Learning Separable Filters with Shared Parts

Bugra Tekin

Master of Science

School of Electrical Engineering

Supervised by

Prof. Pascal Fua
Dr. Vincent Lepetit

Computer Vision Laboratory (CVLAB)

Ecole Polytechnique Fédérale de Lausanne

June 21, 2013

Abstract

Learned image features can provide great accuracy in many Computer Vision tasks. However, when the convolution filters used to learn image features are numerous and not separable, feature extraction becomes computationally demanding and impractical to use in real-world situations. In this thesis work, a method for learning a small number of separable filters to approximate an arbitrary non-separable filter bank is developed. In this approach, separable filters are learned by grouping the arbitrary filters into a tensor and optimizing a tensor decomposition problem. The separable filter learning with tensor decomposition is general and can be applied to generic filter banks to reduce the computational burden of convolutions without a loss in performance. Moreover, the proposed approach is orders of magnitude faster than the approach of a very recent paper based on ℓ_1-norm minimization [34].

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Pascal Fua, for providing me the opportunity to work in this pleasant research environment. I would like to thank Dr. Vincent Lepetit for his supervision, useful suggestions and valuable comments throughout this research. Same way, I would like to thank Amos Sironi for his constant support on this research. With all the great ideas, and unceasing encouragement this project turned into a major learning experience. I would like to also thank all CVLAB family for the enjoyable laboratory environment.

I would like to express my appreciation to Prof. Michael Unser for the opportunities I have been granted to work in his lab during my time at EPFL. It is my great honor to have been supervised by you. Your personality and enthusiasm are a great source of inspiration for me.

I owe my deepest gratitude to all my beloved friends, too numerous to name here, whose support and appreciation make me feel valued and happier at all moments.

Last and foremost, I would like to thank my family, to whom this thesis is dedicated, for their unconditional love, understanding, and support during all times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few years, representing images as sparse linear combinations of learned filters has been proved to be very effective in numerous image processing tasks, such as feature extraction, object recognition and image denoising. Using learned filters is particularly useful where our lack of intuition makes it difficult to engineer good hand-crafted feature extractors.

Most common methods for object recognition utilize hand-designed features, such as SIFT [24] and SURF [3]. A more general and flexible strategy in order to find a concise and meaningful representation would be to use an unsupervised learning method that automatically incorporates all available information of the input data.

In order to learn filters to extract good visual features in an unsupervised manner, there has been an increasing amount of research in sparse coding. The learning algorithms that use sparse coding are commonly trained at the patch level. Patch-based training mostly produces oriented edge detectors, whereas convolutional training produces highly diverse filters such as center-surround filters, corner detectors, cross detectors, and oriented grating detectors [17]. This is important in the sense that the visual world can be represented at many levels such as pixel intensities, edges, shapes or objects, and these convolutional filters are able to represent these diverse modalities in the image. With the convolutional formulation, some of the filters could detect objects by finding low-level features indicative of object parts, whereas more complex information in the image could be resolved by some other sophisticated filters. Moreover, the adoption of convolutional approaches could prove useful in terms of efficiency as using small patches on large images is slow and difficult.

A major drawback of many feature learning systems is their complexity and expense as the filters are numerous and non-separable. In the case of 3D image stacks that are used for biomedical purposes, the computational cost is even more pronounced. The computational complexity associated with convolving non-separable filter sets with 2D images and 3D volumes can be greatly reduced by using separable filters. Separable filters are favored in the sense that a substantial amount of speed-up with no loss of accuracy is obtained when convolving the filters with images.

Separable filters can be used to decompose convolution operations into separate one-dimensional convolutions. Because of its computational efficiency, hand-designed filter banks are often made separable. However, the problem of learning generic separable filters has not been addressed until recently in computer vision literature.

A recent study [34] has investigated two separate schemes to learn separable filters. The first one involves directly learning a separable filter bank by enforcing the separability constraint to the convolutional filter learning framework. The second method starts directly from a non-separable filter bank and approximates them by linear combinations of a small set of separable filters. The optimization problem used to learn separable filters for both schemes relies on the minimization of the nuclear norm of the filters (a convex relaxation of the rank). If the rank of the filters could be made equal to one, then these filters could be represented as separable filters. In these approaches, approximation of a set of separable filters is carried out by forcing each filter to be low-rank independently. A different approach for learning separable filters could be constraining all these learned non-separable filters globally to be the linear combination of a small set of separable filters, where

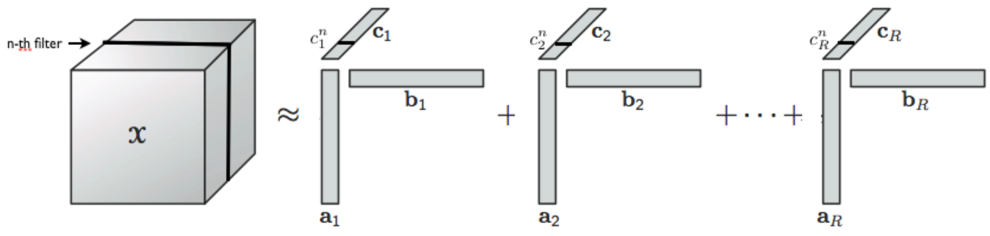

Figure 1.1: Tensor decomposition of a 3-way tensor into a sum of R rank-one tensors. The n-th filter (n-th slice of the tensor) can be expressed as the linear combination of the outer products of vectors $\mathbf{a_i}$ and $\mathbf{b_i}$, where weighting coefficients are c_i^n.

separable filters are written explicitly as the products of one-dimensional filters. This is the problem addressed in this thesis work where a generic framework for learning a separable filter basis is implemented.

It has been shown that the accuracy of the image processing tasks carried out using non-separable filters can be matched using separable filters at a much smaller computational cost [34]. A global optimization approach to learn separable filters would prove useful in terms of computational complexity. The key idea is to group all non-separable filters of the filter bank into a multidimensional array, which is also called a *tensor* and enforce this tensor to be the linear combination of one-rank tensors. An important novelty of the approach is that the optimization problem of learning separable filters is carried out globally for the all filters, at once, instead of doing so for each filter separately. *Tensor decomposition* is the tool that is useful for our aim. An N-dimensional tensor can be factorized into a sum of rank-one tensors, i.e. a tensor that can be represented as the outer product of N vectors. With this kind of decomposition, a tensor can be seen as the weighted sum of separable components. If we stack a 2D non-separable filter bank into a 3-way tensor as illustrated in Fig. 1.1 and decompose it as the sum of rank-one tensors, the n-th slice of the tensor corresponding to the n-th filter can be represented by the outer products of vectors a_i and b_i, which is a separable matrix, weighted by the n-th elements of vectors c_i, c_i^n.

1.1 Thesis Outline

- Chapter 2 gives a background on topics considered in this work, including feature extraction and separable filter learning.
- The theoretical framework of the main concepts of separable filter learning and tensor decomposition is given in Chapter 3 and Chapter 4.
- A new method to learn separable filters using tensor decomposition is presented in Chapter 5.
- The performance comparisons of different separable filter learning algorithms on different computer vision tasks are provided in Chapter 6.
- Finally, conclusions of the thesis work is presented in Chapter 7.

Chapter 2

Related Work

Over a long period of time, a growing amount of research on visual recognition has focused on automatic feature learning. Several techniques, such as Neural Networks [22], Linear Discriminant Analysis [5], Restricted Boltzman Machines [15], Autoencoders [4] have been utilized to learn features in supervised or unsupervised ways.

In recent years, creating overcomplete dictionary of features and coding this dictionary with a sparse set of coefficients has emerged as a useful tool in object recognition [34], image denoising [37] and beyond. Enforcing sparsity constraints has been a popular approach in many image processing and computer vision tasks. This is due to the fact that receptive fields observed in V1, the first layer of the visual cortex in the mammal brain, produces sparse distribution of output activity in the brain in response to natural images, and algorithms based on sparsity constraints on the output activity can produce linear filters similar to these receptive fields [28], [32]. Consequently, it has been assumed that sparse coding algorithms could extract relevant features for image classification [31], [16], [41]. Input data distribution can be expressed with sparse coding assuming a sparse output prior. Deep Belief Networks (DBNs) are one such example in which sparsity constraint on the coefficients yielded convergence on natural images via filters that are similar to the receptive fields in V1. In another study [33], sparsity is shown to be important when learning the image filters that are used in convolutional sparse coding, however it is not necessarily required for classification.

For feature extraction tasks, runtime can be very long as it involves convolving the image with many non-separable filters. Separable filters would speed up the process of extracting features. Using separable filters, convolution can be split into more than one stages of one-dimensional convolutions which is advantageous in terms of runtime efficiency. It was first proposed in [38] that the convolution operations can be split into convergent sums of matrix valued stages. The method proposed in this study is used in [29] to avoid coarse discretizations of the scale and orientation spaces, which, in the end, gives steerable separable 2D edge-detection kernels. These separability approaches are limited in the sense that they are decomposable only in the suggested manner and they do not yield filters that can be found in a learned dictionary or handcrafted to suit particular needs. Separability property has long been neglected until recently [25], [30]. Again, the scope of the filters obtained with these approaches is restricted to particular frameworks, while a learning approach is a more generic and flexible approach.

Runtime efficiency issue of feature extraction is also addressed in frameworks that use parallel capabilities of modern hardware [10], [27]. One way to use parallel processing is FPGA, but programming in FPGA is cumbersome [10]. On the other hand, exploiting Graphics Processing Unit can be another attractive option [27], however, in this case, the time required for memory transfers between CPU and GPU is too long to be used effectively in practical applications.

In order to reduce computational complexity of feature extraction, one recent attempt is to learn a filter bank by composing a few atoms from a handcrafted separable dictionary [35]. A better way to learn the filter bank would be to learn also these atoms which is the problem addressed in [34]. This yields a smaller number of separable filters appropriate for the feature extraction task.

Chapter 3

Learning Separable Filters

In this chapter, a mathematical background for learning non-separable and separable filters is going to be provided. Besides, patch-based and convolutional sparse coding methods will be reviewed.

3.1 Learning Arbitrary Filters

In order to extract features out of an image, filters which are either learned or handcrafted can be used. These features are either obtained from a convolution between the image and the filters, or from a sparse optimization procedure. Olshausen and Field's algorithm (OLS) which is known to converge well on natural image patches can be used to learn filters.

Olshausen and Field proposed in their study [28], that the first layer of the visual cortex V1 produces a sparse representation of the images. With this premise, they formulated the reconstruction of the images from features as follows

$$\min_{M,\{\mathbf{t_i}\}} \sum_i ||\mathbf{t_i}||_0 \ \ s.t. \ \sum_i ||\mathbf{x_i} - \mathbf{M}\mathbf{t}_i||_2^2 = 0 \tag{3.1}$$

where $\mathbf{x_i}$ are training images, $\mathbf{t_i}$ are the corresponding feature vectors. $\mathbf{M}$ is a matrix whose columns form the dictionary, and the ℓ_0-norm, the number of non-zero elements, is the best sparsity measure available. ℓ_0-norm yields a non-convex optimization problem. In order to have a convex optimization problem which produces sparsity, ℓ_1-norm can be used. Then, the dictionary of filters can be learned by minimizing the following objective function:

$$\min_{M,\{\mathbf{t_i}\}} \sum_i ||\mathbf{x_i} - \mathbf{M}\mathbf{t}_i||_2^2 + \lambda_{learn} ||\mathbf{t_i}||_1 \tag{3.2}$$

where ℓ_1-norm enforces sparsity on the feature vectors.

