) Different orientations.

FIGURE 4.5 – Persistence - Experiment 1 : Magnitude of the scattering coefficients obtain for the a realization of the class "brick wall" of the *CUReT* texture dataset at a given index i of the tree against those of its father $\rho(i)$. The scattering network has $M = 3$ layers, $J = 4$ scales and $L = 4$ orientations.

— $\forall i \in \mathcal{T}$, let $\bar{\mathcal{S}}_i = \bar{s}_i$ be the observed sub-tree rooted at node i. By convention $\bar{\mathcal{S}}_0$ denotes the entire observed tree.

— $\forall i \in \mathcal{T}$, let $\bar{\mathcal{S}}_{c(i)} = \bar{s}_{c(i)}$ be the entire -possibly empty collection of observed sub-trees rooted at children of node i (i.e. the sub-tree $\bar{s}_i$ except its root s_i).

— If $\bar{\mathcal{S}}_i$, is a sub-tree of $\bar{\mathcal{S}}_j$, then $\bar{\mathcal{S}}_{j\backslash i} = \bar{s}_{j\backslash i}$ is the sub-tree rooted at node j except the sub-tree rooted at node i.

— $\forall i \in \mathcal{T}$ let $\bar{\mathcal{S}}_{0\backslash c(i)} = \bar{s}_{0\backslash c(i)}$ be the entire tree except for the sub-trees rooted at children of node i.

Note that those notations transpose to the hidden state and for instance $\bar{\mathcal{H}}_i = \bar{h}_i$ is the state sub-tree rooted at node i.

4.3.1 E-Step :

The smoothed version of the E-step requires the computation of the conditional probability distributions $\xi_i(k) = P(H_i = k|\bar{\mathcal{S}}_i = \bar{s}_i)$ (smoothed probability) and $P(H_i = k, H_{\rho(i)} = g|\bar{\mathcal{S}}_i = \bar{s}_i)$ for each node $i \in \mathcal{T}$ and states k and g. The smoothed probability adapted to the HMT structure can be decomposed as,

$$\xi_i(k) = \frac{P(\bar{\mathcal{S}}_{0\backslash i} = \bar{s}_{0\backslash i}|H_i = k)}{P(\bar{\mathcal{S}}_{0\backslash i} = \bar{s}_{i\backslash i}|\bar{\mathcal{S}}_1 = \bar{s}_i)} P(H_i = k|\bar{\mathcal{S}}_i = \bar{s}_i) \tag{4.10}$$

The smoothed upward-downward algorithm requires the introduction the following quantities,

Classification results		
Signal :	Correlation mean	Correlation variance
diagonal :	0.909	0.260
Square :	0.811	0.300
Circle :	0.876	0.164
uiuc brick :	0.647	0.241
Mandrill :	0.503	0.255
Lena :	0.727	0.236

TABLE 4.1 – Persistence - Experiment 2 : Average correlation across nodes of the scattering transform applied to different signals. The scattering network has $M = 3$ layers, $J = 4$ scales and $L = 4$ orientations.

$$\beta_i(k) = P(H_i = k|\bar{\mathcal{S}}_i = \bar{s}_i) \tag{4.11}$$

$$\beta_{\rho(i)i}(k) = \frac{P(\bar{\mathcal{S}}_i = \bar{s}_i|H_{\rho(i)} = k)}{P(\bar{\mathcal{S}}_i = \bar{s}_i)} \tag{4.12}$$

$$\alpha_i(k) = \frac{P(\bar{\mathcal{S}}_{0\setminus i} = \bar{s}_{0\setminus i}|H_{\rho(i)} = k)}{P(\bar{\mathcal{S}}_{0\setminus i} = \bar{s}_{0\setminus i}|\bar{\mathcal{S}}_i = \bar{s}_i)} \tag{4.13}$$

The smoothed upward-downward algorithm also requires the preliminary knowledge of the marginal state distributions $P(H_i = k)$ for each node i. However this can simply be achieved by a downward recursion initialized for the root node with $P(H_0 = k) = \pi_0(k)$ and then cascading the information down the tree using the recursive Formula 4.8.

