Learning Separable Filters

Amos Sironi*, Bugra Tekin*, Roberto Rigamonti, Vincent Lepetit and Pascal Fua, *IEEE Fellow*

Abstract—Learning filters to produce sparse image representations in terms of overcomplete dictionaries has emerged as a powerful way to create image features for many different purposes. Unfortunately, these filters are usually both numerous and non-separable, making their use computationally expensive.
In this paper, we show that such filters can be computed as linear combinations of a smaller number of separable ones, thus greatly reducing the computational complexity at no cost in terms of performance. This makes filter learning approaches practical even for large images or 3D volumes, and we show that we significantly outperform state-of-the-art methods on the curvilinear structure extraction task, in terms of both accuracy and speed. Moreover, our approach is general and can be used on generic convolutional filter banks to reduce the complexity of the feature extraction step.

Index Terms—Convolutional sparse coding, filter learning, features extraction, separable convolution, segmentation of linear structures, image denoising, convolutional neural networks, tensor decomposition.

1 Introduction

It has been shown that representing images as sparse linear combinations of learned filters [26] yields effective approaches to image denoising and object recognition, which outperform those that rely on hand-crafted features [39]. Among these, convolutional formulations have emerged as particularly appropriate to handle whole images, as opposed to independent patches [15], [21], [30], [40]. Unfortunately, because the filters are both numerous and not separable, they tend to be computationally demanding, which has slowed down their acceptance. Their computational cost is even more damaging when dealing with large 3D image stacks, such as those routinely acquired for biomedical purposes.

In this paper, we show that we can preserve the performance of these convolutional approaches while drastically reducing their cost by learning and using separable filters that approximate the non-separable ones. Fig. 1 depicts this behavior in the case of filters designed to classify whether or not a pixel belongs to a blood vessel in retinal scans. Using the learned separable filters is much faster than using either the original non-separable ones or a state-of-the-art implementation of the FFT for all practical filter sizes. We will demonstrate that this is consistently true over a wide range of images.

As we will see, such a result could be achieved by enforcing the separability constraint as part of a convolutional, ℓ_1-based learning framework to directly learn a set of separable filters. However, we have found that an even better result could be obtained by first learning a set of non-separable filters, such as those of Fig. 1(a), and then a second smaller set of separable filters, such as those of Fig. 1(b), whose linear combinations can be used to represent the original ones. This makes the approach very efficient because it only requires convolutions with a small set of separable filters, which greatly reduces the run-time computational requirements at no loss in accuracy. It also makes the approach very general because it can be applied to any non-separable filter bank, no matter how it has been learned or designed.

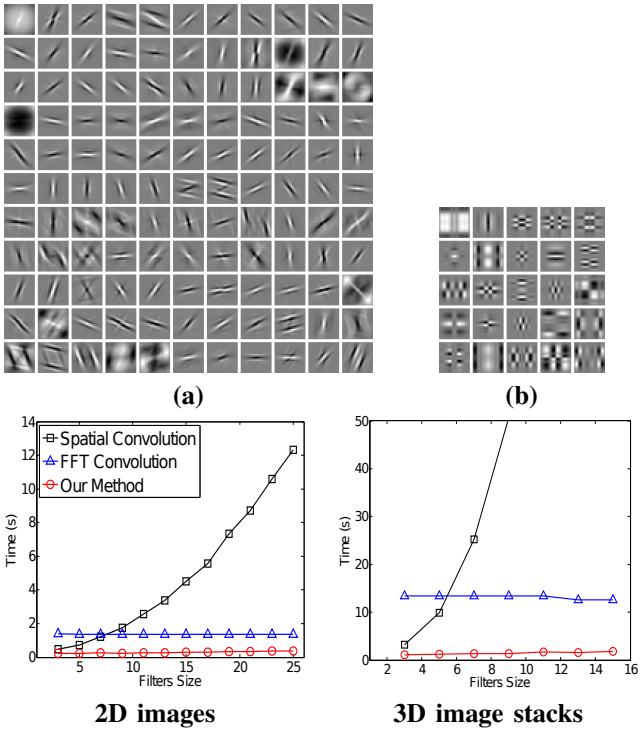


Fig. 1: Convolutional filter bank **(a)** learned for the extraction of curvilinear structures in retinal scan images, along with its separable approximation **(b)**. The full-rank filters of **(a)** can be approximated very precisely as linear combinations of the far fewer separable filters of **(b)**. This allows us to use this property to considerably speed up extraction of learned image features compared with convolutions with the original non-separable filters, even when Fast Fourier Transform is used for both the 2D and the 3D case, as it is shown by the figures in the bottom row.

• *The authors are with the Computer Vision Laboratory, IC Faculty, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne CH-1015, Switzerland.*
E-mail: firstname.lastname@epfl.ch
* *indicates equal contribution.*

We first introduced this idea in [31], where the separable filters are learned by minimizing an objective function that includes low-rank constraints. This approach delivers the desired run-time accuracy and efficiency but at a high-computational cost during the learning phase. Moreover, the low-rank constraints, being soft constraints, do not guarantee that the resulting filters have rank one. In this situation, separability can be imposed after the fact by truncating the singular values of the filters. Here, we introduce a second approach that explicitly treats the separable filters as products of 1D components, which we learn by tensor decomposition. Our experiments show that this second approach is faster at learning time and more accurate.

In the remainder of the paper, we first discuss related work. We then introduce our approach to separable approximation, first in 2D and then in 3D. Finally, we test our methods on different applications—pixel and voxel classification as well as image denoising and neural networks—and show that the speed-up is systematically significant at no loss in accuracy.

2 Related work

Automatic feature learning has long been an important area in Machine Learning and Computer Vision. Neural networks [18], Restricted Boltzmann Machines [14], Auto-Encoders [4], Linear Discriminant Analysis [5], and many other techniques have been used to learn features in either supervised or unsupervised ways. Recently, creating an overcomplete dictionary of features—sparse combinations of which can be used to represent images—has emerged as a powerful tool for object recognition [8], [15], [39] and image denoising [9], [22], among others.

However, for most such approaches, run-time feature extraction can be very time-consuming because it involves convolving the image with many non-separable non-sparse filters. It was proposed several years ago to split convolution operations into convergent sums of matrix-valued stages [36]. This principle was exploited in [28] to avoid coarse discretization of the scale and orientation spaces, yielding steerable separable 2D edge-detection kernels. This approach is powerful but restricted to kernels that are decomposable in the suggested manner, which precludes the potentially arbitrary ones that can be found in a learned dictionary or a handcrafted one to suit particular needs. After more than a decade in which the separability property has been either taken for granted or neglected, there is evidence of renewed interest [23], [29]. The scope of those papers is, however, limited in that they are restricted to specific frameworks, while our approach is completely generic. Nonetheless, they prove a growing need for fast feature extraction methods.

Among other recent feature-learning publications, very few have revisited the run-time efficiency issue. The majority of those advocate exploiting the parallel capabilities of modern hardware [10], [25]. However, programming an FPGA unit as in [10] is cumbersome. Exploiting the Graphics Processing Unit as in [25] is an attractive alternative, but the time required for memory transfers between the CPU and the GPU is often prohibitive in practice.

An interesting recent attempt at reducing computational complexity is the approach of [33], which involves learning a filter bank by composing a few atoms from an handcrafted separable dictionary. Our own approach is in the same spirit but is much more general as we also learn the atoms. As shown in the results section, this results in a smaller number of separable filters that are tuned for the task at hand.

[13] learns separable dictionaries in the case of classical sparse coding, i.e., not in a convolution-based approach. The authors show that using separable items as compared to unstructured ones, it is possible to deal with larger images. However the dimensions of the images used are smaller than those typically handled by convolutional sparse coding approaches. Moreover, we will show that directly learning separable filters yields results worse than those of their unstructured counterpart. Here we overcome this limitation by introducing separability at a later stage of the learning process. We first learn a set of non-separable filters and then approximate them as linear combinations of a small set of separable ones, which are specific for the particular application.

Finally, the authors of [7] propose a way to reduce the time it takes to learn non-separable filters. This makes their approach complementary to ours as it could be used to speed up the first step of our algorithm.