In Eq. (3.2), the images $\mathbf{x_i}$ are reconstructed from a few columns of $\mathbf{M}$ since a sparsity constraint on $\mathbf{t_i}$ is enforced by the last term. λ_{learn} is a regularization parameter that determines the relative importance of the reconstruction error, $||\mathbf{x_i} - \mathbf{M}\mathbf{t}_i||_2^2$ with respect to the regularization term $||\mathbf{t_i}||_1$. $\mathbf{M}$ has more columns than rows, thus it is overcomplete and it gives us the freedom to choose among all possible representations, a sparse one.

Using Eq. (3.2) for large images is slow and difficult; it is appropriate only for small image patches as many coefficients in $\mathbf{M}$ should have to be optimized simultaneously. A convolutional approach used in [42] and [23] where the matrix vector product is replaced by convolution would prove more useful for large images. The optimization problem in Eq. (3.2) then takes the form:

$$\min_{M,\{\mathbf{t_i}\}} \sum_i \left(\left|\left| \mathbf{x_i} - \sum_{j=1}^{N} \mathbf{f^j} * \mathbf{t_i^j} \right|\right|_2^2 + \lambda_{learn} ||\mathbf{t_i}||_1 \right) \tag{3.3}$$

where $\mathbf{f}^j$'s are linear filters and $*$ denotes the convolution operator. The $\mathbf{t_i^j}$'s can now be seen as a set of images whose sizes are equal to that of $\mathbf{x_i}$ images. Similar intermediate representations have been used in Convolutional Neural Networks literature and have been called feature maps. The optimization problem posed in Eq. (3.3) can now be solved via alternatively optimizing $\mathbf{f^j}$ and $\mathbf{t_i^j}$. The optimization for coefficients $\mathbf{t_i^j}$ is carried out by stochastic gradient descent with clipping. For $\mathbf{f^j}$, also stochastic gradient descent optimization is applied.

3.2 Learning Separable Filters

In [33], it has been shown that the features computed with direct convolution of images with the learned filters yields competitive recognition rates while reducing complexity compared to the case where a sparse optimization procedure is applied to extract features. The computational complexity is high when dealing with large amount of data since the resulting convolution filters obtained with Eq. (3.3) are not separable. If we assume $\mathbf{x_i} \in \mathbb{R}^{\mathbf{pxq}}$ and $\mathbf{f_i^j} \in \mathbb{R}^{\mathbf{sxt}}$, in order to extract the features, the non-separable convolution requires $\mathcal{O}(\mathbf{p}.\mathbf{q}.\mathbf{s}.\mathbf{t})$ operations, whereas a separable convolution requires $\mathcal{O}(\mathbf{p}.\mathbf{q}.(\mathbf{s}+\mathbf{t}))$ operations.

If we could learn separable filters in tasks that use convolutions with filter banks, the computational complexity would be much less. To handle this problem, following two approaches have been used in [34]. The first one directly forces the filters to be low-rank by adding one more constraint (minimization of the nuclear norm) to the optimization problem given in Eq. (3.3). This approach has been found to degrade the classification performance because of the additional constraints applied on the filters. The second approach yields a faster and better solution to separable filter learning problem. The arbitrary non-separable filters are replaced with linear combinations of filters that are forced to be separable by lowering their rank. This approach is also justified by [29] which suggests that arbitrary filters of rank R can be expressed as linear combinations of R separable filters. This solution preserves the discriminative power of the non-separable filter bank while providing a more generic framework for learning separable filters.

3.2.1 Forcing the Learned Filters to Be Low-Rank

In order to learn separable filters, one straight approach would be to add one more constraint to the optimization problem posed in Eq. (3.3) that enforces the filters to be separable by minimizing their nuclear norm:

$$\underset{\{\mathbf{s^j}\},\{\mathbf{t_i^j}\}}{\operatorname{argmin}} \sum_i \left(\left|\left| \mathbf{x_i} - \sum_{j=1}^{N} \mathbf{s^j} * \mathbf{t_i^j} \right|\right|_2^2 + \mathbf{\Gamma^i_{m,s}} \right), \tag{3.4}$$

$$\text{with } \mathbf{\Gamma^i_{t,s}} = \lambda_1 \sum_{j=1}^{N} \left|\left| \mathbf{t_i^j} \right|\right|_1 + \lambda_* \sum_{j=1}^{N} ||\mathbf{s^j}||_* \tag{3.5}$$

In this formulation, $\mathbf{s^j}$'s are learned linear filters, $||.||_*$ is the nuclear norm, and λ_* is an additional regularization parameter. Nuclear norm of the matrix is defined as the sum of the matrix singular values obtained from singular value decomposition. It is a convex relaxation of the rank [11]. Thus, minimizing the nuclear norm is equivalent to lowering the rank of the matrix. It has been suggested in [34] that for high values of λ_*, the $\mathbf{s^j}$ filters become effectively rank-one, therefore it can be written as products of one-dimensional filters, i.e. they become separable filters.

The solution to the optimization problem posed above does not differ substantially from the solution of the optimization problem of Eq. (3.3). The only extra effort is applying a soft-thresholding operation to the singular values of the filter matrices (proximal operator of the nuclear norm) after stochastic gradient descent procedure on the filters, $\mathbf{s^j}$. To make sure that separable filters are obtained, all singular values are set to 0 except the largest one, at convergence. In practice, the second largest singular value is already very close to 0. One drawback of this optimization procedure is that choosing suitable parameters for λ_1 and λ_* is difficult. Actually, because of these additional constraints the performance of the optimization is shown to be degraded in [34].

3.2.2 Approximating a Non-separable Filter Bank as Linear Combinations of Separable Filters

In this method, the learned arbitrary non-separable filters from Eq. (3.3) are written as linear combinations of M separable filters $\{\mathbf{s}_k\}_{1\leq k \leq M}$, i.e. a set of weights w_k^j are found such that each arbitrary non-separable filter $\mathbf{f^j}$ is equal to $\sum_{k=1}^{M} w_k^j \mathbf{s_k}$. Consequently, convolving the image with $\mathbf{f^j}$'s amounts to convolving the image with separable filters $\mathbf{s_k}$'s and combining these convolution results with the obtained set of weights, w_k^j. Thus, learning of arbitrary non-separable filters is decoupled from learning of separable filters in this approach. The approximation of non-separable filters with linear combinations of separable filters is carried out by optimizing the following objective function:

$$\underset{\{\mathbf{s_k}\},\{w_k^j\}}{\operatorname{argmin}} \sum_i \left|\left|\mathbf{x_i} - \sum_{k=1}^{M} w_k^j \mathbf{s_k}\right|\right|_2^2 + \lambda_* \sum_{k=1}^{M} ||\mathbf{s_k}||_* \tag{3.6}$$

This may seem suboptimal as the learning procedure is decoupled into two steps, however, in practice, it gives superior results as the optimization procedure is split into two simpler tasks that are easier to schedule.

3.2.3 Approximating a 3D Non-separable Filter Bank as Linear Combinations of Separable Filters

The computational complexity of feature extraction becomes more pronounced when it is carried out on three-dimensional image stacks. The formalism to extract features for two-dimensional images can be adapted to three-dimensional volumes. In the previous case, minimizing the nuclear norm was handled by soft-thresholding the singular values obtained from singular value decomposition (SVD). In order to decompose a three-dimensional array, there exist decomposition methods [19] such as Canonical Polyadic Decomposition (CPD). In this method, an N-dimensional, R-rank tensor can be factorized into a sum of R rank-one tensors (a tensor that can be represented as the outer product of N vectors).

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{3.7}$$

It is not possible to infer R from the the tensor, therefore a suitable R is given as a parameter to the decomposition scheme. The framework for the optimization scheme applied to learn two-dimensional filters remains the same in this case, the only difference is that soft-thresholding is applied on the decomposition coefficients, λ_r, not the singular value coefficients.

Chapter 4

Tensor Decomposition

The tensor decomposition [7], [14] is a higher-order generalization of the matrix singular value decomposition (SVD). It has been proven to be useful in many applications such as signal processing, neuroscience and web analysis [2], [19].

In this chapter, firstly, a review of the concept of tensor will be presented. Afterwards, the mathematical background of the tensor decomposition will be provided.

4.1 Overview of Tensors

Tensor is defined as a multidimensional array. The *order* of the tensor is the number of dimensions, also known as ways or modes. For instance, vectors are order-one tensors and matrices are order-two tensors. Tensors are mathematical tools to manipulate multidimensional arrays.

The notational convention of [19] is used throughout the thesis. Vectors are denoted by boldface lowercase letters, e.g $\mathbf{a}$. Matrices are denoted by boldface capital letters, e.g. $\mathbf{A}$. Higher-order tensors are denoted by Euler script letters, e.g., $\mathcal{X}$. Scalars are denoted by lowercase letters, e.g., a. The ith entry of a vector $\mathbf{a}$ is denoted by a_i, element (i, j) of a matrix $\mathbf{A}$ is denoted by a_{ij}, and element (i, j, k) of a third-order tensor $\mathcal{X}$ is denoted by x_{ijk}. The

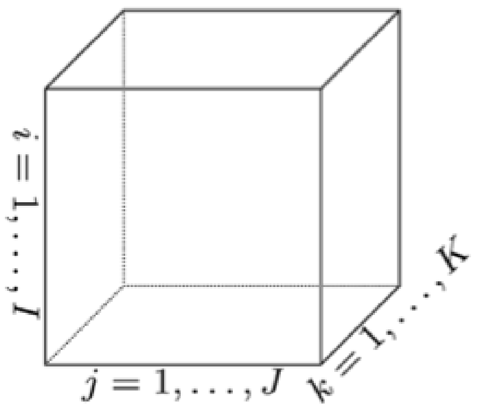

Figure 4.1: A third order tensor.

jth column of a matrix $\mathbf{A}$ is denoted by $\mathbf{a}_j$.

An overview of the mathematical background that is going to be used for tensor decomposition operations is presented in the following. The reader familiar with tensor manipulations could skip to the next section.

The *inner product* of two same-sized tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}$ is the sum of the products of their corresponding entries and it is given mathematically as follows:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=2}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N} \tag{4.1}$$

The *norm* of the tensor is defined as

$$||\mathcal{X}||= \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle} \tag{4.2}$$

An N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is a *rank-one tensor* if it can be written as the outer product of N vectors as in the following:

$$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \ldots \circ \mathbf{a}^{(N)} \tag{4.3}$$

where '$\circ$' represents the vector outer product. This operation means that each element of the tensor is the multiplication of corresponding vector elements.

$$x_{i_1 i_2 \dots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \dots a^{(N)}_{i_N} \text{ for all } 1 \le i_n \le I_n \tag{4.4}$$

The *Khatri-Rao product* is defined as the 'matching columnwise' Kronecker product, where Kronecker product of two vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$ is defined as follows:

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1\mathbf{b} \\ a_2\mathbf{b} \\ \vdots \\ a_I\mathbf{b} \end{bmatrix} \tag{4.5}$$

The Khatri-Rao product of two matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ is denoted by $\mathbf{A} \odot \mathbf{B}$. It results in a $(IJ) \times K$ matrix and is computed as in the following:

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \;\; \mathbf{a}_2 \otimes \mathbf{b}_2 \;\; \dots \;\; \mathbf{a}_N \otimes \mathbf{b}_N] \tag{4.6}$$

