Final Graduation Project Report

Institut de Recherche en Informatique de Toulouse - IRIT

SAMoVA

# Statistical methods vs. Neural network

# -

# Comparison of methods for learning units using Dictionary Learning and Sparse Coding vs Auto-encoders

*Author*
Thomas Rolland

*Supervisors*
Thomas Pellegrini
Adrian Basarab
Carine Jauberthie

August 27, 2018

# Remerciements

Je voudrais remercier toute l'équipe SAMoVA sans laquelle ce stage n'aurait jamais eu lieu. Merci de leurs accueils ainsi que leurs précieux conseils pendant ces cinq mois. Je souhaiterais notamment remercier Thomas Pellegrini d'avoir encadré ce stage, pour ses conseils, ses idées, sa patience et son accueil. Il mérite amplement son nom de Monsieur "Deep Learning" de SAMoVA.

Je voudrais aussi remercier Adrian Basarab d'être intervenu comme expert des méthodes de Sparse Coding, ses conseils ont été précieux dans la réussite de ce stage.

Ce stage n'aurais pas été ce qu'il a été sans la présence des différents doctorants et stagiaires des différentes équipes du Thème 1. Merci à eux de m'avoir permis de travailler dans une bonne ambiance.

Je souhaiterais naturellement remercier aussi mes proches, ma famille, mes amis.

Certain d'oublier quelqu'un, je remercie aussi tous ceux qui ont participés à ce stage, de près ou de loin.

J'ai aussi une pensée pour Mamie Brest et Lili.

# Contents

# Chapter 1

# Introduction

## 1.1 Hosting Laboratory

In the context of my master's degree, I did 5 months of internship in the SaMova team in IRIT laboratory.

The IRIT (**I**nstitut de **R**echerche en **I**nformatique de **T**oulouse – **I**nformatics **R**esearch **I**nstitute of **T**oulouse) represents one of the major laboratories of the French research in computer science, with a workforce of more than 700 members including 272 researchers and teachers 204 PhD students, 50 post-doc and researchers under contract and also 32 engineers and administrative employees.
The 21 research groups of the laboratory are dispatched in seven scientific themes covering all the computer science domains of today :

- 1 : Information Analysis and Synthesis
- 2 : Indexing and Information Search
- 3 : Interaction, Autonomy, Dialogue and Cooperation
- 4 : Reasoning and Decision
- 5 : Modelization, Algorithms and High-Performance Calculus
- 6 : Architecture, Systems and Networks
- 7 : Safety of Software Development

SAMoVA (**S**tructuration, **A**nalysis, **Mo**delling of **V**ideo and **A**udio documents) team, include into the Topic 1 - Information Analysis and Synthesis, focuses its research activities mainly on audiovisual contents. Those works are applied on different kinds of content such as audiovisual content analysis, spoken content analysis, analysis of pathological voice, ...

## 1.2 Context

Since time immemorial, speech has been the most important means of communication among humans. By definition, speech is the ability or act of expressing or describing thoughts, feelings or perceptions through the articulation of words. To do this, humans use their vocal apparatus (lungs, glottis, larynx, tongues, lips, jaws, ...) to produce "syllables" that fuse to form words and phrases.

Speech recognition allows the captured human voice to be analyzed and transcribed into machine-readable text. Over the past decades, automated speech recognition has become one of the major fields of artificial intelligence and signal processing. Indeed, since the introduction to our daily lives of artificial personal assistants like Siri or Google Assistant, speech recognition systems must answer more complex demands. To achieve this level of complex problem solving, these systems must first understand what the user has said, i.e. do good speech recognition.

Recently, the speech recognition field has benefited from advances in deep learning with neural network methods to improve results and advanced technologies.
Speech recognition is traditionally based on two steps (see figure 1.1):

- First, the raw signal is transformed by a feature extractor to give a new representation of this signal, with more relevant information (generally we use Mel Frequency Cepstral Coefficients as a new representation for speech processing).
- Second, for speech recognition, this new representation is given as the input of a classifier that will give us the text translation of our input signal.

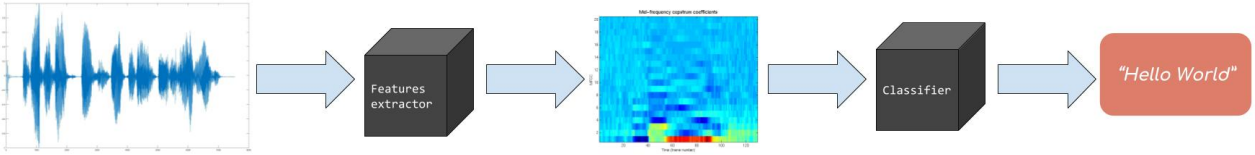


Figure 1.1: Basic pipeline of speech recognition

The goal of my internship is to find a new kind of feature extractor using deep learning and statistical methods. Compare the discriminating power of each of them for signal processing in a generic approach, whether we work on images or sounds. Our work was mainly focused on images using the MNIST dataset since working on the sound level is similar. Indeed the sound signal is transformed into a spectrogram like that will be used as an image to feed the models.

The chosen methods to replace the traditional feature extractor are:

- Auto-Encoder as a Deep learning approach
- Sparse Coding a Statistical approach

However, the main focus of this internship has been on the statistical methods due to the recent interest of the signal processing community in this kind of method.

