On the Relevance of Sparsity for Image Classification

Roberto Rigamonti[a], Vincent Lepetit[a], Germán González[b], Engin Türetken[a], Fethallah Benmansour[a], Matthew Brown[c], Pascal Fua[a]

[a]*Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne (EPFL), EPFL/IC/ISIM/CVLab, Station 14, CH-1015 Lausanne, Switzerland.*
[b]*Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139-4307, USA*
[c]*Department of Computer Science, University of Bath, BA2 7AY, UK.*

Abstract

In this paper we empirically analyze the importance of sparsifying representations for classification purposes. We focus on those obtained by convolving images with linear filters, which can be either hand designed or learned, and perform extensive experiments on two important Computer Vision problems, image categorization and pixel classification. To this end, we adopt a simple modular architecture that encompasses many recently proposed models.

The key outcome of our investigations is that enforcing sparsity constraints on features extracted in a convolutional architecture does not improve classification performance, whereas it does so when redundancy is artificially introduced. This is very relevant for practical purposes, since it implies that the expensive run-time optimization required to sparsify the representation is not always justified, and therefore that computational costs can be drastically reduced.

Keywords: Sparse representations, image descriptors, image categorization, pixel classification

Email addresses: `roberto.rigamonti@epfl.ch` (Roberto Rigamonti), `vincent.lepetit@epfl.ch` (Vincent Lepetit), `ggonzale@mit.edu` (Germán González), `engin.turetken@epfl.ch` (Engin Türetken), `fethallah.benmansour@epfl.ch` (Fethallah Benmansour), `m.brown@bath.ac.uk` (Matthew Brown), `pascal.fua@epfl.ch` (Pascal Fua)

1. Introduction

Sparse image representations are at the heart of many modern approaches to classification, such as [1, 2, 3, 4]. Some neurophysiological evidence [5, 6] supports their presence in the human visual cortex. Although this evidence is still in dispute [7], the fact that sparsity constraints can be used to derive filters exhibiting a structure very close to that of receptive fields in V1 [8, 9] has played a major role in their widespread acceptance.

On a more practical note, the usefulness of sparsity for image processing purposes is widely recognized [10, 11, 4] along with its suitability as a regularizer for general inverse problems [12]. Part of the appeal of sparse representations is that they are believed to be easily separable in high-dimensional spaces [1, 13, 14]. They have also been successfully used for classification and shown to improve performance in specific cases [15].

In this paper, we will show that the reported classification performance increases [15] stem from the specific setup in which the experiments were performed and that, under different experimental conditions, they do not materialize. More specifically, we will demonstrate that in a shallow recognition architecture and when using convolutional features [16, 17] that rely on the now classic functional proposed by Olshausen and Field [9], no gain arises from sparsifying the representations prior to classification. Similar or better results are obtained by directly feeding the features to a classifier. In this setup, sparsity remains key to learning effective features but becomes unnecessary at run-time. By contrast, if we replace the convolutional features by features derived from overlapping patches, which introduce additional redundancy, run-time sparsity helps as reported in [15].

This analysis validates in a systematic manner casual observations about convolutional architectures that appeared in the literature over the years [1, 18]. It also has important practical consequences since eliminating the run-time sparsifying step can result in substantial computational savings and markedly increase the size of the problems that can be handled. This is because sparsifying remains computationally expensive, even though many recent efforts [19, 20, 21], driven in part by the needs of the Compressed Sensing community [22, 23], have produced efficient algorithms.

In this paper we operate in the context of two key Computer Vision tasks, image categorization and pixel classification. While these two problems might seem only loosely related, state-of-the-art solutions to both involve computing image descriptors either at given locations or densely, post-processing

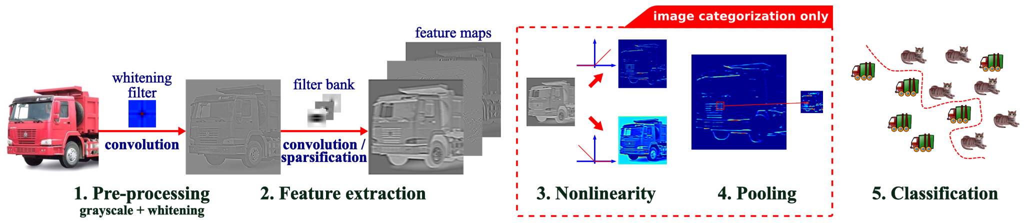


Figure 1: Our image categorization and pixel classification pipeline. Each module can be changed independently to encompass different architectures proposed in literature. For categorization purposes we use the whole pipeline while bypassing the nonlinearity and pooling modules for pixel classification purposes.

them, and performing a final classification step.

This work extends the investigations performed in [16] by comparing our results with related studies available in literature, in particular [15]. Moreover, the inclusion of the pixel classification task in our analysis allows us to validate our claims in two different settings, thus helping us to discount domain-specific biases.

Our investigation relies on the modular classification pipeline depicted by Fig. 1, which is designed to encompass representative state-of-the-art methods and to allow for comparisons. In the following section we briefly review these methods. We then describe and analyze our experiments in the fields of image categorization and pixel classification.

2. Related work

Sparsity constraints have featured many image modeling papers [13, 24, 25, 26, 27]. In fact, they pervade the modern Computer Vision and Pattern Recognition literature, where they are used both as a means to tune feature extractors to the statistics of the data, and as a feature encoding scheme. A comprehensive review of the applications of sparsity is presented in [4]. However, the authors' claim that sparsity is helpful for classification is supported by only few experiments in a very constrained, supervised or semi-supervised setting, and not in an unsupervised one. A more systematic investigation on the different training and encoding schemes is reported in [15]. It analyzes different dictionary learning techniques and the corresponding sparsity-promoting encoders, and concludes that the main benefit of sparse coding lies in its nonlinear encoding scheme. The performance of sparse coding is, therefore, closely matched by simple soft-thresholding, except when very few training samples are used. Note that the conclusions of [15] depend on the use of overlapping patches, while we propose a scheme that can operate efficiently on whole images and avoids stitching artifacts. A similar choice has been made, independently and concurrently, by [28, 29, 30, 31]. The use of overlapping patches introduces unwanted redundancies which, as will be discussed below, explains some of the apparent discrepancies between the outcome of the earlier study [15] and ours.

We now briefly review the relevant literature specific for the two tasks we used to investigate our claims.

2.1. Image categorization architectures

Image categorization is a well-researched topic. A recent trend focuses on the analysis of modular architectures, where each component is tuned to improve the final performance [32, 26, 27]. In particular, the system developed by Jarrett *et al.* [26] shares many similarities with ours. In their work they show both the importance of using the absolute value as a nonlinear operation between the feature extraction and the pooling stages depicted by Fig. 1 and the power of stacking multiple layers. They do not, however, present an evaluation of the effects of sparsity, as they just compare filters learned under sparsity constraints with random filters.

The image categorization literature contains some works, such as [1, 33], where sparse representations enforced at learning time but relaxed at test time to improve performances. None of these works, however, systematically investigates the issue. An interesting approach which avoids the sparsification costs is proposed in [29, 26], where a regressor is trained to approximate the sparse code that is obtained by the optimization process, but no formal guarantees on the approximation error are given.

2.2. Pixel classification architectures

Tubular structures, such as blood vessels or dendritic arbors, are pervasive in biological images and their modeling is critical for analysis purposes. Automated delineation techniques are thus key to exploiting the endless streams of image data that modern imaging devices produce. Among them there is a whole class of approaches, such as [34, 35, 36], that take as input image segmentations in which pixels or voxels within linear structures are labeled as one and others as zero. The better the initial segmentation, the more effective these methods are. To generate them, most approaches start by making strong assumptions on the shape of the corresponding signal. For example, assuming the intensity profile is U-Shaped, optimal steerable filters for neurite tracing can be derived [37]. An even more widespread approach is to rely on the Hessian matrix of the image and its eigenvalues [38, 39, 40, 41]. To detect filaments of various widths, a range of variances for the Gaussian derivative filters must be used and compared. Other models use differential kernels [42], look for parallel edges [43], or fit superellipsoids to the image [44, 45]. Of particular interest is the Optimally Oriented Flux Filter (OOF) [46], obtained by convolving the second derivatives of the image with the indicator of a sphere, which is a steerable filter designed for detecting ideal sharp ridges. Compared to Hessian-based detectors, the OOF is simpler to normalize over scale and less sensitive to adjacent features of filaments. Real linear structures, however, do not necessarily conform to these *ad hoc* models, and this can drastically impact performance. As a result, machine learning-based approaches that can learn complex appearances are an attractive alternative. In [47], the distribution of the eigenvalues of the structure tensor are estimated via Expectation Maximization. Probabilistic Boosting Trees with sparse rotational features have also been demonstrated for vessel segmentation purposes [48]. Support Vector Machines operating on the Hessian's eigenvalues have been used to discriminate between filament and non-filament pixels [49].

In our own earlier work [50], we compute the responses of steerable filters at every pixel and feed them to an SVM to classify pixels as filament-like or not. Because the filters are separable, they can be implemented very efficiently, which is critical when dealing with very large data volumes. However, as we will see in the result section, they are less expressive than the ones we derive here.

3. Image categorization

To properly discuss the influence of sparsity on recognition rates, we rely on the shallow modular architecture of Fig. 1, which is very similar to the ones used in recent works, such as [26, 51, 24, 32, 52]. In particular, it can be considered as the first of a sequence of layers that constitute a Deep Network architecture [53, 54]. These models recently gained relevance for their effectiveness in solving multiple Computer Vision problems [55]. Understanding the behavior of a layer as we do in this paper is therefore important for these promising approaches.

In our architecture, after a pre-processing step, we extract features by using filters that are either learned or handcrafted. These dense features result from a simple convolution between the image and the filters, and their sparsified version can be obtained using a sparse optimization procedure. The usual modules of a biologically-inspired classification architecture, namely a nonlinearization and a pooling step, follow.

