Multi-layer Dictionary Learning for Image Classification

Stefen Chan Wai Tim, Michèle Rombaut, Denis Pellerin

Multi-layer Dictionary Learning for Image Classification

Stefen Chan Wai Tim, Michele Rombaut, and Denis Pellerin

Univ. Grenoble Alpes, GIPSA-Lab
F-38000 Grenoble, France

Abstract. This paper presents a multi-layer dictionary learning method for classification tasks. The goal of the proposed multi-layer framework is to use the supervised dictionary learning approach locally on raw images in order to learn local features. This method starts by building a sparse representation at the patch-level and relies on a hierarchy of learned dictionaries to output a global sparse representation for the whole image. It relies on a succession of sparse coding and pooling steps in order to find an efficient representation of the data for classification. This method has been tested on a classification task with good results.

1 Introduction

Sparse coding is the approximation of an input signal by a linear combination of a few number of dictionary elements. Dictionary learning and sparse representations have received a lot of focus in the recent years because they have led to state-of-the-art results in many applications, in particular in image processing. One reason for its success is that it can efficiently learn the underlying patterns in the data, leading to good performances for example in denoising [6], [12] or inpainting [1]. The sparse codes obtained can also be seen as a new representation of the input or as features in classification tasks [11], [15], [2], [19], [3]. In such cases, the dictionary is often learned in an unsupervised way. Then, the sparse codes obtained can either be used directly for classification [14], or as features fed to a classifier [3] (i.e SVM).

Recent researches have emphasized the advantages of learning **discriminative** sparse models [11], [2], [20] instead of purely **reconstructive** ones. It is usually done by learning conjointly the sparse representation and the classifier. 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 doing so in denoising, however, in the case of classification, a mean to fuse efficiently the representation of the patches into an image-level descriptor is needed (i.e pooling [20] or Bag of words [18]).

In this paper, we propose to learn discriminative dictionaries for classification (similarly as in [20]) while working at a patch-level in a supervised framework by

using an architecture which combines many layers of sparse coding and pooling in order to reduce the dimension of the problem.

Method framework In the proposed multi-layer architecture, the sparse codes obtained by encoding signals on a dictionnary 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. In Section 3, we introduce our multi-layer dictionary learning setup. In Section 4, we present the experiments and their results.

2 Dictionary learning

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

2.1 Unsupervised dictionary learning

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. Let's consider a set of n signals $\mathbf{Y} = [\mathbf{y}_1, ..\mathbf{y}_n]$. A dictionary $\mathbf{D}$ can be learned by solving:

$$\min_{\mathbf{D},\mathbf{x}_k} \quad \sum_{k=1}^{n} \|\mathbf{y}_k - \mathbf{D}\mathbf{x}_k\|_2^2 \; + \lambda \|\mathbf{x}_k\|_1 \tag{1}$$

with $\mathbf{D} = (d_{ij})_{i\in[1,m],j\in[1,K]}$ being a dictionary, K the number of dictionary atoms $\mathbf{d}_j$, m the dimension of $\mathbf{y}_k$, and $\mathbf{x}_k$ a sparse vector containing the coefficients to reconstruct $\mathbf{y}_k$. $\|\cdot\|_2$ and $\|\cdot\|_1$ denote ℓ_2-norm and ℓ_1-norm respectively. 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 get both dictionary $\mathbf{D}$ and coefficients $\mathbf{x}_k$ [1], [13], [17].

In this formulation, the reconstruction error is minimized and the sparsity can be controlled with the value of the parameter λ (a higher λ increases sparsity). Using such unsupervised approach can yield good results in reconstruction problem and even in classification tasks [15], [19], because it can often find the underlying patterns in the data. However, it has been shown that better results could be obtained by tuning the dictionaries for a specific task [11], [2].

2.2 Supervised dictionary learning

Supervised dictionary learning methods began to be investigated [11], [20] in order to take advantage of parsimony in classification tasks. 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 set 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) and l_k, $k \in [1, n]$ is the corresponding label for $\mathbf{y}_k$. We define $\mathcal{L}$, a differentiable classification loss function and $\mathbf{W}$, its set of parameters.

