Is Sparsity Really Relevant for Image Classification ?*

Roberto Rigamonti (roberto.rigamonti@epfl.ch)
http://cvlab.epfl.ch/~rigamont

Matthew Brown (matthew.brown@epfl.ch)
http://cvlab.epfl.ch/~brown

Vincent Lepetit (vincent.lepetit@epfl.ch)
http://cvlab.epfl.ch/~lepetit

School of Computer and Communication Sciences
Swiss Federal Institute of Technology, Lausanne (EPFL)

Technical Report

April 06, 2010

*This work has been supported in part by the Swiss National Science Foundation.

Abstract

Recent years have seen an increasing interest in sparseness constraints for image classification and object recognition, probably motivated by the evidence of sparse representations internal in the primate visual cortex. It is still unclear, however, whether or not sparsity helps classification. In this paper we analyze the image classification task on CIFAR-10, a very challenging dataset, and evaluate the impact of sparseness on the recognition rate using both standard and learned filter banks in a modular architecture. In our experiments, enforcing sparsity constraints is not required at run-time, since no performance improvement has been reported by indiscriminatively sparsifying the descriptors. This observation complies with the most recent findings on human visual cortex suggesting that a feed-forward mechanism underlies object recognition, and is of practical interest, as enforcing these constraints can have a heavy computational cost. Our best method outperforms the state-of-the-art, from 64.84% success rate on color images to 71.53% on grayscale images.

Contents

Chapter 1

Introduction

Probably inspired by massive work from the neuroscience community [5, 1, 6, 32, 45, 33], many recent Computer Vision algorithms rely on sparse image representations, including [8, 38, 39, 40, 44, 37, 27, 28, 18, 41, 22, 3, 23, 17, 49]. The effectiveness of sparse coding from a generative point of view, for image restoration for example, is justified by observing that natural images represent only a tiny part of the image space [33]. While sparse representations have been regarded as more likely to be separable in high-dimensional spaces, and therefore suitable for classification, it is still not clear if they are actually needed for recognition tasks [19].

Understanding if sparse representations are really relevant for image classification is not only interesting from a theoretical point of view, but also from a practical point of view. Their encoding requires indeed the solution of an optimization algorithm that has proven to be NP-hard [15]. Despite the signal processing community has developed a stir of algorithms facing a very similar problem in the Compressive Sensing field, this task remains extremely expensive in computational terms.

In this paper we aim to evaluate the actual importance of sparsity in image classification, by focusing on the hard recognition problem posed by the CIFAR-10 dataset [47], made of 32×32 pixel images of different object categories. Using the architecture presented in Fig. 1.1 and made of a feature extraction stage and a pooling stage, similar to those used in recent image classification algorithms [26, 42, 30, 4, 48, 40, 17], we analyze the impact that sparsity has on the recognition rate.

The results of our experiments advocate that no advantage is gained by imposing sparsity at run-time, at least when this sparsity is not tailored in a discriminative fashion, and the currently employed expensive optimization procedures can be skipped without affecting the success rate. Our claim complies with the most recent evidences from studies on the human visual cortex [25, 43], suggesting that a feed-forward mechanism underlies the initial stages of object recognition.

Even though our study was not focused on improving the state-of-the-art success rate on the CIFAR-10 dataset, our best method outperforms the current state-of-the-art result [20], moving from a 64.84% to a 71.53% success rate. Moreover, our result has been achieved on grayscale images, whereas [20] considered color images.

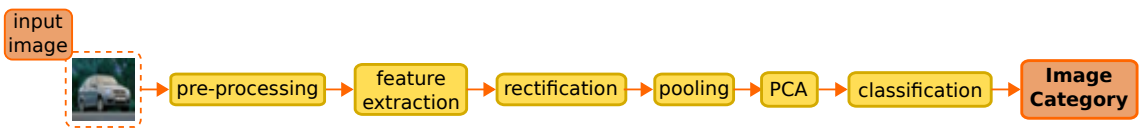


Figure 1.1: Our pipeline, similar to many recent approaches. We tried different options for each of the different steps, in order to evaluate the importance of sparsity constraints in the Feature Extraction step, and the impact of the overall architecture on the recognition rate.

Chapter 2

Related Work

Imposing sparsity constraints has recently become popular in Computer Vision, especially for image recognition tasks. This is probably due to evidence for sparse representations in the mammal brain, and because simple algorithms based on such constraints can reproduce linear filters similar to the receptive fields of neurons in V1, the first layer of the visual cortex [32, 38]. These approaches are also attractive because they require only unlabeled data, which are much simpler to produce than labeled data, for their training.

As a result, such algorithms have been used to extract features that are assumed to be relevant for classification tasks [39, 37, 41, 22, 17]. Sparsity is also convenient to constrain over-complete linear representations, which would be ill-posed without such constraints [23].

Sparse coding can be interpreted as learning the input data distribution with a sparse prior [32]. Restricted Bolzmann Machines (RBMs) have been used to learn the data distribution and extract features in an unsupervised way [10], and [22] showed that sparsity constraints are useful to make them converge on natural images toward V1 receptive fields-like filters.

Despite their popularity and to the best of our knowledge, sparsity constraints have not really been evaluated in terms of classification performances. Is it really important to enforce sparsity constraints to learn how to extract good features ? Is it important to enforce sparsity constraints when extracting the features at run-time, even though it can be costly ?

Very recently, [17] developed an architecture very close to ours. They showed the importance of the rectification function that we also use as a non-linear operation between the feature extraction and the pooling stage, and the power of stacking multiple layers. Nonetheless, they did not really evaluate the effects of sparsity, as they only compared learned sparse features against convolutions with random filters. [49] has shown the clear value of sparsity constraints for face recognition, but in conditions that are difficult to generalize to other problems.