3 Learning 2D Separable Filters

Most dictionary learning algorithms operate on image patches [8], [22], [26], but convolutional approaches [15], [21], [30], [40] have been recently introduced as a more natural way to process arbitrarily-sized images. They generalize the concept of *feature vector* to that of *feature map*, a term borrowed from the Convolutional Neural Network literature [19]. In our work, we consider the convolutional extension of Olshausen and Field's objective function proposed in [30].

Formally, J filters $\{\mathbf{f}^j\}_{1\leq j\leq J}$ are computed as

$$\underset{\{\mathbf{f}^j\},\{\mathbf{m}_i^j\}}{\operatorname{argmin}} \sum_i \left(\left\| \mathbf{x}_i - \sum_{j=1}^{J} \mathbf{f}^j * \mathbf{m}_i^j \right\|_2^2 + \lambda_1 \sum_{j=1}^{J} \left\| \mathbf{m}_i^j \right\|_1 \right), \quad (1)$$

where

- $\mathbf{x}_i$ is an input image;
- $*$ denotes the convolution product operator;
- $\{\mathbf{m}_i^j\}_{1\leq j\leq J}$ is the set of feature maps extracted during learning;
- λ_1 is a regularization parameter.

A standard way to solve Eq. (1) is to alternatively optimize over the $\mathbf{m}_i^j$ representations and the $\mathbf{f}^j$ filters. Stochastic gradient descent is used for the latter, while the former is achieved by first taking a step in the direction opposite to the ℓ_2-penalized term gradient and then applying the soft-thresholding operation[1] on the $\mathbf{m}_i^j$s.

While this formulation achieves state-of-the-art results [32], the required run-time convolutions are costly because the

1. Soft-thresholding is the proximal operator for the ℓ_1 penalty term [3]; its expression is $\text{prox}_\lambda(x) = \text{sgn}(x)\max(|x| - \lambda, 0)$. Proximal operators allow to extend gradient descent techniques to some nonsmooth problems.

resulting filters are not separable. Quantitatively, if $\mathbf{x}_i \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{f}^j \in \mathbb{R}^{d_1 \times d_2}$, extracting the feature maps requires $\mathcal{O}(n_1 \cdot n_2 \cdot d_1 \cdot d_2)$ multiplications and additions. By contrast, if the filters were separable, the computational cost would drop to a more manageable $\mathcal{O}(n_1 \cdot n_2 \cdot (d_1 + d_2))$. This cost reduction becomes even more desirable in biomedical applications that require processing large 3D image stacks.

Our goal is therefore to make our filters separable without compromising their descriptive power. One way to do this would be to explicitly write the $\mathbf{f}^j$ filters as products of 1D filters and to minimize the objective function of Eq. (1) in terms of their coefficients. Unfortunately, this would involve a quartic objective function in terms of the unknowns in addition to the ℓ_1 norm of the feature maps and, therefore, a very difficult optimization problem.

In the remainder of this section, we introduce two different solutions to overcoming this problem. The first involves a slight modification of the objective function of Eq. (1) to make the learned filters separable by lowering their rank. The second exploits the fact that arbitrary filters of rank R can be expressed as linear combinations of R separable filters [28]. This second solution is more general than the first as it can be applied to any filter bank. It involves approximating the $\mathbf{f}^j$ filters of Eq. (1) with linear combinations of separable filters, and we give two methods two obtain these filters. We will show in the Results section that the resulting separable filters, unlike those obtained using the first solution, retain the full discriminative power of the full-rank ones.

3.1 Penalizing High-Rank Filters

A straightforward approach to finding low-rank filters is to add a penalty term to the objective function of Eq. (1) and to solve

$$\underset{\{\mathbf{s}^j\},\{\mathbf{m}_i^j\}}{\operatorname{argmin}} \sum_i \left(\left\| \mathbf{x}_i - \sum_{j=1}^{J} \mathbf{s}^j * \mathbf{m}_i^j \right\|_2^2 + \Gamma_{\mathbf{m},\mathbf{s}}^i \right) , \tag{2}$$

$$\text{with } \Gamma_{\mathbf{m},\mathbf{s}}^i = \lambda_1 \sum_{j=1}^{J} \left\| \mathbf{m}_i^j \right\|_1 + \lambda_* \sum_{j=1}^{J} \left\| \mathbf{s}^j \right\|_* , \tag{3}$$

where the $\mathbf{s}^j$s are the learned linear filters, $\|\cdot\|_*$ is the nuclear norm, and λ_* is an additional regularization parameter. The nuclear norm of a matrix is the sum of its singular values and is a convex relaxation of the rank [11]. Thus, forcing the nuclear norm to be small amounts to lowering the rank of the filters. Experimentally, for sufficiently high values of λ_*, the $\mathbf{s}^j$ filters become effectively rank 1 and they can be written as products of 1D filters.

Solving Eq. (2), which has the nuclear norm of the filters as an additional term compared to Eq. (1), requires minimal extra effort. After taking a step in the direction opposite of that of the gradient of the filters, as described in the previous section, we just have to apply the proximal operator of the nuclear norm to the filters. This amounts to performing a Singular Value Decomposition (SVD) $\mathbf{s} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ on each filter $\mathbf{s}$, soft-thresholding the values of the diagonal matrix $\mathbf{D}$ to obtain a new matrix $\widehat{\mathbf{D}}$, and replacing $\mathbf{s}$ by $\mathbf{U}\widehat{\mathbf{D}}\mathbf{V}^\top$. At convergence, to make sure we obtain separable filters, we apply a similar SVD-based operation but set to 0 all the singular values but the largest one. In practice, the second largest singular value is already almost zero even before clipping.

Choosing appropriate values for the optimization parameters, the gradient step size, λ_1, and λ_*, is challenging because they express contrasting needs. We have found it effective to start with a low value of λ_*, solve the system, and then progressively increase it until the filter ranks are close to one.

3.2 Linear Combinations of Separable Filters

We will show in the Results section that the separable filters obtained using the method discussed above give less accurate classification results than the non-separable ones. This is probably because the additional constraints imposed on the filters make the problem too hard to solve optimally, as observed in [13] when learning separable dictionaries for denoising purposes.

We therefore introduce a second approach that exploits the fact that a filter $\mathbf{f}^j$ of rank R can always be written as

$$\mathbf{f}^j = \sum_{k=1}^{R} w_j^k \mathbf{s}^{j,k} , \tag{4}$$

where the $\mathbf{s}^{j,k}$ filters are separable, or equivalently of rank one, and the w_j^k are scalar weights.

In the 2D case, such a representation could be obtained by SVD decomposition of each one of the J $\mathbf{f}^j$ filters of Eq. (1) independently. However, this would yield a different bank of filters for each $\mathbf{f}^j$ and would be inefficient at run-time. To avoid this, we impose that each $\mathbf{f}^j$ be written as

$$\mathbf{f}^j = \sum_{k=1}^{K} w_j^k \mathbf{s}^k , \tag{5}$$

where the separable filters $\mathbf{s}^k$ are shared among all the non-separable ones and only the coefficients w_j^k's change. In this way, convolving the image with all the $\mathbf{f}^j$'s at run-time amounts to convolving it with the separable $\mathbf{s}^k$ filters and then linearly combining the results, without any further convolutions.

3.2.1 Minimizing the Nuclear Norm

A most direct way to learn the separable filters $\mathbf{s}^k$ and scalar coefficients w_j^k of Eq. 5 would be to solve

$$\underset{\substack{\{\mathbf{m}_i^j\} \\ \{\mathbf{s}^k\},\{w_k^j\}}}{\operatorname{argmin}} \sum_i \left(\left\| \mathbf{x}_i - \sum_{j=1}^{J} \left(\sum_{k=1}^{K} w_j^k \mathbf{s}^k \right) * \mathbf{m}_i^j \right\|_2^2 + \Gamma_{\mathbf{m},\mathbf{s}}^i \right) , \tag{6}$$

where $\Gamma_{\mathbf{m},\mathbf{s}}^i$ is defined in Eq. (3) and includes the nuclear norm. This is a reformulation of Eq. (2) with $w_j^k = 1$ if $j = k$ and 0 otherwise. Unfortunately, this objective function is difficult to optimize because its first term contains products of three unknowns, in addition to the second term involving the ℓ_1 norm of the feature maps and the nuclear norm of the filters.