The supervised dictionary learning problem can be written as follows:

$$\hat{\mathbf{x}}_k = \underset{\mathbf{x}}{\operatorname{argmin}} \quad \|\mathbf{y}_k - \mathbf{D}\mathbf{x}\|_2^2 \ + \lambda\|\mathbf{x}\|_1 \tag{2}$$

$$\min_{\mathbf{W},\mathbf{D}} \quad \sum_{k=1}^{n} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) \tag{3}$$

The objective here, is to minimize the suitable cost function $\mathcal{L}$ with respect to its parameters $\mathbf{W}$ and a dictionary $\mathbf{D}$ (Fig. 1) using gradient descent for example. The main difference between this formulation 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 $\mathbf{D}$ and $\mathbf{W}$, it is possible to use a method similar to the backpropagation algorithm [10] used in neural networks.

Computing the gradient of $\mathcal{L}$ with respect to the parameters $\mathbf{W}$ is usually simple. The main difficulty when solving (Eq. 3) is the minimization of the cost function $\mathcal{L}$ with respect to dictionary $\mathbf{D}$ because it does not appear explicitly and involves another optimization problem (solving for $\hat{\mathbf{x}}_k$, (Eq. 2)). 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 [11], [2].

In this paper, we follow the ideas of [11] which show the differentiability of $\mathcal{L}$ and give the steps to compute its gradient with respect to the parameters $\mathbf{W}$ and the dictionary $\mathbf{D}$. According to the paper, the desired gradient can be computed as follows :

$$\nabla_{\mathbf{D}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) = \mathbf{D}\beta\hat{\mathbf{x}}_k^\top + (\mathbf{y} - \mathbf{D}\hat{\mathbf{x}}_k)\beta^\top \tag{4}$$

We define $\Lambda = \{i | x_i \neq 0\})$, the set of non-zero coefficients of the considered code $\mathbf{x}_k$.

Let's consider a vector $\beta \in \mathcal{R}^K$. β_Λ which is the vector β restricted to the indices in Λ is defined as follows :

$$\beta_\Lambda = (\mathbf{D}_\Lambda^\top \mathbf{D}_\Lambda)^{-1} \nabla_{\hat{\mathbf{x}}_{k\Lambda}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W}) \tag{5}$$

where $\hat{\mathbf{x}}_{k\Lambda}$ corresponds to $\hat{\mathbf{x}}_k$ restricted to its non-zero coefficients and $\beta_j = 0$, if $j \notin \Lambda$.

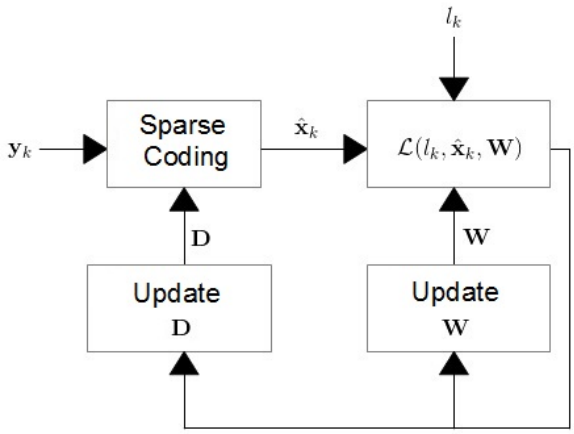

Fig. 1. A signal $\mathbf{y}_k$ associated to a label l_k is encoded by a dictionary $\mathbf{D}$, the resulting code $\hat{\mathbf{x}}_k$ is used as an input to a cost function with parameters $\mathbf{W}$. The cost $\mathcal{L}(l_k, \hat{\mathbf{x}}_k, \mathbf{W})$ is computed. Then, the dictionary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

2.3 Patch decomposition and pooling

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 an image (Section 2.2) or a patch reshaped as a column vector containing the pixel values (Section 3.2).