We perform extensive experiments on the challenging CIFAR-10 dataset [56, 57], and we validate the resulting insights on the Caltech-101 dataset [58] for which a thorough analysis would be prohibitively costly. Besides illuminating the role played by sparsity in convolutional models, this methodical exploration of the architecture and parameter spaces allows us to get useful insights on the structure of an effective classification model.

We detail below the filter learning algorithms and the individual components of our framework. We introduce acronyms for these different modules, which we will use in our result tables. Finally, we describe the datasets we use and the comparative results we obtain.

3.1. Learning the filters

Olshausen and Field [9] suggested that V1, the first layer of the visual cortex, builds a sparse image representation. Under this assumption, and the hypothesis that a perfect reconstruction is attainable, the problem one would like to solve can be stated as

$$\underset{\mathbf{M},\{\mathbf{t}_i\}}{\operatorname{argmin}} \sum_i \|\mathbf{t}_i\|_0 \quad \text{s.t. } \forall i, \mathbf{M}\mathbf{t}_i = \mathbf{x}_i, \tag{1}$$

where $\mathbf{x}_i$ are training images, $\mathbf{t}_i$ are the corresponding feature vectors, and $\mathbf{M}$ is a matrix whose columns form the dictionary. The ℓ_0 norm formulation in Eq. (1) is, however, non-convex, making the optimization very difficult. Even more importantly, the perfect reconstruction premise is never satisfiable with real images. The version proposed in [9] solves therefore a relaxed problem that, under certain assumptions, converges to the true solution. In particular, in many recent works such as [25, 26, 4], a dictionary of filters is learned by optimizing the objective function

$$\underset{\mathbf{M},\{\mathbf{t}_i\}}{\operatorname{argmin}} \sum_i \|\mathbf{x}_i - \mathbf{M}\mathbf{t}_i\|_2^2 + \lambda_{learn} \|\mathbf{t}_i\|_1 , \tag{2}$$

where the ℓ_1 norm enforces sparsity on the $\mathbf{t}_i$ vectors and has other desirable properties that have been thoroughly investigated in the Compressed Sensing literature [22, 23, 12, 20].

Solving Eq. (2) yields a dictionary $\mathbf{M}$ such that the images $\mathbf{x}_i$ can be reconstructed from only a few columns of $\mathbf{M}$ by computing the product $\mathbf{M}\mathbf{t}_i$. The sparseness in the $\mathbf{t}_i$ vectors is enforced by the last term. λ_{learn} is a regularization parameter that establishes the relative importance of the reconstruction error $\|\mathbf{x}_i - \mathbf{M}\mathbf{t}_i\|_2^2$ against the regularization term $\|\mathbf{t}_i\|_1$. To prevent the algorithm from decreasing the ℓ_1 norm of the coefficients by increasing the magnitude of the filters, each column of $\mathbf{M}$ is normalized at each optimization step [9]. Moreover, the dictionary is overcomplete: $\mathbf{M}$ has more columns than rows, and this gives us the degrees of freedom we need to choose a representation among all the possible ones. The resulting filter bank contains many filters that differ just by a translation [59]. Note that solving Eq. (2) for large images would be slow and difficult because many coefficients in $\mathbf{M}$ have to be optimized simultaneously. In earlier approaches it was therefore done only for relatively small patches. In this work, to handle whole images, we instead adopt a convolutional approach where the matrix-vector product is replaced by a convolution. We will refer to it with the acronym *OLS* in the remainder of the paper. An underlying assumption is that local image properties are translation invariant, which seems reasonable. As a side effect we get a strongly overcomplete representation, since all the possible translations of the non-zero components of each filter are implicitly taken into account. The optimization problem in Eq. (2) hence becomes

$$\operatorname*{argmin}_{\{\mathbf{f}^j\},\{\mathbf{t}_i^j\}} \sum_i \left(\left\| \mathbf{x}_i - \sum_j \mathbf{f}^j * \mathbf{t}_i^j \right\|_2^2 + \lambda_{learn} \sum_j \left\| \mathbf{t}_i^j \right\|_1 \right), \tag{3}$$

where the $\mathbf{f}^j s$ are linear filters and $*$ denotes the convolution operator. The $\mathbf{t}_i^j s$ can now be seen as a set of images with the same size as the $\mathbf{x}_i$ images, whose cardinality is equal to that of the filter bank. Similar intermediate representations have been called *"feature maps"* in the Convolutional Neural Networks literature [60]. The relationship between Eq. (2) and Eq. (3) is readily understood by analyzing separately the two terms composing the equations: Since the convolutions in Eq. (3) are linear transformations, they could actually be written as a matrix-vector product of the form $\mathbf{M}\mathbf{t}_i$, where $\mathbf{M}$ would be an extremely large matrix, and $\mathbf{t}_i$ a vector obtained concatenating the $\mathbf{t}_i^j$ feature maps together. The first terms of Eq. (2) and Eq. (3) are therefore equivalent. The sum of the ℓ_1-norm of the $\mathbf{t}_i^j$ maps is equal to the ℓ_1-norm of the corresponding $\mathbf{t}_i$ vector. The second terms are therefore equivalent as well.

The optimization problem of Eq. (3) is not convex, but the two subproblems obtained by alternatively minimizing on the filters and on the feature maps, keeping the other variables fixed, are convex [61]. We therefore optimize on the feature maps using a proximal method [19] and on the filters with Stochastic Gradient Descent [62]. Proximal methods allow to extend gradient descent techniques to some nonsmooth problems, and in the case of ℓ_1-norm regularization the corresponding proximal operator [19] used in the minimization is the soft-thresholding, whose expression is $\text{prox}_\lambda(x) = \text{sgn}(x)\max(|x| - \lambda, 0)$. The optimization on the feature maps thus reduces to performing a step in the direction opposite to the gradient of the ℓ_2-regularized term, followed by a component-wise soft-thresholding of the argument of the ℓ_1-penalized term. This algorithm is also known in literature as Iterative Thresholding [63].

The resulting filter banks learned on the CIFAR-10 and on the Caltech-101 dataset are depicted by Fig. 2. While there is no guarantee that the algorithm converges to a global optimum, the optimization consistently converges from random initializations for a wide range of λ_{learn} values. Nevertheless, the algorithm exhibits a strong sensitivity to the gradient descent step size both for filters and coefficients; We manually tuned these step sizes. Each filter in a filter bank being optimized independently from the others, nothing prevents a subset of them becoming identical. However, in practice, the large variety of structures present in the used datasets partially mitigates this problem and we have observed that the presence of some replicated filters does not severely affect the performance for image categorization purposes. As we will discuss later, for pixel classification purposes replication is more of a problem and we had to devise a strategy to prevent it.

3.2. Classification architecture

3.2.1. Pre-processing

We only use grayscale images and the first pre-processing step therefore maps input color images into a grayscale representation in $[-1, 1]$. For convolution purposes, we replicate the image borders.

To speed up convergence, we found it helpful to whiten the data. Whitening also happens in the human visual system, where it is performed by the Lateral Geniculate Nucleus [64]. As we will discuss later, we have observed that whitening plays an important role in artificial classification systems too. A whitening operator can be learned from the covariance matrix $\mathbf{C}$ of the original data [64]. By applying an eigenvalue decomposition to $\mathbf{C}$, $\mathbf{C} = \mathbf{E}\mathbf{D}\mathbf{E}^\top$, a whitening matrix $\mathbf{W}$ can be computed as $\mathbf{W} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^\top$.

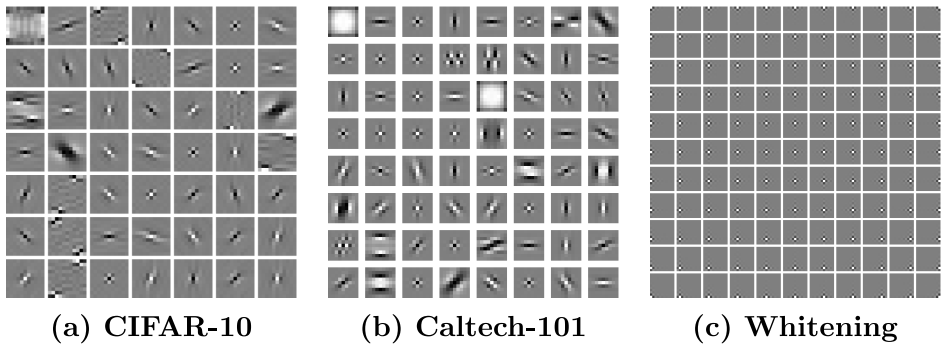


Figure 2: **(a),(b)** Some of the filter banks we have learned using the *OLS* algorithm on the CIFAR-10 and Caltech-101 datasets. Filter values are normalized in $[-1, 1]$. **(c)** Whitening filter learned from the data. To whiten arbitrarily sized images we pick the filter in the middle of the filter bank and we convolve it with the images.

However, as in Eq. (2), this is not really practical for large images. Fortunately, owing to the shift invariance of image statistics, $\mathbf{W}$ describes a per pixel linear operation that is independent of translation, we can therefore efficiently implement whitening as a convolution.

3.2.2. Feature extraction

We evaluate different types of filter banks for feature extraction. As mentioned earlier, the abbreviations in parentheses are used to denote the different possible modules:

- Filter banks made of filters learned as discussed in Section 3.1 (*OLS*). As the learning procedure depends on several parameters, many such filter banks are possible.
- The Leung-Malik (*LM*) filter bank [65].
- A filter bank constituted by randomly generated filters (*RND*), with values sampled from $\mathcal{N}(0, 1)$.

