# Descriptor extraction based on a multilayer dictionary architecture for classification of natural images

Stefen Chan Wai Tim, Michèle Rombaut, Anuvabh Dutt, Denis Pellerin

# Descriptor extraction based on a multilayer dictionary architecture for classification of natural images.

Stefen Chan Wai Tim[a], Michele Rombaut[a,**], Denis Pellerin[a], Anuvabh Dutt[a,b]

[a] *Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-Lab, F-38000 Grenoble France*
[b] *Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble France*

ABSTRACT

This paper presents a descriptor extraction method in the context of image classification, based on a multilayer structure of dictionaries. We propose to learn an architecture of discriminative dictionaries for classification in a supervised framework using a patch-level approach. This method combines many layers of sparse coding and pooling in order to reduce the dimension of the problem. The supervised learning of dictionary atoms allows them to be specialized for a classification task. The method has been tested on known datasets of natural images such as MNIST, CIFAR-10 and STL, in various conditions, especially when the size of the training set is limited, and in a transfer learning application. The results are also compared with those obtained with Convolutional Neural Network (CNN) of similar complexity in terms of number of layers and processing pipeline.

## 1. Introduction

Dictionary learning and sparse representations have received a lot of focus in recent years because they have led to state-of-the-art results in many applications, in particular in image processing. One reason for their success is that they can efficiently learn the underlying patterns in the image, leading to good performances for example in denoising (Elad and Aharon (2006); Mairal et al. (2008)) or inpainting (Aharon et al. (2006)). The sparse codes obtained can also be seen as a new representation of the input or as features in image classification tasks (Mairal et al. (2012); Raina et al. (2008); Bradley and Bagnell (2008); Wright et al. (2009); Chan Wai Tim et al. (2015)). In such cases, the dictionary is often learned in an unsupervised way, meaning that the dictionary is learned without taking into account the class of the images. Then, the sparse codes obtained can either be used directly for classification (Qiu et al. (2011)), or as features fed to a classifier i.e SVM (Chan Wai Tim et al. (2015)).

Recent research has emphasized the advantages of learning **discriminative** sparse models (Mairal et al. (2012); Bradley and Bagnell (2008); Yang et al. (2010)) instead of purely **reconstructive** ones. It is usually done by learning the sparse representation and the classifier conjointly. In practice, each input image is matched with a label and the dictionary is learned in a supervised setup.

Generally, in image processing applications, dictionary learning and sparse coding are computed on a small portion of an image (i.e image patches) because learning a dictionary directly on high resolution images is computationally intensive. There is no particular problem in doing so in denoising, however, in the case of classification, a means to fuse efficiently the representation of the patches into an image-level descriptor is needed such as pooling (Yang et al. (2010)) or Bag of words (Wang et al. (2009)).

*Method framework*

In the proposed multilayer architecture, the sparse codes obtained by encoding signals on a dictionary are used as inputs to a subsequent coding layer. Each additional layer of dictionary encoding changes the representation by projecting the features into a new space. The prospective objective is to increase the discriminability of the features by building a hierarchy of dictionaries.

In Section 2, we recall the dictionary learning framework going from the unsupervised setup to the supervised dictionary learning setup, and we detail the patch decomposition of the input data. In Section 3, we introduce our multilayer dictionary

**Corresponding author:
*e-mail:* Michele.Rombaut@gipsa-lab.grenoble-inp.fr (Michele Rombaut)

learning setup. In Section 4, we present the multilayer learning process and finally, in section 5, we present the experiments and their results, especially when the training set is small and in the context of transfer learning when the architecture is trained on one dataset and applied to a different dataset.

## 2. Dictionary learning

In this section, we recall various formulations of the dictionary learning problem, starting with an unsupervised method more suited to reconstruction of an image and followed by the supervised method tailored around a specific task such as classification.

### *2.1. Unsupervised dictionary learning*

Unsupervised dictionary learning has been widely used in reconstruction tasks. In its classical formulation, the goal is to learn a set of atoms directly from data in order to optimise the reconstruction of the image. Let's consider a set of $N$ signals $\mathbf{Y} = [\mathbf{y}_1, ..\mathbf{y}_N]$ where each signal $\mathbf{y}_k$ is extracted from an image $I_k$. In this case, column vector $\mathbf{y}_k$ of the image $k$ can be the gray level of the $m$ pixels of the image. Dictionary $\mathbf{D}$ is represented by the matrix $\mathbf{D} = (d_{ij})$ with $i \in [1, m]$, $j \in [1, K]$ where $K$ is the number of dictionary atoms $\mathbf{d}_j$. The atom $\mathbf{d}_j$ has the same dimension $m$ as the dimension of each vector $\mathbf{y}_k$.

Dictionary $\mathbf{D}$ can be learned by solving:

$$\min_{\mathbf{D},\mathbf{x}_k} \quad \sum_{k=1}^{N} \|\mathbf{y}_k - \mathbf{D}\mathbf{x}_k\|_2^2 + \Phi(\mathbf{x}_k) \tag{1}$$

with $\mathbf{x}_k$ a sparse vector of dimension $K$ containing the coefficients to reconstruct $\mathbf{y}_k$. We denote $\|\cdot\|_2$ and $\|\cdot\|_1$ the $\ell_2$-norm and $\ell_1$-norm respectively. In this formulation, the reconstruction error is minimized and the sparsity can be controlled with the function $\Phi(\mathbf{x})$ which can be:

- for the LASSO approach as in Tibshirani (2013): $\Phi(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$
- for the Elastic Net approach as in Zhou and Hastie (2005):

$$\Phi(\mathbf{x}) = \lambda_1\|\mathbf{x}\|_1 + \frac{\lambda_2}{2}\|\mathbf{x}\|_2^2 \tag{2}$$

A higher $\lambda$ (or $\lambda_1$) increases sparsity. Once a sparse code $\mathbf{x}_k$ is obtained, the original signal can be approximated by computing $\hat{\mathbf{y}}_k \approx \mathbf{D}\mathbf{x}_k$.

This problem has been widely studied and many approaches exist in order to obtain both dictionary $\mathbf{D}$ and coefficients $\mathbf{x}_k$ (Aharon et al. (2006); Olshausen and Field (1996); Tibshirani (1996)). Using such an unsupervised approach can yield good results in image reconstruction problems. Code $\mathbf{x}_k$ can be seen as a sparse representation of $\mathbf{y}_k$. Since it can find the underlying patterns in the data, some authors have proposed using these codes in classification tasks (Raina et al. (2008); Wright et al. (2009)).

### *2.2. Supervised dictionary learning*

The idea is to optimize the dictionary learning for reconstruction and also for classification. Supervised dictionary learning methods began to be investigated by Mairal et al. (2012); Yang et al. (2010) in order to take advantage of parsimony in classification tasks. In this case, the learning process requires a training set made up of labeled elements $\mathbf{y}_k$. Encoding a datum using a dictionary can be seen as a projection into another coordinate system. The objective is to obtain projected features that are discriminative in the new space.