A supervised dictionary learning approach can successfully learn patterns for a classification task. However, in practice, some limitations can be observed: in particular, if we take the example of image classification, these methods are the most effective when the input images are relatively small and the object of interest homogeneously localized. Otherwise, the dictionary used would need a huge number of atoms in order to obtain an efficient sparse decomposition. 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 of the set of patches is needed.

This particular problem has been studied by Yang et al. [20]. In the paper, the authors proposed 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 obtain a multi-scale representation.

To be able to deal with large images, we recall the pooling function: the pooling operation is used to insert robustness and translation invariance to the features and are an effective way of reducing the dimensions. It is a mean of

aggregating spatial local features in an image. Let's consider the classification of an image I. First, it can be decomposed into local overlapping patches $\mathbf{y}_k$ which are encoded on dictionary by computing the coefficients $\hat{\mathbf{x}}_k$. These patches can be spatially localized on a grid which gives the relative position of all patches. Then the codes $\hat{\mathbf{x}}_k$ obtained can be, for example, averaged over a small group of patches reducing the total number of patches.

3 Multi-layer supervised dictionary learning

3.1 Multi-layer framework

Intuitively, sparse coding can extract important characteristics for reconstruction in unsupervised frameworks and for classification in supervised methods.

The idea of the contribution is to reiterate the sparse coding layer in order to increase the discriminability of the features. The method is inspired by the convolutional networks [16]: the convolution by a filter is replaced by a sparse coding step.
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.

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 a new input for the following layer. So, adding another dictionary encoding step is akin to doing another projection in a new coordinate system.
Each dictionary is learned in a supervised manner. The process can be repeated many times, with as much dictionaries as the number of layers in the architecture.

The proposed multi-layer can be described as follows (Fig. 2):

1. An input image is decomposed into a set of overlapping patches ordered to retain their spatial localization.
2. Each patch $\mathbf{y}_k$ is encoded on a first dictionary $\mathbf{D}^{(1)}$. Since the spatial localization of the patches has been retained, the set of encoded sparse codes $\hat{\mathbf{x}}_k$ can be represented as a 3D volume $\mathbf{X}$ with a depth equal to the number of atoms in dictionary $\mathbf{D}^{(1)}$.
3. The resulting 3D volume is treated as a 3D image input for the next layer. (1) and (2) can be repeated for the chosen number of layers.

To complement 3), for example, if we consider the q-th layer, we can write: $\mathbf{Y}^{(q)} = \hat{\mathbf{X}}^{(q-1)}$, meaning that the stacked codes at layer $q-1$ are used as an input image $\mathbf{Y}^{(q)}$ in layer q (see Fig. 2). The image $\mathbf{Y}^{(q)}$ is then decomposed into 3D patches $\hat{\mathbf{y}}_k^{(q)}$ and $\hat{\mathbf{x}}_k^{(q)}$ are obtained by encoding $\mathbf{y}_k^{(q)}$ with $\mathbf{D}^{(q)}$. More generally, we can replace $\mathbf{Y}^{(q)} = \hat{\mathbf{X}}^{(q-1)}$ by $\mathbf{y}_k^{(q)} = f(\hat{\mathbf{X}}^{(q-1)})$ where f can denote a transformation on the codes coupled with the patch decomposition.

In order to optimize the multi-layer architecture for classification, it is needed to find the optimal dictionaries at each layer.