These filters are used to extract features $\mathbf{t}^j$ from an image $\mathbf{x}$ in two different ways:

- Features computed by direct convolution (*CONV*). The $\mathbf{t}^j s$ result from direct convolution with the filters, as

$$\mathbf{t}^j = \mathbf{f}^j * \mathbf{x}, \; \forall j. \tag{4}$$

- Sparse features with Iterative Thresholding (*SPARSEIT*). The $\mathbf{t}^j s$, initialized by direct convolution, are then sparsified using Iterative Thresholding to solve

$$\underset{\{\mathbf{t}^j\}}{\operatorname{argmin}} \left\| \mathbf{x} - \sum_j \mathbf{f}^j * \mathbf{t}^j \right\|_2^2 + \lambda_{extract} \sum_j \left\| \mathbf{t}^j \right\|_1 . \tag{5}$$

  This optimization is the same as the one posed in Eq. (3) after fixing the filters $\mathbf{f}^j$ and considering only the given image. In this setting, the problem we are solving is convex [61], and therefore the correctness of the optimization scheme is easily verifiable. We consider a termination condition for the algorithm based on the amount of variation in the functional value between two subsequent steps.

3.2.3. Non-linearity

Before the pooling stage we apply a nonlinear transformation to the feature maps $\mathbf{t}^j$, as is usually done in multilayer architectures. This operation gives a new set of feature maps $\mathbf{u}^j$. Again, we try different possibilities:

- Taking the absolute values of the coefficients of the $\mathbf{t}^j$ vectors (*ABS*). The m-th coefficient $\mathbf{u}^j[m]$ is simply taken to be: $\mathbf{u}^j[m] = |\mathbf{t}^j[m]|$. This operation is identified as very effective in [26] for recognition performance despite its simplicity.

- Separating the negative coefficients from the positive ones (*POSNEG*). The values in $\mathbf{t}^j$ are spread over $\mathbf{u}^{2j}$ and $\mathbf{u}^{(2j+1)}$ according to:

$$\mathbf{u}^{2j}[m] = [\mathbf{t}^j[m]]^+, \mathbf{u}^{(2j+1)}[m] = [-\mathbf{t}^j[m]]^+, \tag{6}$$

  where $[x]^+ = x$ if $x > 0$ and 0 otherwise. This operation doubles the descriptor's size.

3.2.4. Pooling

This stage pools the coefficients of the $\mathbf{u}^j$ vectors to provide invariance to small displacements and distortions. Having a pooling stage is advisable for two reasons:

- From a biological perspective, the pooling stage corresponds to a complex cells' layer in Hubel and Wiesel's model of the V1 cortex [66].

The role of pooling is to enable a certain degree of invariance to minor pose and appearance changes. The importance of pooling layers is also acknowledged by their employment in Convolutional Neural Networks [60].

- From a computational perspective, plain descriptors have a dimensionality which is too high for practical applications. The downsampling step is therefore vital for subsequent operations.

We test three different pooling mechanisms found in literature:

- Gaussian pooling (*GAUSS*). This is used in [67]: the $\mathbf{u}^j s$ are first convolved with a Gaussian filter, then downscaled by a factor that is a multiple of 2.
- Average pooling (*BOXCAR*). This is similar to *GAUSS*, except that we use a boxcar filter.
- Maximum value pooling (*MAX*). We retain the maximum absolute value in a given neighborhood. This is used for example in [24, 26], and also evaluated in [27].

3.2.5. Classifiers

The final step is to apply a classifier to the unitary normalized vectors obtained from the previous stages. We report results using two different methods [1]:

- Approximate Nearest Neighbor classification (*NN*). It provides a direct measure of the discriminative capabilities of the derived descriptor.
- Support Vector Machines (*SVM*). They are commonly adopted in pipelines similar to ours and usually achieve the best results [2]. In particular, we use an RBF-SVM, since theoretical results show that it is better than a sigmoid-SVM [68]. Since we explore thoroughly the parameter space, we do not need to explicitly consider a linear-SVM [69].

[1]We have also tried other classifiers: Feed-Forward Neural Networks, ensembles of Classification Trees, and Naive Bayes classifiers. As they do not give better results than *SVM*s, we do not report them.

[2]We performed our experiments with the LIBSVM library (`http://www.csie.ntu.edu.tw/~cjlin/libsvm`).

- Logistic Regression (*LOG-REG*). While having generally worse performances than Support Vector Machines, it is very fast and can efficiently operate on large feature vectors; These characteristics made it suitable for our experiments with the Caltech-101 dataset. We have used the implementation provided by the authors of [26].

3.3. Image categorization datasets

Solving the image categorization problem involves the derivation of a mapping from the feature space to the label space, so as to assign to a given input image the label of the corresponding category. Recent analysis demonstrated the difficulties in the choice of a dataset that truly gauges the capabilities of a classification system [70, 71]. We have opted for CIFAR-10 [56, 57] as our reference dataset, because it avoids the pitfalls exposed by [70], while at the same time the reduced dimensionality of its images enabled us to perform extensive experimentations. We have then validated our insights on the renowned Caltech-101 dataset [58], which is commonly adopted in other works in the field.

3.3.1. CIFAR-10

The CIFAR-10 dataset is composed of 32×32 pixels images, yet it exhibits a large variability in pose, appearance, scale, and background composition, making it an ideal test case. Despite the low resolution of the input images, the feature maps after pooling $\mathbf{v}^j$ are very large, and therefore a dimensionality reduction step before classification is desirable. We investigate the following methods:

- No dimensionality reduction (*NONE*).
- Principal Component Analysis (*PCA*).
- Local Discriminant Embedding (*LDE*) [72]. We use a power regularization fixing the signal to noise ratio to 15% as was done in the original paper since it was performing well in our experiments.
- Random Projections (*RP*). We try random projections because they can be applied to sparse signals with limited information loss. [23].

In both the *PCA* and the *LDE* case we normalize the feature maps to unit norm after the projection, as this is deemed to give significant improvements on the final result [72]. To choose the best size of the eigenspace we perform for each specific configuration an extensive cross-validation for all dimensions in a range $d = \{8, 10, \ldots, 256\}$, and we select the value that scores best in an Approximate Nearest Neighbor classification. The dimensionality of the eigenspace, as selected by the procedure above, usually ranges between 20 and 70. To perform more extensive experimentations, we first downsample the dataset to 16×16 pixels and identify the various trade-offs and the best components of the pipeline. Once the most effective choices are determined, we validate the resulting architectures on the original 32×32 images.

3.3.2. Caltech-101

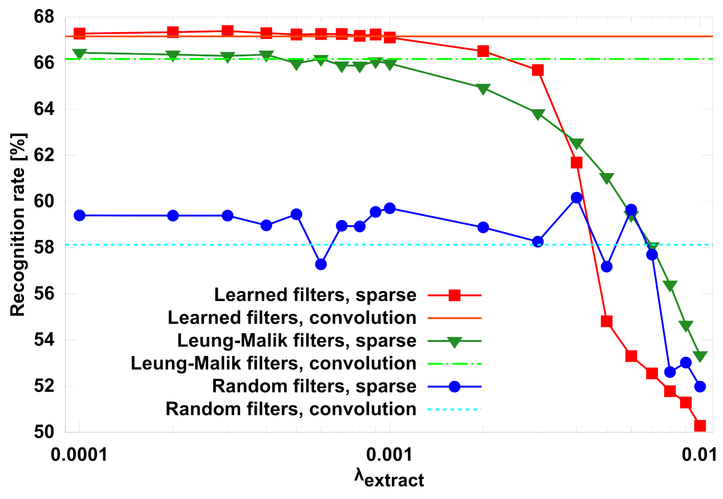
We perform additional experiments using the Caltech-101 dataset, which is widely acknowledged as a reference dataset in the Computer Vision community, and has been used in the related works [26, 28]. We have adopted the same testing methodology of [26]; At first, we have learned, on the grayscale Caltech-101 images resized to 151×151 pixels, the filter bank composed by 64 9×9 filters depicted by Fig. 2(b). We then extracted the features, eventually imposing sparsity via *SPARSEIT*, followed by rectification using the *ABS* function, and boxcar pooling with a 10×10 filtering and a $5\times$ downscaling. The resulting features are passed to the logistic regression classifier provided by the authors of [26], as their high-dimensionality makes them unsuitable for SVM classification. Our approach corresponds therefore to the $64.F_{CSG}^{9\times9}$-R_{abs}-P_A-*log_reg* architecture of [26]. Experiments have been performed with 30 training and 30 test images, with a fixed choice of the images in both sets across the different experiments.

3.4. Results and discussion

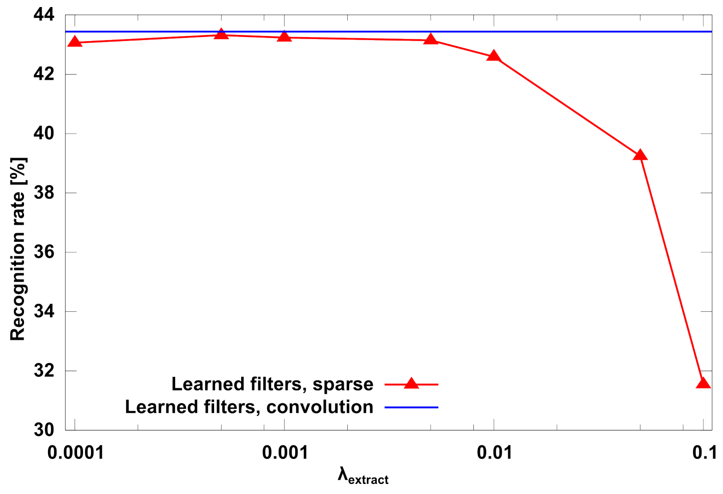
Our first experiment aims at evaluating the influence of the way the features are extracted on the recognition rate. Fig. 3 reports the results of our classification pipeline for different filter banks and different feature extraction methods. For the experiment in this section we use either 49 (*OLS,RND*) or 48 (*LM*) 11×11 filters. The other components of the model are set to *POSNEG*, *GAUSS*, *PCA*, *SVM*, which is one of the best combinations we have found [3].