Upward recursion :

The upward algorithm is initialized at all the leaves of the tree, by computing $\beta_i(k)$ using,

$$\begin{aligned}\beta_i(k) &= P(H_i = k|\mathcal{S}_i = s_i) \\ &= \frac{P(S_i = s_i|H_i = k)P(H_i = k)}{P(S_i = s_i)} \\ &= \frac{P_{\theta_{k,i}}(s_i)P(H_i = k)}{N_i},\end{aligned} \tag{4.14}$$

where the normalization factor for the leaves N_i is given by,

$$N_i = P(S_i = s_i) = \sum_{k=1}^{K} P_{\theta_{k,i}}(s_i)P(H_i = k). \tag{4.15}$$

Then one can recursively —upward recursion— compute $\beta_i(k)$ for the remaining noades of the tree using,

$$
\begin{aligned}
\beta_i(k) &= P(H_i = k|\bar{\mathcal{S}}_i = \bar{s}_i) \\
&= \left[\prod_{j\in c(i)} P(\bar{\mathcal{S}}_j = \bar{s}_j|H_i = k)\right] P(S_i = s_i|H_i = k)\frac{P(H_i = k)}{P(\bar{\mathcal{S}}_i = \bar{s}_i)} \\
&= \left[\prod_{j\in c(i)} \frac{P(\bar{\mathcal{S}}_j = \bar{s}_j|H_i = k)}{P(\bar{\mathcal{S}}_j = \bar{s}_j)}\right] P(S_i = s_i|H_i = k)P(H_i = k)\frac{\prod_{j\in c(i)} P(\bar{\mathcal{S}}_j = \bar{s}_j)}{P(\bar{\mathcal{S}}_i = \bar{s}_i)} \\
&= \frac{\left[\prod_{j\in c(i)} \beta_{ij}(k)\right] P_{\theta_{k,i}}(s_i)P(H_i = k)}{N_i},
\end{aligned}
\tag{4.16}
$$

where the normalization factor for the non-leaf nodes N_i is given by,

$$
\begin{aligned}
N_i &= \frac{P(\bar{\mathcal{S}}_i = \bar{s}_i)}{\prod_{j\in c(i)} P(\bar{\mathcal{S}}_j = \bar{s}_j)} \\
&= \sum_{k=1}^{K}\left[\prod_{j\in c(i)} \beta_{ij}(k)\right] P_{\theta_{k,i}}(s_i)P(H_i = k).
\end{aligned}
\tag{4.17}
$$

For all node i, the quantities $\beta_{\rho(i)i}(k)$ can be extracted from β_i using,

$$
\begin{aligned}
\beta_{\rho(i)i}(k) &= \frac{P(\bar{\mathcal{S}}_i = \bar{s}_i|H_{\rho(i)} = k)}{P(\bar{\mathcal{S}}_i = \bar{s}_i)} \\
&= \frac{\sum_{g=1}^{K} P(\bar{\mathcal{S}}_i = \bar{s}_i|H_i = g)P(H_i = g|H_{\rho(i)} = k)}{P(\bar{\mathcal{S}}_i = \bar{s}_i)} \\
&= \sum_{g=1}^{K} \frac{P(H_i = g|\bar{\mathcal{S}}_i = \bar{s}_i)}{P(H_i = g)} P(H_i = g|H_{\rho(i)} = k) \\
&= \sum_{g=1}^{K} \frac{\beta_i(g)\epsilon_i^{(kg)}}{P(H_i = g)}.
\end{aligned}
\tag{4.18}
$$

Those relationships one can derive the the upward Algorithm 1.

Downward recursion :

The downward recursion can either be built on the basis of the quantities $\alpha_i(k)$ or on the basis of the smoothed probabilities $\xi_i(k) = P(H_i = k|\bar{\mathcal{S}}_i = \bar{s}_i)$. The downward recursion on ξ_i is initialized at the root node with,

$$
\xi_0(k) = P(H_0 = k|\bar{\mathcal{S}}_0 = \bar{s}_0) = \beta_0(k). \tag{4.19}
$$

The quantity can then be computed recursively for each node of the tree using,

Meta-parameters :
K
Initialization :
// $P_{\theta_{k,i}}(s_i)$:
for *All the node i of the tree $\mathcal{T}$* **do**
  $P_{\theta_{k,i}}(s_i) = \mathcal{N}(s_i|\mu_{k,i}, \sigma_{k,i})$
end
// Loop over the leaves i of the tree :
for *All the leaf i of the tree $\mathcal{T}$* **do**
  $\beta_i(k) = \frac{P_{\theta_{k,i}}(s_i)P(H_i=k)}{\sum_{g=1}^{K} P_{\theta_{g,i}}(s_i)P(H_i=g)}$
  $\beta_{i,\rho(i)}(k) = \sum_{g=1}^{K} \frac{\beta_i(g)\epsilon_i^{(kg)}}{P(H_i=g)}.P(H_{\rho(i)} = k)$
  $l_i = 0$
end
Induction :
// Bottom-Up loop over the nodes of the tree :
for *All non-leaf node i of the tree $\mathcal{T}$* **do**
  $M_i = \sum_{k=1}^{K} P_{\theta_{k,i}}(s_i) \prod_{j\in c(i)} \frac{\beta_{j,i}(k)}{P(H_i=k)^{n_i-1}}$
  $l_i = \log(M_i) + \sum_{j\in c(i)} l_j$
  $\beta_i(k) = \frac{P_{\theta_{k,i}(s_i)} \prod_{j\in c(i)}(\beta_{j,i}(k))}{P(H_i=k)^{n_i-1}M_i}$
  for *All the children node j of node i* **do**
    $\beta_{i\setminus c(i)}(k) = \frac{\beta_i(k)}{\beta_{i,j}(k)}$
  end
  $\beta_{i,\rho(i)}(k) = \sum_{g=1}^{K} \frac{\beta_i(g)\epsilon_i^{(kg)}}{P(H_i=g)}.P(H_{\rho(i)} = k)$
end