A standard way to handle this difficulty is to introduce auxiliary unknowns, making the formulation linear by introducing additional parameters. Parameter tuning is, however, already

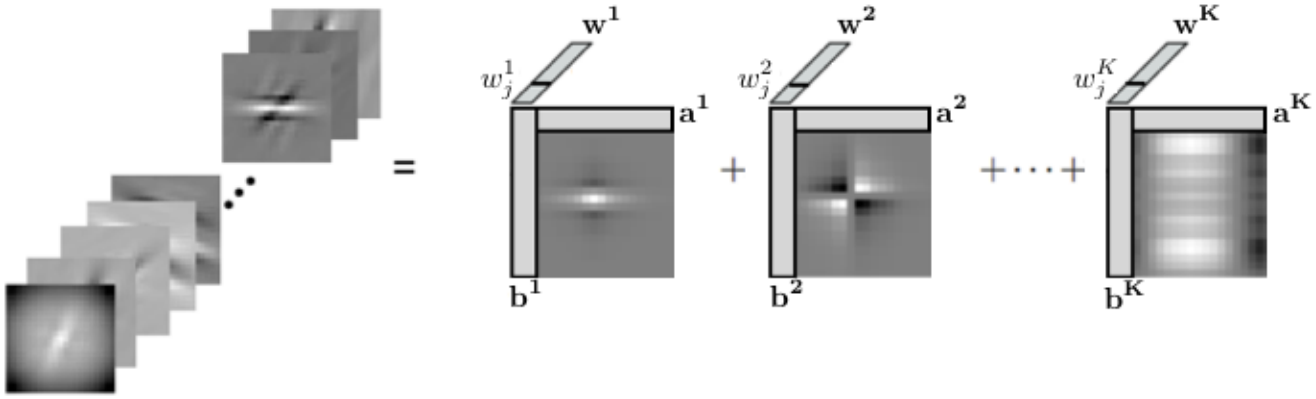


Fig. 2: Tensor decomposition for learning separable filters. Left: A bank of two-dimensional filters is stacked together to form a 3-dimensional tensor, Right: The tensor is decomposed in the sum of K rank-one tensors. Thus, the original filters are approximated by the weighted sum of the separable filters $\mathbf{s}_k = \mathbf{a}_k \circ \mathbf{b}_k$.

difficult in the formulation of Eq. (1), and this would therefore only worsen the situation. We tried instead a simpler approach, which has yielded better results by decoupling the computation of the non-separable filters from that of the separable ones. We first learn a set of non-separable filters $\{\mathbf{f}^j\}_j$ by optimizing the original objective function of Eq. (1). We then look for separable filters whose linear combinations approximate the $\mathbf{f}^j$ filters by solving

$$\underset{\{\mathbf{s}^k\},\{w_k^j\}}{\operatorname{argmin}} \sum_j \left\| \mathbf{f}^j - \sum_{k=1}^{K} w_j^k \mathbf{s}^k \right\|_2^2 + \lambda_* \sum_{k=1}^{K} \left\| \mathbf{s}^k \right\|_* . \tag{7}$$

Even though this may seem suboptimal when compared to the global optimization scheme of Eq. (6), it gives superior results in practice because the optimization process is split into two easier tasks and depends on just two parameters, easing their scheduling.

To solve Eq. (7), we alternately optimize on the filters $\mathbf{s}^k$ and on the weights w_j^k using gradient descent. At each iteration, after moving in the gradient direction, we apply the proximal operator of the nuclear norm to the filters, as described in Section 3.1 to minimize the criterion of Eq. (2).

3.2.2 Tensor Decomposition

The method described above produces a small set of separable filters that approximate the original ones. However, it requires introducing an additional regularization parameter λ_* that can be difficult to tune. Moreover, we have experimentally found that its convergence rate is slow, especially when trying to approximate high-rank filters. We therefore introduce here an alternative approach to finding the separable filters $\mathbf{s}^k$ and weights w_j^k of Eq. (5).

We start by stacking the J filters $\mathbf{f}^j \in \mathbb{R}^{d_1 \times d_2}$ into a 3-dimensional tensor $\mathcal{F} \in \mathbb{R}^{d_1 \times d_2 \times J}$, where the j-th slice of $\mathcal{F}$ corresponds to the j-th filter $\mathbf{f}^j$, as shown in Fig. 2.

Writing the slices of $\mathcal{F}$ as linear combinations of rank-one matrices is equivalent to writing the tensor $\mathcal{F}$ as a linear combination of rank-one tensors

$$\mathcal{F} = \sum_{k=1}^{K} \mathbf{a}^k \circ \mathbf{b}^k \circ \mathbf{w}^k \ , \tag{8}$$

where $\mathbf{a}^k$ is a vector of length d_1, $\mathbf{b}^k$ a vector of length d_2 and $\mathbf{w}^k$ a vector of length J. The symbol $\circ$ corresponds to the tensor product, that for vectors is also referred to as outer product. Such a decomposition is called Canonical Polyadic Decomposition (CPD) [16] of the tensor $\mathcal{F}$ and the right-hand side of Eq. (8) is called Kruskal form of the tensor. We will refer to K as the rank of the Kruskal tensor.

Thus, if tensor $\mathcal{F}$ can be written in the form of Eq. (8), we obtain

$$\mathbf{f}^j = \sum_{k=1}^{K} w_j^k \mathbf{s}^k, \quad \forall j,$$

where the separable filters are given by the $\mathbf{a}^k$ and $\mathbf{b}^k$ components of the CPD, that is, $\mathbf{s}^k = \mathbf{a}^k \circ \mathbf{b}^k$. The coefficients w_j^k necessary to reconstruct the filter j are given by the j-th component of the $\mathbf{w}^k$ vectors, as shown in Fig. 2.

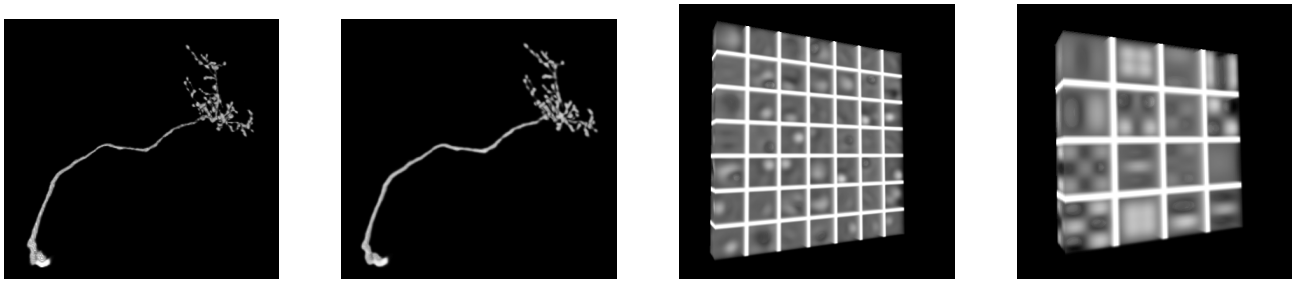
In general, for a given K, we have no guarantee that the decomposition of Eq. (8) exists. Thus, we will compute the best approximation of this form by solving the optimization problem

$$\min_{\{\mathbf{a}^k,\mathbf{b}^k,\mathbf{w}^k\}_k} \left\| \mathcal{F} - \sum_{k=1}^{K} \mathbf{a}^k \circ \mathbf{b}^k \circ \mathbf{w}^k \right\|_2^2 . \tag{9}$$

To this end, we use the CP-OPT algorithm of [1], implemented in the MATLAB tensor toolbox, in which Eq. (9) is solved by conjugate gradient descent. Details about the optimization procedure are given in the Appendix.

The rank K of the decomposition is the only parameter of the method. It determines the number of separable filters used to approximate the original filter bank.