As shown in Fig. 3 the key experimental result is that performing simple convolutions (*CONV*) at detection-time works just as well as enforcing spar-

[3] For more results, as well as for details on the parameters, please refer to the supplemental material.

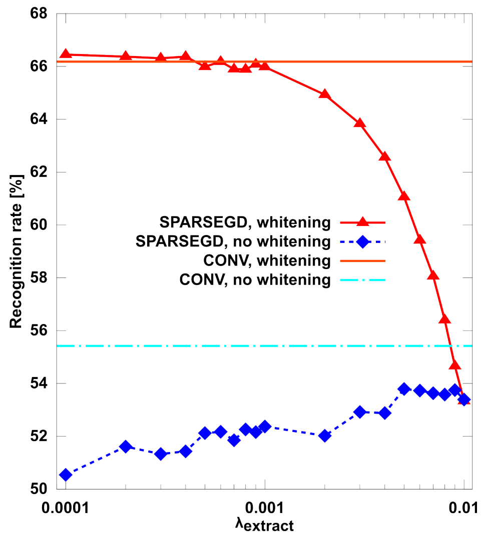


CIFAR-10

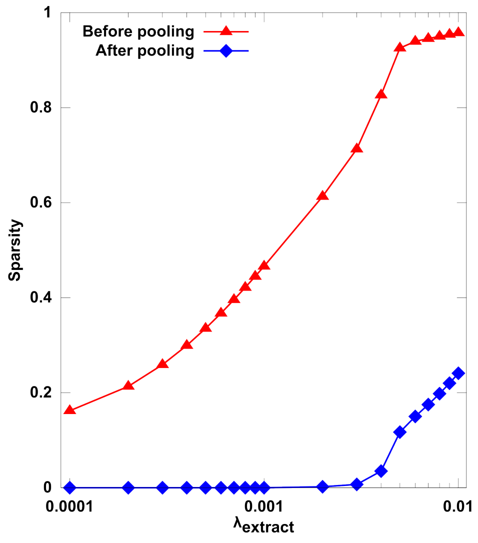


Caltech-101

Figure 3: **(Top)** Classification results on the CIFAR-10 dataset. Straight markerless lines depict the results obtained using simple convolutions ($CONV$) while the other curves represent recognition rates as a function of $\lambda_{extract}$, when enforcing sparsity ($SPARSEIT$) at detection-time. Red curves corresponds to results obtained using filters learned under sparsity constraints with $\lambda_{learn} = 2$, green to handcrafted ones, and blue to random ones. Note that the red curves and lines are above the others and very close to each other for low values of $\lambda_{extract}$. By contrast, for high values of $\lambda_{extract}$ the performance drops abruptly. The same behavior can be observed for the green and blue curves. **(Bottom)** Classification results obtained on the Caltech-101 dataset with a logistic regression classifier. The filters were obtained under sparsity constraints with $\lambda_{learn} = 0.02$.



(a)



(b)

Figure 4: **(a)** Analysis of the effects of whitening with an handcrafted filter bank (the Leung-Malik filter bank) on the CIFAR-10 dataset. **(b)** Sparsity of the descriptor, measured as the fraction of zeroes in the representation, before and after Gaussian pooling. After pooling, sparsity is completely lost.

sity (*SPARSEIT*), no matter how the filters were derived in the first place. Furthermore, imposing too much sparsity by increasing the $\lambda_{extract}$ parameter eventually results in a severe performance loss. To prevent this loss, the $\lambda_{extract}$ used for *SPARSEIT* must be much smaller than the λ_{learn} used to learn the filter bank, as also noted in [33].

By contrast, enforcing sparsity at learning time is very useful, as evidenced by the fact that filters learned in this way perform better than handcrafted or random ones.

To investigate further when sparsity can be useful, we ran the same experiments on images from the CIFAR-10 dataset after corruption by noise. The most significant results are reported in Tab. 1. We experiment with both Gaussian and structured noise, where the latter consists of randomly generated lines superimposed to the images (see Fig. 5). In all these experiments, we worked with the original 32×32 images of CIFAR-10, in order to avoid that the signal is prevailed by the noise. *SPARSEIT* performs well in presence of strong Gaussian noise, but it does not help for structured noise, as it focuses its efforts around the noisy area skipping the parts of the images that

Table 1: Classification rates on the noisy CIFAR-10 dataset for different feature extraction methods, using learned filters and a SVM as a classifier

Method	$\lambda_{extract}$	$\|\mathbf{t}\|_0$	**Rec. Rate** [%]
small Gaussian noise ($\sigma = 0.01$)			
CONV		**1.00**	**69.44**
SPARSEIT	0.0001	0.83	68.66
SPARSEIT	0.0005	0.58	67.07
SPARSEIT	0.001	0.43	64.54
SPARSEIT	0.005	0.11	54.37
strong Gaussian noise ($\sigma = 0.14$)			
CONV		1.00	60.30
SPARSEIT	0.0001	0.88	61.89
SPARSEIT	**0.0005**	**0.69**	**63.54**
SPARSEIT	0.001	0.55	63.28
SPARSEIT	0.005	0.17	59.94
small structured noise (1 random line)			
CONV		**1.00**	**48.53**
SPARSEIT	0.0005	0.51	47.00
SPARSEIT	0.005	0.09	31.75
strong structured noise (1 to 3 random lines)			
CONV		**1.00**	**35.20**
SPARSEIT	0.0005	0.49	33.51
SPARSEIT	0.005	0.09	15.08

convey discriminative information. This is reasonable, as the sparse coding equations in [9] were derived under a Gaussian prior on the noise. Since the original images of the dataset are mostly noise free, the denoising capabilities of sparsity are a property unexploited when evaluating categorization algorithms on these benchmarks.

Individual choices for the different pipeline components bear a strong influence on the final outcome. In Tab. 2 the classification rates for different pooling/subspace projection methods are reported, and it can be seen that Gaussian pooling outperforms the highly acclaimed MAX pooling strategy [27]. Tab. 3 evaluates the two nonlinearities, namely *POSNEG* and *ABS*, for the different choices of the subspace projections and with both learned and handcrafted filters. *POSNEG* scores consistently better than *ABS*. In Fig. 4(a), we compare the performance of handcrafted filters applied to im-

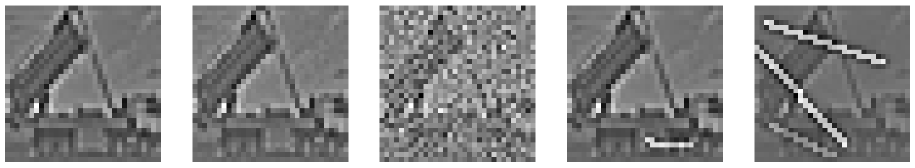

(a) **(b)** **(c)** **(d)** **(e)**

Figure 5: Examples from the noisy version of the CIFAR-10 dataset. **(a)** Original, noise free image. **(b)** Image corrupted by small Gaussian noise. **(c)** Image corrupted by strong Gaussian noise. **(d)** Image corrupted by small structured noise. **(e)** Image corrupted by strong structured noise.

Table 2: Comparison between pooling strategies for different subspace projections, (*OLS-CONV-POSNEG-*-*-SVM*), CIFAR-10 dataset

Method	**Rec. Rate** [%]		
	PCA	*LDE*	*RP256*
GAUSS	**67.16**	**67.13**	**66.07**
MAX	62.62	61.91	59.92
BOXCAR	63.33	63.33	61.33

ages with and without whitening. Since the convolution operator is commutative, applying whitening to an image and then convolving it with a filter bank is equivalent to applying whitening to the filters and then convolving them with the original image. From the graph it can be deduced that there is a huge gap, more than 10%, between the two results. The performance of the Leung-Malik filter bank without whitening is below that of random filters with whitening. These structural insights have been confirmed in two recent papers, namely Coates *et al.* [52] for what concerns the importance of the architecture and of the whitening step, and Saxe *et al.* [73] for the amazing performance of random filters.

In all of our experiments and irrespective of the chosen feature extraction and pooling strategies the results after pooling are dense, as shown in Fig. 4(b) for Gaussian pooling. We have observed a similar behavior with MAX pooling, despite its alleged sparsity-preserving properties. This suggests that, in architectures that employ pooling stages, sparsity is a temporary condition only.

Despite its simplicity, our best architecture performs well on the CIFAR-10 dataset, yielding a 75.18% classification rate (average over 5 random

Table 3: Comparison between the tested subspace projections, for both learned and handcrafted filter banks, and for both *POSNEG* and *ABS* (**-CONV-*-GAUSS-*-SVM*). *PCA* and *LDE* perform equally well in our experiments

Method	**Rec. Rate** [%]			
	OLS		*LM*	
	POSNEG	*ABS*	*POSNEG*	*ABS*
PCA	**67.16**	63.17	66.18	62.83
LDE	67.13	**63.62**	**66.34**	**62.93**
RP256	66.07	62.93	64.10	61.26

dataset splits, with standard deviation 0.27%) by using grayscale images only, whereas competing methods also use color information. On the Caltech-101 dataset, however, the performance are well below the state-of-the-art results of [74], which achieves an exceptional 84.3%. Nonetheless, the architecture we propose is not aimed at achieving high classification scores by exploiting, for instance, prior knowledge about the image content, but at studying systematically a property of feature descriptors. Similar architectures that were developed with the same goal in mind, such as that of [26], achieve comparable classification rates.