## 1.3 Organization

The present report is organized as follows:

- **Chapter 2**: Explain the principle of traditional Sparse Coding, how it is done and give an example on MNIST dataset.
- **Chapter 3**: Show the problem with traditional Sparse coding for classification and bring a new solution to obtain a discriminative Sparse code for classification.
- **Chapter 4**: Here, we will explain the natural extension of traditional Sparse Coding to handle shift-variance.
- **Chapter 5**: As we did in chapter 3 for traditional Sparse coding, we will explain one method to obtaining discriminatory Sparse coding in Convolution Sparse Coding.
- **Chapter 6**: We present the deep learning methods with which we compare Sparse Coding.
- **Chapter 7**: Explain our results and the different perspectives we have.

# Chapter 2

# Traditional Sparse Coding

## 2.1 *Sparseland*

Modeling data play a central role in signal processing and machine learning. Sparse representation model also refers as *Sparseland* [9], suggests a description of every signal as sparse linear combinations of basic elements called *atoms*, which come from a redundant matrix called a *dictionary*. Redundant matrix means that we have more atoms than the size of the corresponding signal.
The Sparseland model became one of the most important modelizations in image processing, and more generally in signal processing. Its give state-of-the-art results in a wide variety of tasks: Denoising, inpainting, image separation, deblurring,...
There is one fundamental property on which this theory is based, the same property which allows us to compress a signal, to denoise, enhance a degraded signal, .... All meaningful data sources (i.e signal) are structured.
Each source of information: image, audio, video, 3D object (meshes or points cloud), text, financial time series, data on a graph,...have an inner structure that is unique to them which can be characterized in various ways and allow redundancy.

The first signs of *Sparseland* appear in the 1990s with the greedy and relaxed pursuit algorithm [7, 3] and the introduction of dictionary learning [8].

### 2.1.1 The idea behind *Sparseland*

The first step in *Sparseland* is to find (i.e learn) a dictionary with a relevant set of atoms (this step is called **Dictionary Learning**). Then the model assumption is that every incoming signal could be represented as a linear combination of only a few atoms from the dictionary (this step is called **Sparse Coding**).

> **Definition:** Sparse
>
> *Occurring at widely spaced intervals, not thick or dense.*

Unlike decomposition based on principal component analysis, *Sparseland* does not impose that learned representation to be orthogonal, allowing more flexibility to adapt the representation to the signal.
The idea of finding new representations of an input signal is not new. Signal processing's scientists already use this approach with predefined dictionaries instead of a learned one, for example, wavelets transform or Fourier transform. With a learned dictionary, we can expect to get more relevant information about the inner structure of the signal. For that reason, we can expect better performances in a wide range of applications.

## 2.2 Mathematical formulation

Consider a signal $x_i \in \mathbb{R}^m$ in a set of signal $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{m \times n}$ ( generally $n$ is large and $m$ is small: $n >> m$), the aim of this method is to find a linear combination of overcomplete basis elements

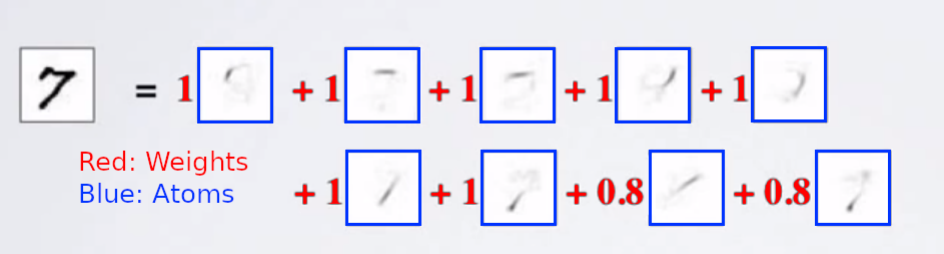


Figure 2.1: Example of Sparse Coding reconstruction

$D = [d_1, d_2, \ldots, d_k]$ under sparsity constraints which reconstruct the input signal (see an example at figure 2.1), overcomplete dictionary mean that $k > m$:

$$x_i = D\gamma_i$$

With $\gamma_i$ the sparse coefficients of the sparse decomposition for the signal $x_i$

The traditional way to get $\gamma_i$ is to optimize the empirical cost function.

$$f_n(D) = \frac{1}{n}\sum_{i=1}^{n} l(x_i, D)$$

Where $l$ is the cost function which is small where D is good for representing the signal. Usually, we use :

$$l(x_i, D) = \min_{\gamma_i} \frac{1}{2}\underbrace{\|x_i - D\ px\gamma_i\|_2^2}_{Squared\ error} + \lambda \underbrace{\|\gamma_i\|_0}_{Sparsity\ term}$$

Note that the cost function has two terms, which fight each other. The first term corresponds to the squared error, while that the second term corresponding to the sparsity of the weight vector. $\lambda$ is a positive constant which controls the importance of the sparse term relative to the square error term.
The norm of all columns of D must be equal or less than 1 , otherwise, D could grow big while $\gamma$ becomes small to satisfy the prior.
This problem is also known as a *basic pursuit* or the *Lasso*. However, it is important to notice that the problem isn't convex and his optimization is NP-complete (due to the $l_0$ norm). Fortunately, it is well known that for kind of problem use $l_1$ norm instead of $l_0$ norm yields a sparse solution for $\gamma_i$.
Then the optimization problem can be rewritten as:

$$\min_D \frac{1}{n}\sum_{i=1}^{n} \min_{\gamma_i} \frac{1}{2}\|x_i - D\ px\gamma_i\|_2^2 + \lambda\|\gamma_i\|_1$$

Before presenting our experimentations we will explain how traditional Sparse Coding works. In particular, how this method learns redundant properties and patterns in a signal set. This is the learning phase.