In Section 5 we will show that the two strategies based on the linear combination of separable filters, presented in Section 3.2, yields the best results and that they are really close in terms of accuracy and efficiency. In Section 5.6 we will show that the Tensor Decomposition approach of Eq. (9) has faster convergence and gives a better approximation of the original filter bank, compared to the approach of Eq. (7) and therefore should be preferred in practice.



(a) **(b)** **(c)** **(d)**

Fig. 3: Examples of 3D filter banks learned on the OPF dataset [2]. **(a)** A 3D test image stack. **(b)** Response of the classifier trained on the separable filter bank output **(d)**. **(c)** Non-separable filter bank learned by optimizing Eq. (1). **(d)** The separable filter bank learned by optimizing Eq. (7).

4 Learning N-D Separable Filters

The computational complexity of feature extraction becomes even more daunting when handling multidimensional data, such as the 3D volume in Fig. 3. Fortunately, our approach to learning separable filters generalizes naturally to any dimension.

As will be shown in the Results section, the formalism of Sections 3.2.1 and 3.2.2 yields the best results in the 2D case and we therefore rely on it for the proposed extension.

Minimizing the Nuclear Norm: The structure of the optimization scheme of Eq. (7) of Section 3.2.1 is unchanged. It simply involves a CPD of the filters instead of the SVD decomposition used for the 2D filters, during the optimization procedure.

Tensor Decomposition: Similarly, the generalization of the method of Section 3.2.2 to N-D filters is straightforward. Let $\{\mathbf{f}^j\}_{j=1}^J$ be a set of filters, with $\mathbf{f}^j \in \mathbb{R}^{d_1 \times \cdots \times d_N}\ \forall j$. Let $\mathcal{F}$ be the $(N+1)$-D tensor formed by stacking the $\mathbf{f}^j$'s along the $(N+1)$-th dimension, that is $\mathcal{F}_{i_1,i_2,\ldots,i_N,j} = f^j_{i_1,i_2,\ldots,i_N}$. Applying CPD of rank K to $\mathcal{F}$, yields

$$\mathcal{F} \approx \sum_{k=1}^{K} \mathbf{a}^{k,1} \circ \mathbf{a}^{k,2} \circ \cdots \mathbf{a}^{k,N} \circ \mathbf{w}^k.$$

Therefore, for all $j = 1, \ldots, J$, $\mathbf{f}^j \approx \sum_{k=1}^{K} w_j^k \mathbf{s}^k$, with $\mathbf{s}^k = \mathbf{a}^{k,1} \circ \mathbf{a}^{k,2} \circ \cdots \mathbf{a}^{k,N}$.

In Section 5.6 we compare the tensor decomposition approach against the approach of Eq. (7), showing that the first one converges faster and gives lower approximations errors.

5 Results and Discussion

In this section, we compare the performance and computational complexity that results from using either separable filters or non-separable ones and different strategies for deriving them.

We first introduce these strategies and provide a theoretical analysis of their respective computational complexities. We then test them for three very different purposes:

- classifying pixels and voxels in biomedical images as belonging to linear structures or not;
- denoising;
- performing handwritten digit recognition using convolutional neural networks.

We will show that our separable filters systematically deliver a very substantial speed-up at no loss in performance in line with our theoretical analysis. The code and parameters for all these experiments are publicly available at `http://cvlab.epfl.ch/software`.

5.1 Competing Strategies

In the following, we will refer to the non-separable filters obtained by minimizing the objective function of Eq. (1) as *NON-SEP* and the separable ones learned using the technique of Section 3.1 as *SEP-DIRECT*. These essentially constitute baselines that reflect the current state-of-the-art. To provide additional baselines, we also compute separable *SEP-SVD* and *SEP-CPD* filters by approximating each *NON-SEP* filter by the outer product of its first left singular vector with its first right singular vector computed using SVD in 2D and by rank-1 CPD in 3D, which is the simplest way to approximate a non-separable filter by a separable one. For completeness' sake, we reimplemented *NON-SEP* using the Fast Fourier Transform to perform the convolutions. This approach is known to speed-up convolutions for large enough filters and we will refer to it as *NON-SEP-FFT*.

SEP-COMB and *SEP-TD* will denote the separable ones we advocate in this paper and whose linear combinations can be used to approximate the non-separable *NON-SEP* filters as described in Section 3.2. More specifically, *SEP-COMB* will refer to those that have been learned by minimizing the nuclear norm, as described in Section 3.2.1 and *SEP-TD* to those obtained by tensor decomposition, as discussed in Section 3.2.2.

Finally, although the *SEP-COMB* and *SEP-TD* filters can be used to write the non-separable ones as linear combinations of them, explicitly computing the coefficients of these combinations is not always necessary. For example, when the filters' output is to be fed to a linear classifier for classification purposes, this classifier can be trained directly on the separable-filters' output instead of that of the non-separable ones. This approach, which we will refer to as *SEP-COMB** and *SEP-TD**, further simplifies the run-time computations because the linear combinations' coefficients are then learned implicitly at training-time.

TABLE 1: Summary of the different methods used for our experiments, as described in Section 5.3.

Method Name	Filter Bank	Run Time Computations
NON-SEP	Non-separable filters learned from Eq. (1)	Spatial convolutions
NON-SEP-FFT	Non-separable filters learned from Eq. (1)	FFT convolutions
SEP-SVD	Approximation of *NON-SEP* by truncated SVD in 2D	Separable convolutions
SEP-CPD	Approximation of *NON-SEP* by rank-one CPD in 3D	Separable convolutions
SEP-DIRECT	Separable filters learned from Eq. (2)	Separable convolutions
SEP-COMB	Separable filters learned from Eq. (7)	Separable convolutions + linear combinations
SEP-TD	Separable filters learned from Eq. (9)	Separable convolutions + linear combinations
*SEP-COMB** and *SEP-TD**	As *SEP-COMB* and *SEP-TD*	Separable convolutions

5.2 Computational Complexity

The different methods described in the previous section are summarized in Table 1. Here, we provide an analysis of their computational complexities in terms of the number of multiplications required to perform the necessary run-time convolutions.

5.2.1 The 2D Case

Let $\mathbf{x} \in \mathbb{R}^{n_1 \times n_2}$ be an image we want to convolve with a filter bank $\{\mathbf{f}^j\}_{j=1}^J$, with $\mathbf{f}^j \in \mathbb{R}^{d_1 \times d_2}$.

In the *NON-SEP* case, J convolutions are computed in the spatial domain and each one requires $n_1 \cdot n_2 \cdot d_1 \cdot d_2$ multiplications, for a total of

$$\textit{NON-SEP}_{nop} = J \cdot n_1 \cdot n_2 \cdot d_1 \cdot d_2 \tag{10}$$

multiplications.

In the *NON-SEP-FFT* case, the convolutions are performed in the frequency domain, which involves the following steps:

- Padding $\mathbf{x}$ and $\mathbf{f}^j$ with zeros to have the same size $m_1 \times m_2$, where m_i is the closest power of 2 larger than $(n_i + d_i - 1)$, for $i = 1, 2$;
- Computing real-to-complex FFT on the padded image and the J filters;
- Multiplying the resulting discrete Fourier transform (DFT) of the image by that of each filter;
- Computing complex-to-real Inverse FFT (IFFT) on the results.

To decrease the total computational cost of the convolutions, we can precompute the FFT of the filters, at the cost of using more memory.

Assuming that each FFT and IFFT requires $c \cdot m_1 \cdot m_2 \cdot \log_2(m_1 \cdot m_2)$ complex multiplications where c depends on the specific FFT algorithm being used and that each complex multiplication requires 3 real multiplications, this yields a total of

$$\begin{aligned} \textit{NON-SEP-FFT}_{nop} &= 3 \cdot c \cdot m_1 \cdot m_2 \cdot \log_2(m_1 \cdot m_2) \\ &+ 3 \cdot J \cdot m_1 \cdot m_2 \\ &+ 3 \cdot J \cdot c \cdot m_1 \cdot m_2 \log_2(m_1 \cdot m_2) \end{aligned} \tag{11}$$

multiplications. In practice, the value used for the constant c is 2.