3.5. Comparison with patch-based architectures

Traditional, sparsity-based image categorization architectures operate on small overlapping image patches extracted on a regular grid. This is mostly an heritage of the original optimization scheme for obtaining sparse representations proposed by Olshausen and Field [9]. Recently, this approach has been subject to an accurate analysis where different training and encoding schemes have been chained and the resulting combinations evaluated in terms of their recognition capabilities [15].

At first sight, its conclusions appear to contradict ours. In particular, while soft-thresholding performs comparably with sparse coding most of the times, some form of sparsification in the encoding is always required to achieve good performance. Also, using an architecture strongly resembling ours but using just a linear SVM classifier, they obtain a classification rate of over 80% on the CIFAR-10 dataset.

Starting with the publicly available source code, we first validated these insights by observing that, with default parameters – 1600 filters with size 6×6, linear SVM classifier – setting to zero the threshold parameter α in the soft-thresholding encoding, which corresponds to *CONV-POSNEG* in our architecture, negatively affects the performances, moving from 78.18% when $\alpha = 0.25$ to 75.80% when $\alpha = 0$. We then investigated the apparent discrepancy between our findings.

An obvious difference is that we operate on grayscale images instead of color ones. While color information is mostly redundant, it still has an impact on the classification rate. Simply converting the images to grayscale makes the results drop from 78.18% to 74.08%. Please note that the same reasoning applies in the comparison between our architecture and other color-based machine learning architectures which have been specifically tuned to operate on the CIFAR-10 dataset, such as the factorized third-order Boltzmann Machine proposed in [75] or the improved version of the 2009 PASCAL image classification challenge winning system presented in [76].

A more subtle difference concerns the feature extraction process. In [15] filters are applied on patches extracted on a regular grid with a stride of one, while we apply our filters convolutionally. In a convolutional architecture all the extracted coefficients contribute to the reconstruction of the input image, and the learned filters account for this. The resulting redundancy is therefore lower compared to using overlapping patches [29], as altering even a single coefficient stymies the final image reconstruction. An approach which exhibits the same characteristics but in a patch-wise setting has to constrain the patches to be non-overlapping.

To verify how the extraction procedure affects the final score, we have analyzed the classification rate for different degrees of sparsity imposed by soft-thresholding the coefficients computed on both overlapping and distinct patches. We have then compared these results with those obtained by plugging the feature maps computed by the extraction step of our convolutional architecture in the same code. The results are reported in Tab. 4.

As can be observed, the basic architecture of [15] benefits from a soft-thresholding of its features when the source image patches are extracted in an overlapping way (Tab. 4(a)). This is also true when the patches are distinct, but color information is used (Tab. 4(b)). Note that the number of filters in the color case has been divided by three, to account for the difference in size of the descriptors compared with the grayscale case. However, when either grayscale non-overlapping patches, convolutional extraction, or very few filters are considered (Tab. 4(b-e)), sparsely-encoded features do not perform better than non-thresholded ones. When just 500 training samples per category are considered (Tab. 4(f)), if the same number of filters as in

Tab. 4(b) is considered, sparse encoding is again relevant. The same applies when fewer filters are used.

These results suggest that, when redundant information is introduced in the feature extraction step, an encoding which removes feeble components and therefore promotes sparsity has to be preferred. However, when this redundancy is absent, experimental results do not support the sparsification. Moreover, by comparing the results in Tab. 4(c-d), for a given number of filters and total operations, convolutional feature extraction appears to perform better than schemes based on overlapping patches, at least when linear SVMs — which give a significantly better result than Nearest Neighbor classification – are used as classifiers. Finally, a feature extraction scheme based overlapping patches scores much better than one based on distinct patches for a fixed descriptor size.

4. Pixel classification

Starting with [77], pixel classification has become a popular way to address the image segmentation problem. A particular case of segmentation is represented by the extraction of extended linear structures, such as those present in the images of Fig. 6. In this case the image is not subdivided into regions, but the elements of interest are enhanced with respect to a background. It is therefore natural to interpret each pixel as either belonging to the target structure or not, and it makes sense to express class membership in probabilistic terms.

We explore here the classification of pixels as belonging or not to extended linear structures such as those of Fig. 6, in the same spirit of [77, 50]. The target structures appear at many different scales and in many different contexts, such as micrometer scale dendrites in light microscopy image-stacks, millimeter-scale blood vessels in retinal scans, or meter-scale road networks in aerial images, and are of fundamental importance in many applications. To this end, we use a simplified version of the shallow modular architecture of Fig. 1. It forgoes the whitening and the pooling steps, as we have empirically found them to negatively affect the classification score. Our interpretation is that whitening removes important information from the data by eroding the vessels' profiles, while pooling drops their localization and erases the thinner ones. Nonlinearization has been removed as well, since the absence of a pooling step made it unnecessary.

As in the previous section, we detail below the filter learning algorithms and the individual components of our framework, describe the datasets we use, and the comparative results we obtain.

4.1. Learning the filters

We first tried using the unsupervised filter learning algorithm of Section 3. As discussed, one key weakness of this formulation is that, even though the filter replication due to translations is avoided, nothing prevents two filters from independently converging to an identical solution. This is usually caused by strong gradients, which dominate the reconstruction error term. This is particularly true in images containing neat, curvilinear profiles, such as those of Fig. 6. While the regularization term pushes for an economical representation, the regularization parameter λ_{learn} cannot make the sparsity penalty prevail over the reconstruction error without trivial filters appearing. Furthermore, the ℓ_1 regularizer penalizes similarly all cases where a certain amount of energy is equally split among similar filters. In fact, this is the main difficulty in using the ℓ_1 norm in place of the ℓ_0 norm for sparsity promotion. The precondition requiring the original image to be truly sparse, which is requested for the solutions obtained by the two norms to be equivalent [22, 23], is indeed generally satisfied by natural images [9]. We therefore introduce an additional term in the objective function of Eq. (3) that penalizes filters that are too similar, where the similarity is expressed in terms of the squared dot product. We look for

$$\underset{\{\mathbf{f}^j\},\{\mathbf{t}_i^j\}}{\operatorname{argmin}} \sum_i \left(\left\| \mathbf{x}_i - \sum_j \mathbf{f}^j * \mathbf{t}_i^j \right\|_2^2 + \xi \sum_j \sum_{k \neq j} \left(\langle \mathbf{f}^j, \mathbf{f}^k \rangle \right)^2 + \lambda_{learn} \sum_j \left\| \mathbf{t}_i^j \right\|_1 \right) . \quad (7)$$

Even though this does not completely prevent replication, it makes it much rarer. A related approach has been independently proposed in [78]. Fig. 7 depicts the filter banks learned on the three datasets of Fig. 6. Unsurprisingly, the resulting shapes match the structures present in each image type, i.e., curvilinear profiles with ridges for the DRIVE dataset, pointwise structures for the noisy neurons images, and straight, parallel elements for the more geometrically defined roads dataset.

4.2. Pixel classification datasets

We used three very different datasets.

The first one is the publicly available DRIVE dataset of retinal images, where the aim is to automatically segment blood vessels [79]. It is composed of 40 RGB-formatted retinal scans, which were originally obtained for the diagnosis of diabetic retinopathy. In our experiments we used only the green channel, since it has been shown to give the highest contrast between background and vessels [80]. Fig. 6(a) shows an example retinal scan from this dataset. The images typically have a uniform background with the vessels appearing as dark linear structures. We use segmentations of the underlying vasculatures provided by expert ophthalmologists as ground truth for performing our evaluations.

The second dataset is made of minimum intensity projections of bright-field micrographs, such as that of Fig. 6(b), paired up with annotations made by a human expert. The bright-field micrographs are obtained from biocityne-dyed rat neurons. Due to irregularities in the staining process, they contain both structured and unstructured noise that is difficult to distinguish from the dendrites. Also, the minimum intensity projection of points from a 3D stack to a 2D image introduces a significative noise component.

The third dataset is made of aerial images, such as the one of Fig. 6(c), which contain road networks of a residential area in the United States. Segmenting streets from these images is a challenging task as they are often occluded by trees along roadsides and medians. Furthermore the image intensities of the streets vary according to the quality of the asphalt, and the background is cluttered with many complex structures that can be mistaken for roads such as houses, swimming pools, and parking lots. We manually annotated the streets and used these annotations as ground truth for both training and testing.

4.3. Experimental setup

We manually delineated the centerlines of the training images to distinguish between the target linear structures and the background in the supervised training phase. In total we traced 8 training images for the DRIVE dataset and 1 high-resolution image for both the neurons and the roads dataset. Please note that these delineations are only used at training time for the acquisition of training samples. No such tracing is therefore required for the test images. We collected potentially ambiguous negative instances by randomly sampling points within a short distance from the traced centerlines. These examples constitute half of the negative samples, and the other half was also randomly selected from the rest of the background. The same training methodology has been utilized in [50], and therefore the results can be quantitatively compared. To account for contrast and brightness variations across different images we rescale pixel intensity values using a zero-mean unit-variance normalization. For each sample in the dataset we then compute a feature vector by convolving the learned filters with the normalized images. These feature vectors are used to train classifiers at training time and to obtain classification scores at test time. In this paper we use Support Vector Machines as baseline classifiers.

4.4. Results and discussion

We compare our results against the very widely used Hessian-based technique of [39], the Oriented Flux Filter of [46], and our earlier supervised learning approach [50] that relies on steerable filters instead of the learned filters presented here. We use multiscale implementations for all the competing methods and compare their output to that of our filter banks learned at a single scale.

Fig. 8 summarizes the results on our three datasets by using Precision/Recall curves, while the corresponding F-measure values are reported in Tab. 5. Our method consistently scores better than our three baselines. As a final remark, the non-monotonic shape of some curves in Fig. 8 can be explained by strong responses due to the high contrast present in some areas, such as the image boundaries or, for the DRIVE dataset case, the optical disc, as discussed in [81].