Algorithm 1: Smoothed upward algorithm.

$$
\begin{aligned}
\xi_i(k) &= P(H_i = k|\bar{\mathcal{S}}_0 = \bar{s}_0) \\
&= \sum_{g=1}^{K} \frac{P(H_i = k, H_{\rho(i)} = g, \bar{\mathcal{S}}_0 = \bar{s}_0)}{P(H_{\rho(i)} = g, \bar{\mathcal{S}}_0 = \bar{s}_0)} P(H_{\rho(i)} = g|\bar{\mathcal{S}}_0 = \bar{s}_0) \\
&= P(\bar{\mathcal{S}}_i = \bar{s}_i|H_i = k) \sum_{g=1}^{K} \frac{P(H_i = k|H_{\rho(i)} = g)}{P(\bar{\mathcal{S}}_i = \bar{s}_i|H_{\rho(i)} = g)} P(H_{\rho(i)} = g|\bar{\mathcal{S}}_0 = \bar{s}_0) \\
&= \frac{\beta_i(k)}{P(H_i = k)} \sum_{g=1}^{K} \frac{\epsilon_i^{(gk)} \xi_{\rho(i)}(g)}{\beta_{\rho(i),i}(g)}.
\end{aligned}
\tag{4.20}
$$

Using the fact that for all $i \in \mathcal{T} \quad \xi_i(k) = \beta_i(k)\alpha_i(k)$ and the relationship from Equation 4.20, one can express the downward pass as presented in Algorithm 2.

Meta-parameters :
K
Initialization :
$\alpha_0(k) = 1$
Induction :
// Top-Down loop over the nodes of the tree :
for *All node i of the tree* $\mathcal{T}\backslash\{0\}$ **do**
  $\alpha_i(k) = \frac{1}{P(H_i=k)} \sum_{g=1}^{K} \alpha_{\rho(i)}(g)\epsilon_i^{(gk)}\beta_{\rho(i)\backslash i}(g)P(H_{\rho(i)} = g)$
end

Algorithm 2: Smoothed downward algorithm.

Conditional properties :

To complete the E-step one needs to compute the conditional probabilities for each node. This is done simply by using,

$$
\forall i \in \mathcal{T} \quad P(H_i = k|\bar{\mathcal{S}}_0 = \bar{s}_0) = \alpha_i(k)\beta_i(k), \tag{4.21}
$$

and

$$
\forall i \in \mathcal{T}\backslash\{0\} \quad P(H_{\rho(i)} = g, H_i = k|\bar{\mathcal{S}}_0 = \bar{s}_0) = \frac{\beta_i(k)\epsilon_i^{(gk)}\alpha_{\rho(i)}(g)\beta_{\rho(i)}(g)}{P(H_i = k)\beta_{\rho(i)i}(g)}. \tag{4.22}
$$

4.3.2 M-Step :

The *Maximization* step of the EM algorithm aims to maximize the log-likelihood of the observation with regards to the parameters and then use those pseudo-optimal parameters for the next *Expectation* step. In other words the iteration m of the EM process, the M-step carries out the update,

$$
\Theta^{m+1} = \underset{\Theta}{\operatorname{argmax}} \left(E[\ln f(\mathbf{x}, H|\Theta)|\mathbf{x}, \Theta^l] \right). \tag{4.23}
$$

The Θ maximizing the log-likelihood in Equation 4.23 can be expressed analytically and this yields the Algorithm 3