Predictive Sparse Decomposition [18] is an attempt to overcome the issue posed by the synthesis of sparse codes by learning a regressor that is able to approximate the optimal reconstruction. This algorithm has been used as a building block in [17], and represents a viable strategy to impose sparseness without having to solve a costly optimization problem, but the solution obtained through the learned regressor is only an approximation of the true, sparse solution.

This technical report represents an attempt to fill the gap represented by the absence of a thorough evaluation of the real contribution of sparseness in the image classification task, in order to be able to focus future analyses towards the most relevant aspects of an image classification architecture.

Chapter 3

Evaluation Framework

Our framework for evaluation is very similar to the ones used in recent works [26, 42, 30, 4, 48, 40] and particularly to [17]. We rely on a first stage to extract features, which are either learned or hard-wired. We apply a non-linear operation to the output, then the second stage "pools" the features to obtain some robustness to small translations and deformations. We try different pooling methods.

Before going through the first stage the images are pre-processed with a whitening operation, as this improves the convergence of the learning algorithm and the recognition performances. We linearly project the output of the second stage into some subspace for dimensionality reduction. Again, we try different possible projections. The result is finally fed to different types of classifier.

We detail below these different parts of our framework, while the results of the evaluation are given in the next subsection. The different combinations we tried are called by concatenating the names of their parts together. The possible part names are given below in capital letters. For example, "SPARSEGD-MAX-PCA-SVM" represents the pipeline where we first extract sparse features with the Olshausen and Field's algorithm [32], use the absolute value function for rectification, use a max-pooling operation, project the result into an eigenspace, and finally use a Support Vector Machine for classification.

3.1 Feature Extraction Stage

3.1.1 Learned Features

To learn image features, we chose to adopt the Olshausen and Field's algorithm [32], used in many recent works [37, 27, 17] and known to converge well on natural image patches. We have only slightly modified it for more efficiency when processing images.

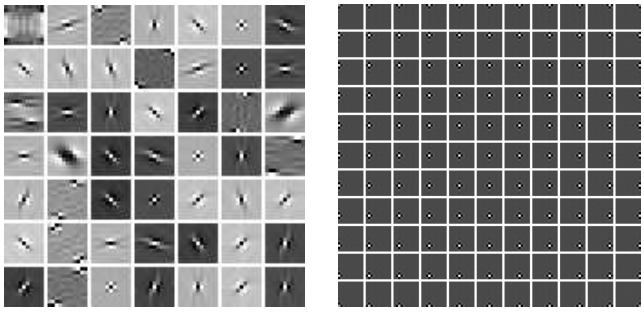
In [32], Olshausen and Field suggested that V1, the first layer of the visual cortex, builds a sparse representation of the images. Under this assumption and the hypothesis that a perfect reconstruction in attainable, the problem one would like to solve can be stated as

$$\begin{aligned} &\min_{\mathbf{M},\{\mathbf{t}_i\}} \sum_i \|\mathbf{t}_i\|_0 \\ &\text{s.t.} \sum_i \|\mathbf{x}_i - \mathbf{M}\mathbf{t}_i\|_2^2 = 0 \,, \end{aligned} \tag{3.1}$$

where the ℓ_0-norm, the number of non-zero elements, is the best sparsity measure available. The formulation in Eq. (3.1) is, however, non-convex, making the optimization much more difficult. The version proposed in [32] therefore learns a dictionary of features by optimizing the following objective function:

$$\min_{\mathbf{M},\{\mathbf{t}_i\}} \sum_i \|\mathbf{x}_i - \mathbf{M}\mathbf{t}_i\|_2^2 + \lambda \|\mathbf{t}_i\|_1 \;, \tag{3.2}$$

where $\mathbf{x}_i$ are training images, $\mathbf{t}_i$ are the corresponding feature vectors, $\mathbf{M}$ is a matrix whose columns form the dictionary, and $\|\cdot\|_1$ is the ℓ_1-norm used to enforce sparsity on the vectors $\mathbf{t}_i$.



(a) (b)

Figure 3.1: Extracting features and whitening. **(a)** The 49 filters we obtain with our modified version of the Olshausen and Field's algorithm. Some filters look like the often encountered localized and oriented filters, but some are made of disconnected components. We believe that this happens because the convolutional approach is equivalent to have a very large dictionary. As a result, some filters can be used to capture less important distant dependencies. **(b)** Filter bank representation of the columns of the whitening matrix $\mathbf{W}$, obtained as described in Subsection 3.1.3. We kept only the filter in the middle and convolved it with input images in order to whiten them.

Eq. (3.2) looks for a dictionary $\mathbf{M}$ so 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. Moreover the dictionary is overcomplete: $\mathbf{M}$ has more columns than rows, and this gives us the degress of freedom that we need in order to be able to choose among all the possible representations the sparsest one. λ is a meta-parameter that establishes the relative importance of the reconstruction error $\|\mathbf{x}_i - \mathbf{M}\mathbf{t}_i\|_2^2$ with respect to the penalty term $\|\mathbf{t}_i\|_1$.

The problem introduced by a straightforward application Eq. (3.2) is that it was developed for small patches only, and using it on possibly large images is slow and difficult, as many coefficients in $\mathbf{M}$ would have to be optimized simultaneously. We therefore adopt here a convolutional approach, where the matrix-vector product is replaced by a convolution. This is possible if we assume that the local properties of images are translation invariant, which seems reasonable. [23] uses a similar approach with RBMs. Trivial solutions to the optimization problem, where filter norms are increased in order to reduce the penalty term, are avoided by performing a proper normalization step over the filters. Even though more recent investigations focused on finding a principled approach to solve this issue [34], it has been shown that the aforementioned approximation provides a reasonable solution [31].