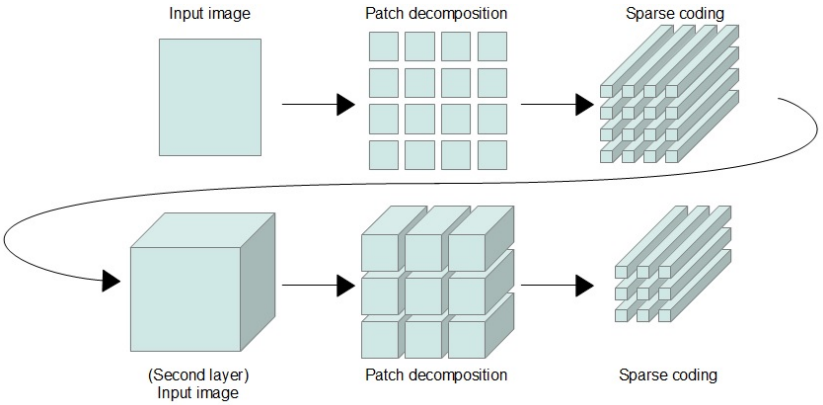

Fig. 2. Example of an architecture with 2 layers. An input image in presented to the first layer. The image is decomposed into patches $\mathbf{y}_k^{(1)}$ which are encoded by dictionnary $\mathbf{D}^{(1)}$. The codes $\mathbf{x}_k^{(1)}$ are processed and restructured into a 3D volume $\mathbf{X}^{(1)}$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

3.2 Formulation

Let's consider a set of input features $\mathbf{Y} = \{\mathbf{y}_1^{(1)}, \cdots, \mathbf{y}_p^{(1)}\}$ associated to a label l. This set is obtained by decomposing an input image into p patches. In the 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. Let's consider the proposed algorithm with Q layers, its formulation is as follows:

For $q \in [1, Q-1]$ (same as in (Eq. 2))

$$\hat{\mathbf{x}}_k^{(q)} = \underset{\mathbf{x}}{\operatorname{argmin}} \quad \|\mathbf{y}_k^{(q)} - \mathbf{D}^{(q)}\mathbf{x}\|_2^2 \; + \lambda_q \|\mathbf{x}\|_1 \tag{6}$$

$\mathbf{y}_k^{(q+1)}$ is obtained from the output of the previous layer $\hat{\mathbf{x}}_k^{(q)}$ by applying a transformation f (see [20] for an example with f being the max-pooling function) followed by a new patch decomposition step. In this paper, we propose to use the average-pooling function.

To optimize the classification cost function with respect to the dictionaries at each layer, we use the backpropagation algorithm. Therefore, we need to compute the gradients with respect to the various dictionaries $\mathbf{D}^{(1)}, \cdots, \mathbf{D}^{(Q)}$.

By extending the formulation given in (Eq. 3), we obtain:

$$\min_{\mathbf{W}, \mathbf{D}^{(1)}, \cdots, \mathbf{D}^{(Q)}} \quad \sum_{k=1}^{n} \mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) \tag{7}$$

We only use the output $\hat{\mathbf{x}}_k^{(Q)}$ of the last layer as features for the classification (Fig. 3) and the cost function is minimized over the entire training set of n images.

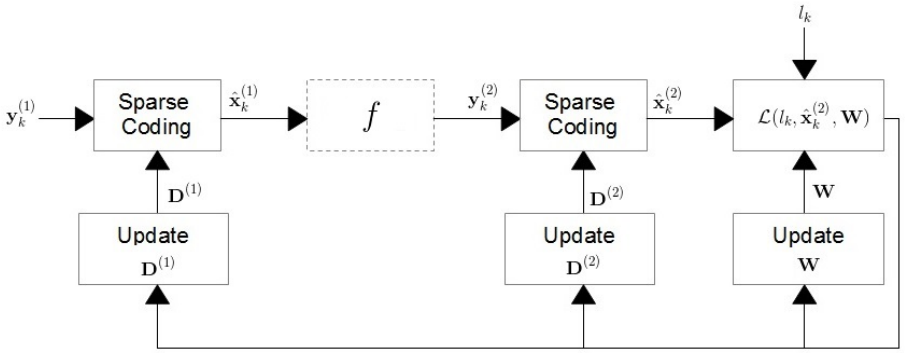

Fig. 3. Proposed structure with 2 layers. Input vectors go through a sparse coding step. We have $\mathbf{y}_k^{(2)} = f(\hat{\mathbf{X}}^{(1)})$ as input to the second layer. Backpropagation is then used to update both dictionaries simultaneously.