## 2.3 Learning step

A natural approach to solve this optimization problem is to alternate between the optimization of $D$ and $\gamma$, minimize over one while keeping the other one fixed.

**Algorithm 1** Learning step

**Require:** X the input signal
  $D_0$ initilized randomly, $\gamma$ is a zeros matrix
  **while** D and $\gamma$ not converged **do**
    Fix D
    Find $\gamma$ /* *Sparse Coding step* */ (1)
    Fix $\gamma$
    Find D /**Dictionary Learning* */ (2)
  **end while**

### 2.3.1 Inference of Sparse code

We will focus here on the $\gamma$ calculation during the sparse coding step of learning algorithm (i.e. part (1) of algorithm 1). There are different ways to obtain the sparse coefficients: Some based on the $l_0$ norm, other on the $l_1$ norm.

#### $l_0$ norm based

Matching pursuit algorithm proposed in [7] intended to approximately solve the basic pursuit problem which is normally unacceptable in terms of calculation (if $D$ is large):

$$\min_{\gamma_i} \|\underbrace{x_i - D\ px\gamma_i}_{r}\|_2^2\ s.t.\ \|\gamma_i\|_0 \leq L$$

(with $r$ the residual)

Matching pursuit algorithm is a greedy algorithm which iteratively generates a sorted list of atom and weighting scalars that represent the sub-optimal solution. An improvement of this algorithm is called Orthogonal Matching Pursuit which Orthogonalizes all the chosen elements.

**Algorithm 2** Orthogonal Matching Pursuit Algorithm

**Require:** D, $x$
  $\Gamma =$
  **while** $\|\gamma\|_0 < L$ **do**
    /* *Pick the element that most reduces the objective* */
    $e \leftarrow \arg\min_{i \in \Gamma}\{\min_{\gamma}\|x - D_{\Gamma \cup \{e\}}\gamma\|_2^2\}$
    /* *Update the active set* */
    $\Gamma \leftarrow \Gamma \cup \{e\}$
    /* *Update $\gamma$ and the residual* */
    $\gamma \leftarrow (D_\Gamma^\intercal D_\Gamma)^{-1} D_\Gamma^\intercal x$
    $r \leftarrow x - D\gamma_\Gamma$
  **end while**

#### $l_1$ norm based

With this relaxation of the $l_0$ norm, we could use a gradient descent method to solve this optimization problem:

$$\Delta_\gamma l(x^{(t)}) = D^\intercal(D\gamma - x) + \lambda sign(\gamma)$$

However, it's well known that $l_1$ norm is not differentiable at 0, in this case, if $\gamma$ changes sign because of $l_1$ norm gradient then clamp to 0:

Gradient descent step for each $k_th$ element of $\gamma$:

$$\gamma_k = \gamma_k - \alpha(D_{.,k})^\intercal(D\gamma - x)$$

Clamp to 0:

$$\text{if } \text{sign}(\gamma_k) \neq \text{sign}(\gamma_k - \alpha\lambda \text{ sign}(\gamma_k)) \text{ then: } \gamma_k = 0$$
$$\text{else } \gamma_k = \gamma_k - \alpha\lambda \text{ sign}(\gamma_k)$$

This algorithm is called Iterative Shrinkage and Thresholding Algorithm:

**Algorithm 3** ISTA (Iterative Shrinkage and Thresholding Algorithm)

initialize $\gamma$
**while** $\gamma$ has not converged **do**
  **for all** $\gamma_k$ **in** $\gamma$ **do**
    $\gamma_k = \gamma_k - \alpha(D[:,k])^\intercal(D\gamma - x)$
    $\gamma_k = shrink(\gamma_k, \alpha\lambda)$
  **end for**
**end while**
**return** $\gamma$

Here **shrink(a,b)** = [..., sign($a_i$) max($|a_i| - b_i$, 0), ...]

### 2.3.2 Inference of Dictionary

A well learned Dictionary is the key for sparse coding method because if the dictionary is not well learned, the sparse coding step will have trouble in properly reconstructing the input data.
Here we expose only a few methods which aim to well learn a dictionary: The first two algorithms learn the dictionary D using the whole training set, unlike the third one wich learn D by using an iterative batch procedure. The last method is a well-known algorithm based on singular value decomposition.

**Algorithm 1: Gradient Descent**

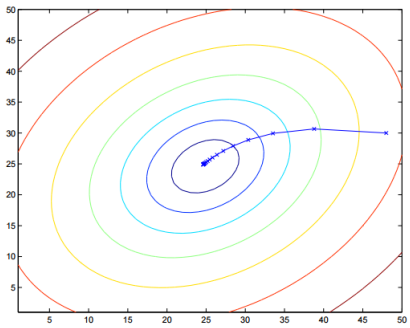


Figure 2.2: Example of gradient descent

To minimize Sparse Coding and dictionary learning's cost function we must use all our knowledge of mathematical optimization. There are many optimizations algorithm that can find an approximation which minimizes Sparse Coding and dictionary learning's cost function. Most famous one is the gradient descent, its simplicity of implementation and its results are no longer to be proved. It becomes an indispensable tool for Machine learning scientist.
The main idea is to find the local minimum of a cost function iteratively. At each step, the parameters are updating proportionally (this is step size, controlled by a learning rate $\lambda$) in the opposite direction of the gradient of the objective function (see figure 2.2).
As a reminder, our objective function is:

$$\min_D \frac{1}{n} \sum_{i=1}^{n} \min_\gamma \frac{1}{2}\|x_i - D\ px\gamma\|_2^2 + \lambda\|\gamma_i\|_1$$