While the performance of a classifier on a given dataset is readily established by computing the number of successfully classified items, no such measure exist for the pixel classification task. For this reason we include in Tab. 5 the Area Under Curve (AUC) and two analytic measures of segmentation quality, namely the Variation of Information (VI) [82] and the Rand Index (RI) [83]. Both the VI and the RI require a thresholded image, and we automatically pick the best threshold identified by the F-measure. The results are consistent with the Precision/Recall curves. More extensive results, including the ROC curves corresponding to the Precision/Recall curves of Fig. 8, are included in the appendices.

The method in [50] uses a richer vocabulary of filters than those of [39, 46], which can account for irregularities in the data. Nonetheless, these filters being weighted sums of Gaussians and Gaussian derivatives, they only have limited expressive power. Our filters are learned on the data itself and they are therefore more expressive, especially for non-standard profiles which cannot be reliably detected by methods such as [39]. The main drawback of our filters, compared with steerable ones, is that they can adapt to the data only at the cost of losing the separability of the Gaussian filters.

In Fig. 8, note the good performance of random filters in both the neurons and the roads datasets. This result can be easily explained by the fact that both datasets are heavily corrupted by noise, up to a point that even the human segmentation presents gross mistakes. The SVM is at ease with the representation provided by the random filters of such images, which is a sort of Compressively Sensed representation of them, while it gets confused by the unstable representation obtained when, for example, the smooth Gaussianly shaped filters adopted by [50] are fitted to the given data. A visual inspection of the resulting pixel classifications reveals that the profiles extracted by random filters are not as sharply defined as those obtained by learned filters or rotational features (see Fig. 9). Also, the performance of random filters drops quickly as the number of filters decreases.

Using the result obtained with learned filters as a baseline, we investigate whether our approach to learning the filters can also be used to optimize the feature maps as was done for the image categorization task. We therefore compare the classification scores for the plain convolution case against those achieved by the Iterative Thresholding algorithm for different levels of sparsity by solving the minimization problem of Eq. (5). Since the results of the learning-based approaches depend on the samples collected during the supervised training, we fix these points to provide a fair comparison. The most significant results are reported in Fig. 8(d), and they show that feature vectors computed by convolution perform better than the ones computed from sparsified feature maps.

5. Conclusion

We performed an in-depth analysis of the role of sparsity for image categorization and pixel classification. The consistency of our results for these two very different tasks suggests that sparsity is essential to learn effective filter banks at training time but that enforcing it at run-time is not particularly useful in convolutional architectures, at least when the level of noise remains reasonable. On the other hand, sparsity turns out to be important when redundancy is either introduced (e.g., by extracting features on overlapping patches) or already present in the data (e.g., by considering strongly correlated image channels). Given the high computational burden involved in the enforcement of sparsity, these findings should be taken into account when building actual recognition systems designed to work on large images.

One weakness of our approach is that, since the filters are not separable, the convolutions are difficult to compute very efficiently, and generalizing this approach to cubes of data as opposed to images as in [50] would be prohibitively expensive. Future work will therefore focus on optimizing the filters so that this difficulty can be overcome.

References

[1] M. A. Ranzato, F. Huang, Y.-L. Boureau, Y. LeCun, Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2007.

[2] R. Raina, A. Battle, H. Lee, B. Packer, A. Y. Ng, Self-taught Learning: Transfer Learning from Unlabeled Data, in: Int. Conf. on Mach. Learn. 2007.

[3] J. Yang, K. Yu, Y. Gong, T. Huang, Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2009.

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

[5] R. Baddeley, L. F. Abbott, M. C. Booth, F. Sengpiel, T. Freeman, E. A. Wakeman, E. T. Rolls, Responses of neurons in primary and inferior temporal visual cortices to natural scenes, Proc. R. Soc. Lond. [Biol.] 1997.

[6] W. E. Vinje, J. L. Gallant, Sparse Coding and Decorrelation in Primary Visual Cortex During Natural Vision, Science 2000.

[7] P. Berkes, B. L. White, J. Fiser, No evidence for active sparsification in the visual cortex, in: Adv. Neural Inf. Process. Syst. 2009.

[8] B. A. Olshausen, D. J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature 1996.

[9] B. A. Olshausen, D. J. Field, Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?, Vision Res. 1997.

[10] M. Elad, M. Figueiredo, Y. Ma, On the Role of Sparse and Redundant Representations in Image Processing, IEEE 2010.

[11] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Non-local Sparse Models for Image Restoration, in: Int. Conf. on Comput. Vis. 2009.

[12] J.-L. Starck, M. J. Fadili, An Overview of Inverse Problem Regularization using Sparsity, in: Int. Conf. on Image Processing. 2009.

[13] M. A. Ranzato, C. Poultney, S. Chopra, Y. LeCun, Efficient Learning of Sparse Representations with an Energy-Based Model, in: Adv. Neural Inf. Process. Syst. 2006.

[14] X. Glorot, A. Bordes, Y. Bengio, Deep Sparse Rectifier Neural Networks, J. Mach. Learn. Res. 2011.

[15] A. Coates, A. Y. Ng, The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization, in: Int. Conf. on Mach. Learn. 2011.

[16] R. Rigamonti, M. Brown, V. Lepetit, Are Sparse Representations Really Relevant for Image Classification?, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2011.

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

[18] M. D. Zeiler, G. W. Taylor, R. Fergus, Adaptive Deconvolutional Networks for Mid and High Level Feature Learning, in: Int. Conf. on Comput. Vis. 2011.

[19] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, Convex optimization with sparsity-inducing norms, MIT Press. 2011.

[20] M. Zibulevsky, M. Elad, L1-L2 Optimization in Signal and Image Processing, IEEE Signal Process. Mag. 2010.

[21] A. M. Bruckstein, D. L. Donoho, M. Elad, From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images, SIAM Rev. 2009.

[22] E. J. Candès, J. Romberg, T. Tao, Stable Signal Recovery from Incomplete and Inaccurate Measurements, Comm. Pure Appl. Math. 2006.

[23] D. L. Donoho, Compressed Sensing, IEEE Trans. Inform. Theory 2006.

[24] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust Object Recognition with Cortex-Like Mechanisms, IEEE Trans. Pattern Anal. Mach. Intell. 2007.

[25] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Discriminative Learned Dictionaries for Local Image Analysis, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2008.

[26] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, Y. LeCun, What is the Best Multi-Stage Architecture for Object Recognition?, in: Int. Conf. on Comput. Vis. 2009.

[27] Y.-L. Boureau, F. Bach, Y. LeCun, J. Ponce, Learning Mid-Level Features for Recognition, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2010.

[28] M. D. Zeiler, D. Krishnan, G. W. Taylor, R. Fergus, Deconvolutional Networks, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2010.

[29] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, Y. LeCun, Learning Convolutional Feature Hierarchies for Visual Recognition, in: Adv. Neural Inf. Process. Syst. 2010.

[30] M. A. Ranzato, V. Mnih, G. E. Hinton, Generating more realistic images using gated MRF's, in: Adv. Neural Inf. Process. Syst. 2010.

[31] Q. V. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, A. Y. Ng, Tiled convolutional neural networks, in: Adv. Neural Inf. Process. Syst. 2010.

[32] M. Brown, G. Hua, S. Winder, Discriminative Learning of Local Image Descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 2010.

[33] M. A. Ranzato, Y.-L. Boureau, Y. LeCun, Sparse Feature Learning for Deep Belief Networks, in: Adv. Neural Inf. Process. Syst. 2007.

[34] T. Lee, R. Kashyap, C. Chu, Building Skeleton Models via 3D Medial Surface Axis Thinning Algorithms, Graphical Models and Image Processing 1994.

[35] Z. Vasilkoski, A. Stepanyants, Detection of the Optimal Neuron Traces in Confocal Microscopy Images, Journal of Neuroscience Methods 2009.

[36] P. Chothani, V. Mehta, A. Stepanyants, Automated tracing of neurites from light microscopy stacks of images, Neuroinformatics 2011.

[37] E. Meijering, M. Jacob, J.-C. Sarria, P. Steiner, H. Hirling, M. Unser, Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images, Cytometry A 2004.

[38] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, R. Kikinis, 3D Multi-Scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images, Med. Image Anal. 1998.

[39] A. F. Frangi, W. J. Niessen, K. L. Vincken, M. A. Viergever, Multiscale vessel enhancement filtering, in: Med. Image Comput. Comput. Assist. Interv. 1998.

[40] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, Y. Trousset, Model Based Detection of Tubular Structures in 3D Images, Comput. Vis. Image Underst. 2000.

[41] G. J. Streekstra, J. van Pelt, Analysis of tubular structures in three-dimensional confocal images, Network-Comp. Neural 2002.

[42] K. A. Al-Kofahi, S. Lasek, D. H. Szarowski, C. J. Pace, G. Nagy, J. N. Turner, B. Roysam, Rapid Automated Three-Dimensional Tracing of Neurons From Confocal Image Stacks, IEEE Trans. Inf. Technol. Biomed. 2002.

[43] A. Dima, M. Scholz, K. Obermayer, Automatic Segmentation and Skeletonization of Neurons from Confocal Microscopy Images Based on the 3D Wavelet Transform, IEEE Trans. Image Process. 2002.

[44] S. Schmitt, J.-F. Evers, C. Duch, M. Scholz, K. Obermayer, New Methods for the Computer-Assisted 3D Reconstruction of Neurons from Confocal Image Stacks, Neuroimage 2004.

[45] J. A. Tyrrell, E. di Tomaso, D. Fuja, R. Tong, K. Kozak, R. K. Jain, B. Roysam, Robust 3-D Modeling of Vascular Imagery Using Superellipsoids, IEEE Trans. Med. Imag. 2007.

[46] M. Law, A. Chung, Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux, in: Europ. Conf. on Comput. Vis. 2008.