The Khatri-Rao product has the following property [6]:

$$(\mathbf{A} \odot \mathbf{B})^T (\mathbf{A} \odot \mathbf{B}) = \mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B} \tag{4.7}$$

in which, $*$ is used to represent the element-wise product. Moreover, the pseudo-inverse of the Khatri-Rao product can be written as [6]:

$$(\mathbf{A} \odot \mathbf{B})^\dagger = ((\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}))^\dagger (\mathbf{A} \odot \mathbf{B})^T \tag{4.8}$$

where $\mathbf{A}^\dagger$ is used to represent Moore-Penrose pseudo-inverse of $\mathbf{A}$ [12]. Knowing that pseudo-inverse of the transpose is the transpose of the pseudo-inverse,

$$((\mathbf{A} \odot \mathbf{B})^T)^\dagger = (\mathbf{A} \odot \mathbf{B})((\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}))^\dagger \tag{4.9}$$

The process of reordering the elements of an N-way tensor into a matrix is called *matricization*, or *unfolding*. The mode-n matricization of the tensor

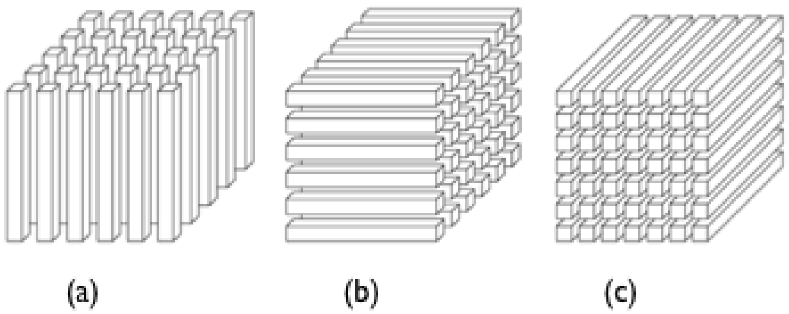

Figure 4.2: Mode-n fibers of a tensor. (a) Mode-1 (column) fibers, (b) Mode-2 (row) fibers, (c) Mode-3 (tube) fibers

$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is represented with $\mathbf{X}_{(n)}$ and arranges the mode-n one-dimensional fibers to be the columns of the resulting matrix. Mode-n fibers of a three-dimensional tensor are illustrated in Fig. 4.2.

The *n-mode (vector)* product of a tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a vector $\mathbf{v}^{(n)} \in \mathbb{R}^{I_n}$ is represented with $\mathcal{X} \times_n \mathbf{v}$. The result is of order $N-1$ and the size is $I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N$. The *n-mode* product can be represented element-wise as in the following:

$$(\mathcal{X} \times_n \mathbf{v})_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} v_{i_n} \tag{4.10}$$

Tensors can also be multiplied by multiple vectors all at once. For instance, for $\mathbf{v}^{(n)} \in \mathbb{R}^{I_n}$, $n = 1, \ldots, N$, a new notation to represent multiplication in multiple modes is introduced in Eq. (4.11). Multiplication in all modes results in a scalar.

$$\mathcal{X} \bigtimes_{n=1}^{N} \mathbf{v}^{(n)} = \mathcal{X} \times_1 \mathbf{v}^{(1)} \times_2 \mathbf{v}^{(2)} \ldots \times_N \mathbf{v}^{(N)} \tag{4.11}$$

$$= \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N} v_{i_1}^{(1)} v_{i_2}^{(2)} \ldots v_{i_N}^{(N)} \tag{4.12}$$

And lastly, multiplication in every mode except mode n results in a vector of length I_n,

$$\mathcal{X} \bigtimes_{m=1, m\neq n}^{N} \mathbf{v}^{(m)} = \mathbf{X}_{(n)} \mathbf{v}^{(-n)} \tag{4.13}$$

where

$$\mathbf{v}^{(-n)} = \mathbf{v}^{(N)} \otimes \ldots \otimes \mathbf{v}^{(n+1)} \otimes \mathbf{v}^{(n-1)} \otimes \ldots \otimes \mathbf{v}^{(1)} \tag{4.14}$$

4.2 Tensor Decomposition

Canonical tensor decomposition decomposes a tensor into sum of rank-one tensors, hence it is analogous to matrix singular value decomposition. Assuming that $\mathcal{Z}$ is a real-valued three-way tensor of size $I \times J \times K$ and rank R, its CP decomposition is given as

$$\mathcal{Z} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{4.15}$$

This tensor decomposition is defined for three-way tensors. In order to generalize to higher dimensions, where $\mathcal{Z}$ is a real-valued N-way tensor of size $I_1 \times I_2 \times \ldots \times I_N$ and rank R, tensor decomposition can be written as

$$\mathcal{Z} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r^{(1)} \circ \ldots \circ \mathbf{a}_r^{(N)}. \tag{4.16}$$

where $\mathbf{a}_r^{(N)} \in \mathbb{R}^{I_n}$ for $n = 1, \ldots, N$ and $r = 1, \ldots, R$. 'Kruskal operator' defined as below can also be used to represent tensors:

$$[[\mathbf{A}^{(1)} \ldots \mathbf{A}^{(N)}]] = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r^{(1)} \circ \ldots \circ \mathbf{a}_r^{(N)}. \tag{4.17}$$

where *factor matrices* are defined as

$$\mathbf{A}^{(n)} = [\mathbf{a}_1^{(n)} \ldots \mathbf{a}_R^{(n)}] \tag{4.18}$$

Factor matrices $\mathbf{A}^{(n)}$, have a size of $I_n \times R$, for $n = 1, \ldots, N$. The columns of $\mathbf{A}^{(n)}$ are the *factors* of mode n. $[[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]]$ can be expressed in matricized form [18]:

$$([[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]])_{(n)} = \mathbf{A}^{(\mathbf{n})}(\mathbf{A}^{(-\mathbf{n})})^T \tag{4.19}$$

where

$$\mathbf{A}^{(-n)} = \mathbf{A}^{(N)} \odot \ldots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \ldots \odot \mathbf{A}^{(1)} \tag{4.20}$$

Given a tensor $\mathcal{Z}$ and a suitable rank R, computing CP amounts to finding the factor matrices $\mathbf{A}^{(n)}$. The corresponding columns of each $\mathbf{A}^{(n)}$, i.e. $\mathbf{a}_i^{(1)}, \mathbf{a}_i^{(2)}, \ldots, \mathbf{a}_i^{(N)}$, form a rank-one matrix. Combining all these rank-one tensors for $i = 1, \ldots, R$, the tensor is approximated by the linear combination of rank-one tensors. Hence, the problem of approximating CP decomposition can be posed as a least-squares optimization problem:

$$\min f(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) = \frac{1}{2} \left|\left|\mathcal{Z} - [[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]]\right|\right|^2 \tag{4.21}$$

Alternating Least Squares (ALS) method proposed in [7] and [14] is a preferred way to decompose a tensor into its rank-one components. It is based on the idea of optimizing one factor matrix at a time instead of solving for all factor matrices simultaneously. Thus, at each inner iteration, the following objective function is optimized:

$$\min_{\mathbf{A}^{(n)}} f(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) \tag{4.22}$$

for a fixed n, while all the other factor matrices are constant. Using matrix notation, the objective function can be expressed as,

$$\min_{\mathbf{A}^{(n)}} \frac{1}{2} \left|\left|\mathcal{Z}_{(n)} - \mathbf{A}^{(n)}(\mathbf{A}^{(-n)})^T\right|\right| \tag{4.23}$$

With all the factor matrices fixed except one, this optimization problem is a least-squares problem and the solution is given by

$$\mathbf{A}^{(n)} = \mathcal{Z}_{(n)} \left((\mathbf{A}^{(-n)})^T \right)^\dagger \tag{4.24}$$

In order to find the solution, the pseudo-inverse of a matrix of size $\prod_{m=1,m\neq n}^{N} I_m \times R$ should be evaluated. Using Eq. (4.9), this computation can be simplified by defining a $\mathbf{\Upsilon}^{(n)}$ as in the following:

$$\mathbf{\Upsilon}^{(n)} = \mathbf{A}^{(n)T}\mathbf{A}^{(n)} \text{ for } n = 1, \ldots, N \tag{4.25}$$

Then, $\mathbf{A}^{(n)}$ can be written as,

$$\mathbf{A}^{(n)} = \mathcal{Z}_{(n)}\mathbf{A}^{(-n)}(\mathbf{\Gamma}^{(n)})^\dagger \tag{4.26}$$

where

$$\mathbf{\Gamma}^{(n)} = \mathbf{\Upsilon}^{(1)} * \ldots * \mathbf{\Upsilon}^{(n-1)} * \mathbf{\Upsilon}^{(n+1)} * \ldots * \mathbf{\Upsilon}^{(N)} \tag{4.27}$$

In this case, it is only needed to compute the pseudo-inverse of the matrix of size $R \times R$.

4.3 Optimization Scheme for Tensor Decomposition

As an alternative to the alternating least squares solution, a gradient-based optimization approach is proposed in [1]. f in Eq. (4.21) is stated as a function of matrices. It can also be devised as a scalar valued function where all the matrices are vectorized and stacked into a vector $\mathbf{x}$.

$$\mathbf{x} = \begin{bmatrix} \mathbf{a}_1^{(1)} \\ \vdots \\ \mathbf{a}_R^{(1)} \\ \vdots \\ \mathbf{a}_1^{(N)} \\ \vdots \\ \mathbf{a}_R^{(N)} \end{bmatrix} \tag{4.28}$$

With this kind of formulation, it is straightforward to compute the gradient. Once the derivatives are known, a first-order optimization method can be applied. In the proposed scheme, a generic nonlinear conjugate gradient method is applied.

The gradient can be assembled by calculating the partial derivatives with respect to each $\mathbf{a}_r^{(n)}$, i.e. by computing $\frac{\partial f}{\partial \mathbf{a}_r^{(n)}}$ for $r = 1, \ldots, R$ and $n = 1, \ldots, N$. The partial derivative is a vector of length I_n.

Theorem 4.1. The partial derivatives of the objective function f can be found via

$$\frac{\partial f}{\partial \mathbf{a}_r^{(n)}} = -\left(\mathcal{Z} \bigtimes_{m=1, m \neq n}^{N} \mathbf{a}_r^{(m)} \right) + \sum_{\ell=1}^{R} \gamma_{r\ell}^{(n)} \mathbf{a}_\ell^{(n)} \tag{4.29}$$

where $r = 1, \ldots, R$ and $n = 1, \ldots, N$ with $\gamma_{r\ell}^{(n)}$ defined as

$$\gamma_{r\ell}^{(n)} = \prod_{m=1, m \neq n}^{N} \mathbf{a}_r^{(m)T} \mathbf{a}_\ell^{(m)} \tag{4.30}$$

The proof of this theorem is provided in Appendix A of the thesis.

Corollary 4.2 The partial derivatives of the objective function f in Eq. (4.21) are given by

$$\frac{\partial f}{\partial \mathbf{A}^{(n)}} = -\mathbf{Z}_{(n)} \mathbf{A}^{(-n)} + \mathbf{A}^{(n)} \Gamma^{(n)} \tag{4.31}$$

for $n = 1, \ldots, N$ and $\mathbf{\Gamma}^{(n)}$ is defined in Eq. (4.27).

Proof. Eq. (4.29) in Theorem 4.1 can be written as in the following:

$$\frac{\partial f}{\partial \mathbf{a}_r^{(n)}} = -\mathbf{Z}_{(n)}\mathbf{a}^{(-n)} + \mathbf{A}^{(n)}\gamma_r^{(n)}, \tag{4.32}$$

for $r = 1, \ldots, R$. This expression is obtained by exploiting the fact that $\mathbf{\Gamma}^{(n)}$ is symmetric. Associating each $r = 1, \ldots, R$ with the column of a matrix yields Eq. (4.31). $\square$

Knowing the objective function given in Eq. (4.21) and the gradient given in Eq. (4.31), a gradient based optimization can be performed in order to find the factor matrices, $[[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]]$. Then, using Eq. (4.18), all vectors constituting rank-one matrices can be found and hence, the tensor can be written as a linear combination of these rank-one matrices (separable matrices) according to Eq. (4.17).