In the dictionary learning step we assume that $\gamma$ is fixed and D variable. We can simplify our problem without all terms which do not depend on the dictionary D, let's define a function $f$ such that:

$$f(D) = \min_D \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \|x_i - D\ px\gamma_i\|_2^2$$

Then we can compute the gradient of $f(D)$:

$$\bigtriangledown f(D) = \frac{1}{n} \sum_{i=1}^{n} (x_i - D\gamma_i)\gamma_i^\intercal$$

**Algorithm 4** Dictionary Learning: Gradient descent

**Require:** x, $\gamma$, $\alpha$
  **while** D not converged **do**
    *// Perform gradient descent update of D*
    $D = D - \alpha * (X - D\gamma) * \gamma^\intercal$
    *// Renormalize columns of D*
    **for** each column j of D **do**
      $D[:, j] = \frac{D[:,j]}{\|D[:,j]\|}$
    **end for**
  **end while**
  **return** D

**Algorithm 2: Block-coordinate descent**

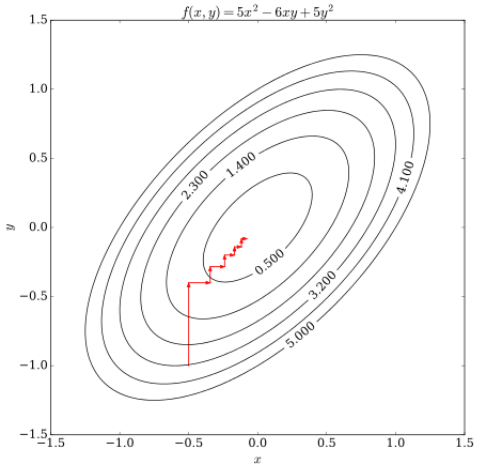


Figure 2.3: Example of Block-Coordinate descent

To avoid the learning rate there are other algorithms that can be applied instead of gradient descent. One of them is called Block-coordinate descent. The idea is to find the minimum of our objective's function for each direction in a cycle, in this case, a direction is a column of $D_{.,j}$ (see figure 2.3). Firstly, we have to set the gradient of $D_{.,j}$ to zero.

$$\frac{1}{n} \sum_{i=1}^{n} (x_i - D\gamma_i)\ \gamma_{[i,j]} = 0$$

We separe $D_{.,j}$ from the rest of D:

$$\iff 0 = \frac{1}{n} \sum_{i=1}^{n} (x_i - (\sum_{l \neq j} D_{.,l}\ \gamma_{[i,l]}\ ) - (D_{.,j}\ \gamma_{[i,j]})\ )\ \gamma_{[i,j]}$$

Our aim is to find the value of $D_{.,j}$, we must isolate $D_{.,j}$ :

$$\iff 0 = \frac{1}{n} \sum_{i=1}^{n} (x_i \gamma_{[i,j]} - (\sum_{l \neq j} D_{.,l}\ \gamma_{[i,l]}\ \gamma_{[i,j]}) - (D_{.,j}\ \gamma_{[i,j]}^2))$$

$$\iff 0 = (\sum_{i=1}^{n}(x_i\gamma_{[i,j]} - (\sum_{l\neq j} D_{.,l}\ \gamma_{[i,l]}\ \gamma_{[i,j]})\ ) - (\sum_{i=1}^{n}(D_{.,j}\ \gamma_{[i,j]}^2)))$$

$$\iff \sum_{i=1}^{n}(D_{.,j}\ \gamma_{[i,j]}^2) = \sum_{i=1}^{n}(x_i\gamma_{[i,j]} - (\sum_{l\neq j} D_{.,l}\ \gamma_{[i,l]}\ \gamma_{[i,j]})\ )$$

$$\iff D_{.,j}\sum_{i=1}^{n}\gamma_{[i,j]}^2 = \sum_{i=1}^{n}(x_i\gamma_{[i,j]} - (\sum_{l\neq j} D_{.,l}\ \gamma_{[i,l]}\ \gamma_{[i,j]})\ )$$

$$\iff D_{.,j} = \frac{1}{\sum_{i=1}^{n}\gamma_{[i,j]}^2}\sum_{i=1}^{n}(x_i\gamma_{[i,j]} - (\sum_{l\neq j} D_{.,l}\ \gamma_{[i,l]}\ \gamma_{[i,j]})\ )$$

$$\iff D_{.,j} = \underbrace{\frac{1}{\sum_{i=1}^{n}\gamma_{[i,j]}^2}}_{A_{j,j}}\underbrace{\sum_{i=1}^{n}(x_i\gamma_{[i,j]})}_{B_{.,j}} - \sum_{l\neq j} D_{.,l}(\underbrace{\sum_{i=1}^{n}\gamma_{[i,l]}\alpha_{[i,j]}}_{A_{i,j}})$$

$$D_{.,j} = \frac{1}{A_{j,j}}(B_{.,j} - DA_{.,j} + D_{.,j}A_{j,j})$$

**Algorithm 5** Dictionary Learning: Block-coordinate descent

**Require:** X, $\gamma$
  **while** D not converged **do**
    **for** each column j of D **do**
      *// For each column D[:,j] perform update*
      $D[:,j] = \frac{1}{A[j,j]}(B[:,j] - DA[:,j] + D[:,j]A[j,j])$
      *// Normalization*
      $D[:,j] = \frac{D[:,j]}{\|D[:,j]\|}$
    **end for**
  **end while**
  **return** D