3.3 Computation of the gradients

To use the backpropagation 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 $\mathbf{y}_k^{(q)}$.

For a pair $(\mathbf{Y}, l)$, the gradient of $\mathcal{L}$ with respect to dictionary $\mathbf{D}^{(Q)}$ (the dictionary of the last layer) is computed using equations (Eq. 4) and (Eq. 6). If we introduce the notation using the layer number, the equation for the last layer becomes:

$$\nabla_{\mathbf{D}^{(Q)}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) = \mathbf{D}^{(Q)} \beta \hat{\mathbf{x}}_k^{(Q)\top} + (\mathbf{y}_k^{(Q)} - \mathbf{D}^{(Q)} \hat{\mathbf{x}}_k^{(Q)}) \beta^\top \tag{8}$$

with β defined as:
for the indexes contained in the set Λ,

$$\beta_\Lambda = (\mathbf{D}_\Lambda^{(Q)\top} \mathbf{D}_\Lambda^{(Q)})^{-1} \nabla_{\hat{\mathbf{x}}_{k\Lambda}^{(Q)}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) \tag{9}$$

and $\beta_j = 0$, if $j \notin \Lambda$.

To compute the gradient of the layer q, the computations for the dictionary become:

$$\nabla_{\mathbf{D}^{(q)}} \mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) = \mathbf{D}^{(q)} \beta \hat{\mathbf{x}}_k^{(q)\top} + (\mathbf{y}_k^{(q)} - \mathbf{D}^{(q)} \hat{\mathbf{x}}_k^{(q)}) \beta^\top \tag{10}$$

$$\beta_\Lambda = (\mathbf{D}_\Lambda^{(q)\top}\mathbf{D}_\Lambda^{(q)})^{-1}\nabla_{\hat{\mathbf{x}}_{k\Lambda}^{(q)}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W}) \tag{11}$$

We underline that only the last layer Q intervenes in the classification step that is why the term is $\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W})$ and not $\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(q)}, \mathbf{W})$: by choice, the output of the last layer is a single code vector. To compute the gradient of the cost function $\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W})$ with respect to the dictionary $\mathbf{D}^{(q)}$ of the q-th layer using backpropagation, we need to compute $\nabla_{\hat{\mathbf{x}}_{k\Lambda}^{(q)}}\mathcal{L}(l_k, \hat{\mathbf{x}}_k^{(Q)}, \mathbf{W})$. The computation of this gradient can be decomposed as follows:

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{(q)}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{(q+1)}}\frac{\partial \hat{\mathbf{x}}^{(q+1)}}{\partial \mathbf{y}^{(q+1)}}\frac{\partial \mathbf{y}^{(q+1)}}{\partial \hat{\mathbf{x}}^{(q)}} \tag{12}$$

where $\frac{\partial \hat{\mathbf{x}}^{(q)}}{\partial \mathbf{y}^{(q)}}$:

$$\frac{\partial \hat{\mathbf{x}}_\Lambda^{(q)}}{\partial \mathbf{y}^{(q)}} = (\mathbf{D}_\Lambda^{(q)\top}\mathbf{D}_\Lambda^{(q)})^{-1}\mathbf{D}_\Lambda^{(q)} \tag{13}$$

and 0 elsewhere.

We remind that the transformation f between $\hat{\mathbf{x}}^{(q)}$ and $\mathbf{y}^{(q+1)}$ can be the identity, a pooling operation or a more general transformation (see 3.1) combined with a decomposition into patches, so the backpropagation includes a image "reconstruction" step to reverse the patch decomposition (Fig. 4).

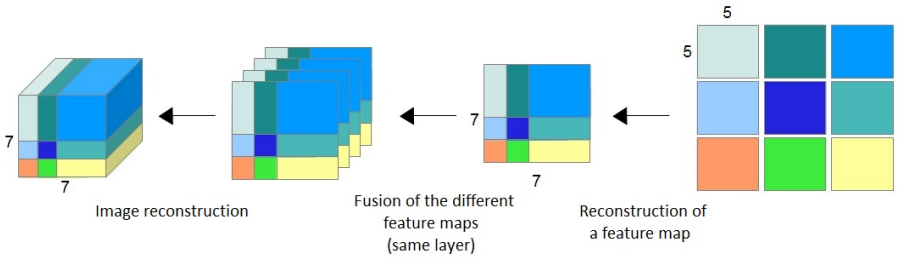

Fig. 4. Example of the reconstruction step from 5×5 patches to $7 \times 7 \times K$ volume.