The optimization problem in Eq. (3.2) therefore becomes:

$$\min_{\{\mathbf{f}^j\},\{\mathbf{t}_i^j\}} \sum_i \left(\left\| \mathbf{x}_i - \sum_j \mathbf{f}^j * \mathbf{t}_i^j \right\|_2^2 + \lambda \sum_j \left\| \mathbf{t}_i^j \right\|_1 \right) , \tag{3.3}$$

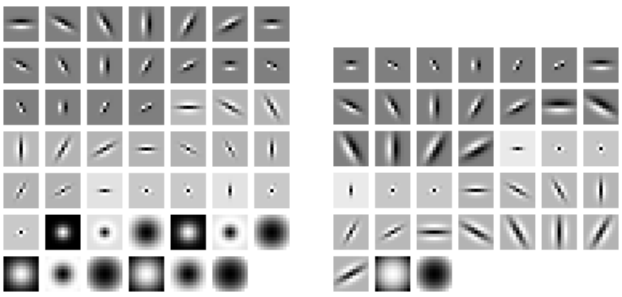
where $\mathbf{f}^j$ are linear filters and $\mathbf{t}_i^j$ can now be seen as a set of images with the same size as the $\mathbf{x}_i$ images, whose cardinality is equal to that of the filter bank. Similar intermediate representations have been called "feature maps" in the Convolutional Neural Networks literature.

The original problem in Eq. (3.2) was optimized using stochastic gradient descent, and it is easy to also use stochastic gradient descent to find the coefficients of the $\mathbf{f}^j$ filters. We learned the 49 11×11 filters shown in Fig. 3.1(a) in this way. Details about the derivation of this filter bank are reported in Appendix A.

In our evaluations, we used these filters $\mathbf{f}^j$ to extract features $\mathbf{t}^j$ from a image $\mathbf{x}$ in three different ways:

- Sparse features with gradient descent (SPARSEGD). The $\mathbf{t}^j$s are obtained by minimizing by gradient descent the following objective function:

$$\min_{\{\mathbf{t}^j\}} \left\| \mathbf{x} - \sum_j \mathbf{f}^j * \mathbf{t}^j \right\|_2^2 + \lambda \sum_j \left\| \mathbf{t}^j \right\|_1 \tag{3.4}$$



(a) (b)

Figure 3.2: Hard-wired filter banks. **(a)** The hard-wired Leung-Malik filter bank [24]. **(b)** The hard-wired Maximum Response 8 filter bank [9].

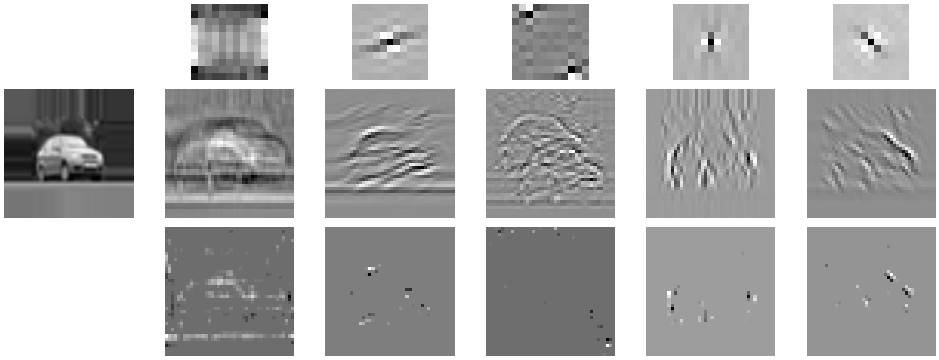


Figure 3.3: Feature extraction stage with learned features. **First row:** 5 of the 49 filters learned on the dataset. **Second row:** The original image, and the corresponding $\mathbf{t}^j$ vectors obtained by simple convolution, as done by the CONV method. **Third row:**. The corresponding $\mathbf{t}^j$ vectors obtained when enforcing the sparsity constraints with gradient descent, as it was done for the SPARSEGD method.

This optimization is the same as the one posed in Eq. (3.3) after fixing the filters $\mathbf{f}^j$ and considering only the given image.

- Sparse features with Matching Pursuit (SPARSEMP). The $\mathbf{t}^j s$ are obtained by minimizing the same function in Eq. (3.4) but using the Matching Pursuit algorithm [29]. Matching Pursuit is usually able to find much sparse feature vectors than gradient descent for a similar reconstruction error. In particular, fixing the number of non-zero coefficients is equivalent to choosing a proper value for the λ coefficient in Eq. (3.3). However, it is slower, and its slowness prevented us to use it when optimizing Eq. (3.3).

- Features computed by direct convolution (CONV). The $\mathbf{t}^j s$ are obtained by direct convolution, without any sparsity constraint:

$$\mathbf{t}^j = \mathbf{f}^j * \mathbf{x}, \ \forall j \ . \tag{3.5}$$

  This is much faster than the two previous options. It is not, however, unrelated to SPARSEGD, the first option: it corresponds to the first iteration of gradient descent required to solve Eq. (3.4) when the $\mathbf{t}^j s$ are initialized to $\mathbf{0}$.

Border replication was performed in the pre-processing step to exploit the full image information when performing convolutions. The $\mathbf{t}^j s$ corresponding to a few filters are shown Fig. 3.3 for SPARSEGD and CONV.