Meta-parameters :
K,
Distribution family for P_θ ; // Here Gaussian
N ; // Number of observed realizations of the signal
Initialization :
$\pi_0(k) = \frac{1}{N}\sum_{n=1}^{N} P(H_0^n = m|s_0^n, \Theta^l)$
Induction :
// Loop over the nodes of the tree :
for *All node i of the tree* $\mathcal{T}\backslash\{0\}$ **do**
$P(H_i = k) = \frac{1}{N}\sum_{n=1}^{N} P(H_i^n = k|\bar{s}_0^n, \Theta^l)$,
$\epsilon_i^{gk} = \frac{\sum_{n=1}^{N} P(H_i^n=k, H_{\rho(i)}^n=g|\bar{s}_0^n,\Theta^l)}{NP(H_{\rho(i)}=k)}$,
$\mu_{k,i} = \frac{\sum_{n=1}^{N} s_i^n P(H_i^n=k|\bar{s}_0^n,\Theta^l)}{NP(H_i=k)}$,
$\sigma_{k,i}^2 = \frac{\sum_{n=1}^{N}(s_i^n-\mu_{k,i})^2 P(H_i^n=k|\bar{s}_0^n,\Theta^l)}{NP(H_i=k)}$.
end

Algorithm 3: M-step of the EM algorithm.

4.3.3 EM algorithm :

Finally the EM algorithm iterates over the *E-step* and the *M-step* as described by the Algorithm 4.

Meta-parameters :
K ;
Distribution family for P_θ ;
Convergence criteria ; // Iteration limit or information based
Initialization method for Θ ; // Random or prior knowledge
Initialization :
$l = 0$; // Iteration counter
Initialize(Θ)
Iteration :
while *Not convergence* **do**
E-step : Calculate $P(\bar{\mathcal{H}}|\bar{\mathcal{H}}, \Theta^l)$.
M-step : Set $\Theta^{m+1} = \text{argmax}_\Theta \left(E[\ln f(\mathbf{x}, H|\Theta)|\mathbf{x}, \Theta^l]\right)$.
l = l+1
end

Algorithm 4: EM algorithm.

4.4 Classification :

Let Θ_c now be a set of parameters for an SCHMT $\mathcal{T}$ learned using the EM algorithm described in Section 4.3 on a training set $\{\bar{S}_{0,c}^n\}_{n\in[\![1,N]\!]} = \{ST_{(\psi,J,M,L)}(\mathbf{x}_c^n)\}_{n\in[\![1,N]\!]}$ composed of the scattering representation of N realizations of a signal of class c . Let also $\mathbf{x}^{new}$ be another realization of this signal, not used for training and $\mathcal{T}^{new}$ be the instance of the SCHMT generated by this realization.

In this context the MAP algorithm aims at finding the optimal hidden tree $\hat{\hat{h}}_0^{new} = (\hat{h}_0^{new} \dots \hat{h}_{I-1}^{new})$ maximizing the probability of this sequence given the model's parameters $P(\mathcal{H}_0 = \hat{\hat{h}}_0^{new} | \mathcal{T}^{new}, \Theta_c)$. The MAP framework also provides $\hat{P}$ the value of this maximum.

For SCHMT the MAP algorithm has the form described by the Algorithm 5.

Meta-parameters :
K ;
Initialization :
for *all leaf* i *of* $\mathcal{T}$ **do**
  $\gamma_i(k) = \beta_i(k)$; // The gamma for all k must be computed before the next step
  $\gamma_{i,\rho(i)}(k) = \max_{1 \leq g \leq K} \gamma_i(g) \epsilon_i^{kg}$
  $\xi_i(k) = \text{argmax}_{1 \leq g \leq K} \gamma_i(g) \epsilon_i^{kg}$
end
Induction :
// Top-Down loop over the nodes of the tree :
for *All node* i *of the tree* $\mathcal{T} \backslash \{0\}$ **do**
  $\gamma_i(k) = P_{\theta_{k,i}}(s_i) \prod_{j \in c(i)} \gamma_{j,i}(k)$
  $\gamma_{i,\rho(i)}(k) = \max_{1 \leq g \leq K} \gamma_i(g) \epsilon_i^{kg}$; // except at root node
  $\xi_i(k) = \text{argmax}_{1 \leq g \leq K} \gamma_i(g) \epsilon_i^{kg}$
end
Termination :
$\hat{P} = \max_{1 \leq g \leq K} \gamma_0(g)$
$\hat{h}_0 = \text{argmax}_{1 \leq g \leq K} \gamma_0(g)$
Downward tracking :
// Creation of the hidden tree from the root node
for *All node* i *of the tree* $\mathcal{T} \backslash \{0\}$ **do**
  $\hat{h}_i = \xi_i(\hat{h}_{\rho(i)})$
end

Algorithm 5: MAP algorithm.

The MAP Algorithm 5 can be used in a multi-class classification problem by training an SCHMT model per class and then when presented with anew realization $\mathbf{x}^{new}$ comparing the MAP provided by each model as described by Algorithm 6.