[47] G. Agam, C. Wu, Probabilistic Modeling-Based Vessel Enhancement in Thoracic CT Scans, IEEE Conf. on Comput. Vis. and Pattern Recogn. 2005.

[48] R. Socher, A. Barbu, D. Comaniciu, A Learning-Based Hierarchical Model for Vessel Segmentation, in: IEEE Int. Symp. Biomed. Imaging. 2008.

[49] A. Santamaría-Pang, C. M. Colbert, P. Saggau, I. A. Kakadiaris, Automatic Centerline Extraction of Irregular Tubular Structures Using Probability Volumes from Multiphoton Imaging, in: Med. Image Comput. Comput. Assist. Interv. 2007.

[50] G. González, F. Fleuret, P. Fua, Learning Rotational Features for Filament Detection, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2009.

[51] D. G. Lowe, Distinctive Image Features from Scale-Invariants Keypoints, Int. J. Comput. Vision 2004.

[52] A. Coates, H. Lee, A. Y. Ng, An Analysis of Single-Layer Networks in Unsupervised Feature Learning, in: Adv. Neural Inf. Process. Syst. 2010.

[53] G. E. Hinton, Learning to represent visual input, Phil. Trans. R. Soc. B 2010.

[54] D. Cireşan, A. Giusti, L. Gambardella, J. Schmidhuber, Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images, in: Adv. Neural Inf. Process. Syst. 2012.

[55] L. Quoc, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, A. Ng, Building High-Level Features Using Large Scale Unsupervised Learning, in: Int. Conf. on Mach. Learn. 2012.

[56] A. Torralba, R. Fergus, W. T. Freeman, 80 million tiny images: a large dataset for non-parametric object and scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. 2008.

[57] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Master's thesis, University of Toronto, 2009.

[58] L. Fei-Fei, R. Fergus, P. Perona, Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2004.

[59] D. Grimes, R. P. Rao, Bilinear Sparse Coding for Invariant Vision, Neural Computat. 2005.

[60] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-Based Learning Applied to Document Recognition, IEEE 1998.

[61] H. Lee, A. Battle, R. Raina, A. Y. Ng, Efficient sparse coding algorithms, in: Adv. Neural Inf. Process. Syst. 2006.

[62] Y. LeCun, L. Bottou, G. B. Orr, K.-R. Müller, Efficient BackProp, Springer. 1998.

[63] I. Daubechies, M. Defrise, C. D. Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Comm. Pure Appl. Math. 2004.

[64] H. A, J. Hurri, P. O. Hoyer, Natural Image Statistics, Springer-Verlag, 2009.

[65] T. Leung, J. Malik, Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons, Int. J. Comput. Vision 2001.

[66] D. H. Hubel, T. N. Wiesel, Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex, J. Physiol. 1962.

[67] E. Tola, V. Lepetit, P. Fua, DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo, IEEE Trans. Pattern Anal. Mach. Intell. 2010.

[68] H.-T. Lin, C.-J. Lin, A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods, Technical Report, National Taiwan University, 2003.

[69] S. S. Keerthi, C.-J. Lin, Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel, Neural Computat. 2003.

[70] N. Pinto, D. D. Cox, J. J. DiCarlo, Why is Real-World Visual Object Recognition Hard?, PLoS Comput. Biol. 2008.

[71] A. Torralba, A. A. Efros, Unbiased Look at Dataset Bias, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2011.

[72] G. Hua, M. Brown, S. Winder, Discriminant Embedding for Local Image Descriptors, in: Int. Conf. on Comput. Vis. 2007.

[73] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, A. Y. Ng, On Random Weights and Unsupervised Feature Learning, in: Adv. Neural Inf. Process. Syst. 2010.

[74] J. Yang, Y. Li, Y. Tian, L. Duan, W. Gao, Group-sensitive multiple kernel learning for object categorization, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2009.

[75] M. A. Ranzato, G. E. Hinton, Modeling Pixel Means and Covariances Using Factorized Third-Order Boltzmann Machines, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2010.

[76] K. Yu, T. Zhang, Improved Local Coordinate Coding using Local Tangents, in: Int. Conf. on Mach. Learn. 2010.

[77] X. Ren, J. Malik, Learning a Classification model for segmentation, in: Int. Conf. on Comput. Vis. 2003.

[78] I. Ramirez, P. Sprechmann, G. Sapiro, Classification and Clustering via Dictionary Learning with Structured Incoherence and Shared Features, in: IEEE Conf. on Comput. Vis. and Pattern Recogn. 2010.

[79] J. Staal, M. D. Abràmoff, M. Niemeijer, M. A. Viergever, B. van Ginneken, Ridge-Based Vessel Segmentation in Color Images of the Retina, IEEE Trans. Med. Imag. 2004.

[80] M. Patasius, V. Marozas, D. Jegelevicius, A. Lukoševičius, Ranking of color space components for detection of blood vessels in eye fundus images, in: Proc. ECIFMBE. 2009.

[81] M. Sofka, C. V. Stewart, Retinal Vessel Centerline Extraction Using Multiscale Matched Filters, Confidence and Edge Measures, IEEE Trans. Med. Imag. 2006.

[82] M. Meilă, Comparing Clusterings - an information based distance, J. Multivariate Anal. 2007.

[83] R. Unnikrishnan, C. Pantofaru, M. Hebert, Toward Objective Evaluation of Image Segmentation Algorithms, IEEE Trans. Pattern Anal. Mach. Intell. 2007.

Table 4: Analysis on the relevance of sparsity according to the feature extraction method performed with the architecture and the code of [15]. Unless otherwise stated, the results have been obtained on grayscale images. The original pooling scheme of [15] (average pooling on 16×16 regions) has been used, except for the features extracted in a convolutional way where reducing the region size to 6×6 gave better results. Additional results are available as supplemental material

a. *PATCH-WISE OVERLAPPING, 900 filters (size* 6×6*)*

	Coates et al. [15], grayscale		**Coates et al. [15], color**	
α	SVM	NN	SVM	NN
0	70.08%	38.54%	75.32%	43.89%
0.0025	70.11%	38.41%	75.34%	44.67%
0.025	70.11%	38.41%	75.70%	44.52%
0.25	73.15%	40.26%	77.51%	44.45%
1	74.65%	37.64%	68.81%	37.71%

d. *CONVOLUTIONAL, 25 filters (size* 11×11*)*

	Coates et al. [15]		**Learned with Eq.** (3)	
α	SVM	NN	SVM	NN
0	59.69%	32.26%	61.57%	39.77%
0.0025	59.83%	32.09%	61.47%	39.85%
0.025	59.74%	32.40%	57.28%	35.29%
0.25	36.42%	15.06%	15.87%	10.97%

b. *PATCH-WISE DISTINCT,* 6×6 *filters from [15]*

	900 filters, grayscale		**300 filters, color**	
α	SVM	NN	SVM	NN
0	46.90%	19.66%	56.57%	27.77%
0.0025	45.97%	19.09%	56.73%	27.52%
0.025	46.77%	18.94%	58.19%	27.90%
0.25	42.38%	19.34%	58.08%	26.13%
1	40.12%	15.04%	55.39%	19.77%

e. *CONVOLUTIONAL, 400 filters (size* 7×7*)*

	Coates et al. [15]		**Random filters**	
α	SVM	NN	SVM	NN
0	56.82%	39.30%	54.67%	29.41%
0.0005	56.75%	39.26%	54.69%	29.19%
0.005	56.58%	39.65%	54.80%	28.28%
0.01	56.01%	38.93%	54.63%	28.70%

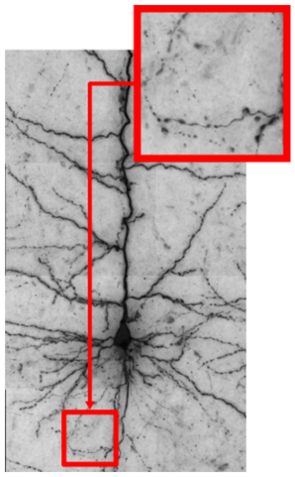
c. *PATCH-WISE OVERLAPPING, 25 filters (size* 11×11*)*

	Coates et al. [15]		**Learned with Eq.** (3)	
α	SVM	NN	SVM	NN
0	54.39%	41.74%	51.27%	37.94%
0.0025	53.32%	40.88%	51.31%	38.65%
0.025	53.32%	41.37%	51.21%	38.22%
0.25	53.28%	40.91%	50.89%	36.76%

f. *Same as (b), 500 train samples*

	900 filters		**90 filters**	
α	SVM	NN	SVM	NN
0	32.51%	16.57%	35.94%	18.31%
0.025	32.56%	17.01%	36.01%	17.93%
0.25	34.48%	17.27%	36.18%	18.71%
0.5	35.23%	17.13%	35.54%	18.30%
1	35.75%	13.76%	34.24%	15.69%

(a) DRIVE dataset


(b) Neurons dataset

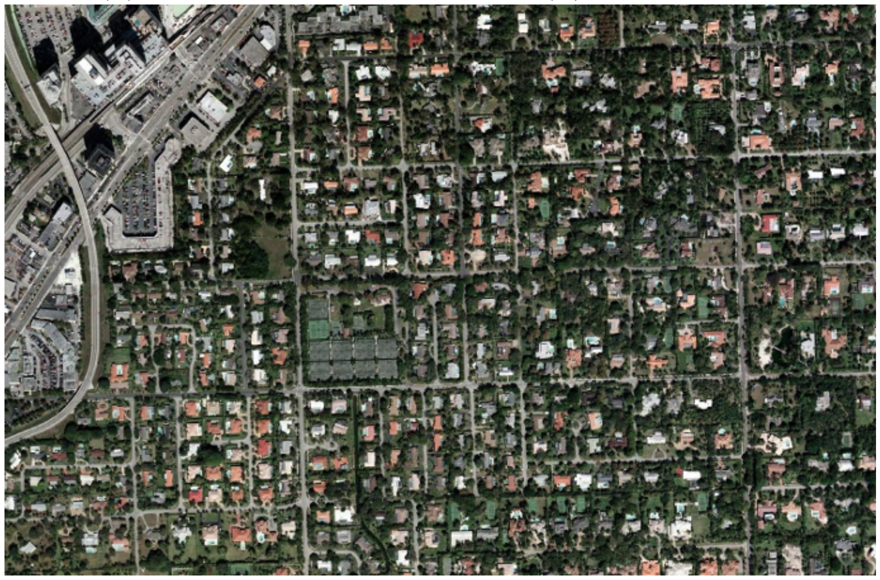

(c) Roads dataset

Figure 6: Sample images from the DRIVE dataset **(a)**, the neurons dataset **(b)**, and the roads dataset **(c)**. The red square in **(b)** outlines the difficulty of the dendritic images by showing the point-wise nature of the target structures, which can be easily confused with the superimposed noise or with segments from the adjacent layers in the image stack. Stitching artifacts due to the imaging process are also present in the image.