Compared to Alternating Least Squares(ALS) approach, this tensor decomposition approach [1] solves for all factor matrices simultaneously. In this particular implementation, a nonlinear conjugate gradient (NCG) method with Polak-Ribiere (PR) updates is used.

4.3.1 Regularization of the optimization formulation of tensor decomposition

Canonical Tensor Decomposition is unique up to permutation and scaling when it satisfies Kruskal conditions [20]. Therefore, tensor decomposition is the same when it is permuted, i.e.,

$$[[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]] = [[\mathbf{A}^{(1)}\mathbf{\Pi}, \mathbf{A}^{(2)}\mathbf{\Pi}, \ldots, \mathbf{A}^{(N)}\mathbf{\Pi}]] \tag{4.33}$$

where $\mathbf{\Pi}$ is an $R \times R$ permutation matrix. Besides, the tensor decomposition is the same when it is scaled, i.e.,

$$[[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]] = [[2\mathbf{A}^{(1)}, \frac{1}{2}\mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]] \tag{4.34}$$

Uniqueness up to scaling and permutation means that there are more than one solution for different scaling constants and permutation matrices, this situation makes it difficult for the optimization scheme to find the solution. In order to have a unique solution that the optimization scheme can converge, a regularization term can be included in the objective function [1].

$$\hat{f}(\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}) = \frac{1}{2} \left|\left| \mathcal{Z} - [[\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}]] \right|\right|^2 + \frac{\lambda}{2} \sum_{n=1}^{N} \left|\left| \mathbf{A}^{(n)} \right|\right|_2^2 \tag{4.35}$$

Tikhonov regularization applied in Eq. (4.35), promotes the norm equivalency [1]:

$$||\mathbf{A}^{(1)}|| = ||\mathbf{A}^{(2)}|| = \cdots = ||\mathbf{A}^{(N)}|| \tag{4.36}$$

Chapter 5

Learning Separable Filters with Tensor Decomposition

So far, two generic approaches to learn separable filters have been presented in Section 3.2.1 and 3.2.2. In these methods, dictionary learning with sparse coding has been used to force each filter to be low-rank independently. The dictionary learning approach employed in these methods is easy and effective, however there is the drawback of converging slowly. Rather than handling each filter independently, another viewpoint for learning separable filters would be grouping all the filters of an arbitrary filter bank into a tensor which has one more dimension than the dimension of the filters and constraining this tensor to be the linear combination of separable tensors. One way to achieve this would be to explicitly write the linearly combined tensors as products of one-dimensional filters and minimize the distance with respect to the original tensor. This global optimization problem can be carried out using tensor decomposition techniques as will be explained in this chapter. Moreover, the tensor decomposition problem could be regularized to include sparsity constraint on the coefficients of the tensor, this extension is also explained in the remainder of the chapter.

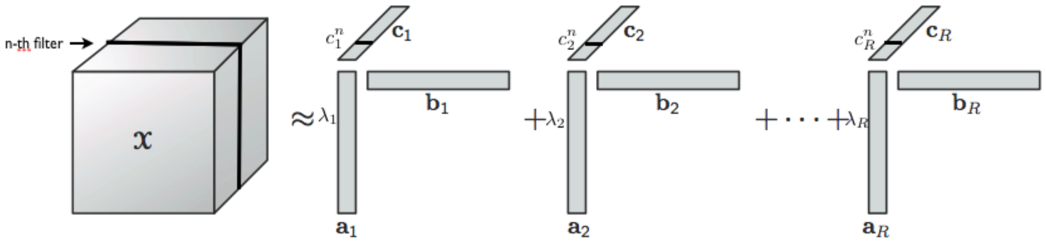

Figure 5.1: Tensor decomposition of a 3-way tensor into a sum of R rank-one tensors. Rank-one tensors can be writen as the outer product of vectors, hence they are separable. The n-th filter, i.e. the n-th slice of the tensor, can be expressed as the linear combination of the outer products of vectors $\mathbf{a_i}$ and $\mathbf{b_i}$, which is a two-dimensional separable filter. The weighting coefficients are $\lambda_i c_i^n$.

5.1 Approximating Arbitrary Filter Sets with Separable Filters

We have seen that the accuracy of the image processing tasks carried out using non-separable filters can be matched using separable filters at a much smaller computational cost using learning based approaches. Optimizing all the filters in a filter bank globally to learn separable filters would also prove useful in terms of computational efficiency. The idea is that the filters can be stacked together to form a tensor and this tensor can be factorized into sum of rank-one matrices as depicted in Fig. 5.1. Assuming that, the non-separable filters in the filter bank share some parts, these shared parts can also be expressed using a smaller set of separable basis filters, i.e., the space spanned by the non-separable filters could also be spanned by a smaller set of separable filters. In this case, the advantages are two-fold. On one hand, a smaller set of filters are used; and on the other hand, filters are separable. These two properties would increase substantially the computational efficiency of the convolutional operations.

The mathematical background for tensor decomposition is given for N-way tensors in Chapter 4. When the 2D filters are stacked into a 3D tensor, the optimization problem for the tensor decomposition can be set as

$$\min f(\mathbf{A},\mathbf{B},\mathbf{C}) = \frac{1}{2}\left\|\mathcal{Z} - \sum_{r=1}^{R} \lambda_r\, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\right\|^2 \tag{5.1}$$

$\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are factor matrices and can be written in Kruskal form to represent the 3-way tensor.

$$[[\mathbf{A},\mathbf{B},\mathbf{C}]] = \sum_{r=1}^{R} \lambda_r\, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{5.2}$$

where *factor matrices* are defined as,

$$\mathbf{A} = [\mathbf{a}_1 \ldots \mathbf{a}_R] \tag{5.3}$$

$$\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_R] \tag{5.4}$$

$$\mathbf{C} = [\mathbf{c}_1 \ldots \mathbf{c}_R] \tag{5.5}$$

$[\mathbf{a}_1 \ldots \mathbf{a}_R]$, $[\mathbf{b}_1 \ldots \mathbf{b}_R]$ and $[\mathbf{c}_1 \ldots \mathbf{c}_R]$ are illustrated in Fig. 5.1 and Fig. 5.2.

Using the optimization scheme explained in Section 4.3, an N-dimensional tensor can be factorized into a sum of rank-one tensors, i.e. a tensor that can be represented as the outer product of N vectors and thus, any arbitrary filter in a filter bank can be approximated by a linear combination of separable filters as illustrated in Fig. 5.2. The n-th filter in the filter bank can be expressed mathematically as follows:

$$\mathbf{f}^n = \lambda_1\, c_1^n\, \mathbf{a_1} \circ \mathbf{b_1} + \lambda_2\, c_2^n\, \mathbf{a_2} \circ \mathbf{b_2} + \cdots + \lambda_R\, c_R^n\, \mathbf{a_R} \circ \mathbf{b_R} \tag{5.6}$$

With this kind of formulation, convolving the image with $\mathbf{f^j}$'s amounts to convolving the image with separable filters ($\mathbf{a_i} \circ \mathbf{b_i}$'s for $i = 1, \ldots, R$) and linearly combining the resulting convolution images with the obtained set of weights ($\lambda_i\, c_i^j$'s for $i = 1, \ldots, R$ and $j = 1, \ldots, I_n$ where I_n is the size of the vector c_i). The generic tensor decomposition method is also applicable for three or more-dimensional tensors.

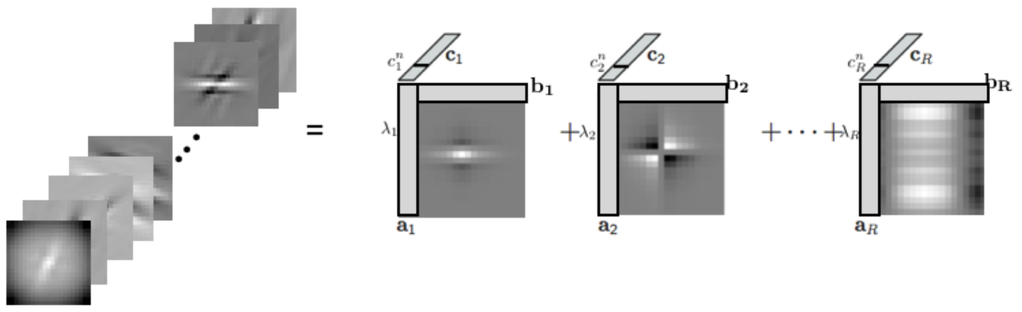

Figure 5.2: Illustration of the tensor decomposition for learning separable filters for two-dimensional biomedical DRIVE dataset. The learned 2D arbitrary non-separable filters are grouped into a 3-way tensor and the tensor is decomposed into a sum of R rank-one tensors. The outer product of the corresponding columns of first two factor matrices, **A** and **B** form the separable basis and spans the space of the non-separable filters. Hence, a 'shared basis' of separability is learned.

5.2 Sparse Regularization of the Tensor Decomposition Problem

The formulation of the tensor decomposition can also be regularized to include sparsity constraints. In this case, ℓ_1-norm which enforces sparsity on the factor matrices could be used.

$$\hat{f}(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}) = \frac{1}{2}\left|\left|\mathcal{Z} - [[\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}]]\right|\right|^2 + \frac{\lambda}{2}\sum_{n=1}^{N}\left|\left|\mathbf{A}^{(n)}\right|\right|_1 \tag{5.7}$$

The minimization of this objective function can be obtained via a first-order optimization method with thresholding as suggested in [8]. This decomposition yields sparse rank-one tensors which could speed-up the process of arithmetic operations carried out on tensors as there are many zero coefficients in the tensor. Regularization parameters can be optimized for different factor matrices, separately.

Chapter 6

Results and Discussion

Algorithmic descriptions of the separable filter learning problem using dictionary learning and tensor decomposition approaches were provided in addition to their mathematical derivations in Chapter 3 and 5. In this chapter, visual and numerical results of the computer vision tasks using non-separable and separable filters learned with different approaches are going to be provided.

We will demonstrate that a set of learned separable filters can deliver a significant amount of speed-up in tasks that require convolution at no loss in performance as compared to a set of arbitrary non-separable filters. We, then, illustrate that a set of separable filters can effectively approximate an arbitrary non-separable filter bank. We will provide comparisons for the performances of different separable filter learning schemes with tensor decomposition and dictionary learning in different computer vision tasks.

Firstly, we compare the performance of the separable filters against that of non-separable filters for classifying pixels and voxels in order to understand whether they belong to linear structures or not. Thus, the performance of the separable filters is demonstrated on both two-dimensional and three-dimensional data. The notation followed in order to depict different kinds of filter banks is as follows: NON-SEP is used to denote the learned arbitrary non-separable filters obtained by minimizing the objective function given in Eq. (3.3). SEP-DIRECT is used to denote the learned separable filters obtained by minimizing the objective function given in Eq. (3.4). SEP-COMB is used to denote the learned separable filters obtained by minimizing the objective function given in Eq. (3.6). We call our approach to learn separable filters with tensor decomposition SEP-TD. In another approach to learn separable filters, a sparse regularization is applied on the tensor decomposition optimization problem as explained in Section 5.2, i.e., the separable filters are obtained by minimizing the objective function given in Eq. (5.7). The classification is, then, carried out using the feature maps obtained by the filters of this formulation. Enforcing this constraint on the optimization problem results in sparse tensors. The sparse representation could be of use in increasing the computational efficiency by not considering zero-coefficients in the convolution operations. This procedure to learn separable filters is denoted as SEP-TD-SP. Regularization parameters can be chosen separately for different factor matrices.