**Algorithm 3: Online learning algorithm**

Today with the improvement of datasets size, it is impossible to train the dictionary over the entire dataset. To address this problem, machine learning scientist uses the online learning methods. Instead of learning on the entire dataset its update the model for each sample. In our case, we will update the dictionary after visiting each $x_i$. Mairal proposed [5] this online approach for the dictionary learning :

**Algorithm 6** Dictionary Learning: Online learning algorithm

**Require:** X, $\alpha$ (learning rate), T (number of iterations)
  $A = 0$, $B = 0$ (reset the "past information"‘)
  Initialize $D$ randomly (not to 0)
  **for** $t = 1$ to T **do**
    Infer code $\gamma$ from X
    $A = A + \gamma\gamma^\intercal$
    $B = B + X\gamma^\intercal$
    **for** $i = 1$ to n **do**
      **for** each column j of D **do**
        $D[:,j] = \frac{1}{A[j,j]} * (B[:,j] - DA[:,j] + D[:,j]A[j,j])$
        *// Normalization*
        $D[:,j] = \frac{D[:,j]}{\|D[:,j]\|}$
      **end for**
    **end for**
  **end for**
  **return** D

**Optimizing the Algorithm** In practice, it's possible to improve the convergence speed of this algorithm by using a Mini-batch extension: By drawing $\eta > 1$ signals at each iteration instead of a single one. Thus we have:

$$\begin{cases} A_t = \beta A_{t-1} + \sum_{i=1}^{\eta} \alpha_{t,i}\alpha_{t,i}^T \\ B_t = \beta B_{t-1} + \sum_{i=1}^{\eta} x\alpha_{t,i}^T \end{cases}$$

With $\beta = \frac{\theta+1-\eta}{\theta+1}$, where $\theta = t\eta$ if $t < \eta$ and $\eta^2 + t - \eta$ if $t \geq \eta$.

**Algorithm 4: K-SVD**

K-SVD is an algorithm proposed by Aharon, Elad, and Bruckstein [1] that generalizes the K-mean clustering algorithm via a singular value decomposition (SVD) approach:

**Algorithm 7** K-SVD algorithm

**Require:** X
  Initialize D randomly
  Ifer code $\gamma$ from X
  **for** each column j of D **do**
    $GammaActive = \emptyset$
    $ActiveSet = \emptyset$
    $ErrorActiveSet = \emptyset$
    **for** i=1 to n **do**
      **if** $\gamma_{[i,j]} \neq 0$ **then**
        $GammaActive = GammaActive \bigcup gamma_{[i,j]}$
        $ActiveSet = ActiveSet \bigcup X_i$
        $temp = X_i$
        **for** each $l$ such that $\gamma_{i,l} \neq 0$ and $l \neq j$ **do**
          $temp = temp - D[:, l]$
        **end for**
        $ErrorActiveSet = ErrorActiveSet \bigcup temp$
      **end if**
    **end for**
    $D[:, j] = \min_{D[:,j]} \|ErrorActiveSet - D[:, j] * GammaActive)\|_2^2$ $\Rightarrow$ This can be done by using SVD.
  **end for**
  **return** D

## 2.4 Dataset and Toolbox

### 2.4.1 MNIST

Firstly, we want to check if the previously given algorithms work for a simple database. At the beginning of this internship, we had to rewrite a prototype with the full Sparse Coding pipeline (i.e. Algorithm 1 and 3 to 6). The goal here was to understand the underlying principles behind Sparse Coding.
To test this prototype and all our models, we have chosen the MNIST database, which is widely used in machine learning field. The MNIST database is composed of handwritten digits, available from Yann Lecun's website[1]. MNIST has a training set of 50,000 examples and a test set of 10,000 examples (see figure 2.4). This is a subset of a larger dataset: NIST.
The digits have been size-normalized and centered in a fixed size of $28 \times 28$ pixels.
Generally having a model working on MNIST is the first step before to test this model on other types of signal, such as speech.

**That's why during this internship, we focused mainly on this dataset to test our models and the power of extracting discriminant features of Sparse Coding and Autoencoders and not directly on Speech.**

Figure 2.4: Example of MNIST's handwritten digits

The prototype, written in Python 3.6, computes Sparse Coding on a subset of the MNIST dataset to save time. This prototype is available on our GitHub repository [2] under the name: `SparseCoding.py`. However, this prototype is not capable of training on all MNIST dataset, as we only used simple, non-optimized algorithms. In order to use the entire MNIST dataset, we used a toolbox called SPAMS that already includes some optimized algorithms for Sparse Coding.

#### SPAMS

SPAMS[3](**SPA**rse **M**odeling **S**oftware) is an optimization toolbox for solving various sparse estimation problems.

- Dictionary learning and matrix factorization (NMF, sparse PCA, ...)
- Solving sparse decomposition problems with LARS, coordinate descent, OMP, SOMP, proximal methods
- Solving structured sparse decomposition problems (l1/l2, l1/linf, sparse group lasso, tree-structured regularization, structured sparsity with overlapping groups,...).