Let's assume we know a training set $\mathbf{E}$ of pairs $(\mathbf{y}_k, l_k)$ where $\mathbf{y}_k$, $k \in [1, N]$ is a set of signals representing the image and $l_k$ is the corresponding label of the image attached to the vector $\mathbf{y}_k$. We define $\mathcal{L}$, a differentiable classification loss function and $\mathbf{W}$, its set of parameters.

The supervised dictionary learning problem can be written with the two following equations:

$$\hat{\mathbf{x}}_k = \underset{\mathbf{x}}{\operatorname{argmin}} \quad \|\mathbf{y}_k - \mathbf{D}\mathbf{x}_k\|_2^2 + \Phi(\mathbf{x}_k) \tag{3}$$

$$\min_{\mathbf{W},\mathbf{D}} \quad \sum_{k=1}^{N} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) \tag{4}$$

Eq.3 gives the optimal sparse code $\hat{\mathbf{x}}_k$ for the reconstruction of signal $\mathbf{y}_k$ with the learned dictionary $\mathbf{D}$. The Eq.4 has the objective of minimizing the suitable cost function $\mathcal{L}$ with respect to its parameters $\mathbf{W}$ and a dictionary $\mathbf{D}$ as represented in Fig. 1. This cost function depends on the sparse codes $\hat{\mathbf{x}}_k$ used for classification, the associated labels $l_k$, and the parameters $\mathbf{W}$ of the classifier. The main difference between this formulation (Eq.3 and Eq.4) and the previous one (Eq.1) is that the goal now is to minimize the classification loss instead of a reconstruction error term.

To minimize the cost function $\mathcal{L}$ with respect to the parameters of the dictionary $\mathbf{D}$ and the parameters $\mathbf{W}$ of the cost function, it is possible to use a method similar to the backpropagation algorithm used in neural networks (LeCun et al. (1998)). The minimization can be done using gradient descent on $\mathbf{W}$ with step $\eta_1$ and $\mathbf{D}$ with step $\eta_2$.

Computing the gradient of $\mathcal{L}$ with respect to parameters $\mathbf{W}$ is usually classical and simple. The main difficulty when solving Eq.4 is the minimization of the cost function $\mathcal{L}$ with respect to dictionary $\mathbf{D}$ because it does not appear explicitly and involves the other optimization problem of solving Eq.3 for $\hat{\mathbf{x}}_k$. To overcome this problem, a way to compute the gradient of the cost function $\mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$ with respect to the dictionary $\mathbf{D}$ is needed. This problem has been tackled in Mairal et al. (2012); Bradley and Bagnell (2008).

In this work, we follow the ideas of Mairal et al. (2012) which show the differentiability of $\mathcal{L}$ and give the steps to compute its gradient with respect to parameters $\mathbf{W}$ and dictionary $\mathbf{D}$. According to the paper, the desired gradient can be computed as follows:

$$\nabla_{\mathbf{D}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) = \mathbf{D}\beta\hat{\mathbf{x}}_k^\top + (\mathbf{y} - \mathbf{D}\hat{\mathbf{x}}_k)\beta^\top \tag{5}$$

We define the set $\Lambda = \{i | x_i \neq 0\}$ made up of non-zero coefficients of the considered code $\mathbf{x}_k$ and $\hat{\mathbf{x}}_{k\Lambda}$ corresponds to $\hat{\mathbf{x}}_k$ restricted to its non-zero coefficients.

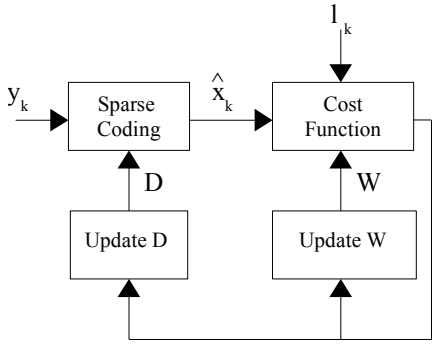


**Fig. 1. A signal $\mathbf{y}_k$ associated to a label $l_k$ is encoded by a dictionary D, the resulting code $\hat{\mathbf{x}}_k$ is used as an input to a cost function with parameters W. The cost function $\mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$ is computed. Then, the dictionary D and the parameters W are updated to fit the classification problem.**

Vector $\beta \in \mathcal{R}^N$ is computed as $\beta_j = 0$, if $j \notin \Lambda$ and $\beta_\Lambda$, which is vector $\beta$ restricted to the indices in $\Lambda$, is defined as follows:

$$\beta_\Lambda = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1} \nabla_{\hat{\mathbf{x}}_{k\Lambda}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) \tag{6}$$

where $\mathbf{D}_\Lambda$ is the dictionary restricted to the set $\Lambda$.

Now, it is necessary to compute the gradient $\nabla_{\hat{\mathbf{x}}_{k\Lambda}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$ (with respect to $\hat{\mathbf{x}}$ which explicitly appears in $\mathcal{L}$, instead of $\mathbf{D}$).

The optimisation process consists of:

1. Inputs: set $\mathbf{E}$ of pairs $(\mathbf{y}_k, l_k)$ where:
   - $\mathbf{y}_k$, $k \in [1, N]$ is a set of signals (i.e images represented as column vectors)
   - $l_k$, $k \in [1, N]$ is the corresponding label for $\mathbf{y}_k$
2. Initialization: compute an initial $\mathbf{D}$ for reconstruction (Eq.1)
3. While *Convergence not reached*
   3..1 Draw a signal randomly $\mathbf{y}_k \in \mathbf{E}$
   3..2 Compute $\hat{\mathbf{x}}_k$, solution of (Eq.3) with the dictionary $\mathbf{D}$
   3..3 Compute
   $$\nabla_{\mathbf{W}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$$
   $$\nabla_{\hat{\mathbf{x}}_{k\Lambda}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$$
   $$\beta_\Lambda$$
   $$\nabla_{\mathbf{D}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$$
   3..4 Update $\mathbf{D}$ and $\mathbf{W}$ by gradient descent:
   $$\mathbf{D} \leftarrow \mathbf{D} - \eta_1 \nabla_{\mathbf{D}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$$
   $$\mathbf{W} \leftarrow \mathbf{W} - \eta_2 \nabla_{\mathbf{W}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$$
4. Return $\mathbf{D}$ and $\mathbf{W}$

In this work, we have chosen to use the cross-entropy function (Eq.7) for the classification loss as it has proven to give good results in multiclass classification problems. The chosen classifier is a linear classifier coupled with Softmax for the output. If we consider a classification problem with $C$ classes, the cross-entropy loss defined for the input vector $\mathbf{y}_k$ coded by the vector $\hat{\mathbf{x}}_k$ of size $K$ is computed as follows:

$$\mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) = -\sum_{i=1}^{C} l_{ik} log(p_{ik}) + \lambda' ||\mathbf{W}||_2^2 \tag{7}$$

where $\lambda'$ is a parameter that penalizes the vector $\mathbf{W}$ and $l_{ik} = 1$ if the image is associated with the class $i$ and 0 elsewhere. $p_{ik}$ is defined by:

$$p_{ik} = \frac{\exp(\hat{\mathbf{x}}_k^\top \mathbf{w}_i)}{\sum_{j=1}^{C} \exp(\hat{\mathbf{x}}_k^\top \mathbf{w}_j)} \tag{8}$$

where $\mathbf{w}_i$ is a column vector of size $K$ corresponding to the class $i$ of the matrix $\mathbf{W}$.