3.1.2 Hard-wired Filters

In addition to learned filters, we have also considered convolutions with four hard-wired filter banks:

- The Leung-Malik (LM) filter bank [24] (CONVLM). It is shown in Fig. 3.2(a), and it is composed of 2 Gaussian derivative filters at 6 orientations and 3 scales, 8 Laplacian of Gaussian filters and 4 Gaussian filters, for a total of 48 filters.
- The Maximum Response 8 filter bank [9] (CONVMR8), depicted in Fig. 3.2(b). It is made of 38 filters similar to those in LM but with different parameters.
- A filter bank constituted by 49 randomly generated filters, with the exception of the first one that is set to be uniform (CONVRAND).

3.1.3 Pre-processing (Whitening)

Removing the linear dependencies between the coefficients of the images, or whitening, is important for the convergence of Eqs. (3.3) and (3.4). A whitening operation can be learned from the covariance matrix $\mathbf{C}$ of the original data [16]. By applying to $\mathbf{C}$ an eigenvalue decomposition $\mathbf{C} = \mathbf{EDE}^\top$, a whitening matrix $\mathbf{W}$ can be computed as $\mathbf{W} = \mathbf{ED}^{-1/2}\mathbf{E}^\top$.

Similarly to Eq. (3.2), this is not really practical for large images. For this reason, after having observed the peculiar structure of the filter bank defined by the columns of $\mathbf{W}$ and depicted in Fig. 3.1(c), we have kept the central filter only, and used it to whiten the input images by convolving the filter with them.

3.1.4 Rectification

Before the pooling stage, we apply some non-linear operation to the feature vectors $\mathbf{t}^j$ as it is usually done in multi-layer architectures. This operation gives a new set of feature vectors $\mathbf{u}^j$. Again, we tried different possibilities:

- Taking the absolute values of the coefficients of the $\mathbf{t}^j$ vectors (ABS). The coefficients $\mathbf{u}^j[m]$ of the $\mathbf{u}^j$ vectors are simply taken to be: $\mathbf{u}^j[m] = \left|\mathbf{t}^j[m]\right|$. This operation was identified as very effective in [17] for recognition performances despite its simplicity.
- Separating the negative coefficients from the positive ones (POSNEG). If each submap is composed by N coefficients, the coefficients $\mathbf{u}^j[m]$ of the $\mathbf{u}^j$ vectors are taken to be:

$$\mathbf{u}^j[m] = [\mathbf{t}^j[m]]^+, \; \mathbf{u}^{N+j}[m] = [-\mathbf{t}^j[m]]^+ , \tag{3.6}$$

  where $[x]^+ = x$ if $x > 0$ and 0 otherwise. This doubles the number of coefficients in the $\mathbf{u}^j$ vectors. As can be seen in Fig. 3.4, some filters present very similar positive and negative parts. This can be explained by the fact that, when optimizing for the sparseness, high coefficients for a given filter are usually compensated by some other high coefficients with opposite sign in its neighborhood. This phenomenon has been observed before in the signal processing literature, and has motivated the creation of approaches based on the Noise Shaping algorithm, such as [7].

3.2 Pooling Stage

This stage pools the coefficients of the $\mathbf{u}^j$ vectors to provide invariance to small displacements and distortions. The choice of having a pooling stage is based on two relevant aspects:

- From a biological perspective, the pooling stage plays the role of the complex cells' layers in the Hubel and Wiesel's model of the V1 cortex [14, 13]. The role that pooling holds is that of enabling 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 [21].
- From a computational perspective, plain descriptors have a dimensionality that is too high for the current computational power. The presence of a downsampling step is therefore mandatory for enabling the subsequent subspace selection and classification operations.

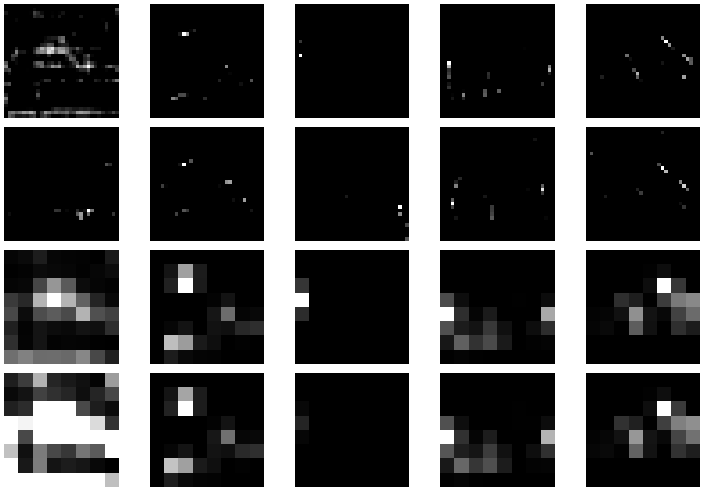


Figure 3.4: Rectification and pooling. **First two rows:** Results of the POSNEG operation applied to the 5 feature vectors on the last row of Fig. 3.3. The first row corresponds to the positive coefficients, the second row to the negative ones. **Last two rows:** Results after pooling with GAUSS. The feature vectors are not as sparse after pooling. The negative and positive parts tend to look similar for some filters.

We tried three different pooling mechanisms found in the literature:

- Gaussian pooling (GAUSS). This was used in [48, 46]: the $\mathbf{u}^j s$ are first convolved with a Gaussian filter, then downscaled by a factor of 4. We have empirically observed that a 5×5 filter with $\sigma = 2.0$ gives the best results in our case, and we therefore adopt these values.
- 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 4×4 neighborhood. This was used for example in [44, 17].

The feature vectors for a given image after pooling will be denoted as $\mathbf{v}^j$ below. Some examples are shown Fig. 3.4.