Meta-parameters :
K ; C ; // Number of class
for *All classes* c **do**
  $\hat{P}_c = \text{MAP}(\mathbf{x}^{new}, \Theta_c, K)$
end
$\hat{P} = \max_{0 \leq c < C} \hat{P}_c$
$l = \text{argmax}_{0 \leq c < C} \hat{P}_c$

Algorithm 6: MAP algorithm applied to multi-class classification problem.

5 Experimental results :

This section presents some experimental results obtained using scattering convolutional hidden Markov trees for classification tasks. First SCHMT are applied to classify seabed and ripples in sonar imagery. Then this model is adapted to perform a very naive segmentation on a sonar imagery.

5.1 Sonar Imagery :

In underwater mine detection recovering a large proportion of the true positives is crucial since missing one of them could be very costly. However, recovering a large proportion of the true positives may incur many false positives. Reducing these to a manageable number is an open problem in marine sciences. Indeed by its design an underwater mine can be mistaken with some natural features of the seabed. One of those natural features generating many false alarm are called "ripple". Those regular patterns drawn in the sand by currents can vary greatly in shape and orientation. To reduce the number of false positive it is thus interesting to be able to tell apart flat seabed from ripply one. This is the task proposed for the SCHMT.

The data used are extracted from the *UDRC MCM* sonar imagery dataset [Dstl, 2009]. This dataset comprises Synthetic Aperture RADAR (SAS imagery) and hyperspectral data —not used in this experiment. From those very high dimensional images, 100 by 100 patches have been extracted and labelled as either seabed or ripple (see respectively Figure 5.1 and Figure 5.2). The classification task at hand is very challenging due to the low informative content of each images and the high intra-class variability.

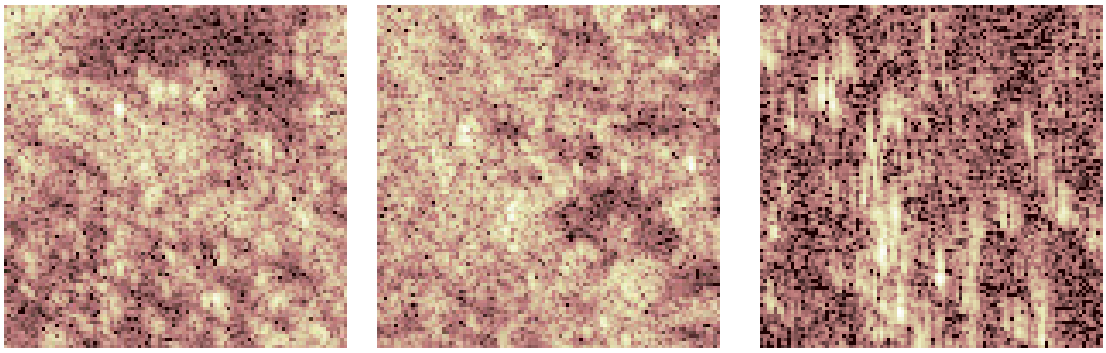


FIGURE 5.1 – Sample of seabed patches.

The scattering transform used has $M = 3$ orders, $J = 5$ scales, $L = 3$ orientations and uses a Morlet wavelet and the hidden Markov tree has $K = 2$ states and is using a mixture of Gaussian to describe the relationship between the scattering coefficients and the hidden states. Two models —one for each class considered— Θ_{ripple} and Θ_{seabed} are trained on 200 realizations of their class signal. The testing is then realized on 80 images —40 of each classes.

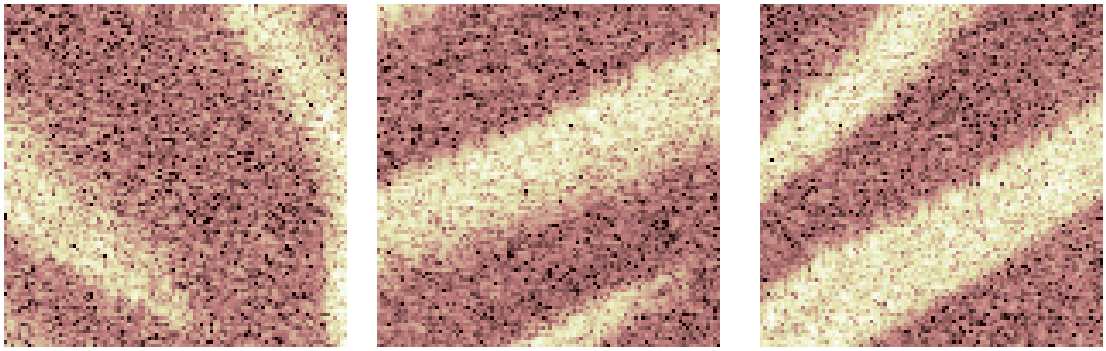


FIGURE 5.2 – Sample of ripple patches.