### *2.3. Patch decomposition*

A supervised dictionary learning approach can successfully learn patterns for an image classification task. Usually, dictionary learning approaches are the most effective when the input images are relatively small and the object of interest homogeneously localized. However, in practice, some limitations can be observed: on the one hand, the larger the size of the image, the further the disparity between images of the same class increases and then the dictionary used would need a huge number of atoms in order to obtain an efficient sparse decomposition. On the other hand, the computational cost for encoding operations (size and number of atoms) and learning will also increase.

Dictionary learning methods have been used for reconstructing or classifying either full image or image patches. It means that in practice, a signal $\mathbf{y}_k$ can be either from an image or from a patch reshaped as a column vector containing the pixel values.

We propose to decompose the image $I$ into a set of patches to obtain a local representation of the image. Image decomposition is often performed using overlapping patches to obtain a more accurate local description. The considered signals $\mathbf{y}_k$ are extracted from patches of a priori defined size, presented in the form of vectors containing the values of the pixels (the representation may be in colors or in gray levels). This type of approach is used, for example, in denoising.

Moreover, classifying a set of patches extracted from an image instead of the image itself is different, from the dictionary methods point of view. Indeed, when dealing with patches, a method to fuse the information extracted from the set of patches composing an image is needed. This particular problem has been studied by Yang et al. (2010) in which he chose to perform a single sparse coding step at patch level (with a patch size smaller than the input image) followed by numerous pooling steps in order to efficiently reduce the dimensions and to obtain a multi-scale representation.

## 3. Multilayer dictionary structure

Intuitively, sparse coding can extract important characteristics for reconstruction in unsupervised frameworks and for classification in supervised methods such as presented in Mairal et al. (2012); Bradley and Bagnell (2008). We could use the extracted codes directly with a classifier but, in order to achieve good performances, we would need large atoms (about the size of the input images) and a huge number of them to cope with the image diversity. In practice, this approach is restricted by the computation time.

In this paper, we propose a multilayer architecture. The method is inspired by convolutional networks (LeCun et al.

| | | | | |
|---|---|---|---|---|
| $\mathbf{X}'^{(q-1)}$ | $\xrightarrow{\boxed{1}}$ | $\mathbf{Y}^{(q)} = \{\mathbf{y}_k^{(q)}\}$ | $\xrightarrow{\boxed{2}}$ | $\mathbf{x}_k^{(q)}$ |
| $\mathbf{x}_k^{(q)}$ | $\xrightarrow{\boxed{3\text{-}4}}$ | $\mathbf{x}'^{(q)}_{k'}$ | $\xrightarrow{\boxed{5}}$ | $\mathbf{X}'^{(q)}$ |

**Table 1. Succession of operations at the q-th layer: Patch decomposition 1, Encoding 2, *ReLU* and *Average-Pooling* 3-4, Concatenation 5**

(1998)): the convolution by a filter is replaced by a sparse coding step. The idea is to reiterate the sparse coding layer in order to increase the discriminability of the features.

The other goal of the proposed method is to control the dimension of the input patches by reducing the sparse coding of a large image, computationally intensive, to the sparse coding of small patches which can be processed more efficiently.

### *3.1. Definition of the multilayer structure*

Encoding a vector on a dictionary is similar to projecting into a new space. The projection is non-linear and the resulting vector is sparse. The vector is then used as new input for the following layer. So, adding another dictionary encoding step is akin to doing another projection in a new coordinate system. The process can be repeated many times, with as many dictionaries as the number of layers in the architecture.

The input of a layer $q \in \{1, ..., Q\}$ is the 2D (or 3D) image $\mathbf{X}'^{(0)}$ for the first layer and 3D "image" $\mathbf{X}'^{(q-1)}$ for the following layers.

We develop below the different processing steps for the layer $q$ as represented in Table 1 and Fig. 2:

1. The input 3D image $\mathbf{X}'^{(q-1)}$ is broken down into a set of paches $\mathbf{Y}^{(q)} = \{\mathbf{y}_k^{(q)}\}$ of size $n \times n \times K^{(q-1)}$ which are overlapping with $stride = 1$.
2. Each patch $\mathbf{y}_k^{(q)}$ gives the code $\mathbf{x}_k^{(q)}$ of size $1 \times 1 \times K^{(q)}$ when encoded with the q-th dictionary $\mathbf{D}^{(q)}$ of $K^{(q)}$ atoms.
3. The $ReLU(x) = max(0, x)$ function is applied to each code (see Yang et al. (2010)). This function eliminates the negative coefficients of a code vector and acts as a step which introduces non-linearity.
4. The *Average-Pooling* function is also used to reduce size. To do that, the $3D$ image from $\mathbf{X}^{(q)} = \{\mathbf{x}_k^{(q)}\}$ is broken down into $m \times m \times K^{(q)}$ small images. Each small image is reduced to a vector $\mathbf{x}'^{(q)}_{k'}$ of size $1 \times 1 \times K^{(q)}$ whose values are the average of the $m \times m$ values associated with the component pixels $\mathbf{x}_k^{(q)}$ of the small image.
5. Since the spatial localization of the patches has been retained, the set of sparse codes $\mathbf{x}'^{(q)}_{k'}$ can be concatenated as a 3D volume $\mathbf{X}'^{(q)}$ with a depth equal to the number $K^{(q)}$ of atoms in dictionary $\mathbf{D}^{(q)}$.
6. This 3D output volume $\mathbf{X}'^{(q)}$ is treated as a 3D image input where each pixel $\mathbf{x}'^{(q)}$ is a vector of dimension $K^{(q)}$.

For the first layer, image $\mathbf{X}'^{(0)}$ is the real 2D (or 3D) image $I$. For the last layer $Q$, input image $\mathbf{X}'^{(Q-1)}$ is broken down into patches $\mathbf{Y}^{(q)}$ which are encoded by the last dictionary $\mathbf{D}^{(Q)}$ giving the codes $\mathbf{X}^{(Q)}$. Only the last codes $\mathbf{X}^{(Q)}$ are used in the classifier (Fig. 2).