When we feed the learned filters' output into a linear classifier, it is not necessary to explicitly compute the linear combination of the separable filters to approximate the non-separable filter bank as the classifier can be directly trained on separable filters' output. Hence, the linear combinations can be implicitly learned by the classifier at training time. This approach has been referred to as SEP-COMB* and SEP-TD* in the remainder of the thesis and it provides a substantial amount of increase in the computational efficiency.

In addition to the classification experiments, the accuracy of approximating a non-separable filter bank with a set of separable filters is evaluated for image denoising tasks. We compare the denoising performance of approximated non-separable filters obtained with different separable filter learning approaches to that of the original filter bank obtained with K-SVD training.

The performance of the learned separable filters is also evaluated on a handwritten digit recognition task using convolutional neural networks (CNNs). We demonstrate that we can increase considerably the computational efficiency of the object recognition task at no loss in performance using separable filters.

6.1 Pixel Classification to Detect Linear Structures

In this section, the efficiency of the developed separable filter learning approach is going to be demonstrated for linear structure detection application. This problem has been researched over a long time and still, there exist some challenges when the image data is noisy. Over the course of the research, Machine Learning techniques have been popular in the past few years. In [36] and [13], Support Vector Machine classifier is applied to the responses of *ad hoc* filters. The recent study of [33], demonstrates that the performance of these methods can be exceeded by convolving the images with non-separable filter banks learned by solving the problem of Eq. (3.3) and training an SVM on the output of those filters. For large images and volumes, the large number of non-separable filters makes convolution a demanding task. We demonstrate that our approach to learn separable filters provides a solution for this difficulty.

For pixel classification, the performance of the separable filter learning with tensor decomposition is evaluated on two different data sets of Fig. 6.1.

- *DRIVE Dataset.* It is a dataset of 40 retinal scans captured in order to diagnose various diseases. There are 20 training images and 20 test images, with two different ground truth sets traced by two different human experts.
- *BF2D Dataset.* It is a dataset composed of minimum intensity projections of bright-field micrographs of neurons. Although images are of high resolution, they have a low signal-to-noise ratio due to irregularities in the staining process. Some parts of the dendrites often seem as point-like structures which can lead to misclassification of linear structures in the presence of noise.

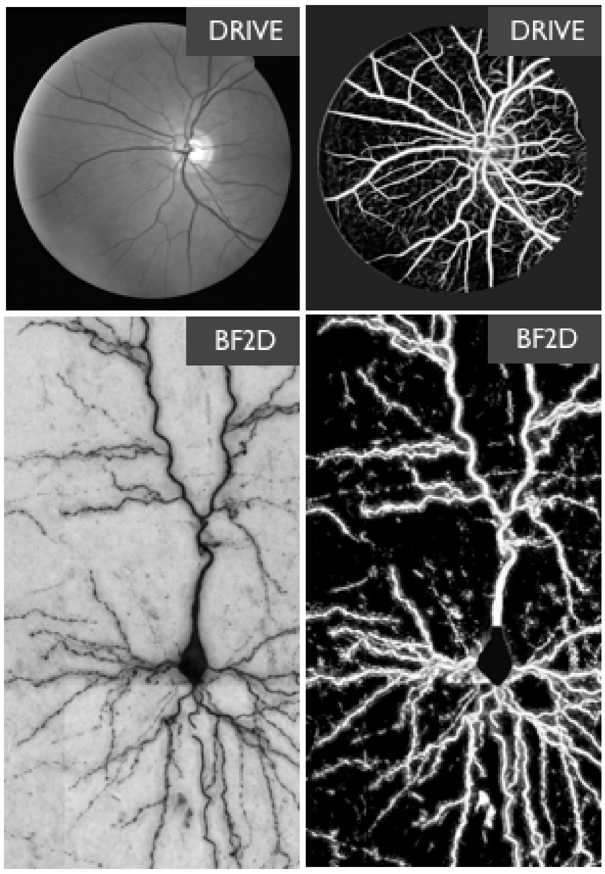

Figure 6.1: Example images from two biomedical datasets along with their classification results obtained with the SEP-TD method. The numerical results yield similar classification results with [34] while the learning procedure is simplified. The run-time speed of learning is orders of magnitude faster than SEP-COMB approach of [34].

All the methods described in Chapter 3 and Chapter 5 have been tested on these datasets for pixel classification along with Optimally Oriented Flux approach [21], which is known to be one of the best techniques to detect linear structures. The feature maps are computed by convolving the images with the learned filter banks and these feature maps are fed to Random Forest classifier. Since this classifier relies on linear projections, computing the linear combination of the separable filters' outputs to approximate the non-separable filter bank is not necessary. The classifier trains the data on

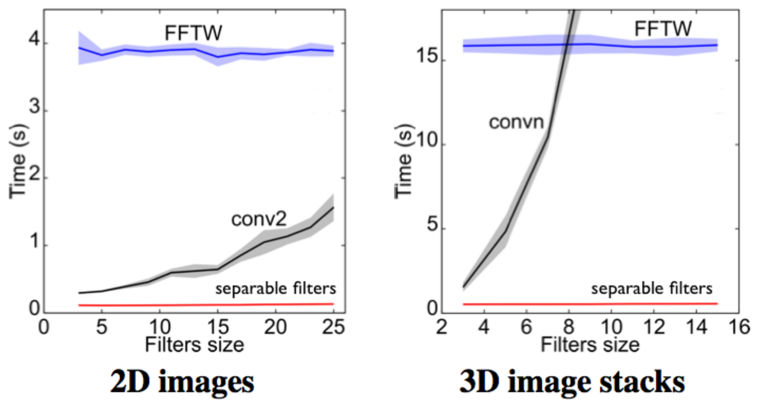

Figure 6.2: Convolution time for FFTW method, non-separable and separable filters [34]. Using a smaller set of separable filters, the extraction of the image features is considerably faster than convolutions with non-separable filters and Fast Fourier Transform.

the feature maps obtained with separable filters and thus, implicitly learns the linear combinations of these separable filters. Therefore, in this case, we opt for SEP-COMB* and SEP-TD* approaches.

NON-SEP method outperforms the methods that rely on Machine Learning [33]. However, it has the drawback of being slow. The motive to use separable filters is to obtain the same level of performance in a much faster way.

The advantage of using separable filters can be clearly seen in Fig. 6.2 based on the previous separable filter learning study [34]. The time needed to convolve a 512×512 image with a set of 121 filters in order to extract feature maps is depicted in the graph for two-dimensional case as a function of filter size by using MATLAB's conv2 function, the FFTW library and separable filter learning approach. The required time for convolving increases sharply for conv2 function whereas there is only a small increase in the convolution time when separable filters are used. Convolution time does not increase with varying filter size in FFTW approach, however it still requires much more time than both approaches. The advantage of using separability is even more pronounced for the three-dimensional data of $128 \times 128 \times 64$ size.

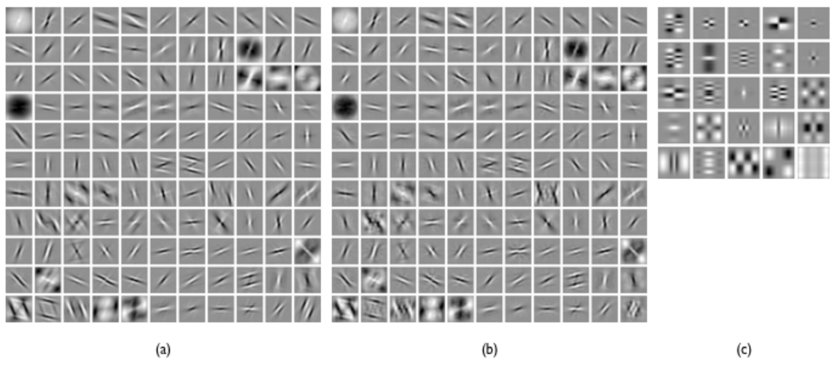

Figure 6.3: Convolutional filter banks for classification in 2D. (a) Learned non-separable filter bank from DRIVE dataset, (b) reconstructed filter bank , (c) separable filter bank learned with the SEP-TD approach. The non-separable filters can be approximated very accurately using a smaller set of separable filters.

Throughout the experimentation, first, the methods proposed in Chapter 3 are used to obtain feature maps for classification. Then, using the tensor decomposition techniques described in Chapter 4 and Chapter 5, separable filters are learned (See Fig. 6.3.(c)), and the feature maps are computed using these separable filter banks. The classification with these different filter learning strategies are compared.

The detailed and comparative pixel classification results are reported in Table 6.1. There have been several methods proposed to measure the classification performance. Among these, the results are tabulated in terms of

- F-measure [40];
- Area Under the Curve (AUC), which represents the area under the ROC curve. It takes values in the range $[0, 1]$, the higher it is the, the better the classification is;
- Variation of Information (VI) [26], which takes values in the range $[0, \infty)$, the lower it is, the better the classification is;
- Rand Index (RI) [39], which takes values in the range $[0, 1]$, the higher it is, the better the classification is;

Table 6.1: Analytic measure of the performance of the pixel classification task over different datasets. The VI and RI values are compared on the classification thresholded at the value found using the F-measure. The values are averaged over 5 random trials and over the whole dataset images. For the learning-based approaches, a training set of 50000 positive and 50000 negative samples and a Random Forests classifier have been used. Approaches that use a separable filter basis have been found to reduce the computational costs by a factor of 10 in classifications tasks.

Method	**AUC**	**F-measure**	**VI**	**RI**	**Time**[s]
		DRIVE			
Ground truth	–	0.788	0.380	0.930	–
OOF	0.933	0.695	0.569	0.770	5.70
NON-SEP(121)	0.959	0.782	0.554	0.890	2.22
SEP-DIRECT(25)	0.948	0.756	0.602	0.879	0.23
SEP-COMB(25)*	0.959	0.785	0.541	0.894	0.23
SEP-TD(25)*	0.960	0.780	0.583	0.885	0.23
SEP-TD-SP(25)	0.959	0.777	0.586	0.884	0.23
		BF2D			
OOF	0.958	0.677	0.325	0.891	15.88
NON-SEP(121)	0.983	0.754	0.300	0.945	11.42
SEP-DIRECT(25)	0.980	0.750	0.306	0.944	1.44
SEP-COMB(25)*	0.981	0.752	0.301	0.944	1.44
SEP-TD(25)*	0.980	0.736	0.340	0.936	1.44
SEP-TD-SP(25)	0.979	0.732	0.344	0.933	1.44

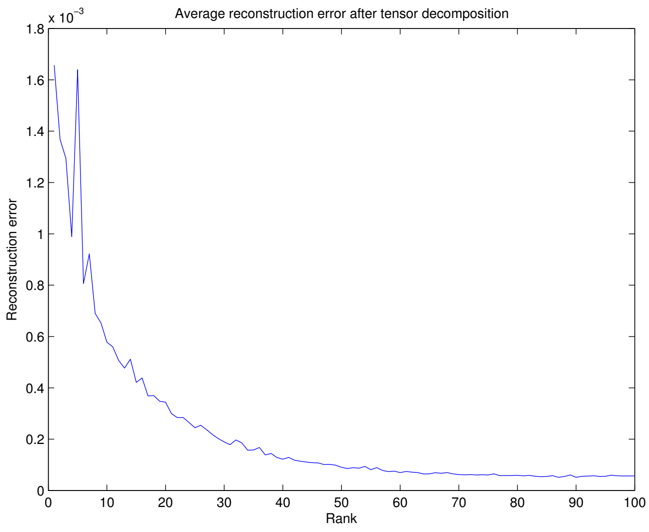

Figure 6.4: Average reconstruction error with respect to the specified rank value for tensor decomposition. As the rank increases the non-separable filter bank can be better approximated. Increasing the rank is equivalent to increasing the number of separable filters, hence there is the trade-off between the accuracy and the speed.