When the filters are separable, the cost of a spatial convolution reduces to $n_1 \cdot n_2 \cdot (d_1 + d_2)$. If the filter bank is composed of K filters, this represents

$$\text{SEP*}_{nop} = K \cdot (n_1 \cdot n_2 \cdot (d_1 + d_2)) \tag{12}$$

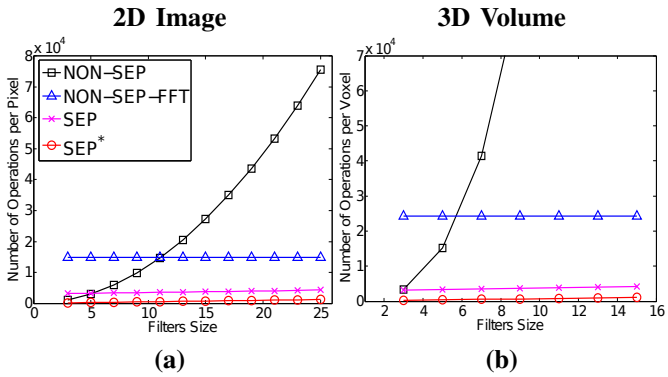


Fig. 4: Number of operations per pixel to compute convolutions, as a function of the filters size. **(a)** An image of 488×488 pixels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. **(b)** A volume of $114 \times 114 \times 50$ voxels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. The theoretical values are very similar to the experimental time shown in Fig. 1 and they show that the number of operations needed to compute the convolutions with our approach is smaller than both spatial convolution and FFT based convolution.

multiplications. This is the total cost for *SEP-COMB** and *SEP-TD**. In the cases of *SEP-COMB* and *SEP-TD*, one must account for the additional cost of linearly combining the results to approximate the J non-separable filters. This requires $n_1 \cdot n_2 \cdot K \cdot J$ more multiplications, for a total of

$$\text{SEP}_{nop} = K \cdot n_1 \cdot n_2 \cdot (J + d_1 + d_2) \tag{13}$$

multiplications.

In Fig. 4(a), we plot the values of $\textit{NON-SEP}_{nop}$, $\textit{NON-SEP-FFT}_{nop}$, SEP*_{nop}, and SEP_{nop}, normalized by the number of pixels in the image, as a function of the size $d = d_1 = d_2$ of the filters between $[3, 25]$. A 2D test image of size 488×488 is considered and convolved with $J = 121$ non-separable filters and $K = 25$ separable ones. Notice that the size of the image is chosen so that the size considered to compute the FFT is a power of 2 for the maximum value of the filters $d = 25$. In this way the zero-padding required is minimal, which is at the advantage of the FFT based approach.

Note that these theoretical curves are very similar to those observed experimentally, shown in Fig. 1.

Our code relies on the MATLAB `conv2` function for spatial 2D convolutions and on the fftw library for the frequency domain convolutions. Observe that these functions can be run in parallel to further reduce the cost of the convolutions, as shown in Fig. 5.

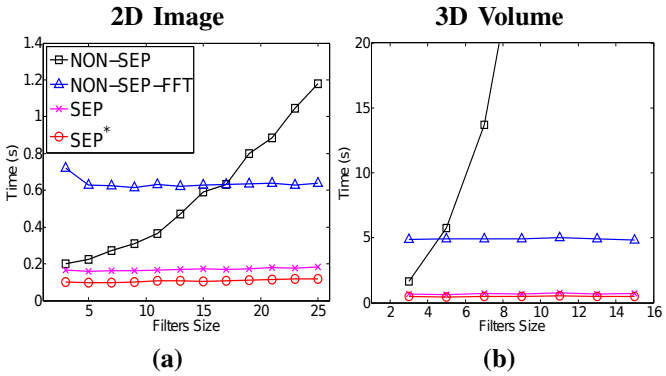


Fig. 5: Time needed to compute convolutions using a multi-thread MATLAB implementation, as a function of the filters size. **(a)** An image of 488×488 pixels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. **(b)** A volume of $114 \times 114 \times 50$ voxels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. Using parallel computation the time needed to compute the convolution is further reduced and our methods are still the most efficient ones. The times are averaged over 5 repetitions.

5.2.2 The N-D Case

The generalization to any dimension is straightforward. If $\mathbf{x} \in \mathbb{R}^{n_i \times,\ldots,\times n_N}$ is an N-dimensional image and $\mathbf{f}^j \in \mathbb{R}^{d_1 \times,\ldots,\times d_N}$ an N-dimensional filter, the cost of a non-separable convolution becomes $\prod_{i=1}^{N} n_1 \cdot d_i$. The cost of a separable convolution is $\left(\prod_{i=1}^{N} n_i\right) \cdot \left(\sum_{i=1}^{N} d_i\right)$ and the cost of a FFT is $c \cdot m \cdot \log_2(m)$, where m is the product of the closest larger powers of 2 of $n_i + d_i - 1$.

Figs. 4(b) and 5(b) illustrate that using separable filters is even more advantageous for the 3D case. Here the size of the filters is between 3 and 15 and a $114 \times 114 \times 50$ volume is considered. Again the size of the volume is taken at the advantage of the FFT based approach.

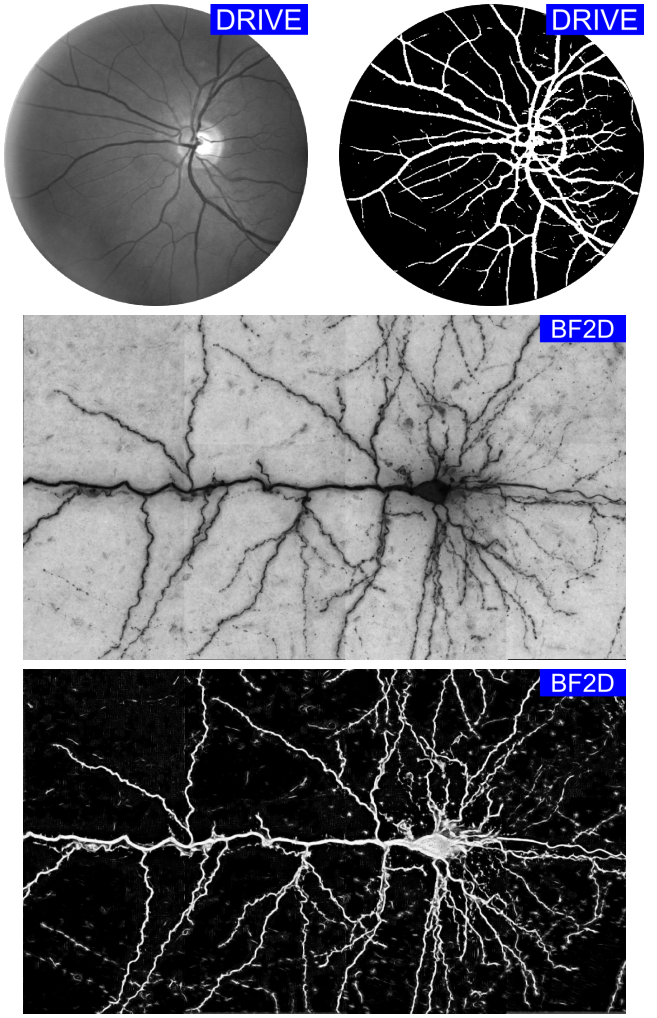


Fig. 6: Representative images from the 2D medical datasets considered, together with the corresponding pixel classification results obtained with our *SEP-COMB** method.

5.3 Detection of Curvilinear Structures

Biomedical image processing is a particularly promising field of application for Computer Vision techniques as it involves large numbers of 2D images and 3D image stacks of ever growing size, while imposing strict requirements on the quality and the efficiency of the processing techniques. Here, we demonstrate the power of our separable filters for the purpose of identifying curvilinear structures, a long-standing Computer Vision problem that still remains wide-open when the image data is noisy.