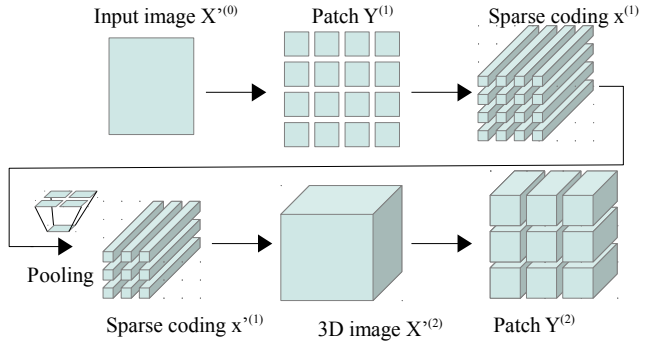


**Fig. 2. Example of an architecture with 2 layers with *ReLU* and *Pooling* operations. An input image $I = X'^{(0)}$ in presented to the first layer. The image is decomposed into patches $\mathbf{y}_k^{(1)}$ which are encoded by dictionary $\mathbf{D}^{(1)}$. The codes $\mathbf{x}_k^{(1)}$ are processed into $\mathbf{x}'^{(1)}$ and restructured into a 3D volume $\mathbf{X}'^{(2)}$ and then decomposed again into 3D patches $\mathbf{y}_k^{(2)}$ in the second layer. These patches are encoded using dictionary $\mathbf{D}^{(2)}$.**

Related works on multilayer dictionary learning have been done by Tariyal et al. (2016). The approach is quite similar with a faster but unsupervised learning process. The idea is to decompose dictionary $\mathbf{D}$ in sub-dictionaries $\mathbf{D}^{(q)}$ as represented in Eq.9. Then the problem is to solve the system represented for each layer $q$. Eq.10 gives the resolution for the first layer $q = 1$. The last step is represented in Eq.11. This approach is a simplified version of our proposal without using the labels of the training set and without the patch-level processing.

$$\min_{\mathbf{D}^{(1)},\cdots,\mathbf{D}^{(Q)},\mathbf{x}_k} \quad \sum_{k=1}^{m} \|\mathbf{y}_k - \mathbf{D}^{(1)} \cdots \mathbf{D}^{(Q)}\mathbf{x}_k\|_2^2 + \phi(\mathbf{x}_k) \tag{9}$$

$$\min_{\mathbf{D}^{(1)},\mathbf{z}_{1k}} \quad \sum_{k=q}^{m} \|\mathbf{y}_k - \mathbf{D}^{(1)}\mathbf{z}_{1k}\|_2^2, \qquad \mathbf{z}_{1k} = \mathbf{D}^{(2)} \cdots \mathbf{D}^{(Q)}\mathbf{x}_k \tag{10}$$

$$\min_{\mathbf{D}^{(Q)},\mathbf{x}_k} \quad \sum_{k=1}^{m} \|\mathbf{z}_{(Q-1)k} - \mathbf{D}^{(Q)}\mathbf{x}_k\|_2^2 + \phi(\mathbf{x}_k) \tag{11}$$

## 4. Dictionary learning step

As the label of image $I$ is $l$, the patches of the image are associated to the same label $l$. In this section, to simplify the notations, each subscript $k$ used for denoting the index of patches or codes is tied to a specific layer $q$ (it can be read as $k_q$). The upper index $(q)$ denotes the $q$-th layer.

In order to optimize the multilayer architecture for classification, it is necessary to find the optimal dictionaries at each layer and the parameters of the classifier. Each dictionary is learned for classification in a supervised manner.

To optimize the classification cost function with respect to the dictionaries at each layer, we will use the back-propagation algorithm. Therefore, we need to compute the gradients with

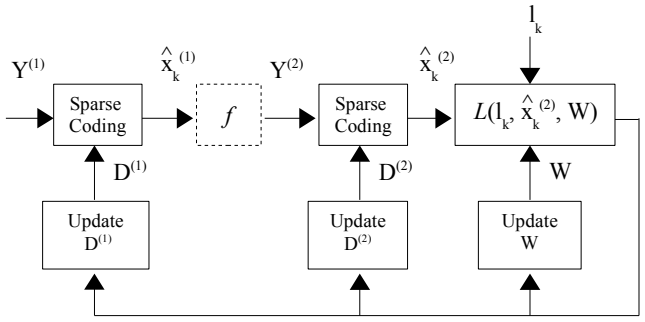


**Fig. 3. Structure with 2 layers. Input vectors $\mathbf{Y}^{(1)}$ go through a sparse coding step. We have $\mathbf{Y}^{(2)} = f(\hat{\mathbf{X}}^{(1)})$ as input to the second layer. $l_k$ is the label of the image and the function $f$ models pulling and concatenation operations. Back-propagation is then used to update both dictionaries $D^{(1)}$, $D^{(2)}$ and the parameters $W$ simultaneously.**

respect to the various dictionaries $\mathbf{D}^{(1)}, \cdots, \mathbf{D}^{(Q)}$ as well as $\mathbf{W}$ the parameters of the classifier.

For $q \in \{1, ..., Q-1\}$ and for the fixed dictionaries $\mathbf{D}^{(q)}$ (same as in (Eq.3))

$$\hat{\mathbf{x}}_k^{(q)} = \underset{\mathbf{x}}{\operatorname{argmin}} \quad \|\mathbf{y}_k^{(q)} - \mathbf{D}^{(q)}\mathbf{x}_k\|_2^2 + \Phi(\mathbf{x}_k) \tag{12}$$

The codes $\hat{\mathbf{x}}_k^{(q)}$ of all layers $q$ could be used by the classifier. However, we decided only to use the output $\hat{\mathbf{x}}_k^{(Q)}$ of the last layer as features for the classification (Fig. 3) in order to limit the size of the descriptors. The cost function is minimized over the entire training set of $N$ images.

To use the back-propagation algorithm (i.e computing each gradient using the chain rule, the gradient is computed the same way as presented in Section 2.2 by replacing $\hat{\mathbf{x}}_k$ by $\hat{\mathbf{x}}_k^{(q)}$ and $\mathbf{y}_k$ by $\mathbf{y}_k^{(q)}$.

By extending the formulation given in Eq.4, we obtain:

$$\min_{\mathbf{W},\mathbf{D}^{(1)},\cdots,\mathbf{D}^{(Q)}} \quad \sum_{k=1}^{N} \mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) \tag{13}$$

where $\hat{\mathbf{x}}_k^{(Q)}$ is computed by Eq.12.

During this learning step, the cost function is modified to penalize high values of $W$ as in Eq.7:

$$\mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) = -\sum_{i=1}^{C} l_{ik} log(p_{ik}) + \lambda' \|W\|_2^2 \tag{14}$$

where $p_{ik}$ is defined Eq.8.

### *4.1. Computation of the gradients at last layer Q*

At first, for an image-label pair $(\mathbf{I}, l)$, the gradient $\nabla_{\mathbf{W}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W})$ of $\mathcal{L}$ with respect to $\mathbf{W}$ is computed.