The performance of the SCHMT are assessed on 100 instances of this experiment and the results are displayed in Table 5.1.

Classification results					
Classification score :	N	Mean	Variance	Maximum	Minimum
Full :	100	0.74	0.101	0.9	0.5
$\geq 60\%$:	91	0.76	0.079	0.9	0.6
$\geq 70\%$:	73	0.79	0.058	0.9	0.7

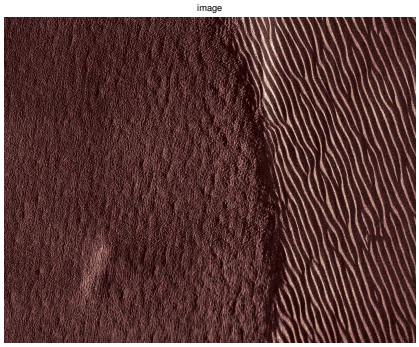
TABLE 5.1 – Classification performance over 100 experiments of Ripple/Seabed classification.

The first line of Table 5.1 displays the results obtained on the all 100 experiments run. Despite a slightly unsatisfying average classification score of 74%, the best models reach a good accuracy of 90%. The lowest score is 50% accuracy and is obtained because all the testing examples are associated with to one class. This can be explained by one class's model having reached a local maxima way less interesting than the other. Those results highly a weakness of the current learning method. At the moment the convergence is not properly tested during the learning phase. However this experiment and its very satisfying best model also validate the assumptions made since when the convergence occurs correctly the discriminative performance are good. The other lines in Table 5.1 simulates the results if a validation criterion based on an imposed accuracy score on a validation set was imposed. This validation set would be a way to address the problem of occasional convergence to a local minima. Another leads would be to work on the EM algorithm itself to make it more robust by adding information based convergence test.

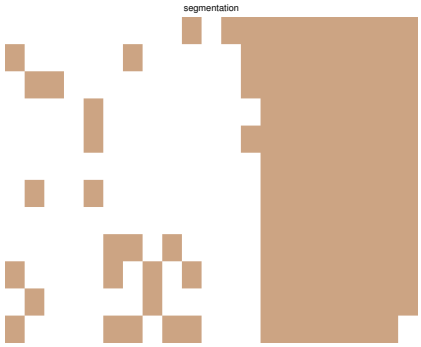
5.2 Segmentation :

On its simplest form a segmentation task can be seen as a set of independent classification tasks on subpart of an image. Hence one can use the models trained in the previous section to realize the segmentation of a full sonar imagery.

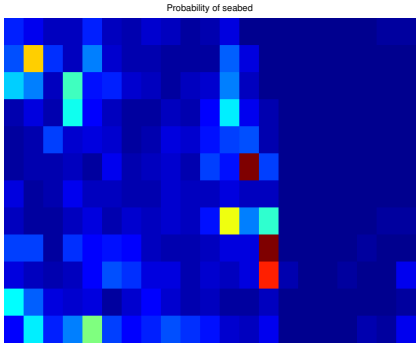
One of the 2000×7300 image from the *UDRC MCM* is cut into a set 100×100 patches —some regions of the original image are not considered. And each of those patches is presented to the classifier. Results of this procedure can be seen in Figure 5.3.



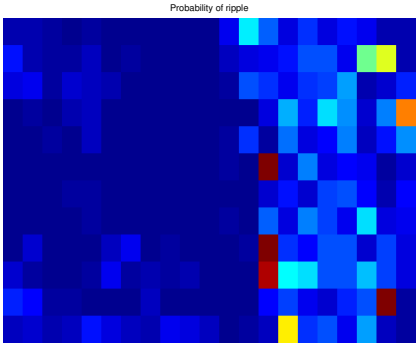
(a) Original image



(b) Segmentation



(c) Probability map for the class "seabed"



(d) Probability map for the class "ripple"

FIGURE 5.3 – Segmentation of a sonar imagery.

Even though this approach to segmentation is very naive and does not introduce any form of spatial smoothing or correlation between nearby patches to improve accuracy, the SCHMT model provides satisfying segmentation of the seabed. Furthermore, as displayed by the Figure 5.3d and 5.3c, it provides a probability map for the confidence in our segmentation decision. Those probability maps are very interesting as they show that the misclassified patches do not have a high probability of belonging to the wrong class. It just appears that given the learned model their probability to belong to their own class is very low.

6 Conclusion :

6.1 Scattering convolutional hidden Markov tree :

This document introduces a framework to process a high-dimensional signal from the raw data to the prediction task. First SCHMTs project the data into a representational space of even higher dimensionality but of reduced volume along the invariants in the data. Then a probabilistic graphical model —hidden Markov tree— is used to create a generative model of the distribution of the data in this space. This model can then be used for inference tasks such as prediction.