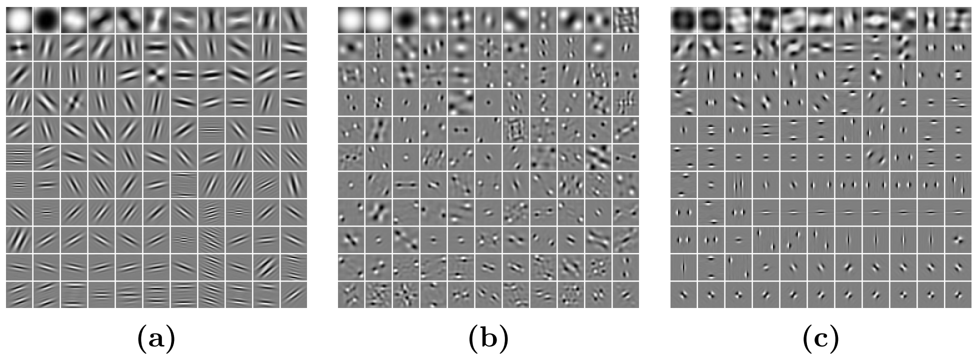
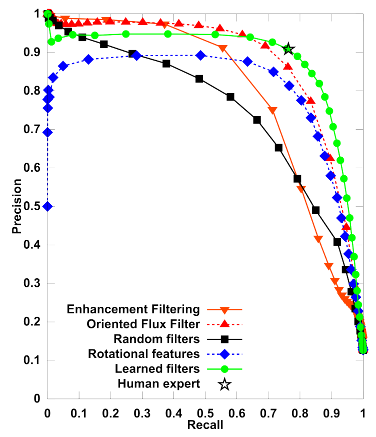
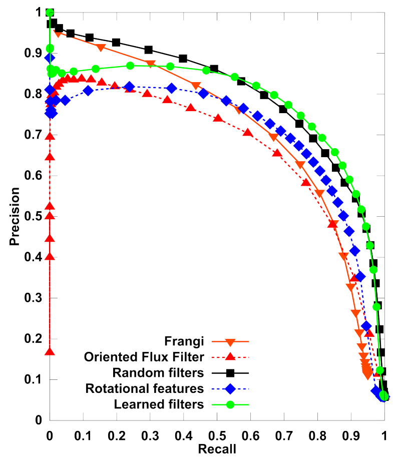


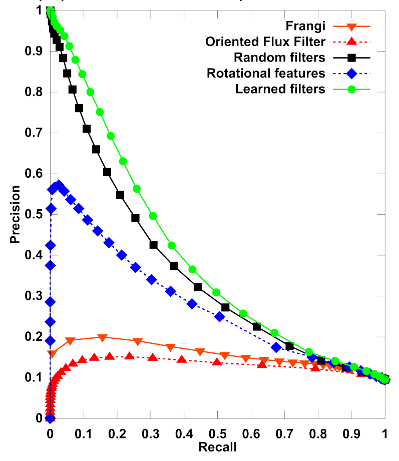
Figure 7: Examples of filter banks learned, in an unsupervised way, on the DRIVE dataset **(a)**, the neurons dataset **(b)**, and the roads dataset **(c)**. The filters are ordered according to the ℓ_2 norm of their responses on a test image.



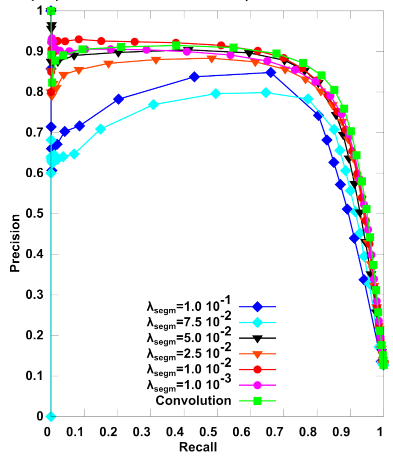
(a) DRIVE, P/R curve



(b) Neurons, P/R curve



(c) Roads, P/R curve

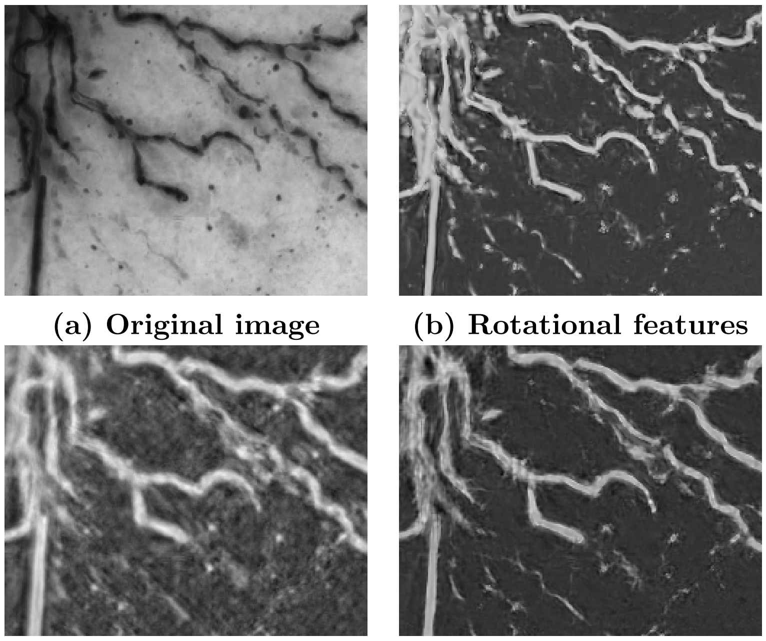


(d) Different refinements

Figure 8: Precision/Recall curves for some images of the used datasets. **(a):** Image 19 of the DRIVE dataset. Our method outperforms those presented in [39, 50, 46], and according to the Precision/Recall curves it provides a pixel classification comparable with that of the second human expert. This point is validated also by the numerical evaluations presented in Tab. 5. **(b):** Neurons dataset. Our method clearly outperforms the results of [39, 50, 46]. Learning a classifier improves the results, but learning both the filter bank and the classifier yields the best classification. Please note the astonishing performance of the random filters on this particular dataset. **(c):** Roads dataset. This dataset is the most challenging one and, as expected, yields the lowest scores. Our method markedly outperforms [39, 50, 46]. Again, random filters perform better than those methods that assume the presence of neat and highly structured components in the image. **(d):** Precision/Recall curves reporting the classification performances on the image 19 of the DRIVE dataset for different degrees of sparsity of the representation (classifier SVM, 2500 positive and 2500 negative samples).

Table 5: Analytic measures of the quality of the pixel classification for the experiments presented in Fig. 8. Both the VI and the RI are computed on the classification thresholded at the value found using the F-measure. Please note that VI assumes values in $[\,0, \infty)$, the lower the better, and RI assumes values in $[0, 1]$, the higher the better. Learned filters with SVM consistently score better than the competing methods in terms of AUC, F-measure, VI, and RI

Method	**AUC**	**F-measure**	**VI**	**RI**
DRIVE, image 19				
Ground truth		*0.8301*	*0.4780*	*0.9099*
Frangi	0.9311	0.7326	0.5890	0.8810
Oriented Flux Filter	0.9663	0.8106	0.4887	0.9098
Random filters, SVM	0.9364	0.6938	0.6759	0.8585
Rotational features,SVM	0.9581	0.7907	0.5347	0.8986
Learned filters,SVM	**0.9717**	**0.8419**	**0.4269**	**0.9245**
Neurons				
Frangi	0.9385	0.6855	0.3792	0.9261
Oriented Flux Filter	0.9561	0.6684	0.3987	0.9208
Random filters, SVM	**0.9782**	0.7371	0.3337	0.9381
Rotational features, SVM	0.9467	0.7070	0.3606	0.9311
Learned filters, SVM	0.9742	**0.7503**	**0.3217**	**0.9411**
Roads				
Frangi	0.6710	0.2414	1.2501	0.6085
Oriented Flux Filter	0.6286	0.2159	1.4278	0.5120
Random filters, SVM	0.7554	0.3731	0.8686	0.7737
Rotational features, SVM	0.7416	0.3378	0.9848	0.7299
Learned filters, SVM	**0.7715**	**0.3939**	**0.8178**	**0.7917**


(a) Original image **(b) Rotational features**

(c) Random filters **(d) Learned filters**

Figure 9: Detail of the pixel classification of an image in the neurons dataset. **(a)** Segment of the original image. **(b)** Classification provided by rotational features. **(c)** Classification provided by random filters. Please note how the contours of the dendrites are not as sharply defined as the other two cases. **(d)** Classification provided by learned filters.