3.3 Classification

The last step of our pipeline applies a classifier on the unit-normalized vectors obtained after subspace selection. We kept two different classification methods:

- Nearest Neighbor classification (NN). It provides a direct measure of the discriminative capabilities of the previous steps.
- Support Vector Machines (SVM). They are commonly adopted in pipelines similar to ours and usually achieved the best results. We used the LIBSVM library [1], and we have performed a C-Support Vector Classification (C-SVC) by using the choosing radial basis functions as kernels, and setting the γ parameter to 10. Our best result, however, was obtained by solving an expensive multi-class bound-constrained SVC problem with $\gamma = 8$ using the BSVM software [11].

We tried other classifiers: Feed-Forward Neural Networks, ensembles of Classification Trees, and a Naïve Bayes classifier. As they did not give better results than SVMs, we do not report them here.

In the case of SVMs the feature vectors after pooling $\mathbf{v}^j$ are still too large, and therefore a dimensionality reduction step using PCA is a mandatory choice. In the case of Nearest Neighbor classification, we use either PCA or no dimensionality reduction step (NONE). We also tried using

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm

Figure 3.5: The CIFAR-10 dataset. It is made of 32×32 images of ten object categories (airplane, automobile, bird, ...) (**First row**). For computational reasons, we had to downscale the images to 16×16 and convert them to grayscale images (**Second row**), which makes the recognition even more challenging.

Local Discriminant Embedding (LDE) [12] with a power regularization fixing the signal to noise ratio to 15%. The results are, however, identical to the PCA case. In both situations, a normalization to unit norm is performed after projection, as it is deemed to give significant improvements on the final result [12]. We have also performed trials with Random Projections [2], which are well justified by the sparsity constraints we imposed to our feature vectors for some Feature Extraction options, and pairwise LDE, but no performance improvement has been reported compared to plain PCA. In order to choose the best size of the eigenspace, we perform an extensive cross-validation for all dimensions in a range $d = \{8, \ldots, 64\}$, and choose the value that scores best in a nearest-neighbor classification over a subset of the training set.

Chapter 4

Results and Discussion

4.1 Evaluation Dataset

As mentioned in the introduction, we adopted the CIFAR-10 dataset [20] to evaluate the different architectures and the influence of sparsity. The CIFAR-10 dataset represents a small, labeled subset of a much bigger, 80-million dataset of 32×32 image patches collected from the Web [47]. The different classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Images from the classes satisfying certain requirements are split in a balanced dataset formed by 50000 samples for training and 10000 samples for testing.

Some examples from this dataset are shown in Fig. 3.5. It is a challenging dataset as its exhibits large variability in pose, appearance, scale, and background composition, and some images are affected by severe distortions. Those characteristics make this dataset suitable for our needs, since it avoids the pitfalls involved with the uncontrolled exploitation of natural images and presented in some recent works by Pinto *et al.* [35, 36]. The results reported in this subsection were obtained on grayscale images. Given the large number of different combinations for our pipeline, we applied an initial downsampling step to 16×16 pixels to speed up the experiments. This downsampling seemed to yield a decrease of about 5% of the recognition rate, with negligible variations across different choices for the extraction stage.

As depicted in Fig. 3.5, downsampled grayscale images are extremely challenging even for a human. [47] reports the result of an experiment performed on humans in a 15-class image classification problem, and the recognition rate for 16×16 grayscale images is approximately $52.5 \pm 2.5\%$.

4.2 Evaluation of the Results and Interpretation

Tables 4.1, 4.2, 4.3, 4.5 group the results of the relevant experiments to make them easier to interpret. Each table gives for each pipeline:

- The ℓ_0-norm of the $\mathbf{t}$ vectors, which are the feature vectors after feature extraction. The ℓ_0-norm is the number of non-zero elements, which we normalize by the length of the vector. The smaller the norm, the sparser the vector is.
- The ℓ_0-norm of the $\mathbf{v}$ vectors, which are the feature vectors after pooling.
- The recognition rate, which is the percentage of correctly classified examples from a randomly chosen, balanced test set constituted by 10000 samples. The test set is disjoint from the training set.

4.2.1 Sparsity is not required for recognition

Table 4.1 compares the recognition rates obtained when using the Olshausen and Field's algorithm (SPARSEGD) with the results achieved by a simple convolution with learned filters

Table 4.1: Comparison between sparse feature vectors and feature vectors obtained by convolution

Method	$\|\mathbf{t}\|_0$	$\|\mathbf{v}\|_0$	Rec. rate
CONV-POSNEG-GAUSS-NONE-NN	1.00	0.99	39.94%
CONV-POSNEG-GAUSS-PCA-NN	″	″	44.50%
CONV-POSNEG-GAUSS-PCA-SVM	″	″	66.04%
SPARSEGD-POSNEG-GAUSS-NONE-NN	0.46	0.98	41.72%
SPARSEGD-POSNEG-GAUSS-PCA-NN	″	″	45.91%
SPARSEGD-POSNEG-GAUSS-PCA-SVM	″	″	65.25%

Table 4.2: Comparisons between sparse feature vectors obtained with the Olshausen and Field algorithm (SPARSEGD), and sparse feature vectors obtained with the Matching Pursuit algorithm (SPARSEMP)