Then, the gradient of $\mathcal{L}$ with respect to last layer dictionary $\mathbf{D}^{(Q)}$ is computed using Eq.5 and Eq.6. If we introduce the notation using the layer number, the equation for the last layer $Q$ becomes:

$$\nabla_{\mathbf{D}^{(Q)}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) = \mathbf{D}^{(Q)}\beta\hat{\mathbf{x}}_k^{(Q)\top} + (\mathbf{y}_k^{(Q)} - \mathbf{D}^{(Q)}\hat{\mathbf{x}}_k^{(Q)})\beta^\top \tag{15}$$

For the indexes contained in the set $\Lambda$, $\beta$ is defined as:

$$\beta_\Lambda = (\mathbf{D}_\Lambda^{(Q)\top}\mathbf{D}_\Lambda^{(Q)})^{-1}\nabla_{\hat{\mathbf{x}}_{k\Lambda}^{(Q)}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) \tag{16}$$

and $\beta_j = 0$, if $j \notin \Lambda$. By choice, the output of the last layer is a single code vector meaning that $k = 1$ for this Q-th layer.

### *4.2. Computation of the gradients at layer q*

We underline that only the last layer $Q$ is used in the classification step. To compute the gradient of the cost function $\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W})$ with respect to dictionary $\mathbf{D}^{(q)}$ of the $q$-th layer using Eq.5 and Eq.6, we need to compute $\nabla_{\hat{\mathbf{x}}_{k\Lambda}^{(q)}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W})$. That is why the gradient of layer $q$ becomes:

$$\nabla_{\mathbf{D}^{(q)}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) = \mathbf{D}^{(q)}\beta\hat{\mathbf{x}}_k^{(q)\top} + (\mathbf{y}_k^{(q)} - \mathbf{D}^{(q)}\hat{\mathbf{x}}_k^{(q)})\beta^\top \tag{17}$$

$$\beta_\Lambda = (\mathbf{D}_\Lambda^{(q)\top}\mathbf{D}_\Lambda^{(q)})^{-1}\nabla_{\hat{\mathbf{x}}_{k\Lambda}^{(q)}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) \tag{18}$$

In Eq.18, only the term $\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W})$ is used and not $\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(q)}, \mathbf{W})$ because only $\hat{\mathbf{x}}_k^{(Q)}$ is used for classification. The computation of this gradient can be broken down in three terms:

$$\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{x}}^{(q)}} = \frac{\partial\mathcal{L}}{\partial\hat{\mathbf{x}}^{(q+1)}} \cdot \frac{\partial\hat{\mathbf{x}}^{(q+1)}}{\partial\mathbf{y}^{(q+1)}} \cdot \frac{\partial\mathbf{y}^{(q+1)}}{\partial\hat{\mathbf{x}}^{(q)}} \tag{19}$$

The computation of the two first terms is quite simple:

- the term $\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{x}}^{(q+1)}}$ is previously computed at layer $q+1$.
- the term $\frac{\partial\hat{\mathbf{x}}^{(q)}}{\partial\mathbf{y}^{(q)}}$ is given by derivation of Eq.12 and 0 elsewhere.

$$\frac{\partial\hat{\mathbf{x}}_\Lambda^{(q)}}{\partial\mathbf{y}^{(q)}} = (\mathbf{D}_\Lambda^{(q)\top}\mathbf{D}_\Lambda^{(q)})^{-1}\mathbf{D}_\Lambda^{(q)} \tag{20}$$

From Eq.19 and Eq.20, it gives the term:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{y}^{(q+1)}} = \frac{\partial\mathcal{L}}{\partial\hat{\mathbf{x}}^{(q+1)}} \cdot \frac{\partial\hat{\mathbf{x}}^{(q+1)}}{\partial\mathbf{y}^{(q+1)}} \tag{21}$$

The computation of the third term $\frac{\partial\mathbf{y}^{(q+1)}}{\partial\hat{\mathbf{x}}^{(q)}}$ is more complex because $\mathbf{y}^{(q+1)}$ is computed from $\hat{\mathbf{x}}_k^{(q)}$ by the sequence of functions *ReLU* and *Average-pooling* giving $\mathbf{x'}_k^{(q)}$, concatenated into $\mathbf{X'}^{(q)}$ and broken down into patches $\mathbf{y}^{(q+1)}$. So the back-propagation includes an image "reconstruction" step to reverse the patch decomposition such as represented in Table 2.

Each patch $\mathbf{y}^{(q+1)}$ comes from the decomposition in patches covering the 3D image $\mathbf{X}^{(q)}$. We have $\frac{\partial\mathcal{L}}{\partial\mathbf{y}^{(q+1)}}$ with the Eq.21: these errors must be transferred to the "pixel" of $\mathbf{X'}^{(q)}$ formed from the patch overlay $\mathbf{y}^{(q+1)}$. We have chosen to add up the errors on the pixels of the same overlap 3.

The concatenation operation 2 can be treated in the same way, but this time without overlapping: each pixel keeps the associated error.

For the *Average − Pooling* 1 function, we just distribute the error of the pixel on the pixels that have been "pooled". For example, if each pixel $\mathbf{x'}^{(q)}$ of $\mathbf{X'}^{(q)}$ comes from four pixels $\hat{\mathbf{x}}^{(q)}$ of $\hat{\mathbf{X}}^{(q)}$, the error associated with $\hat{\mathbf{x}}^{(q)}$ corresponds to a quarter of the error associated with $\mathbf{x'}^{(q)}$.

Finally, for the *ReLU* function, the errors associated with $\hat{\mathbf{x}}^{(q)}$ are maintained where the coefficients of $\hat{\mathbf{x}}^{(q)}$ are greater than 0, and equal to 0 everywhere else.

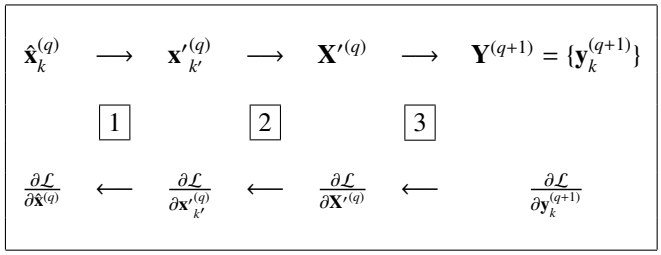


**Table 2. The first line represents the data processing. The second line represents the differential computation: Pooling 1, Concatenation 2, Patch decomposition 3**

### *4.3. Back-propagation of the gradient*

We have opted for a stochastic gradient descent (SGD) as in Bottou (2010) where, at each iteration, small groups of samples (about ten) are used instead of a single sample to "smooth" the direction of descent. At each iteration, a small number $b$ of signals are selected in their order of appearance (Bottou (2012)): at the first iteration, the first $b$ signals (after random mixing) are used, at the second iteration, the next $b$, and so on. Thus, the learning set is cycled.

Stochastic Gradient Descent (SGD) is often used in Machine Learning algorithms. Its asymptotic convergence has been proved for convex systems. Since our problem is not convex, the convergence cannot be proved theoretically. However, we rely on the same intuition as in Deep Learning which states that finding a local minima is sufficient in practice.