SCHMT appears to be a powerful tool combining the interesting properties of the scattering transform for signal representation and the representational power of the hidden Markov models. As seen in Section 5, the modelled distribution can be used to perform efficient discrimination/classification tasks on images with low informative content. Even though this document considers only classification, a generative model is much more versatile than a simple —yet efficient— discriminative counterpart. Because they model the full distribution of the data they can express more complex relationships between the observed and the unknown variables than simple discrimination (see Sub-section 3.1.2 for more details).

6.2 Future work :

Despite showing good performance when the learning phase has converged properly, SCHMT's performance is still undermined by occasional convergence issues which have drastic effects on learning quality. This observation is one of the main drivers for the upcoming work. The other main focuses are to keep a well define probabilistic framework in order to be able to express the uncertainty of our model and of our predictions.

The version of the EM algorithm considered in this report uses full Bayesian inference. However this methods is computationally expensive and sometimes yields a poor learning. Furthermore this version of the EM algorithm provides only a point-wise estimate of the model's parameters. It would be interesting to apply variational methods the this problem [Wainwright and Jordan, 2008]. Beside a potential improvement of the learning /performance, variational inference would also provide an estimated distribution for the model's parameters, allowing us to have access to a measure of uncertainty for the learned model and thus discard models according to the uncertainty on their parameters.

Another interesting direction to follow for future works is to integrate the scattering convolutional hidden Markov tree into a hierarchical graphical models [Fine et al., 1998]. This framework would allow the use the SCHMT model as a node of a wider probabilistic graphical model. Using such an architecture yields a tremendous number of possibilities. The performance in the segmentation task could be improved by adding a layer of graphical model encoding the spatial dependencies between the different labels in the scene. One could also use

hierarchical models to describe a network of sensors each providing information on a targeted scene or integrate multiple source of information into the final prediction.

Finally another interesting lead would be to consider other architectures for the graphical model and even make it includes the representation learning step. This would be possible using Bayesian neural networks and probabilistic back-propagation [Hernández-Lobato and Adams, 2015] or a Bayesian flavour of back-propagation [Blundell et al., 2015].

7 Acknowledgements :

We would like to thanks Dstl and UCL for their support on this project. Jean-Baptiste Regli is funded by a Dstl/UCL Impact studentship. Members from DSTL also provided useful insights on the weakness of the actual classification and segmentation tools as well as a fresh look to the work done.

Bibliography :

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., and Penn, G. (2012). Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4277–4280. IEEE.

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines*. *Cognitive science*, 9(1) :147–169.

Andén, J. and Mallat, S. (2011). Multiscale scattering for audio classification. In *ISMIR*, pages 657–662.

Baker, M. J., Trevisan, J., Bassan, P., Bhargava, R., Butler, H. J., Dorling, K. M., Fielden, P. R., Fogarty, S. W., Fullwood, N. J., Heys, K. A., et al. (2014). Using fourier transform ir spectroscopy to analyze biological materials. *Nature protocols*, 9(8) :1771–1791.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, pages 164–171.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful ? In *Database Theory—ICDT'99*, pages 217–235. Springer.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv :1505.05424*.

Bracewell, R. (1965). The fourier transform and iis applications. *New York*.

Brailean, J. C. and Katsaggelos, A. K. (1996). Recursive map displacement field estimation and its applications. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 917–920. IEEE.

Bruna, J. (2012). Operators commuting with diffeomorphisms.

Bruna, J. and Mallat, S. (2010). Classification with scattering operators. *arXiv preprint arXiv :1011.3023*.

Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8) :1872–1886.

Chudacek, V., Talmon, R., Anden, J., Mallat, S., Coifman, R., Abry, P., and Doret, M. (2014). Low dimensional manifold embedding for scattering coefficients of intrapartum fetale heart rate variability. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 6373–6376. IEEE.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1) :21–27.

Crouse, M. S., Nowak, R. D., and Baraniuk, R. G. (1998). Wavelet-based statistical signal processing using hidden markov models. *Signal Processing, IEEE Transactions on*, 46(4) :886–902.

De Chazal, P., Celler, B., and Reilly, R. (2000). Using wavelet coefficients for the classification of the electrocardiogram. In *Engineering in Medicine and Biology Society, 2000. Proceedings of the 22nd Annual International Conference of the IEEE*, volume 1, pages 64–67. IEEE.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Devijver, P. A. (1985). Baum's forward-backward algorithm revisited. *Pattern Recognition Letters*, 3(6) :369–373.

DeVore, R. A., Jawerth, B., and Lucier, B. J. (1992). Image compression through wavelet transform coding. *Information Theory, IEEE Transactions on*, 38(2) :719–746.