Over the years, models of increasing complexity and effectiveness have been proposed, and attention has recently turned to Machine Learning techniques. In [12], [34] a Support Vector Machine classifier is applied to the responses of *ad hoc* filters. In particular, [34] considers the Hessian's eigenvalues while [12] relies on steerable filters. In [32], we showed that convolving images with non-separable filter banks learned by solving the problem of Eq. (1) and training an SVM on the output of those filters outperforms these other methods. Unfortunately, this requires many such non-separable filters, making it an impractical approach for large images or image stacks, whose usage is becoming standard practice in medical imaging. We show that our approach solves this issue.

5.3.1 Pixel Classification

In the 2D case we considered the two biomedical datasets of Fig. 6:

- The DRIVE dataset [35] is a set of 40 retinal scans captured for the diagnosis of various diseases. The dataset is split into 20 training images and 20 test images, with two different ground truth sets traced by two different human experts for the test images.
- The BF2D dataset is composed of minimum intensity projections of bright-field micrographs of neurons. The images have a very high resolution but exhibit a low signal-to-noise ratio, because of irregularities in the staining process. Furthermore, parts of the dendrites often appear as point-like structures that can be easily mistaken for the structured and unstructured noise affecting the images.

As mentioned above, we showed in [32] that the *NON-SEP* approach outperforms other recent approaches [12], [34] that

rely on Machine Learning but is slow. Our goal is therefore to achieve the same level of performance but much faster. For completeness, we also compare our results to those obtained using the Optimally Oriented Flux [17], which we will refer to as *OOF*, which is widely acknowledged to be one of the best techniques for finding curvilinear structures using hand-designed filters.

In these experiments we replaced the SVM classifiers we used earlier [32] by Random Forests [6]. They not only brought a considerable speed improvement, but also resulted in better performance. Note that we do not need to compute the linear combination of the filter outputs in the case of *SEP-COMB* and *SEP-TD*, since the Random Forest classifier relies on linear projections. We will therefore opt for *SEP-COMB** and *SEP-TD**.

We first learned a filter bank with 121 learned filters of size 21×21 on the DRIVE dataset and one on the BF2D dataset, as these parameters provided us with the best results. We have then learned other filter banks of reduced cardinality, both full-rank and separable, to assess the impact of the filter bank size on the final classification performance.

As there is no universally accepted metric to evaluate pixel classification performance, we used several to compare our method to others. In particular, we considered the following:

- Area Under Curve (AUC): represents the area subtended by the ROC curve. It assumes values in $[0, 1]$, the higher the better;
- F-measure [38]: assumes values in $[0, 1]$, the higher, the better;
- Variation of Information (VI) [24]: assumes values in $[0, \infty)$;
- Rand Index (RI) [37]: assumes values in $[0, 1]$, the higher, the better.

The classification performance is tabulated using these different metrics in Table 2. *SEP-COMB** and *SEP-TD** performs consistently best, closely matching the performance of *NON-SEP* but with a significant speed-up. *SEP-DIRECT* is just as fast but entails a loss of accuracy. Somewhat surprisingly, *SEP-SVD* falls between *SEP-DIRECT* and *SEP-COMB** in terms of accuracy but is much slower than both. Finally, *NON-SEP-FFT* yields exactly the same results as *NON-SEP* as it should be, but it is much slower than plain 2D convolutions. The costs of the Fourier Transform are indeed amortized only for extremely large image and filter sizes. Hence, it can be inferred that expressing the learned full-rank filter bank in terms of few separable filters leads to a significant speed-up at no cost in terms of accuracy.

The accuracy of *OOF* is significantly lower than that of filtering-based approaches.

TABLE 2: Analytic measure of the performance of the pixel classification task over different datasets. For the DRIVE dataset two ground truth are provided. We use the first one as reference and compute the analytic measure for the second one. For the other methods, 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			
Human Expert #2	–	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
NON-SEP-FFT(121)	0.959	0.782	0.554	0.890	4.79
SEP-SVD(121)	0.955	0.773	0.563	0.887	1.02
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)*	0.962	0.786	0.572	0.888	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
NON-SEP-FFT(121)	0.983	0.754	0.300	0.945	23.04
SEP-SVD(121)	0.982	0.749	0.306	0.943	6.67
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	0.749	0.304	0.936	1.44

5.3.2 Voxel Classification

We also evaluated our method on classifying voxels as belonging or not to curvilinear structures in 3D volumes of Olfactory Projection Fibers (OPF) from the DIADEM challenge [2], which were captured by a confocal microscope. We first learned the non-separable 3D filter bank made of 49 $13 \times 13 \times 13$ pixel filters depicted by Fig. 10(a) using Eq. (1) and then the 16 separable filters. Figs. 3(d) and 10(b) show the separable filters learned using the *SEP-COMB* and *SEP-TD* approaches respectively.

Separable filters are learned as described in Section 4. As in the 2D case, we then trained classifiers to use these filters, but we used ℓ_1-regularized logistic regressors instead of Random Forests since they have proved faster without significant performance loss. For training we used a set of 200000 samples, randomly selected from 4 train images. Since these classifiers do not require us to compute the linear combination of the separable filter outputs, we chose again the *SEP-COMB** and *SEP-TD** approach for our experiments.

We use *NON-SEP* as our baseline. We compare *SEP-COMB** and *SEP-TD** against*NON-SEP-FFT*, a 3D version of *OOF* and *SEP-CPD*.

The results are essentially the same as in the 2D-case. *SEP-COMB** and *SEP-TD** are 30 times faster than *NON-SEP-FFT* for virtually the same accuracy. They are 4 times faster than *SEP-CPD*. As before, *OOF* is even worse in terms of accuracy.

5.4 Denoising

To evaluate how good our approaches are at representing a set of generic filters in a very different context, we used them to approximate the 256 denoising filters computed by the K-SVD algorithm [9], some of which are depicted by Fig. 7(b). We experimented with different sizes of the approximating separable filter bank, and reported the results in Table 4. As can be seen, the 36 separable filters shown in Fig. 7(a) are already enough to obtain a very accurate approximation, giving a perfect reconstruction of the original filters up to a

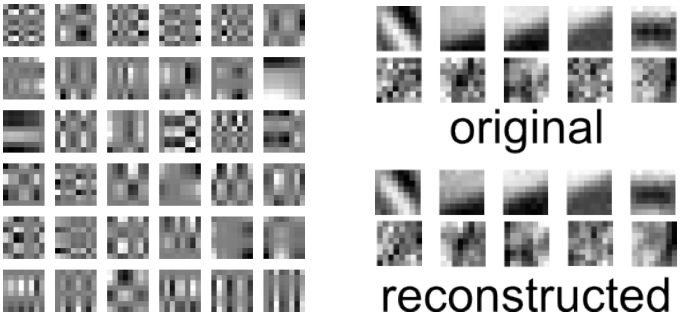
TABLE 3: Analytic measure of the performance of the voxel classification task over the OPF dataset. The VI and RI values are compared on the classification thresholded at the value found using the F-measure. For the learning-based approaches, a training set of 200000 randomly selected samples and a ℓ_1-regularized logistic regressor classifier have been used. Approaches that use a separable filter basis have been found to reduce the computational costs by a factor of 30 in classifications tasks.

Method	AUC	F-measure	VI	RI	Time[s]
		OPF:Image 4			
OOF	0.997	0.531	0.012	0.998	193.05
NON-SEP-FFT(49)	0.997	0.571	0.013	0.998	339.01
SEP-CPD(49)	0.997	0.567	0.013	0.998	40.06
SEP-COMB(16)*	0.997	0.570	0.013	0.998	11.08
SEP-TD(16)*	0.997	0.567	0.013	0.998	11.08

nearly imperceptible smoothing of the filters with many high-frequency components.

Table 4 reports the denoising scores, measured using the Peak Signal-to-Noise Ratio (PSNR). [33] also considered the approximation of filter banks learned with the K-SVD algorithm by using sparse linear combinations of separable filters computed from a 1D DCT basis. However, we need significantly fewer separable filters, only 36 compared to the 100 required by [33].