Method	$\|\mathbf{t}\|_0$	$\|\mathbf{v}\|_0$	Rec. rate
SPARSEGD-POSNEG-GAUSS-NONE-NN	0.65	0.99	41.46%
SPARSEGD-POSNEG-GAUSS-PCA-NN	″	″	45.11%
SPARSEGD-POSNEG-GAUSS-PCA-SVM	″	″	**65.92%**
SPARSEGD-POSNEG-GAUSS-NONE-NN	0.46	0.98	41.72%
SPARSEGD-POSNEG-GAUSS-PCA-NN	″	″	45.91%
SPARSEGD-POSNEG-GAUSS-PCA-SVM	″	″	65.25%
SPARSEGD-POSNEG-GAUSS-NONE-NN	0.12	0.81	35.30%
SPARSEGD-POSNEG-GAUSS-PCA-NN	″	″	41.81%
SPARSEGD-POSNEG-GAUSS-PCA-SVM	″	″	59.62%
SPARSEMP-POSNEG-GAUSS-NONE-NN	0.04	0.33	17.49%
SPARSEMP-POSNEG-GAUSS-PCA-NN	″	″	28.42%
SPARSEMP-POSNEG-GAUSS-PCA-SVM	″	″	46.26%
SPARSEMP-POSNEG-GAUSS-NONE-NN	0.06	0.38	17.65%
SPARSEMP-POSNEG-GAUSS-PCA-NN	″	″	27.41%
SPARSEMP-POSNEG-GAUSS-PCA-SVM	″	″	45.24%

(CONV). The vectors after feature extraction are much sparser in the case of SPARSEGD. After pooling, however, the two sets of vectors are not sparse anymore. In particular, in all the experiments we have carried out and irrespectively of the chosen feature extraction and pooling strategies, the results after pooling are dense. This suggests that, in all the architectures that employ pooling stages, sparsity is a temporary condition only.

For the same combination of Subspace Selection and Classification, the two feature extraction strategies give the same recognition rates. Enforcing the sparsity clearly does not help here.

To check this surprising results, we tried the Matching Pursuit algorithm to impose the sparsity constraints. Matching Pursuit is slower than gradient descent but produces much sparser vectors for the same reconstruction error. The results are presented Table 4.2. After feature extraction with Matching Pursuit (SPARSEMP) the feature vectors are indeed sparser, and still remain very sparse after pooling. The recognition rate, however, degrades significantly! Our interpretation is that the Matching Pursuit algorithm makes the feature vectors "unstable": the number of features shared by different samples of the same class becomes too small as well, which makes the classification difficult.

A visual inspection of the feature maps seems to suggest that the strongly-responding features in an image are usually related to patch-specific details that correspond to intra-class dissimilarities.

4.2.2 Sparsity is interesting when learning the filters

Although sparsity is not important for the feature extraction stage, and can even hurt the recognition rate when Matching Pursuit is used, it is still important for *learning* the filters. Table 4.3 com-

Table 4.3: Comparisons between different filter banks

Method	$\|\mathbf{t}\|_0$	$\|\mathbf{v}\|_0$	Rec. rate
CONV-POSNEG-GAUSS-NONE-NN	1.00	0.99	39.94%
CONV-POSNEG-GAUSS-PCA-NN	″	″	44.50%
CONV-POSNEG-GAUSS-PCA-SVM	″	″	**66.04%**
CONVLM-POSNEG-GAUSS-NONE-NN	1.00	1.00	38.95%
CONVLM-POSNEG-GAUSS-PCA-NN	″	″	41.00%
CONVLM-POSNEG-GAUSS-PCA-SVM	″	″	63.60%
CONVMR8-POSNEG-GAUSS-NONE-NN	1.00	1.00	38.66%
CONVMR8-POSNEG-GAUSS-PCA-NN	″	″	39.92%
CONVMR8-POSNEG-GAUSS-PCA-SVM	″	″	61.31%
CONVRAND-POSNEG-GAUSS-NONE-NN	1.00	0.99	34.30%
CONVRAND-POSNEG-GAUSS-PCA-NN	″	″	36.29%
CONVRAND-POSNEG-GAUSS-PCA-SVM	″	″	54.36%
GRAD-POSNEG-GAUSS-NONE-NN	0.96	n.a.	37.17%
GRAD-POSNEG-GAUSS-PCA-NN	″	″	37.39%
GRAD-POSNEG-GAUSS-PCA-SVM	″	″	52.31%

Table 4.4: Comparisons of the results obtained by different strategies using randomly generated filters

Method	$\|\mathbf{t}\|_0$	$\|\mathbf{v}\|_0$	Rec. rate
CONVRAND-POSNEG-GAUSS-NONE-NN	1.00	0.99	34.30%
CONVRAND-POSNEG-GAUSS-PCA-NN	″	″	36.29%
CONVRAND-POSNEG-GAUSS-PCA-SVM	″	″	54.36%
GDRAND-POSNEG-GAUSS-NONE-NN	0.81	1.00	34.95%
GDRAND-POSNEG-GAUSS-PCA-NN	″	″	37.30%
GDRAND-POSNEG-GAUSS-PCA-SVM	″	″	56.63%
SPARSEGDRAND-POSNEG-GAUSS-NONE-NN	0.45	0.99	34.95%
SPARSEGDRAND-POSNEG-GAUSS-PCA-NN	″	″	37.33%
SPARSEGDRAND-POSNEG-GAUSS-PCA-SVM	″	″	56.14%

pares the recognition rates for convolution with different filter banks: the filters obtained with the Olshausen and Field algorithm (CONV), the Leung-Malik filter bank (CONVLM), the Maximum Response 8 filter bank (CONVMR8), and a bank made of random filters (CONVRAND). None of these filter banks produced sparse feature vectors, but even CONVRAND obtains decent classification rates already noticed in [17]. Convolution with the learned features (CONV) obtains the best results. Learning filters while imposing sparsity constraints therefore improves the classification results, even when done in an unsupervised way. As a comparison term for the classification rate, we have also performed an experiment passing to the classifier the countours extracted by gradient detectors in the x and y direction (GRAD). It is noticeable that, when Gradient Descent is applied to the feature maps obtained by convolution with random filters in order to improve the reconstruction quality, no improvement is obtained by enforcing sparsity (SPARSEGDRAND) compared with the case when a good reconstruction only is sought (GDRAND) (see Table 4.4).