4 Experimentations

We tested the proposed algorithm on the MNIST [10] dataset and CIFAR-10 dataset [9]. The first well-known MNIST dataset used for classification regroups a set of handwritten digits divided in 10 classes (i.e 0 - 9) and contains 60000 (28×28) pixels images for training and 10000 for testing. These images have been rescaled to (32×32) pixels to be able to fit the CIFAR-10 dataset.

We have chosen to use the cross-entropy function (Eq. 14) 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 is computed as follows:

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

where p_{ik} is defined by:

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

To demonstrate the proposed method, we choose to use an architecture with $Q = 3$ layers. For the first layer, we decompose the input image into patches of (5×5) pixels with a stride of 1 pixels. The second layer use patches of size (5×5). Then, a pooling step is done on a 2×2 region without overlap. We follow last layer with size (5×5). After the last layer, only one code remains for the whole image and this code is used as input for the classification.

The optimization method used is the stochastic gradient descent with a batch size of 10. The training set is shuffled randomly and the training samples for each batch are used in order. The learning step is initially fixed at 0.3 and divided by 2 every 3 passes on the full dataset.

For parameter λ (Eq. 1), we choose $\lambda = 0.1$ for all three layers. Empirically, this value leads to good reconstruction while giving very sparse codes. Increasing this value too much can lead to patches not being reconstructed (only zeros) while reducing the value (i.e by an order of magnitude) increases the number of non-zero coefficients and the computation costs without necessarily improving the performances.

We tested two configurations of architecture: we used $K = 25$, 25, 50 atoms and $K = 50$, 50, 100 atoms for the three layers. Increasing the number of atoms in the intermediate layers leads to very high dimensional input for the subsequent layer hindering the computational performances. The tests has been run on Matlab and we could not test very large dictionaries.

The results in Table 1 are obtained with the classifier directly learned by the algorithm. We used the original training and test sets with no data augmentation. Supervised learning greatly improves the perfomances for the tested configurations. Moreover, the gain in perfomances between the two choices of the number of atoms is small even though the number of atoms is doubled. It may be explained by the fact that the images in the MNIST dataset are not really complex so additional atoms are not needed to describe more features.

Table 2 regroups some performances of state of the art methods on the MNIST dataset. The results for the proposed method are obtained with no data augmentation as opposed to [11] and [20].

Methods	Dictionary size K	Error rate
1 layer (unsupervised)	700	3.71%
3 layers (unsupervised)	25, 25, 50 (5×5), (5×5), (5×5)	4.93%
3 layers (unsupervised)	25, 25, 50 (5×5), (5×5), (5×5)	2.2%
3 layers (supervised)	25, 25, 50 (5×5), (5×5), (5×5)	0.46%
3 layers (supervised)	50, 50, 100 (5×5), (5×5), (5×5)	0.41%

Table 1. Performance comparison between supervised and unsupervised learning with a linear classifier on the MNIST dataset.

Methods	Error rate
Yang et al. [20]	0.84%
Mairal et al. [11]	0.54%
Proposed method	0.41%

Table 2. Performance comparison on the MNIST dataset.

We also tested our method on the CIFAR-10 dataset [9] which is constructed with 60000 real images (50000 images for learning and 10000 images for testing) of size (32×32) pixels, separated into 10 classes. This dataset is more challenging than the MNIST dataset because the variance of view points, poses and localizations of the object of interest is much higher. In particular, the dictionary learning methods which process the whole image at once usually have some difficulties to learn the features efficiently. For this test, we used the same structure as the one used for the MNIST dataset (3 layers) with the same parameters (Table 3).