Learning rate $\eta$ used for the gradient descent decreases periodically after a number of cycles (Krizhevsky et al. (2012)): it is kept constant for two or three cycles and then it is divided by two.

1. From an initial set of dictionaries $\{\mathbf{D_1}, ..., \mathbf{D_Q}\}$ built for example by unsupervised learning for image reconstruction, we compute the optimal codes for the classification $\hat{x}$ with the Eq.12.
2. We compute the gradients with respect to $W$, and with respect to the dictionaries.
3. We update $W$ and the dictionaries with a gradient descent step.
4. The training process is done until the minimum of cost function $\mathcal{L}$ is reached: after each training cycle, loss and accuracy are computed on the validation set.

## 5. Experiments

Numerous experimentations of image classification have been done using different approaches such as Convolutional Neural Networks CNN (Hinton et al. (2012)). Synthetic results of image classification can be found in Mairal et al. (2014). In this paper, we propose to test our approach on particular conditions when the results of the "classical" approaches are less efficient:

1. The size of the training dataset is limited (**experiment 1**)
2. The structure of the CNN is similar to our structure (number of layers, same pooling, ...) (**experiment 2**)
3. Transfer learning application (**experiment 3**).

### *5.1. Datasets used for testing*

In order to compare the results of our approach, we tested the proposed algorithm on the MNIST dataset (LeCun et al. (1998)), CIFAR-10 dataset (Krizhevsky and Hinton (2009)) and STL dataset (Coates et al. (2011)).

The first well-known **MNIST dataset** used for classification is made up a set of handwritten digits divided in 10 classes (digits 0 - 9) and contains $60,000$ $28 \times 28 \times 1$ pixels images of gray level for training and $10,000$ for testing.

The second dataset is **CIFAR-10 dataset** and represents more natural color images. Only a few works present classification results on the CIFAR-10 dataset using a dictionary learning method only. The CIFAR-10 image database is made up of real color images of $32 \times 32 \times 3$ pixels. It contains 10 classes and $60,000$ images distributed in a predefined and balanced way, in $50,000$ learning images and $10,000$ test images.

The **STL dataset** is particular because there are few images labeled. The STL-10 image dataset consists of real color images of $96 \times 96 \times 3$ pixels. It contains 10 classes and contains $10,000$ training images and $8,000$ test images. This database also includes $100,000$ unlabeled images for unsupervised learning.

In order to facilitate the use of the algorithms, we have resized the image databases to the dimensions of the CIFAR-10 dataset. Thus, the images of MINIST have been rescaled to $32 \times 32 \times 1$ pixels and the images of STL are scaled to the size $32 \times 32 \times 3$ pixels.

The images used in these experiments are small because we had some constraints in computation resources. The method proposed could be theoretically used on larger images but it would require a more efficient implementation (i.e Python instead of MATLAB, GPUs).

### *5.2. Parametrization*

Several parameters must be configured in the architecture proposed. An usual method to configure these parameters is to search the parameter space using a cross validation on the training set but this method needs a lot of experiments and computational power. We chose these parameters according to their properties.

The size of atoms and patches influences the patterns that are learned. For example, small atoms ($5 \times 5$ patch) will learn simple patterns while larger atoms can learn "richer" patterns but require larger dictionaries. In this work, we use patches of size $5 \times 5$ pixels.

In order to compute $\hat{x}$ with Eq.12, we choose the "Elastic-net" regularization defined Eq.2 with $\lambda_1 = 0.1$ and $\lambda_2 = 0.01$. The parameter $\lambda_2$ with a small value allows the gradient descent to be smoothed. The classifier chosen is a linear classifier coupled with the Softmax function (Eq.7) where $\mathbf{W}$ is the vector of parameters penalized by the parameter $\lambda'$. The initial step for the gradient descent is $\eta = 0.6$ and the size of the batches for each iteration is 10 signals. The learning step $\eta$ is divided by two every two cycles.

Unlike the context of reconstruction where the number of atoms influences the ability to faithfully represent a signal by a parsimonious code, in classification it is not necessary to have

| | Architecture A-3-25 | Architecture A-3-50 |
|---|---|---|
| Inputs | $32 \times 32 \times n$ | $32 \times 32 \times n$ |
| Layer 1 | $5 \times 5$ - **25** atoms | $5 \times 5$ - **50** atoms |
| $f$ | *ReLU*<br>*Pooling* $2 \times 2$, stride 2 | *ReLU*<br>*Pooling* $2 \times 2$, stride 2 |
| Layer 2 | $5 \times 5$ - **25** atoms | $5 \times 5$ - **50** atoms |
| $f$ | *ReLU*<br>*Pooling* $2 \times 2$, stride 2 | *ReLU*<br>*Pooling* $2 \times 2$, stride 2 |
| Layer 3 | $5 \times 5$ - **50** atoms | $5 \times 5$ - **100** atoms |
| $f$ | *ReLU* | *ReLU* |

**Table 3. Two architectures with 3 layers used with our dictionary approach. $n = 1$ for gray level images. $n = 3$ for color images.**

large dictionaries ("*over-complete*" case). In our architecture, the dimensions of the dictionaries at the different layers are linked, which limits their choice. Moreover, the limitation is essentially due to the explosion of the computation time and often we decide to choose a number of atoms smaller than the dimension of the signal.

We chose to study some architectures (number of layers, pooling, ...) which are specified in Table 3 and Table 4.

| | Architecture A-5-50 |
|---|---|
| Inputs | $32 \times 32 \times n$ |
| Layer 1 | $5 \times 5$ - **50** atoms |
| Layer 2 | $1 \times 1$ - **50** atoms |
| $f$ | *Pooling* $2 \times 2$, stride 2 |
| Layer 3 | $5 \times 5$ - **50** atoms |
| Layer 4 | $1 \times 1$ - **50** atoms |
| $f$ | *ReLU*<br>*Pooling* $2 \times 2$, stride 2 |
| Layer 5 | $1 \times 1$ - **100** atoms |

**Table 4. Architecture with 5 layers used with our dictionary approach. $n = 1$ for gray level images. $n = 3$ for color images.**

These architectures have been defined in order to get only one code at the last layer. For instance for the architecture A-3-25,

- The initial image of gray level is $32 \times 32 \times 1$
- **Layer 1**: Decomposition of the image gives $28 \times 28 \times 1$ patches of $5 \times 5 \times 1$ patches (*stride* = 1 pixel, $28 = 32 - 4$)
- Coding operation gives $28 \times 28$ codes of size $K_1$ leading to a $28 \times 28 \times K_1$ 3D image.
- Pooling operation gives $14 \times 14 \times K_1$ codes
- **Layer 2**: Decomposition of the 3D image gives $10 \times 10 \times K_1$ patches ($10 = 14 - 4$)
- Coding operation gives $10 \times 10$ codes of size $K_2$ leading to a $10 \times 10 \times K_2$ 3D image.
- Pooling operation gives $5 \times 5 \times K_2$ codes
- **Layer 3**: The 3D image is one patch $5 \times 5 \times K_2$
- Coding operation gives a $1 \times 1 \times K_3$ code used for classification

### *5.3. Experiment 1: Limited training dataset*

We have tested our approach on the dataset MNIST (LeCun et al. (1998)) which gives good results with the whole training set.