4.3 Best Results

Table 4.5 compares the recognition rates when varying the pooling method, for the best combination for the other components: convolution with learned filters, PCA projection, and SVM classification. Averaging with a BOXCAR filter slightly outperforms the MAX operation, and using a Gaussian kernel (GAUSS) slightly outperforms the BOXCAR filter. When no non-linearity is introduced, the performance is negatively affected by more than 20%.

Table 4.5 also compares the two rectification methods, ABS and POSNEG, after fixing the other

Table 4.5: Comparisons of the different pooling strategies, and the two different rectification methods

Method	$\|\mathbf{t}\|_0$	$\|\mathbf{v}\|_0$	**Rec. rate**
CONV-POSNEG-GAUSS-PCA-SVM	1.00	1.00	66.04%
CONV-POSNEG-BOXCAR-PCA-SVM	1.00	0.99	63.38%
CONV-POSNEG-MAX-PCA-SVM	1.00	0.99	59.56%
CONV-POSNEG-GAUSS-PCA-SVM	1.00	1.00	66.04%
CONV-ABS-GAUSS-PCA-SVM	1.00	1.00	59.56%
CONV-NONE-GAUSS-PCA-SVM	1.00	1.00	43.96%

components to the best combination: Convolution with learned filters, Gaussian smoothing for pooling, PCA projection, and SVM classification. The POSNEG operation used in [48] outperforms the ABS operation used in [17].

These different evaluations give us the best combination for the full pipeline:
CONV-POSNEG-GAUSS-PCA-SVM yields the highest recognition rates. By running this method on 32×32 grayscale images, we obtained a recognition rate of 71.53%, better than the 64.84% of the state-of-the-art on the original color images of this dataset.

Chapter 5

Conclusion

We have performed an in-depth analysis on the impact of sparsity in image classification. Our experimental results advocate that solely enforcing sparsity is not helpful in terms of recognition rate, while it is important to learn structured filters that yield better recognition performances. Given the high computational burden involved with sparse codes and the increasing interest in biologically inspired multi-layer architectures, this insight heavily impacts on the design strategies for feature extraction algorithms.

As more computational power becomes available, approaches aimed at automatically learn how to face hard problems get an increasing relevance. It is mandatory, however, to devise clever techniques to properly harness these learning systems, in order to fully exploit their capabilities. Although further investigations are required to define a framework in which to cast the image classification task in an effective manner, the evaluation performed in this paper accounts for a set of candidate components of it, assessing their structure and their relative importance in a quantitative way.

The analysis hereby presented has many possible, relevant extensions. In particular, it would be interesting to investigate the behavior of sparsity in multi-layer architectures, and the impact of normalization on the classification performances. The former architecture has, in fact, demonstrated its power in the context of Deep Learning Machines and Convolutional Neural Networks, whereas several recent works reported the effectiveness of subtractive and divisive normalization in several image analysis tasks. Additionally, we argue that a sparse representation should be tuned to provide a rough, class-specific sketch of the considered object, enforcing discriminative traits during the sparsification step. An indepth analysis of discriminative sparsification strategies, combined with sparseness-preserving pooling techniques, is therefore likely to provide powerful, biologically-like classification algorithms.

Bibliography

[1] J.J. Atick. Could information theory provide an ecological theory of sensory processing? *Network*, 3(2):213–251, 1992.

[2] E. Bingham and H. Mannila. Random Projection in Dimensionality Reduction: Applications to Image and Text Data. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 245–250, 2001.

[3] M. Calonder, V. Lepetit, K. Konolige, J. Bowman, P. Mihelich, and P. Fua. Compact Signatures for High-speed Interest Point Description and Matching. In *ICCV*, 2009.

[4] S. Fidler and A. Leonardis. Towards Scalable Representations of Objects Categories. In *CVPR*, 2007.

[5] D.J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America*, 4(12):2379–2394, 1987.

[6] D.J. Field. What Is the Goal of Sensory Coding? *Neural Computation*, 6(4):559–601, 1994.

[7] R.M. Figueras I Ventura and E.P. Simoncelli. Statistically Driven Sparse Image Approximation. In *ICIP*, 2007.

[8] D. Geiger, T.L. Liu, and M.J. Donahue. Sparse Representations for Image Decompositions. *IJCV*, 33(2):139–156, 1999.

[9] J.M. Geusebroek, A.W.M. Smeulders, and J. Van de Weijer. Fast Anisotropic Gauss Filtering. *TIP*, 12(8):938–943, 2003.

[10] G.E. Hinton, S. Osindero, and Y. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18:1391–1415, 2006.

[11] C.W. Hsu and C.J. Lin. A Comparison of Methods for Multiclass Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[12] G. Hua, M. Brown, and S.A.J. Winder. Discriminant Embedding for Local Image Descriptors. In *ICCV*, 2007.

[13] D.H. Hubel. *Eye, Brain, and Vision*. W. H. Freeman, 2nd edition, 1995.

[14] D.H. Hubel and T.N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, 1959.

[15] M. Hyder and K. Mahata. An Approximate L0 Norm Minimization Algorithm for Compressed Sensing. In *ICASSP*, pages 3365–3368, 2009.

[16] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Inc., 2001.

[17] K. Jarrett, K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. What is the Best Multi-Stage Architecture for Object Recognition? In *CVPR*, 2009.

[18] K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition. Technical report, Department of Computer Science-Courant Institute of Mathematical Sciences, New York University, 2008.

[19] B. Krishnapuram, L. Carin, M.A.T. Figueiredo, and A.J. Hartemink. Sparse Multinomial Logistic Regression : Fast Algorithms and Generalization Bounds. *PAMI*, 27(6):957–968, 2005.