As can be interpreted from Table 6.1, the classification performance of SEP-TD*, SEP-COMB* and NON-SEP approaches are very similar. The performance of SEP-TD-SP also closely matches to the classification performance of SEP-TD*. As the regularization parameter, 0.001 is used for **A** and **B** and 0.0005 is used for **C** factor matrices for DRIVE dataset. SEP-DIRECT is as fast as the SEP-COMB* and SEP-TD* approaches, however it entails a loss of accuracy. All the filter-based methods are more accurate than OOF.

The number of separable filters used in SEP-TD approach is determined by the rank specified for the tensor decomposition problem. As it is not possible to know R *a priori*, this becomes a parameter of the decomposition. As the rank increases, the non-separable filter bank is better approximated. The reconstruction error of the non-separable filter bank, therefore, decreases with increasing rank as depicted in Fig. 6.4.

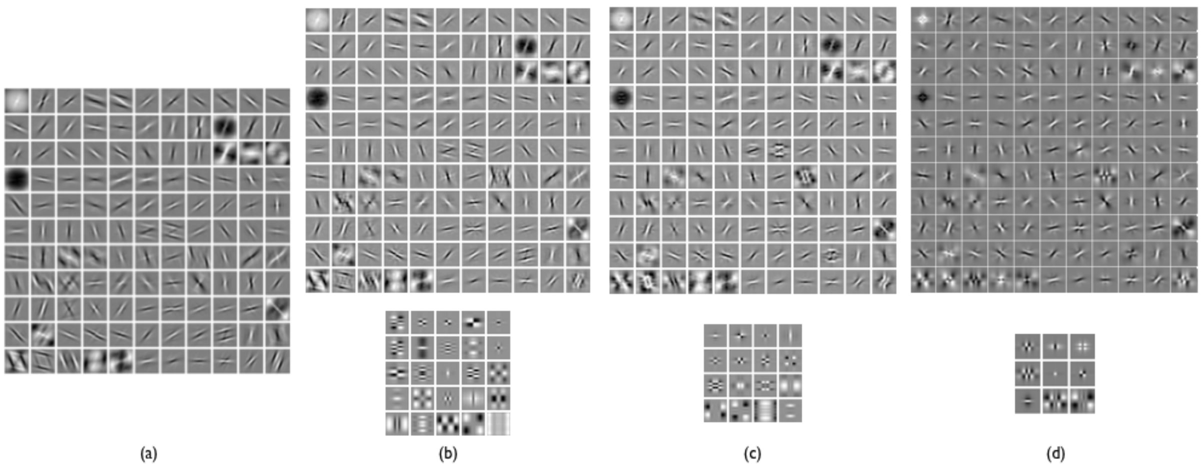

Figure 6.5: Convolutional filter banks for different ranks specified for tensor decomposition. (a) Original non-separable filter bank; (b), (c), (d) learned separable filter banks with SEP-TD approach and their approximations of the original filter bank, $R = 25$, $R = 16$, $R = 9$. As the rank increases, approximation quality also increases.

Several separable filter banks and their approximations of the non-separable filter bank is depicted in Fig. 6.5 along with the original filter bank. Fig. 6.6 and 6.7 , demonstrate the pixel classification results obtained for some images from the considered 2D datasets. The results of our method is not distinguishable from the classifications obtained by the non-separable approach.

As can be seen from the detailed analysis presented in Table 6.1, SEP-TD* approach closely matches the performance of SEP-COMB* approach. The

Table 6.2: Time required to learn a separable filter basis from a non-separable filter bank. Filter banks consist of 25 separable filters. The learning is carried out on DRIVE dataset. Approximation error is given in terms of the mean-squared errors.

Method	**Learning Time[sec]**	**Reconstruction Error(MSE)**	$\|\|\ell\|\|_0$
SEP-TD	18.47	2.67×10^{-4}	36875
SEP-TD-SP	35.89	5.3×10^{-4}	53361

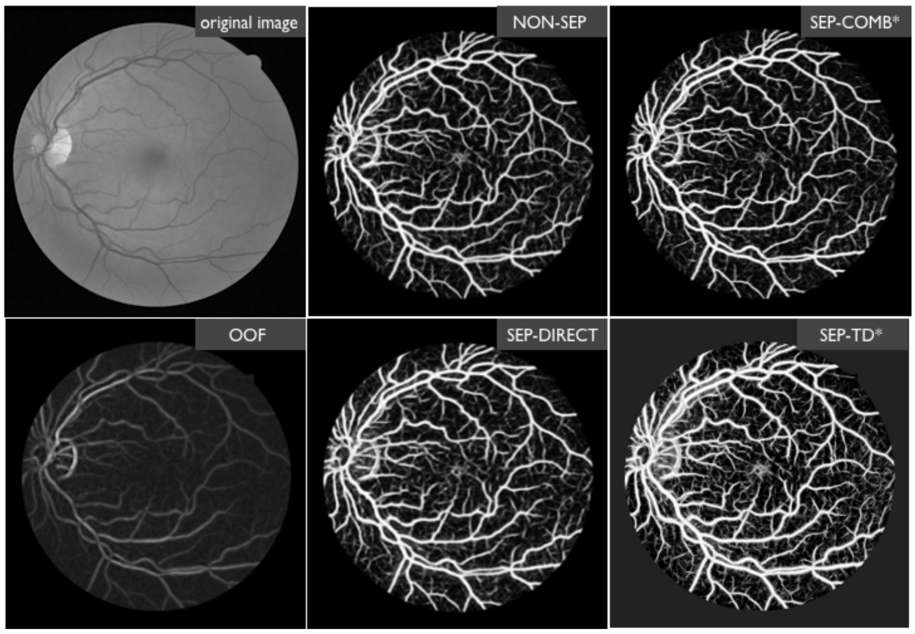

Figure 6.6: Classification examples from DRIVE dataset for different filter learning approaches. Numerically and visually, similar classification results are obtained for SEP-TD*, SEP-COMB* and NON-SEP methods. The feature extraction procedure is fast for SEP-TD* and SEP-COMB* approaches due to the separability of the convolutional filters. The advantage of using SEP-TD* over SEP-COMB* is that the learning speed of the filters with the SEP-TD* is much faster than that of SEP-COMB*.

advantage of using SEP-TD* over SEP-COMB* approach is in its substantial speed-up in the filter learning process. Furthermore, SEP-TD* approach provides a better approximation of the arbitrary non-separable filter bank, i.e. the non-separable filter bank is reconstructed with a smaller error. Table 6.2 presents the results of the reconstruction errors of the non-separable filter banks and the computation time required to learn separable filters. In this experiment, first 121 arbitrary filters are learned from DRIVE dataset, and these filters are approximated by a separable basis using SEP-COMB and SEP-TD approaches.

Commenting on the results tabulated in Table 6.2, SEP-TD approach obtains the separable filter bank 100 times faster than SEP-COMB approach

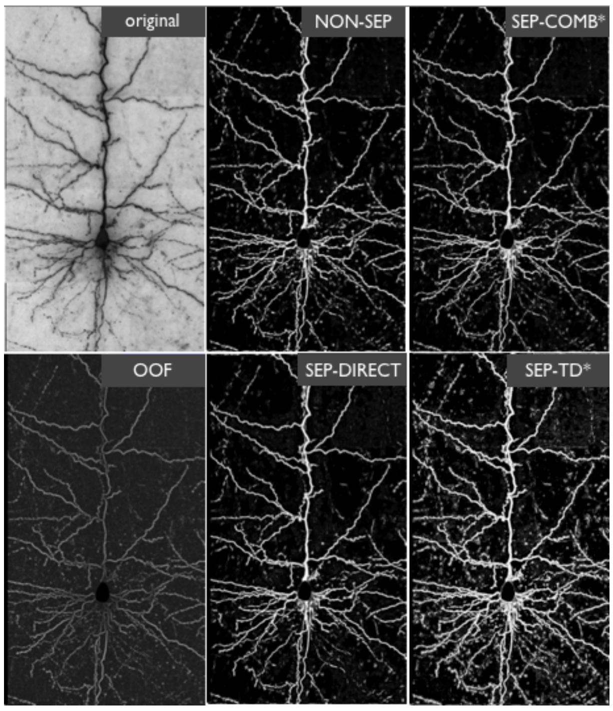

Figure 6.7: Classification examples from BF2D dataset for different filter learning approaches

with twice the accuracy. Efficient tensor formulation of the first derivative of Eq. (4.31) leads to the good performance of the tensor decomposition algorithm.

6.2 Voxel Classification to Detect Linear Structures

Separable filter learning with tensor decomposition has also been evaluated on voxel classification task in order to understand whether the voxels belong to linear structures or not. 3D volumes of Olfactory Projection Fibers (OPF) from DIADEM challenge, which were captured by a confocal microscope, have been used as the experiment dataset. First, a filter bank of 49 13×13×13 pixels have been learned by optimizing the objective function given in Eq.

(3.3). Then, a separable basis consisting of 16 filters has been learned using SEP-COMB and SEP-TD approaches. Learned arbitrary non-separable filter bank consisting of 49 filters, the separable filter bank learned with SEP-TD approach and the reconstructed non-separable filter bank is shown in Fig. 6.8. As in the case for two-dimensional data, the classifiers have been trained on the separable filters' output, therefore we opt for SEP-COMB* and SEP-TD* approaches. For 3D data, ℓ_1-regularized logistic regressors instead of Random Forests have been used since they have been found to be faster with no loss in performance.

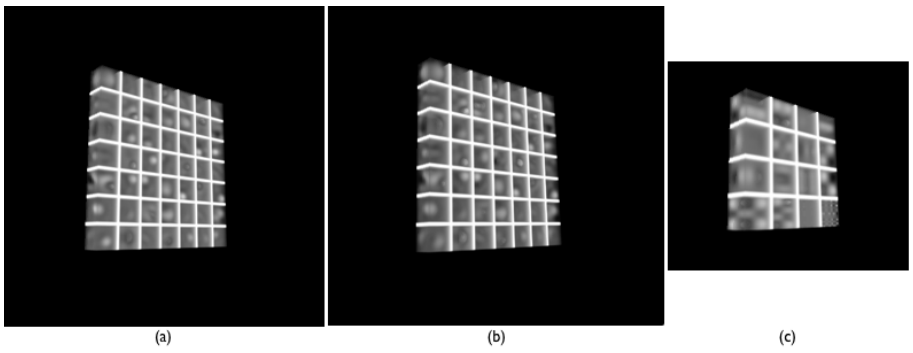

Figure 6.8: Example 3D Non-separable and separable filter banks for voxel classification. (a) Learned Non-separable Filter Bank from OPF dataset, (b) Reconstructed Filter Bank , (c) Separable Filter Bank learned with the SEP-TD approach. SEP-TD approach is accurate in approximating the original non-separable filter bank.