Interestingly, the basis of separable filters we learn seem general. We proved that by taking the filters that were learned to approximate a filter bank of a specific image, and we used them to reconstruct the filter banks of the other images. In other words, we kept the same $\mathbf{s}^k$ filters learned for the Barbara image, and only optimized on the w_j^k weights in Eq. (7) and Eq. (9). The results are summarized in Table 4.



(a) **(b)**

Fig. 7: Approximating an existing filter bank. **(a)** The 36 separable filters learned by *SEP-COMB* to approximate a bank of 256 filters learned with the K-SVD algorithm of [9]. **(b)** Comparison between some of the original filters learned by K-SVD (top row) and their approximations reconstructed by our algorithm (bottom row). While filters with a regular structure are very well approximated, noisy filters are slightly smoothed by the approximation. Their role in the denoising process is, however, marginal, and therefore this engenders no performance penalty.

5.5 Convolutional Neural Networks

We also applied our approach to Convolutional Neural Networks (CNNs) to show that we can speed up the convolutions they require.

TABLE 4: 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	22.11	22.11	22.13	22.11	22.11
K-SVD(256)	30.87	30.39	33.36	32.40	32.34
SEP-COMB(25)	30.10	30.30	33.04	32.37	32.08
SEP-TD(25)	30.21	30.33	33.18	32.39	32.23
SEP-COMB(36)	30.76	30.38	33.23	32.39	32.27
SEP-TD(36)	30.79	30.39	33.27	32.41	32.10
SEP-COMB(49)	30.86	30.40	33.30	32.40	32.30
SEP-TD(49)	30.87	30.40	33.33	32.40	32.31
SEP-COMB(64)	30.87	30.39	33.35	32.40	32.33
SEP-TD(64)	30.87	30.39	33.35	32.40	32.33
SEP-COMB(36)-Barbara	–	30.01	31.76	32.26	31.83
SEP-TD(36)-Barbara	–	30.02	31.71	32.27	31.82
SEP-COMB(64)-Barbara	–	30.04	31.97	32.27	31.93
SEP-TD(64)-Barbara	–	30.05	31.96	32.27	31.93

In our experiment we consider an architecture consisting of 4 hidden layers fully connected to each other and we apply it to the task of hand-written digit recognition. For our experiments, we used the publicly available Matlab toolbox for deep learning of [27].

The first layer consists of 6 feature maps connected to the single input layer via 6 kernels. The second layer is a 2-by-2 downsampling layer. The third layer consists of 12 feature maps connected to the 6 downsampling layers via 72 kernels. The fourth layer is again a 2-by-2 downsampling layer. The feature maps obtained at the last convolution layer are concatenated into feature vectors and fed into the last layer, which has 10 output neurons in order to discriminate between 10 handwritten digit characters.

The CNN was trained using stochastic gradient descent on the full MNIST dataset [20], which is a set of 70000 images of hand-written digits. The training dataset has 60000 images, while the test dataset has 10000 images. A batch size of 50 is used with a learning rate of 1 for 5 epochs.

In order to obtain separable kernels, we apply the *SEP-TD* approach at each convolution layer. In the first layer, the 6 kernels are approximated using 3 separable filters. In the second layer, we group the 12 filters corresponding to each outgoing layer together. Then, each group is approximated by 4 separable filters independently.

The execution times and the misclassification rates are reported in Table 5 for different kernel sizes. In particular, for a kernel size of 9, the classification is two times faster. As the kernel size increases, employing separability in the classification task becomes even more important. Fig. 8 shows the 12 filters from an outgoing node of the second convolutional layer, 4 separable filters learned from them and a visual comparison between the original and the approximated filters.

5.6 Comparison between *SEP-COMB* and *SEP-TD*

In the previous sections we showed that an arbitrary filter bank can be approximated by linear combinations of separable

TABLE 5: Handwritten digit recognition on MNIST dataset with convolutional neural networks. Different kernel sizes are used in the first and second convolutional layers to evaluate the effect of kernel size on the classification performance and the execution time. The classification results and the execution times are reported for separable and non-separable filters. When separable filters are used in convolution, the linear combinations of the convolutions of separable filters are computed in order to approximate the convolutions with non-separable filters. By using the separable filters, we can divide the computational times by 2, at the cost of small loss in accuracy.

Kernel Size		Misclassification Rate		Execution Time	
First Layer	Second Layer	Separable	Non-separable	Separable	Non-separable
5	5	5.27%	5.17%	27.33	28.77
5	9	4.84%	4.17%	24.9	44.61
9	9	3.47%	3.17%	21.9	48.54

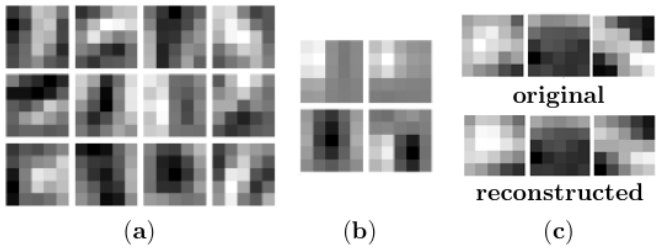


Fig. 8: Filters used in convolutional neural networks. **(a)** 12 filters going out of a node of the second convolutional layer. **(b)** A separable set of 4 filters obtained after tensor decomposition. **(c)** Comparison of the original and the approximated filters

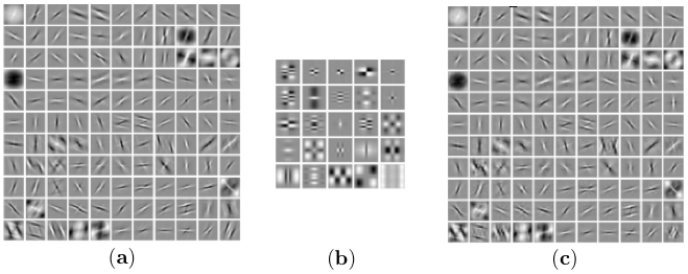


Fig. 9: Convolutional filter banks for classification in 2D. **(a)** Learned non-separable filter bank from DRIVE dataset, **(b)** separable filter bank learned with the *SEP-TD* approach, **(c)** reconstructed filter bank. The non-separable filters can be approximated accurately using a smaller set of separable filters.

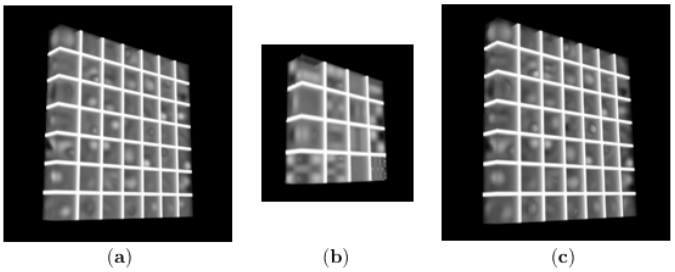


Fig. 10: Convolutional filter banks for classification in 3D. **(a)** Learned non-separable filter bank from OPF dataset, **(b)** separable filter bank learned with the *SEP-TD* approach, **(c)** reconstructed filter bank. As in the 2D case, the non-separable filters can be approximated accurately using a smaller set of separable filters.

filters. We also proved that such decomposition can be used in several Computer Vision tasks to decrease the computational complexity without substantial changes in accuracy.

In this section we compare *SEP-COMB* and *SEP-TD* in terms of approximation error and learning time. In particular we will see that *SEP-TD* has the following advantages compared to *SEP-COMB*:

- **Parameter reduction**: The only parameter of *SEP-TD* is the number of separable filters used to approximate the original one, while *SEP-COMB* relies on a regularization parameter.
- **Faster convergence**: *SEP-TD* approach converges faster than *SEP-COMB*, as can be seen in Fig. 11. The advantage of using *SEP-TD* rather than *SEP-COMB* is more pronounced in the 3D case.
- **Lower approximation error**: This can be explained by the fact that in the *SEP-TD* approach, a non-separable filter is explicitly written as the sum of the products of 1D filters. This approach provides a better approximation quality then *SEP-COMB*, which relies on a soft constraint to make the filter ranks low, as shown in Fig. 11. The visual quality of the approximation of a non-separable filter bank by linear combinations of separable filters obtained with *SEP-TD* is illustrated in Figs. 9 and 10.