The two architectures A-3-25 and A-3-50 of three layers have been tested. The two architectures are similar but the second one has twice the number of atoms at each layer (Table 3).

In a first step, we compare the results obtained for various approaches on MNIST dataset without data augmentation, on the complete training dataset of 60K images (Table 5).

| Type | Approaches | Test error in % |
|---|---|---|
| DL | Tariyal et al. (2016) | 1.36% |
| DL | Yang et al. (2014) | 0.84% |
| CNN | Ranzato et al. (2007) | 0.53% |
| Scattering | Bruna and Mallat (2013) | 0.40% |
| CNN | Mairal et al. (2014) | 0.39% |
| DL | Architecture A-3-25 | 0.46% |
| DL | Architecture A-3-50 | 0.41% |

**Table 5. Classification results for MNIST dataset with full trained dataset of 60K, without data augmentation. The approaches are DL (Dictionary Learning), Scattering and CNN (Convolutional Neural Network)**

The results of the different approaches are relatively similar for a full training set. As Mairal et al. (2014) do, we compare the approaches when the training set is reduced, without data augmentation (Table 6).

| Training Size | 300 | 1K | 10K | 60K |
|---|---|---|---|---|
| Ranzato et al. (2007) | 7.2 | 3.2 | 0.9 | 0.5 |
| Bruna and Mallat (2013) | 4.7 | 2.3 | 0.9 | 0.4 |
| Mairal et al. (2014) | 4.2 | 2.1 | 0.9 | 0.4 |
| A-3-25 $\lambda' = 0$ | 5.6 | 2.7 | 0.8 | 0.4 |
| A-3-25 $\lambda' = 0.01$ | 3.7 | 1.9 | 0.8 | 0.5 |

**Table 6. Test error in % for various approaches on MNIST dataset. The parameter $\lambda'$ is defined in Eq.7**

We can conclude on the importance of the parameter $\lambda'$ which penalizes the vector $\mathbf{W}$ (Eq.7). The explanation is probably that for 10k and 60k, the amount of training data is sufficient (relative to the number of parameters) and penalizing the $\mathbf{W}$-weights prevents the "fitting" of the values to the data, which slightly reduces performance. For smaller sets, it would be the opposite: the $\mathbf{W}$-weights limit "over-fitting".

### *5.4. Experiment 2: Comparison with CNN results with similar architecture*

Because the MNIST dataset is relatively simple, we tested our approach on the CIFAR-10 dataset. The classical CNN approach has good results on CIFAR-10 as Mairal et al. (2014) state. It can be noted that the best performance given in this

paper corresponds to Goodfellow et al. (2013) with 88.32% but with data augmentation. The objective here is to compare results with CNN approaches, but with similar architectures (number of layers, similar pooling, ...).

The number of layers is determined as a function of the size of the images to be processed. In this architecture, the output is fixed to a single vector used for classification. That is why an example of a proposed structure consists of 3 layers using patches of size $5 \times 5$ shifted by 1 pixel and separated by "pooling" operations of size $2 \times 2$.

The choice of the stride of 1 pixel is the smallest possible and also the most expensive in calculation. However, it is also the one that generally provides the best results: the effects of choosing different stride values for classification are presented by Coates et al. (2011). The tests are made for both CNN and Dictionary approaches for architectures with three layers and five layers as represented in Table 7.

Concerning the CNN, the tests are carried out using the Torch library. The descriptors are extracted after the first full connected layers as advocated by Razavian et al. (2014).

| Type | Methods | Accuracy |
|---|---|---|
| DL | Fawzi et al. (2015) | 53.4% |
| Clustering | Coates et al. (2011) | 79.6% |
| DL | Coates and Ng (2011) | 81.5% |
| DL | Lin and Kung (2014) | 81.5% |
| CNN | Mairal et al. (2014) | 82.2% |
| CNN | Hinton et al. (2012) | 83.4% |
| Scattering | Oyallon and Mallat (2015) | 82.3% |
| CNN | A-3-25 like + fully-connected layers | 66.5% |
| CNN | A-3-50 like + fully-connected layers | 83.2% |
| CNN | A-3-50 like + linear SVM | 74.1% |
| DL | A-3-25 + Softmax | 78.9% |
| DL | A-3-50 + Softmax | 83.2% |
| DL | A-5-50 + Softmax | 83.9% |

**Table 7. Performance comparison on the CIFAR-10 dataset without data augmentation. For the proposed architectures - DL: Dictionary + Softmax. CNN: Convolutional Neural Network with same architecture as Dictionary (Table 3 and Table 4) + fully-connected architecture and Softmax or linear SVM**

For CNN architectures, we first chose to use the same architecture as for the Dictionary approach (A-3-25 like, A-3-50 like, A-5-50 like) which means three convolutional layers for feature extraction followed by a fully connected output layer with softmax activation. But the results were bad so we decided to include two additional fully connected layers with ReLU after the convolutional layers, still followed by a fully connected output layer with softmax activation. We also tested the previous CNN architecture for feature extraction followed by a linear SVM classifier (see Table7). As expected, this architecture gives worse results. However for the dictionary learning approach for feature extraction, we only used one output layer with softmax. For the same type of architecture (A-3-25, A-3-50), we can notice that the dictionary learning approach gives better or similar results as the CNN approach.

### *5.5. Experiment 3: Transfer learning*

The transfer learning experiment consists in training the system with one dataset (CIFAR-10 or ImageNet) and using it on a different dataset (STL-10 dataset) to measure the ability of the system to learn globally meaningful descriptors. STL-10 dataset is made up of real color images of $96 \times 96 \times 3$ pixels resized images at the $32 \times 32 \times 3$ pixels (Coates et al. (2011)), and ImageNet (Russakovsky et al. (2015)) is large database (features 1K).

The method is broken down into three steps for the two approaches, CNN and DL:

1. Learning architecture on CIFAR10 or ImageNet (only for CNN):
   - Dictionary structures A-3-50 with linear SVM classifier.
   - Full architecture VGG Net.
2. Learning classifier on STL learning database.
   - Descriptors extracted with learned dictionary and learning of linear SVM classifier.
   - Features extracted with learned VGG Net (after first full connected layer) and learning of linear SVM classifier.
3. Test step on STL test database.
   - Features extracted with learned dictionary and use of learned linear SVM classifier
   - Features extracted with learned VGG Net and use of learned linear SVM classifier.

At steps 2 and 3, no re-learning is done: the descriptors are extracted directly from the first learned architecture step and then coupled with a linear SVM classifier.

Table 8 shows some performance comparisons with the unsupervised methods of the state of the art to obtain descriptors which are not suited to this specific classification task. This allows the relevance of the descriptors learned on other datasets to be estimated.