As in the 2D case, OOF has been used as the baseline. NON-SEP, SEP-COMB* and SEP-TD* approaches have been compared against it. The detailed and comparative pixel classification results are reported in Table 6.3. A representative classified image of the OPF dataset along with the original image is shown in Fig. 6.9. While all the approaches result in similar F-measure values, SEP-COMB* and SEP-TD* approaches are considerably faster in the classification task. The advantage of using SEP-TD over SEP-COMB is that the filter learning process is much faster than that of SEP-COMB. Moreover, SEP-TD approach provides a better approximation

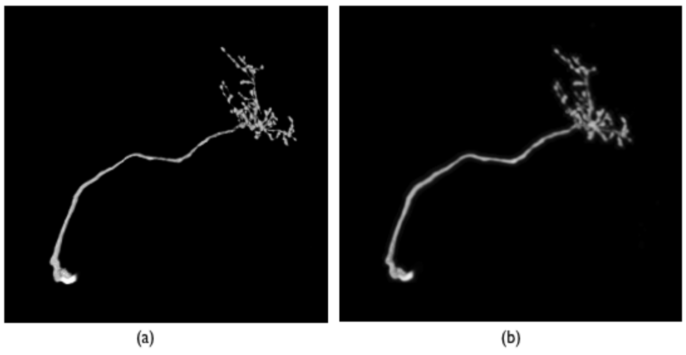

Figure 6.9: Example classification result obtained on OPF dataset. The image 4 of the dataset is used for testing. The ℓ_1 regularized logistic regression classifier is trained on the separable filters' output.

Table 6.3: Analytic measure of the performance of the voxel classification task over different datasets. The VI and RI values are compared on the classification thresholded at the value found using the F-measure. For the learning-based approaches, a training set of 200000 randomly selected

Method	**AUC**	**F-measure**	**VI**	**RI**	**Time**[s]
		OPF:Image 1			
OOF	0.997	0.531	0.012	0.998	193.05
NON-SEP(121)	0.997	0.571	0.013	0.998	339.01
SEP-COMB(25)*	0.997	0.570	0.013	0.998	11.08
SEP-TD(25)*	0.997	0.564	0.013	0.998	11.08

of the arbitrary non-separable filter bank, i.e. the non-separable filter bank is reconstructed with a smaller error. Table 6.4 presents the results of the reconstruction errors of the non-separable filter banks and the computation time required to learn separable filters for SEP-COMB and SEP-TD approaches. Commenting on the results of Table 6.4, SEP-TD learns separable filters 600 times faster than SEP-COMB with twice the accuracy.

Table 6.4: Time required to learn a separable filter basis from a 3D non-separable filter bank. Filter banks consist of 16 separable filters. The learning is carried out on OPF dataset. Approximation error is given in terms of the mean-squared errors.

Method	**Learning Time[sec]**	**Reconstruction Error(MSE)**
SEP-COMB	8370	1.8×10^{-5}
SEP-TD	13.58	0.9×10^{-5}

6.3 Image Denoising

In order to assess how well the separable basis learned by the SEP-TD approach is at representing an arbitrary filter bank, we also carried out experiments in a different task, e.g. image denoising. SEP-TD approach is used to learn the separable basis of 256 denoising filters obtained by K-SVD algorithm [9]. The denoising results obtained by using different sizes for the separable basis is tabulated in Table 6.5 in terms of Peak-Signal-to-Noise-Ratio (PSNR). The original filter bank, the reconstructed filter bank and the separable filter bank consisting of 36 filters are depicted in Fig. 6.10. It is clear that, the reconstructed and the original filter bank are nearly indistinguishable, thus even 36 filters are sufficient to grasp the essence of the information carried by 256 filters. Fig. 6.11 shows an example denoising experiment carried out using our tensor decomposition algorithm.

One more observation is that the separable filters learned with the SEP-TD approach are quite general in the sense that the separable filters learned for one specific image can also be used for another image and still yield satisfactory denoising results which closely matches to the denoising results of K-SVD approach.

Table 6.5: Results for the image denoising task. The denoising results are given in terms of Peak Signal-to-Noise ratio (PSNR) and measured in decibels. The images are corrupted by additive white gaussian noise with a standard deviation of 20. The filters learned by K-SVD algorithm are approximated using different separable filter learning approaches. In the last part of the experiment, the filters learned from the *Barbara* image is used to denoise other images in order to assess how generic the denoising filters are.

	Barbara	**Boat**	**House**	**Lena**	**Peppers**
Noisy image	22.12	22.09	22.06	22.09	22.13
K-SVD	30.94	30.36	33.34	32.42	32.25
SEP-COMB(25)*	30.21	30.26	33.13	32.40	31.99
SEP-TD(25)*	30.27	30.28	33.17	32.42	32.03
SEP-COMB(36)*	30.77	30.36	33.24	32.42	32.08
SEP-TD(36)*	30.79	30.36	33.26	32.42	32.10
SEP-COMB(49)*	30.90	30.36	33.32	32.42	32.17
SEP-TD(49)*	30.91	30.36	33.33	32.42	32.19
SEP-COMB(64)*	30.94	30.36	33.34	32.42	32.25
SEP-TD(64)*	30.94	30.36	33.34	32.42	32.25
SEP-COMB(36)-Barbara*	–	30.28	32.41	32.43	31.97
SEP-TD(36)-Barbara*	–	30.29	32.43	32.43	32.01
SEP-COMB(64)-Barbara*	–	30.36	33.28	32.43	32.23
SEP-TD(64)-Barbara*	–	30.36	32.29	32.43	32.24

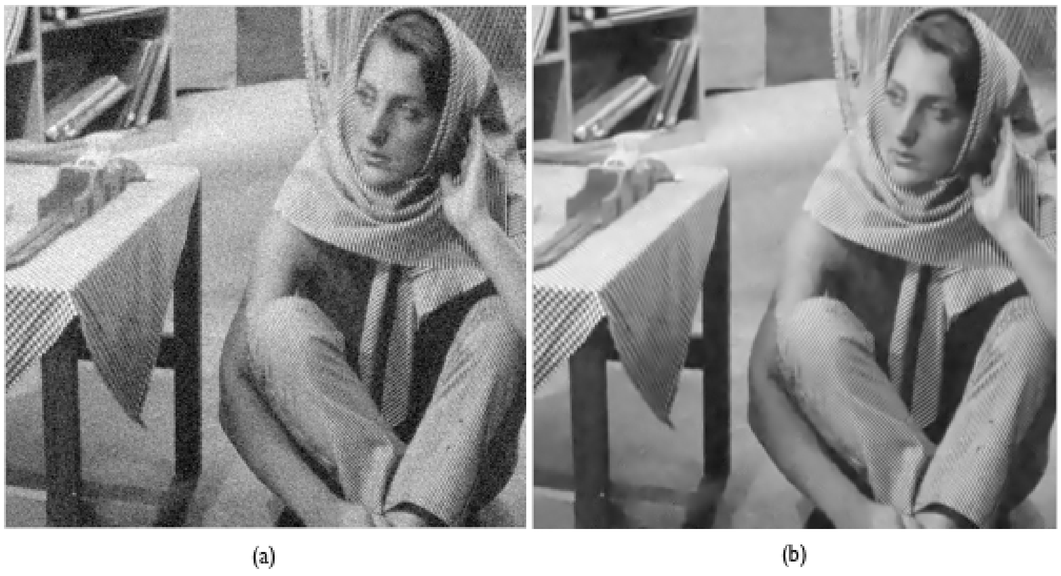

Figure 6.10: An example denoising experiment using the reconstructed filter bank of the SEP-TD approach. (a) Barbara image corrupted with additive white Gaussian noise of standard deviation 20, (b) Denoised image using the the approximated filter bank reconstructed with our SEP-TD approach.

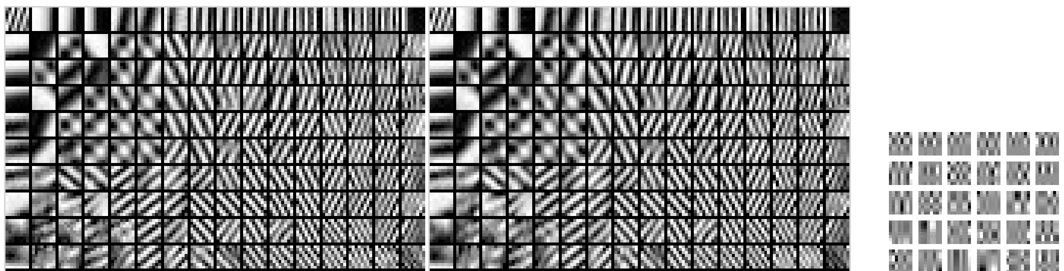

Figure 6.11: Approximation of an existing filter bank. (a) Filter bank learned by the K-SVD algorithm on the Barbara image (b) Filter bank approximated by the SEP-TD approach (c) Separable filter bank consisting of 36 filters learned via SEP-TD. The original and approximated filter banks yield similar denoising results.

6.4 Convolutional Neural Networks

Separable filter learning is important for the tasks that use intense convolutional computations. Convolutional Neural Network is one such example that is used to recognize visual patterns from pixel images. They are able to recognize patterns with too much variability while being robust to distortions and simple geometric transformations. Their training is carried out by back-propagation algorithm as almost all other neural networks. The difference comes from their special architecture. CNN's have alternating layers of convolution layers and sub-sampling/pooling layers in their architecture. The convolution layers constructs feature maps by convolving trained kernels over feature maps in layers below them. The subsampling layers downsample the feature maps in one previous layer by a constant factor. In order to discriminate between C classes, an output layer consisting of connections to C neurons are used. The output layer takes as input the concatenated feature maps of the layer below it (feature vectors) and decides which output class is going to be assigned to the pixel image.

In the experiment, an architecture consisting of 2 convolution and 2 subsampling layers is assumed. The first layer consists of 6 feature maps connected to the single output layer via 6 5×5 kernels. The second layer is a 2×2 mean subsampling layer. The third layer consists of 12 feature maps connected to the 6 mean-pooling layers via 72 5×5 kernels. The fourth layer is a 2×2 subsampling layer. The feature maps obtained at the last convolution layer is concatenated into feature vectors and fed into the last layer which have 10 output neurons in order to discriminate between 10 handwritten digit characters.

Separable filter learning is carried out at each convolution layer. In the first convolution layer, there exist 6 learned arbitrary non-separable kernels. These kernels are stacked together and a tensor is formed. Then by applying tensor decomposition, a separable basis consisting of 3 separable filters is learned for these 6 non-separable filters in the first convolution layer. In the second convolution layer, a tensor is formed for each outgoing node, thus 6 tensors each consisting of 12 filters are obtained. A decomposition is applied on all these tensors. A separable basis consisting of 4 separable filters is learned for each tensor. Also, the coefficients to reconstruct the non-separable filter banks in the two convolution layers are learned during tensor decomposition. The convolutions are carried out with separable filters, then by combining the separable convolution results with the obtained set of weights, an approximation of the non-separable convolution is obtained for each 72 kernel in the second layer. This way the computation time is reduced. Using separable filters the classification can be carried out in less time without losing accuracy. For a kernel size of 5 the computation time for non-separable filters is 5.7 seconds, whereas for separable filters, it is 4.1 seconds with a similar number of misclassifications.

The CNN is trained using stochastic gradient descent on the full MNIST dataset. The MNIST dataset [22] consists of 70000 training images of handwritten digits. The images are grayscale and 28 by 28 pixels. The training dataset has 60000 images and the test dataset has 10000 images. A batch size of 50 is used for 100 epochs and a test score of 1.15% or 115 misclassifications is obtained. The training takes a long time for 100 epochs to be used in practical applications. Fig. 6.12 shows a set of misclassified letters in the object recognition task carried out with CNN. Fig. 6.13 shows the 12 filters from an outgoing node of the second convolutional layer, and 4 separable basis filters learned from them.

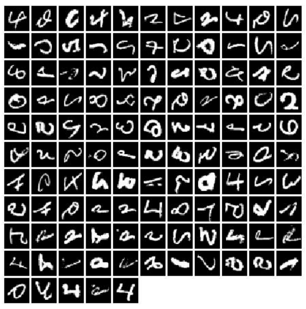

Figure 6.12: A set of misclassified images using SEP-TD approach in Convolutional Neural Networks.