It is developed and maintained by J. Mairal (Inria), and contains sparse estimation methods resulting from collaborations with: F. Bach, J. Ponce, G. Sapiro, R. Jenatton and G. Obozinski, …

[1] `http://yann.lecun.com/exdb/mnist/index.html`
[2] `https://github.com/Usanter/SparseCoding`
[3] `http://spams-devel.gforge.inria.fr/`

## 2.5 Experimentation details

### 2.5.1 Application of traditional Coding on MNIST

Here we use SPAMS toolbox, with algorithm 2 as Sparse Coding step and algorithm 6 for dictionary learning on the MNIST dataset. The choice of lambda (which influence the importance of the sparse term in the optimization) is given by [5] such that : $\lambda = \frac{1.2}{\sqrt{m}}$.

Traditionally, during the learning of an overcomplete (or redundant) dictionary, the number of atoms

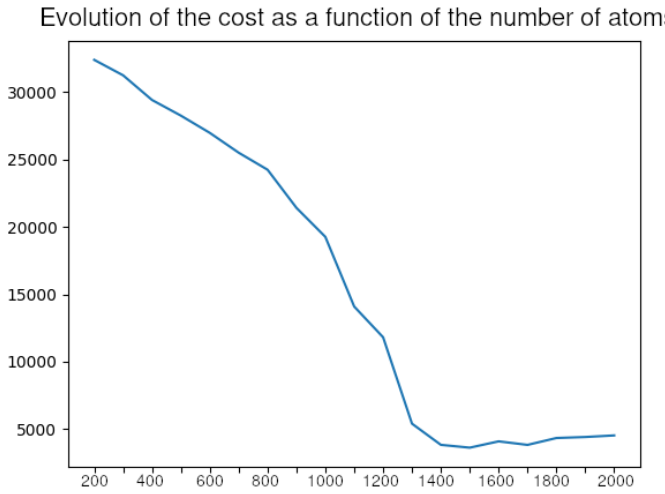


Figure 2.5: Cost's evolution depends on the number of atoms in the dictionary

$k$ so that $k \geq 2 *$ Signal's size.
All samples in the MNIST database have a size of 784 ( $28 \times 28$). Then, we compute for different values of k the cost function. We have obtained the cost function evolution. As expected the minimum of this cost function is reached when the number of atoms is more than 1400 atoms (see figure 2.5). This confirms the traditional assumption of the number of chosen atoms.

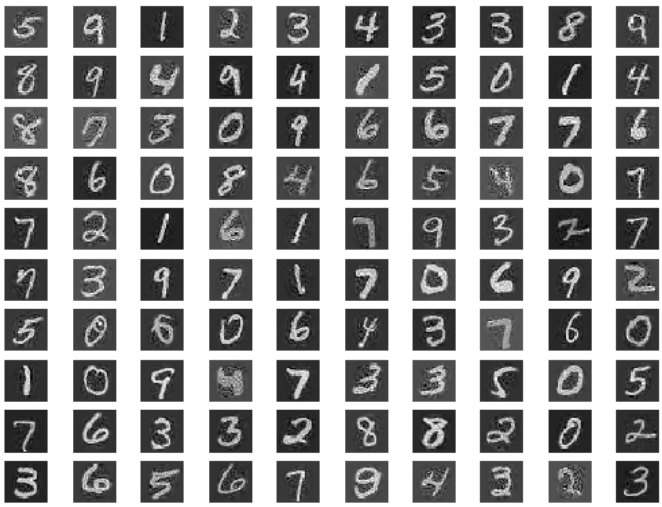

Figure 2.6: Some of the 1400 atoms in the dictionary

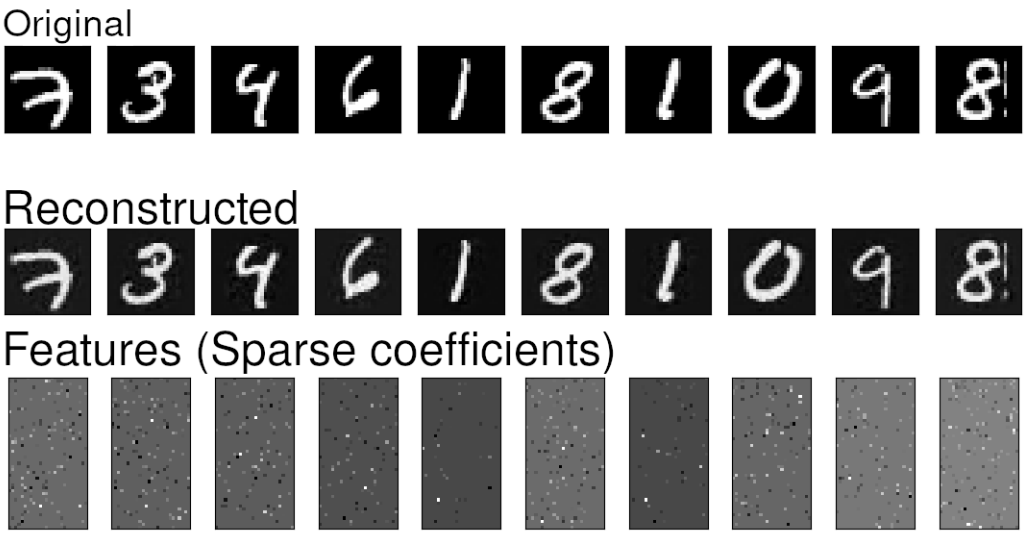


Figure 2.7: Reconstruction using $D\gamma$ and $\gamma$

The results obtained show that the handwritten numbers have a similar structure (for the same handwritten number).Most of the atoms in the dictionary (figure 2.6) directly resemble a handwritten digit (with little noise for some of them). Notice that the reconstruction looks good as shown in figure 2.7 .