Transfer learning is a difficult task. As we can see, the full VGG Net architecture learned on the large ImageNet database (features 1K) gives better results: the classifier manages to give 74% with the test set. It is considered as the base line of transfer learning.

Given the simplicity of the model (A-3-25) and the poor CIFAR-10 training database, the dictionary learning approach gives better results than the CNN approach, meaning that the dictionary approach is more efficient to learn adequate descriptors. This approach also provides better results than the unsupervised learning methods of the state of the art, which illustrates the ability of the method to extract generic and discriminative descriptors for classification.

| Type | Training | Methods | Accuracy STL |
|---|---|---|---|
| Clustering | No | Coates et al. (2011) | 51.5% |
| DL | No | Coates and Ng (2011) | 59.0% |
| DL | No | Lin and Kung (2014) | 60.4% |
| CNN | No | Mairal et al. (2014) | 62.3% |
| DL | No | Bo et al. (2013) | 64.5% |
| CNN | ImageNet | VGG Net | 74.0% |
| CNN | CIFAR 10 | A-3-50 like | 45.0% (*83.2%*) |
| DL | CIFAR 10 | A-3-50 | 66.4% (*83.2%*) |

**Table 8. Comparison of performance of unsupervised (No) or trained methods with the CIFAR-10 and ImageNet databases and applied to the STL-10 database. In brackets, the percentage of the CIFAR-10 classification as Table 7**

## 6. Conclusion

The multilayer dictionary structure used to extract descriptors from an image gives quite satisfactory results compared to CNNs for classifying natural images when the learning set is reduced or in transfer learning applications. The patch decomposition at all layers makes it possible to extract local information that makes the classification more robust.

We still have not fully investigated this method and we will continue to work on the proposed structure in order to study the effects of the choices of the different parameters (dictionary size, sparsity, number of layers). For future work, we will confront this method to more complex datasets, containing larger images, to challenge the limit of this approach.

It could be possible to build hybrid architecture with convolutional and dictionary layers. For example, it may be interesting to replace the first dictionary layer by a convolutional layer to compare their performances in selecting low-level attributes.

The tests of the system were carried out using the same algorithm but with different settings. The Software implementation has not been optimized. It should be noted that depending on the parametrization chosen, the learning phase can be extremely intensive in terms of computation time (experiments carried out on a machine using 8 cores @2.67GHz and MATLAB). As a consequence, the training phase can take up to several days. However, predicting the label of an image can be done in a few seconds.

It will be necessary to consider an implementation with a strong parallelisation, which is quite possible considering the multilayer structure of the system proposed and the decomposition of the images into patches.

## Acknowledgment

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01).

## References

Aharon, M., Elad, M., Bruckstein, A., 2006. K-svd: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing 54, 4311–4322.

Bo, L., Ren, X., D.Fox, 2013. Unsupervised feature learning for RGB-D based object recognition. Springer International Publishing, Heidelberg. pp. 387–402. doi:10.1007/978-3-319-00065-7_27.

Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent, in: 19th International Conference on Computational Statistics (COMPSTAT).

Bottou, L., 2012. Stochastic Gradient Descent Tricks. Technical Report. Microsoft Research.

Bradley, D.M., Bagnell, J.A., 2008. Differential sparse coding, in: Neural Information Processing Systems 21 (NIPS).

Bruna, J., Mallat, S., 2013. Invariant scattering convolution networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 35.

Chan Wai Tim, S., Rombaut, M., Pellerin, D., 2015. Rejection-based classification for action recognition using a spatio-temporal dictionary, in: European Signal Processing Conference (EUSIPCO).

Coates, A., Lee, H., Ng, A., 2011. An analysis of single-layer networks in unsupervised feature learning, in: Artificial Intelligence and Statistics (AISTATS).

Coates, A., Ng, A., 2011. The importance of encoding versus training with sparse coding and vector quantization, in: 28 th International Conference on Machine Learning (ICML).

Elad, M., Aharon, M., 2006. Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing 15, 3736–3745.

Fawzi, A., Davies, M., Frossard, P., 2015. Dictionary learning for fast classification based on soft-thresholding. International Journal of Computer Vision 114, 306–321.

Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y., 2013. Maxout networks. 30 th International Conference on Machine Learning (ICML) 28.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detector, in: arXiv - https://arxiv.org/pdf/1207.0580.pdf.

Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images. Technical Report. University of Toronto.

Krizhevsky, A., Sutskeve, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems (NIPS).

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

Lin, T.H., Kung, H., 2014. Stable and efficient representation learning with nonnegativity constraints, in: 31 st International Conference on Machine Learning (ICML).

Mairal, J., Bach, F., Ponce, J., 2012. Task-driven dictionary learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 791–804.

Mairal, J., Elad, M., Sapiro, G., 2008. Sparse representation for color image restauration. IEEE Transactions on Image Processing 17, 53–69.

Mairal, J., Koniusz, P., Harchaoui, Z., Schmid, C., 2014. Convolutional kernel networks. Adv. Neural Information Processing Systems (NIPS) .

Olshausen, B.A., Field, D.J., 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381, 607 – 609.

Oyallon, E., Mallat, S., 2015. Deep roto-translation scattering for object classification, in: arXiv - https://arxiv.org/pdf/1412.8659v2.pdf.

Qiu, Q., Jiang, Z., Chellappa, R., 2011. Sparse dictionary-based representation and recognition of action attributes, in: IEEE International Conference on Computer Vision (ICCV).

Raina, R., Battle, A., Lee, H., packer, B., Ng, A.Y., 2008. Self-taught learning: Transfer learning from unlabeled data, in: 25th International Conference on Machine Learning (ICML).

Ranzato, M., Huang, F., Boureau, Y., LeCu, Y., 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition, in: Computer Vision and Pattern Recognition Conference (CVPR).

Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S., 2014. Cnn features off-the-shelf: an astounding baseline for recognition, in: Computer Vision and Pattern Recognition (CVPR).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang,

Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fe, L., 2015. Imagenet large scale visual recognition challenge, in: IJCV.

Tariyal, S., Majumdar, A., Singh, R., Vatsa, M., 2016. Deep dictionary learning. IEEE Acces, DOI 10.1109/ACCESS.2016.2611583 4, 10096–10109.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society 58 Issu 1, 267–288.

Tibshirani, R., 2013. The lasso problem and uniqueness. Electronic Journal of Statistics 7., 1456–1490.

Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C., 2009. Evaluation of local spatio-temporal features for action recognition. British Machine Vision Conference (BMVC) .

Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y., 2009. Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 210–227.

Yang, J., Yu, K., Huang, T., 2010. Supervised translation-invariant sparse coding. IEEE Conference on Computer Vision and Pattern Recognition (CVPR2010) .

Yang, M., Zhang, L., Feng, X., Zhang, D., 2014. Sparse representation based fisher discrimination dictionary learning for image classification. International Journal of Computer Vision (IJCV) 109 issu 3, 209–232.

Zhou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, Series B 67, 301–320.