Donoho, D. L. (1993). Unconditional bases are optimal bases for data compression and for statistical estimation. *Applied and computational harmonic analysis*, 1(1) :100–115.

Dstl (2009). Dstl datasets. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/326061/DefenceReporter_Winter2009.pdf`. Accessed : 04-11-2015.

Duncan, T. E., Hu, Y., and Pasik-Duncan, B. (2000). Stochastic calculus for fractional brownian motion i. theory. *SIAM Journal on Control and Optimization*, 38(2) :582–612.

Durand, J.-B., Goncalves, P., and Guédon, Y. (2004). Computational methods for hidden markov tree models-an application to wavelet trees. *Signal Processing, IEEE Transactions on*, 52(9) :2551–2560.

Ephraim, Y. and Merhav, N. (2002). Hidden markov processes. *Information Theory, IEEE Transactions on*, 48(6) :1518–1569.

Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden markov model : Analysis and applications. *Machine learning*, 32(1) :41–62.

Forney Jr, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3) :268–278.

Gilks, W. R. (2005). *Markov chain monte carlo*. Wiley Online Library.

Haveliwala, T. and Kamvar, S. (2003). The second eigenvalue of the google matrix. *Stanford University Technical Report*.

Heckerman, D. (1998). *A tutorial on learning with Bayesian networks*. Springer.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. *arXiv preprint arXiv :1502.05336*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv :1207.0580*.

Hörmander, L. (1971). Fourier integral operators. i. *Acta mathematica*, 127(1) :79–183.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition ? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE.

Jordan, A. (2002). On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14 :841.

Kemeny, J. G. and Snell, J. L. (1960). *Finite markov chains*, volume 356. van Nostrand Princeton, NJ.

Kingsbury, N. (2001). Complex wavelets for shift invariant analysis and filtering of signals. *Applied and computational harmonic analysis*, 10(3) :234–253.

Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T.-W., and Sejnowski, T. J. (2003). Dictionary learning algorithms for sparse representation. *Neural computation*, 15(2) :349–396.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y. (2015). Personal webpage : State of the art on mnist. `http://yann.lecun.com/exdb/mnist/`. Accessed : 04-11-2015.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10).

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE.

Lee, N., Huynh, Q., and Schwartz, S. (1996). New method of linear time-frequency analysis for signal detection. In *Time-Frequency and Time-Scale Analysis, 1996., Proceedings of the IEEE-SP International Symposium on*, pages 13–16. IEEE.

Lin, T. and Zha, H. (2008). Riemannian manifold learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5) :796–809.

Lohmiller, W. and Slotine, J.-J. E. (1998). On contraction analysis for non-linear systems. *Automatica*, 34(6) :683–696.

Mallat, S. (1999). *A wavelet tour of signal processing*. Academic press.

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

Margaritis, D. (2003). *Learning Bayesian network model structure from data*. PhD thesis, US Army.

Nelson, J. and Kingsbury, N. (2010). Fractal dimension based sand ripple suppression for mine hunting with sidescan sonar. In *International conference on synthetic aperture sonar and synthetic aperture radar*.

Nilsson, N. J. (1998). *Artificial intelligence : a new synthesis*. Morgan Kaufmann.

Oyallon, E. and Mallat, S. (2014). Deep roto-translation scattering for object classification. *arXiv preprint arXiv :1412.8659*.

Oyallon, E., Mallat, S., and Sifre, L. (2013). Generic deep networks with wavelet scattering. *arXiv preprint arXiv :1312.5940*.

Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3) :61–74.

Rish, I., Brodie, M., and Ma, S. (2002). Efficient fault diagnosis using probing. In *AAAI Spring Symposium on Information Refinement and Revision for Decision Making*.

Schölkopf, B., Sung, K.-K., Burges, C. J., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. (1997). Comparing support vector machines with gaussian kernels to radial basis function classifiers. *Signal Processing, IEEE Transactions on*, 45(11) :2758–2765.

Sifre, L. and Mallat, S. (2013). Rotation, scaling and deformation invariant scattering for texture discrimination.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE.

Smolensky, P. (1986). Information processing in dynamical systems : Foundations of harmony theory.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*.

Ulusoy, I. and Bishop, C. M. (2005). Generative versus discriminative methods for object recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 258–265. IEEE.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2) :260–269.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2) :1–305.

Waldspurger, I., d'Aspremont, A., and Mallat, S. (2015). Phase recovery, maxcut and complex semidefinite programming. *Mathematical Programming*, 149(1-2) :47–81.

Zhang, Z., Wang, J., and Zha, H. (2012). Adaptive manifold learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(2) :253–265.