### 2.5.2 Classification for traditional Sparse Coding

In the internship's context, we always tried two different classifiers on our data: SVM and K-means. SVM is a supervised classifier, while K-mean is an unsupervised one. The goal here is to highlight the discriminatory power of Sparse coding through a supervised and unsupervised manner.
For the traditional Sparse coding, the obtained accuracies are:

- SVM accuracy: 0.90
- K-means accuracy: 0.1319

As we can see, we have good results for the SVM classifier but very bad results for the K-means, if we take the sparse coefficients in an unsupervised manner, the sparse coefficients are not discriminative.

Whereas the previous tests seem to have a good result in reconstruction, one question appears:

*What makes us confident about the fact that two signals with the same label (for example two handwritten number 3 ) have close sparse coefficients representation $\gamma$ after the Sparse Coding step?*

In fact, nothing guarantees this assumption. The signal processing community asked themselves about this question. To ensure to have discriminative sparse code representation they created an extension of traditional Sparse Coding called Discriminative Sparse coding.

# Chapter 3

# Discriminative Sparse Coding

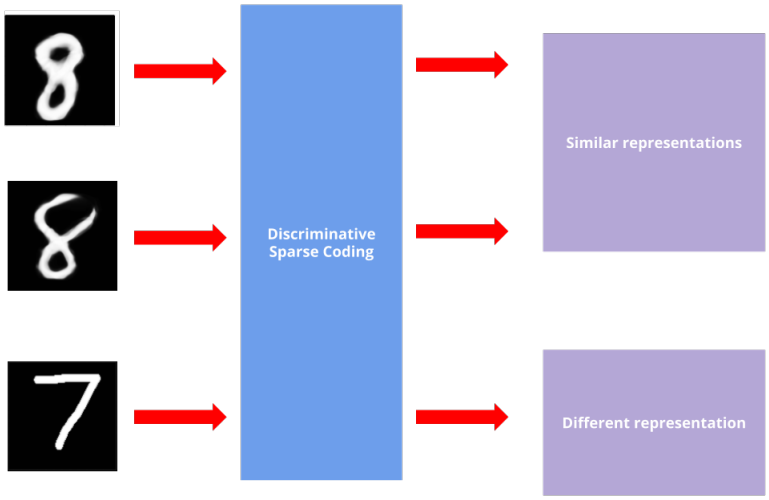


Figure 3.1: Objective of discriminative Sparse Coding

Generally, to get a discriminative sparse code, the first need is a discriminative dictionary. Indeed, during the sparse coding step, if the dictionary is discriminative, the minimization of the lost function will tend to create the sparse code using the right atoms and be disadvantaged for the others atoms in term of reconstruction. The result will be a discriminative sparse code.
However, most methods of discriminative sparse coding are not unsupervised, because they use the label information during the training of the dictionary. [16] enumerate some of these methods:

- In presence of label:
  1. Learn one dictionary per class
  2. Prune large dictionaries
  3. Jointly learn dictionary and classifier
  4. Embed class label into the learning of sparse coefficients
  5. Learn a histogram of dictionary elements over signal constituents
- For weakly supervised:
  Several max margin-based, non-convolutive, synthesis dictionary learning approaches

For the extraction of our features, we are only interested in two methods. We made this choice because both methods are state-of-the-art of discriminative Sparse Coding for feature's extraction (for classification purposes) and because they are simple to implement. We have chosen:

- One Dictionary per class
- Embed class label into the learning of sparse coefficients, through a method named *Label constitent K-SVD*.

## 3.1 One dictionary per class

Suppose we have $G$ classes. When we want to create a discriminative dictionary for each class from our dataset, the simple way to do this is to learn $G$ dictionaries, one for each $G$ class. Indeed, we can hope that during the training, each dictionary will learn all specificities from all data of its associated class. Mathematically, this method is similar to the traditional sparse coding, except that we don't form a unique dictionary on all data, but we form $G$ dictionaries on data of the same label as the dictionary.

In addition, to ensure that the dictionaries learn specific pieces of information of their classes, [10] proposed to add to the traditional cost function a new term $\mathcal{Q}(D_i, D_j)$ that promotes incoherence between the different dictionaries, i.e. this term encourages dictionaries to be more independent as possible :

$$\min_{D_i, C_i} \sum_{i=1}^{G} \sum_{x_j \in C_i} \|x_j - D_i\gamma_j\|_2^2 + \lambda\|\gamma_j\|_0 + \eta \sum_{i \neq j} \mathcal{Q}(D_i, D_j)$$

For example, it is possible to take $\mathcal{Q}(D_i, D_j) = \|D_i^T D_j\|_F^2$ with F denotes Frobenius norm.
Using this method of one dictionary per class to predict the class of an entry signal, we only have to find which dictionary minimizes the cost function for this signal, and the associated dictionary class will be the prediction.
This method is effective when the number of classes is small. However, when the number of classes is large, this method will be time and memory consuming.
In the particular case of speech recognition, the number of phonemes (in the case of the French language) is 36. We concluded that we cannot use this method of discriminative sparse coding. That is why we present the label consistent sparse coding, another state-of-the-art method.