6 Conclusion

We have proposed a learning-based filtering scheme applied to the extraction of curvilinear structures, along with two learning-based strategies for obtaining separable filter banks. The first one directly learns separable filters by modifying the regular objective function. The second one learns a basis of separable filters to approximate an existing filter bank, and not only gets the same performance of the original, but also considerably reduces the number of filters, and thus convolutions, required. We presented two optimization schemes for this second approach. In the first one the separable filters are learned by lowering their ranks. In the second one, which proved to be more efficient and accurate, the filters are obtained by tensor decomposition.

Our techniques bring to learning approaches one of the most coveted properties of handcrafted filters, namely separability, and therefore reduce the computational burden traditionally associated with them. Moreover, designers of handcrafted filter banks do not have to restrict themselves to separable filters anymore: they can freely choose filters for the application at hand, and approximate them using few separable filters with our approach.

References

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda. A Scalable Optimization Approach for Fitting Canonical Tensor Decompositions. *Journal of Chemometrics*, 2011.
[2] G. Ascoli, K. Svoboda, and Y. Liu. Digital Reconstruction of Axonal and Dendritic Morphology DIADEM Challenge, 2010.
[3] F. Bach, R. Jenatton, J. Mairal, and G. Obozienski. Optimization with Sparsity-Inducing Penalties. Technical report, INRIA, 2011.
[4] Y. Bengio. *Learning Deep Architectures for AI*. Now Publishers, 2009.
[5] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
[6] L. Breiman. Random Forests. *Machine Learning*, 2001.

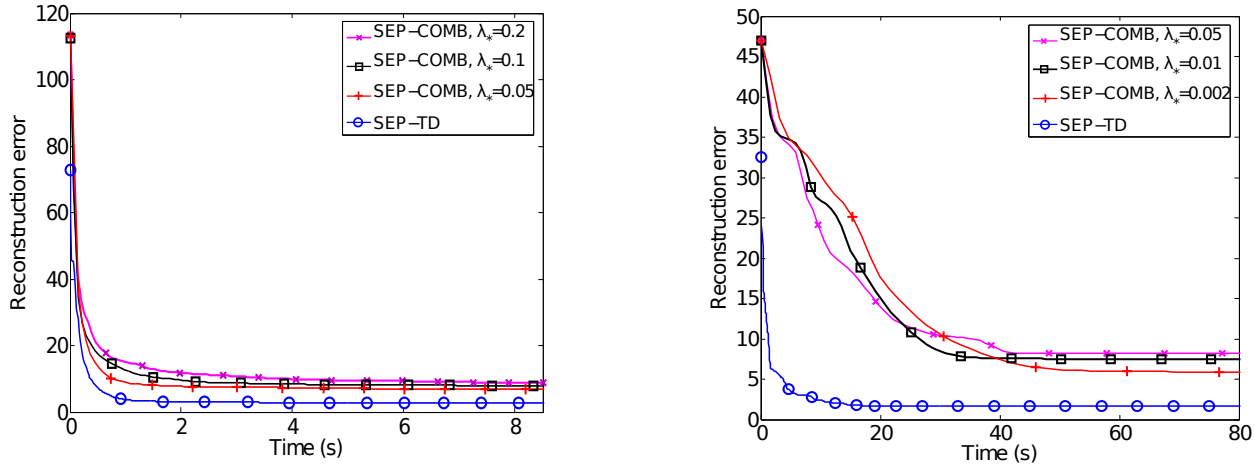


Fig. 11: Comparison of the reconstruction errors of *SEP-TD* and *SEP-COMB* as a function of the learning time for approximating **(a)** a 2D non-separable filter bank and **(b)** a 3D non-separable filter bank. The performance of the *SEP-COMB* approach depends on the specified regularization parameter λ_*. Small regularization parameters yield a smaller reconstruction error. *SEP-TD* does not need to satisfy an additional constraint and yields a smaller reconstruction error compared to *SEP-COMB* with a faster convergence. In the 3D case, the difference is even more pronounced.

[7] H. Bristow, A. Eriksson, and S. Lucey. Fast Convolutional Sparse Coding. In *Conference on Computer Vision and Pattern Recognition*, 2013.

[8] A. Coates and A. Ng. The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization. In *International Conference on Machine Learning*, 2011.

[9] M. Elad and M. Aharon. Image Denoising via Sparse and Redundant Representations over Learned Dictionaries. *IEEE Transactions on Image Processing*, 2006.

[10] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello. Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems. In *International Symposium on Circuits and Systems*, 2010.

[11] M. Fazel, H. Hindi, and S. Boyd. A Rank Minimization Heuristic with Application to Minimum Order System Approximation. In *American Control Conference*, 2001.

[12] G. Gonzalez, F. Fleuret, and P. Fua. Learning Rotational Features for Filament Detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1582–1589, 2009.

[13] S. Hawe, M. Seibert, and M. Kleinsteuber. Separable Dictionary Learning. In *Conference on Computer Vision and Pattern Recognition*, 2013.

[14] G. Hinton. Learning to Represent Visual Input. *Philosophical Transactions of the Royal Society*, 2010.

[15] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning Convolutional Feature Hierarchies for Visual Recognition. In *Advances in Neural Information Processing Systems*, 2010.

[16] T. G. Kolda and B. W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 2009.

[17] M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *European Conference on Computer Vision*, 2008.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

[19] Y. LeCun, L. Bottou, G. Orr, and K. Müller. *Neural Networks: Tricks of the Trade*, chapter Efficient Backprop. Springer, 1998.

[20] Y. Lecun and C. Cortes. The MNIST database of handwritten digits.

[21] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *International Conference on Machine Learning*, 2009.

[22] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-Local Sparse Models for Image Restoration. In *International Conference on Computer Vision*, 2009.

[23] F. Mamalet and C. Garcia. Simplifying Convnets for Fast Learning. In *International Conference on Artificial Neural Networks*, 2012.

[24] M. Meilă. Comparing Clusterings - An Information Based Distance. *JMVA*, 2007.

[25] V. Mnih and G. Hinton. Learning to Detect Roads in High-Resolution Aerial Images. In *European Conference on Computer Vision*, 2010.

[26] B. Olshausen and D. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 1997.

[27] R. B. Palm. Prediction as a candidate for learning deep hierarchical models of data. Master's thesis, 2012.

[28] P. Perona. Deformable Kernels for Early Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[29] H. Pirsiavash and D. Ramanan. Steerable Part Models. In *Conference on Computer Vision and Pattern Recognition*, 2012.

[30] R. Rigamonti, M. Brown, and V. Lepetit. Are Sparse Representations Really Relevant for Image Classification? In *Conference on Computer Vision and Pattern Recognition*, 2011.

[31] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua. Learning Separable Filters. In *Conference on Computer Vision and Pattern Recognition*, 2013.

[32] R. Rigamonti, E. Türetken, G. González, P. Fua, and V. Lepetit. Filter Learning for Linear Structure Segmentation. Technical report, EPFL, 2011.

[33] R. Rubinstein, M. Zibulevsky, and M. Elad. Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation. *IEEE Transactions on Signal Processing*, 2010.

[34] A. Santamaría-Pang, C. Colbert, P. Saggau, and I. Kakadiaris. Automatic Centerline Extraction of Irregular Tubular Structures Using Probability Volumes from Multiphoton Imaging. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2007.

[35] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken. Ridge Based Vessel Segmentation in Color Images of the Retina. *IEEE Transactions on Medical Imaging*, 2004.

[36] S. Treitel and J. Shanks. The Design of Multistage Separable Planar Filters. *IEEE Transactions on Geoscience Electronics*, 1971.

[37] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward Objective Evaluation of Image Segmentation Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[38] C. van Rijsbergen. Foundation of Evaluation. *Journal of Documentation*, 1974.

[39] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse Representation for Computer Vision and Pattern Recognition. *Proc. IEEE*, 2010.

[40] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. In *Conference on Computer Vision and Pattern Recognition*, 2010.