Methods	Dictionary size K	Accuracy
3 layers (unsupervised)	25, 25, 50 (5×5), (5×5), (5×5)	34.98%
3 layers (unsupervised)	25, 25, 50 (5×5), (5×5), (5×5)	42.39%
3 layers (supervised)	25, 25, 50 (5×5), (5×5), (5×5)	78.86%
3 layers (supervised)	50, 50, 100 (5×5), (5×5), (5×5)	83.03%

Table 3. Performance comparison between supervised and unsupervised learning with a linear classifier on the CIFAR-10 dataset.

Only a few works present classification results on the CIFAR-10 dataset using a dictionary learning method only. However, other methods exist (i.e Convolutional neural networks) [8] to deal with this kind of dataset.

Methods	Accuracy
Fawzi et al. [7]	53.44%
Coates et al. [4]	79.6%
Coates et al. [5]	81.5%
Proposed method	83.03%

Table 4. Performance comparison on the CIFAR-10 dataset.

Table 4 shows some results on the CIFAR-10 dataset without data augmentation. Fawzi et al. [7] use a single layer dictionary learning method for comparison. Coates et al. [4], [5] used unsupervised multi-layer sparse coding with large dictionaries (up to 4k atoms). The proposed method performs better using a few number of atoms (undercomplete dictionaries) showing the capability of learning discriminative dictionaries. Better performances may be obtained by increasing the number of atoms in the layers, however at the moment, the MATLAB implementation used is too slow.

During our experiments, we noticed that the number of atoms in the first layer (image layer) is important, for example choosing $K = 15, 25, 50$ (15 instead of 25 in the first layer) could leave to a drop of about 10% in performances. The same can also be said of the number of atoms of the last layer (size of the features) but with a lesser extent.

5 Conclusion

In this paper, we have presented a multi-layer dictionary learning framework. It can potentially handle an image of any size as input and performs the learning of features at the patch level. Its goal is to allow the use of supervised dictionary learning methods on images while evading the computational issues that arise when dealing with large dictionary atoms.

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 with more complex datasets, containing larger images, to challenge the limit of this approach.

Acknowledgement

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

References

1. Aharon, M., Elad, M., Bruckstein, A.: K-svd: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing (2006)
2. Bradley, D.M., Bagnell, J.A.: Differential sparse coding. NIPS (2008)
3. Chan Wai Tim, S., Rombaut, M., Pellerin, D.: Rejection-based classification for action recognition using a spatio-temporal dictionary. In: EUSIPCO (2015)
4. Coates, A., Lee, H., Ng, A.: An analysis of single-layer networks in unsupervised feature learning. In: AISTATS (2011)
5. Coates, A., Ng, A.: The importance of encoding versus training with sparse coding and vector quantization. In: ICML (2011)
6. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing (2006)
7. Fawzi, A., Davies, M., Frossard, P.: Dictionary learning for fast classification based on soft-thresholding. International Journal of Computer Vision 114, 306–321 (2015)
8. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detector. In: arXiv - https://arxiv.org/pdf/1207.0580.pdf (2012)
9. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324 (1998)
11. Mairal, J., Bach, F., Ponce, J.: Task-driven dictionary learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (2012)
12. Mairal, J., Elad, M., Sapiro, G.: Sparse representation for color image restauration. IEEE Transactions on Image Processing 17, 53–69 (2008)
13. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature (1996)
14. Qiu, Q., Jiang, Z., Chellappa, R.: Sparse dictionary-based representation and recognition of action attributes. ICCV (2011)
15. Raina, R., Battle, A., Lee, H., packer, B., Ng, A.Y.: Self-taught learning: Transfer learning from unlabeled data. ICML (2008)
16. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. In: ICLR (2015)
17. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society (1996)
18. Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. British Machine Vision Conference (2009)
19. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 210–227 (2009)
20. Yang, J., Yu, K., Huang, T.: Supervised translation-invariant sparse coding. CVPR (2010)