[20] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Msc. thesis, University of Toronto, 2009.

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

[22] H. Lee, C. Ekanadham, and A.Y. Ng. Sparse deep belief net model for visual area V2. In *NIPS*, 2008.

[23] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *ICML*, 2009.

[24] T. Leung and J. Malik. Recognizing the Visual Appearance of Materials using Three-Dimensional Textons. *IJCV*, 43(1):29–44, 2001.

[25] H. Liu, Y. Agam, J.R. Madsen, and G. Kreiman. Timing, Timing, Timing: Fast Decoding of Object Information from Intracranial Field Potentials in Human Visual Cortex. *Neuron*, 62 (2):281–290, 2009.

[26] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 20(2):91–110, 2004.

[27] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative Learned Dictionaries for Local Image Analysis. In *CVPR*, 2008.

[28] J. Mairal, M. Elad, and G. Sapiro. Sparse Representation for Color Image Restoration. *TIP*, 17(1):53–69, January 2008.

[29] S.G. Mallat and Z. Zhang. Matching Pursuits with Time-Frequency Dictionaries. *SP*, 41(12): 3397–3415, 1993.

[30] J. Mutch and D.G. Lowe. Multiclass Object Recognition with Sparse, Localized Features. In *CVPR*, 2006.

[31] B.A. Olshausen. *Sparse codes and spikes*, chapter 13, pages 257–272. MIT Press, 2002.

[32] B.A. Olshausen and D.J. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[33] B.A. Olshausen and D.J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, 2004.

[34] B.A. Olshausen and K.J. Millman. Learning sparse codes with a mixture-of-Gaussians prior. In *NIPS*, volume 12, pages 841–847, 2000.

[35] N. Pinto, D.D. Cox, and J.J. DiCarlo. Why is Real-World Visual Object Recognition Hard? *PLoS computational biology*, 4(1):151–156, 2008.

[36] N. Pinto, J.J. DiCarlo, and D.D. Cox. How Far Can You Get with a Modern Face Recognition Test Set using Only Simple Features? In *CVPR*, 2009.

[37] R. Raina, A. Battle, H. Lee, B. Packer, and A.Y. Ng. Self-taught Learning: Transfer Learning from Unlabeled Data. In *ICML*, 2007.

[38] M.A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient Learning of Sparse Representations with an Energy-Based Model. *NIPS*, 19:1137–1144, 2006.

[39] M.A. Ranzato, Y.L. Boureau, and Y. LeCun. Sparse Feature Learning for Deep Belief Networks. *NIPS*, 20:1185–1192, 2007.

[40] M.A. Ranzato, F.J. Huang, Y.L. Boureau, and Y. Lecun. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. *CVPR*, 2007.

[41] F. Rodriguez and G. Sapiro. Sparse Representations For Image Classification: Learning Discriminative and Reconstructive Non-Parametric Dictionaries. Technical report, Institute for Mathematics and Its Applications, University of Minnesota, 2008.

[42] T. Serre, L. Wolf, and T. Poggio. Object Recognition with Features Inspired by Visual Cortex. In *CVPR*, 2005.

[43] T. Serre, A. Oliva, and T. Poggio. A Feedforward Architecture Accounts for Rapid Categorization. *Proceedings of the National Academy of Sciences of the United States of America*, 104(15): 6424–6429, 2007.

[44] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust Object Recognition with Cortex-Like Mechanisms. *PAMI*, 29(3):411–426, 2007.

[45] E.P. Simoncelli and B.A. Olshausen. Natural Image Statistics and Neural Representation. *Annual Review Neuroscience*, 24:1193–1216, 2001.

[46] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *PAMI*, 2009.

[47] A. Torralba, R. Fergus, and W.T. Freeman. 80 Million Tiny Images: a Large Data Set for Nonparametric Object and Scene Recognition. *PAMI*, 30(11):1958–1970, 2008.

[48] S.A.J. Winder and M. Brown. Learning Local Image Descriptors. In *CVPR*, 2007.

[49] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma. Robust Face Recognition via Sparse Representation. *PAMI*, 31(2):210–227, 2009.

Appendix A

Filter bank derivation

Given the filter bank $\mathbf{M}$ and an image patch $\mathbf{x}$, we can derive the initial feature map $\mathbf{t}$ by computing the cross-correlation between the patch and the filters,

$$\mathbf{t} = \mathbf{x} \star \mathbf{M} \, . \tag{A.1}$$

The reconstruction $\mathbf{r}$ can be computed as

$$\mathbf{r} = \sum_i \left(\mathbf{t}_i \star \mathbf{m}_i \right) \, , \tag{A.2}$$

where $\mathbf{m}_i \in \mathbf{M}$ is the i-th filter and $\mathbf{t}_i$ is the corresponding feature map. The gradient $\mathbf{g}_i$, needed to perform a stochastic gradient descent update over each $\mathbf{t}_i$ in order to minimize the reconstruction error, can be obtained by the convolution of each filter with $\mathbf{r}$

$$\mathbf{g}_i = \mathbf{m}_i * \mathbf{r} \, . \tag{A.3}$$

To enforce the sparsity of the solution, we periodically update the dictionary while keeping the filter maps fixed. After having computed the reconstructed image $\mathbf{r}$ with Equation (A.2), we estimate the error γ_i for the i-th filter as

$$\gamma_i = \mathbf{t}_i \star \mathbf{r} \, , \tag{A.4}$$

and then alter the filters proportionally to this quantity.

To improve the convergence of the stochastic gradient descent learning, we empirically observed that the regularization parameter λ can be relaxed in the first iterations (low penalty for non-sparse solutions), while it can be increased when the reconstruction provided is sufficiently accurate.