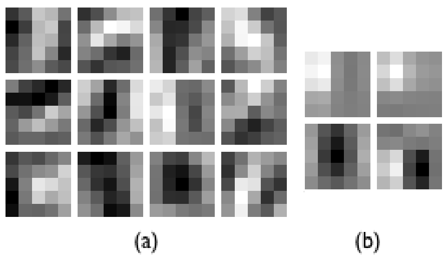

Figure 6.13: A set of learned separable filters for convolutional neural networks. (a)12 kernels outgoing from the first input map of the second convolutional layer, (b)4 separable kernels learned by tensor decomposition

Chapter 7

Conclusions

Convolutional operations are fundamental to many image processing tasks, such as feature extraction. When the convolutional filters are numerous and not separable, the completion of the task becomes computationally demanding, which hinders these applications to be used effectively in practical situations. In this thesis work, a new algorithm for learning separable basis filters from an arbitrary filter bank is developed and implemented. Specifically, the non-separable filters are constrained globally, by grouping them together into a tensor and solving a tensor decomposition optimization problem. By this procedure, the individual filters in the tensor can be expressed as the linear combination of separable filters.

It has been seen by comparative studies that the tensor decomposition approach learns a basis of separable filters that closely approximates an existing filter bank. Separable filters learned by our approach obtains the same performance of the original filter bank in several computer vision tasks while increasing the computational efficiency. The increase in efficiency is achieved through reducing the number of required filters and employing separability.

The approach is applicable to any arbitrary filter bank used in computer vision tasks. One application of this approach would be in designing handcrafted filters. The designers do not need to confine themselves for designing separable filters in the first place. They can select arbitrary filters suitable to the specific application and approximate them with a small number of separable filters.

The approach we have designed to learn separable filters with tensor decomposition is orders of magnitude faster than the previous separable filter learning algorithm using dictionary learning methods. Hence, any generic arbitrary filter bank can be represented with a separable filter basis effectively with our approach.

Appendix A

Partial Derivatives of the Tensor Decomposition Objective Function

In Section 4.3, the partial derivatives of the tensor decomposition of the objective function is computed as in the following theorem [1].

Theorem 4.1. The partial derivatives of the objective function f can be found via

$$\frac{\partial f}{\partial \mathbf{a}_r^{(n)}} = -\left(\mathcal{Z} \bigtimes_{m=1, m\neq n}^{N} \mathbf{a}_r^{(m)} \right) + \sum_{\ell=1}^{R} \gamma_{r\ell}^{(n)} \mathbf{a}_\ell^{(n)} \tag{A.1}$$

where $r = 1, \ldots, R$ and $n = 1, \ldots, N$ with $\gamma_{r\ell}^{(n)}$ defined as

$$\gamma_{r\ell}^{(n)} = \prod_{m=1, m\neq n}^{N} \mathbf{a}_r^{(m)T} \mathbf{a}_\ell^{(m)} \tag{A.2}$$

Proof. Rewriting the objective function given in Eq. (4.21) as three summands, the following is obtained:

$$f(\mathbf{x}) = \frac{1}{2}||\mathcal{Z}||^2 - \langle \mathcal{Z}, [[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]] \rangle + \frac{1}{2} \left|\left|[[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]]\right|\right|^2 \tag{A.3}$$

where the first summand is $f_1(\mathbf{x})$, the second summand is $f_2(\mathbf{x})$ and the third summand is $f_3(\mathbf{x})$.

The first summand does not involve any variables, therefore $\frac{\partial f_1}{\partial \mathbf{a}_r^{(n)}} = \mathbf{0}$. where $\mathbf{0}$ is a zero vector of length I_n. The second summand can be written as

$$f_2(\mathbf{x}) = \langle \mathcal{Z}, [[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]] \rangle \tag{A.4}$$

$$\langle \mathcal{Z}, \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \ldots \circ \mathbf{a}_r^{(N)} \rangle \tag{A.5}$$

$$\sum_{r=1}^{R} \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} z_{i_1 i_2 \ldots i_N} a_{i_1 r}^{(1)} a_{i_2 r}^{(2)} \ldots a_{i_N r}^{(N)} \tag{A.6}$$

$$\sum_{r=1}^{R} \left(\mathcal{Z}) \bigtimes_{m=1}^{(N)} \mathbf{a}_r^{(m)} \right) \tag{A.7}$$

$$\sum_{r=1}^{R} \left(\mathcal{Z}) \bigtimes_{m=1}^{(N)} \mathbf{a}_r^{(m)} \right)^T \mathbf{a}_r^{(n)} \tag{A.8}$$

Writing f_2 in this form, it is clear to conclude that

$$\frac{\partial f_2}{\partial \mathbf{a}_r^{(n)}} = \left(\mathcal{Z} \bigtimes_{m=1, m \neq n} \mathbf{a}_r^{(m} \right) \tag{A.9}$$

The third summand is

$$f_3(\mathbf{x}) = \left|\left|[[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]]\right|\right|^2 \tag{A.10}$$

$$= \langle \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \ldots \circ \mathbf{a}_r^{(N)}, \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \ldots \mathbf{a}_r^{(N)} \rangle \tag{A.11}$$

$$= \sum_{k=1}^{R} \sum_{\ell=1}^{R} \prod_{m=1}^{N} \mathbf{a}_k^{(m)T} \mathbf{a}_\ell^{(m)} \tag{A.12}$$

$$= \prod_{m=1}^{N} \mathbf{a}_r^{(m)T} \mathbf{a}_r^{(m)} + 2 \sum_{\ell=1, \ell \neq r}^{R} \prod_{m=1}^{N} \mathbf{a}_r^{(m)T} \mathbf{a}_\ell^{(m)} + \sum_{k=1, k \neq r}^{R} \sum_{\ell=1, \ell \neq r}^{N} \mathbf{a}_k^{(m)T} \mathbf{a}_\ell^{(m)} \tag{A.13}$$

Thus,

$$\frac{\partial f_3}{\partial \mathbf{a}_r^{(n)}} = 2 \left(\prod_{m=1, m \neq n}^{N} \mathbf{a}_r^{(m)T} \mathbf{a}_r^{(m)} \right) \mathbf{a}_r^{(n)} + 2 \sum_{\ell=1, \ell \neq r}^{R} \left(\prod_{m=1, m \neq n}^{N} \mathbf{a}_r^{(m)T} \mathbf{a}_\ell^{(m)} \right) \mathbf{a}_\ell^{(n)} \tag{A.14}$$

$$= 2 \sum_{\ell=1}^{R} \left(\prod_{m=1, m \neq n}^{N} \mathbf{a}_r^{(m)T} \mathbf{a}_\ell^{(m)} \right) \mathbf{a}_\ell^{(n)}. \tag{A.15}$$

Combining (A.9) and (A.15), the desired result is obtained. □

Bibliography

[1] Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, February 2011.

[2] Evrim Acar and Bulent Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 2008.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[4] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing 2011 edition, October 2007.

[6] R. Bro, A. Smilde, and P. Geladi. *Multi-Way Analysis - Applications in the Chemical Sciences*. Wiley, England, 2004.

[7] J. Carroll and Jih J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of 'Eckart-Young' decomposition. *Psychometrika*, 35(3):283–319, September 1970.

[8] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.*, 57(11):1413–1457, 2004.

[9] M. Elad and M. Aharon. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, December 2006.

[10] Clment Farabet, Berin Martini, Polina Akselrod, Seluk Talay, Yann LeCun, and Eugenio Culurciello. Hardware accelerated convolutional neural networks for synthetic vision systems. In *ISCAS*, pages 257–260. IEEE, 2010.

[11] Maryam Fazel, Haitham Hindi, and Stephen P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *In Proceedings of the 2001 American Control Conference*, pages 4734–4739, 2001.

[12] Gene H. Golub and Charles F. van Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*. The Johns Hopkins University Press, 3rd edition, October 1996.

[13] Germán González, François Fleuret, and Pascal Fua. Learning rotational features for filament detection. In *CVPR*, pages 1582–1589, 2009.

[14] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):84, 1970.

[15] Geoffrey E. Hinton. Learning to represent visual input. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1537):177–184, January 2010.

[16] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153. IEEE, 2009.

[17] Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michal Mathieu, and Yann LeCun. Learning convolutional feature hierarchies for visual recognition. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *NIPS*, pages 1090–1098. Curran Associates, Inc., 2010.

[18] Tamara G. Kolda. Multilinear operators for higher-order decompositions. Technical report, 2006.

[19] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM REVIEW*, 51(3):455–500, 2009.

[20] J. B. Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra Applicat.*, 18, 1977.

[21] Max W. Law and Albert C. Chung. Three dimensional curvilinear structure detection using optimally oriented flux. In *Proceedings of the 10th European Conference on Computer Vision: Part IV*, ECCV '08, pages 368–382, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[23] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Andrea Pohoreckyj Danyluk, Lon Bottou, and Michael L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*, page 77. ACM, 2009.

[24] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[25] Franck Mamalet and Christophe Garcia. Simplifying convnets for fast learning. In *Proceedings of the 22nd international conference on*

Artificial Neural Networks and Machine Learning - Volume Part II, ICANN'12, pages 58–65, Berlin, Heidelberg, 2012. Springer-Verlag.

[26] Marina Meilă. Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98(5):873–895, May 2007.

[27] Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV (6)*, volume 6316 of *Lecture Notes in Computer Science*, pages 210–223. Springer, 2010.

[28] Bruno A. Olshausen and David J. Fieldt. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[29] Pietro Perona. Deformable kernels for early vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(5):488–499, May 1995.

[30] Hamed Pirsiavash. Steerable part models. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3226–3233, Washington, DC, USA, 2012. IEEE Computer Society.

[31] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. Curran Associates, Inc., 2007.

[32] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann Lecun. Efficient learning of sparse representations with an energy-based model. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS 2006*, pages 1137–1144. MIT Press, 2006.

[33] R. Rigamonti, M. A. Brown, and V. Lepetit. Are sparse representations really relevant for image classification? In *Proceedings of the 2011 IEEE*

Conference on Computer Vision and Pattern Recognition, CVPR '11, pages 1545–1552, Washington, DC, USA, 2011. IEEE Computer Society.

[34] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua. Learning separable filters. In *Conference on Computer Vision and Pattern Recognition*, 2013.

[35] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Double sparsity: learning sparse dictionaries for sparse signal approximation. *Trans. Sig. Proc.*, 58(3):1553–1564, March 2010.

[36] A. Santamaría-Pang, C. Colbert, P. Saggau, and I. Kakadiaris. Automatic Centerline Extraction of Irregular Tubular Structures Using Probability Volumes from Multiphoton Imaging. In Nicholas Ayache, Sébastien Ourselin, and Anthony Maeder, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2007*, volume 4792 of *Lecture Notes in Computer Science*, chapter 59, pages 486–494. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2007.

[37] B. Tekin, U. S. Kamilov, E. Bostan, and M. Unser. Benefits of Consistency in Image Denoising with Steerable Wavelets. In *International Conference on Acoustics, Speech and Signal Processing*, 2013.

[38] Sven Treitel and J.L. Shanks. The design of multistage separable planar filters. *Geoscience Electronics, IEEE Transactions on*, 9(1):10–27, 1971.

[39] Ranjith Unnikrishnan, Caroline Pantofaru, and Martial Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):929–944, June 2007.

[40] C. van Rijsbergen. Foundation of evaluation. *"Journal of Documentation"*, 1974.

[41] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition, 2009.

[42] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. In *CVPR*, 2010.