## 3.2 Label Consistent K-SVD

[4] proposed a supervised learning algorithm to learn a compact and discriminative dictionary for sparse coding. They add a new label consistency constraint ( "discriminative sparse-code error") and combining reconstruction error and classification error to form a unified objective function that can be solved with the K-SVD algorithm.

### 3.2.1 Idea

In a supervised setting, during the training phase, all the information about an input signal is given, including its label. The idea is to use this information to learn a discriminative dictionary. To do this, [4] add a new consistency regularization term, a classification error and label consistency regularization term into the objective function:

$$< D, \gamma >= \arg\min_{D,\gamma} \|X - D\gamma\|_2^2 + \lambda\|\gamma\|_0 + \underbrace{\mu\|Q - AX\|_2^2}_{Label\ consistent\ term} + \underbrace{\beta\|H - WH\|_2^2}_{Consistency\ term}$$

Where Q is the discriminative sparse codes of input signals X for classification, A is a linear transformation matrix, W is the classifier parameters and H is the labels corresponding to the input signal X.
This optimization function is called LC-KSVD2. But there is a variant of this method, called LC-KSVD1 which is the same function without the consistency term.
In our setting, we will use LC-KSVD1 because we do not want to train a classifier at this stage.

### 3.2.2 LC-KSVD1

Therefore, the objective function for dictionary learning is defined as:

$$< D, A, \gamma >= \arg\min_{D,A,\gamma} \|X - D\gamma\|_2^2 + \mu\|Q - A\gamma\|_2^2 + \lambda\|\gamma\|_0$$

Where $\mu$ controls the contribution between reconstruction and label consistency regularization. $Q = [q_1, ..., q_N] \in \mathbb{R}^{K \times N}$.
Q is a discriminative sparse code corresponding to an input signal and the dictionary atoms, the nonzero values occur when $signal_i$ and $atom_i$ share the same label (figure 3.2).
Q is chosen by the user, in our setting we have chosen to match the same number of atoms for each class.

.

| Atom \ Signal | signal 1 | signal 2 | signal 3 | signal 4 | signal 5 | signal 6 |
|---|---|---|---|---|---|---|
| k1 | 0 | 1 | 0 | 1 | 0 | 0 |
| k2 | 0 | 1 | 0 | 1 | 0 | 0 |
| k3 | 1 | 0 | 1 | 0 | 0 | 1 |
| k5 | 1 | 0 | 1 | 0 | 0 | 1 |
| k6 | 0 | 0 | 0 | 0 | 1 | 0 |
| k7 | 0 | 0 | 0 | 0 | 1 | 0 |
| k8 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 3.2: Example of Q, each color corresponding to a class (white color is the lack of classes). Signal 1,3 and 6 are from class 1; signal 2 and 5 from class 2 and signal 5 from class 3

It is then possible to have "empty" atoms that are not discriminative (white color's atom in figure 3.2). To use K-SVD algorithm it is necessary to rewrite the previous equation:

$$< D, A, \gamma >= \underset{D,A,\gamma}{\arg\min} \| \begin{pmatrix} X \\ \sqrt{\mu}Q \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\mu}A \end{pmatrix} \gamma \|_2^2 \lambda \|\gamma\|_0$$

We define $X_{new} = \begin{pmatrix} X \\ \sqrt{\mu}Q \end{pmatrix}$ and $D_{new} = \begin{pmatrix} D \\ \sqrt{\mu}A \end{pmatrix}$.
It's important to note that the $D_{new}$ matrix is $L_2$ normalized. Now, it is possible to solve the minimization problem using the K-SVD algorithm:

$$< D, A, \gamma >= \underset{D,A,\gamma}{\arg\min} \|X_{new} - D_{new}\gamma\|_2^2 + \lambda\|\gamma\|_0$$

However, it is impossible to use directly D and A after employing the K-SVD algorithm. It is because D and A are $L_2$ normalized due to $D_{new}$. The desired $\hat{D}$ and $\hat{A}$ can be computed as follows:

$$\hat{D} = (\frac{d_1}{\|d1\|_2} \dots \frac{d_K}{\|d_K\|_2}) \text{ and } \hat{A} = (\frac{a_1}{\|d_1\|_2} \dots \frac{a_K}{\|d_K\|_2})$$

Label-Consistent K-SVD algorithm is given as follow:

**Algorithm 8** Label Consistent K-SVD 1 (LC-KSVD1)

**Require:** $X, Q, \lambda_1$
$D_0 = \text{KSVD}(X, \lambda_1)$
$A_0 = QX^\intercal(XX^\intercal + \lambda_2 I)^{-1}$
$\gamma_0 = \text{OMP}(D_0, X, lambda_1)$
$X_{new} = \begin{pmatrix} X \\ \sqrt{\mu}Q \end{pmatrix}$
$D_{new} = \begin{pmatrix} D \\ \sqrt{\mu}A \end{pmatrix}$
$D = \text{KSVD}(D_{new}, X_{new}, \lambda_1)$
$\hat{D} = (\frac{d_1}{\|d1\|_2} \dots \frac{d_K}{\|d_K\|_2})$
$\hat{A} = (\frac{a_1}{\|d_1\|_2} \dots \frac{a_K}{\|d_K\|_2})$
**return** $\hat{D}, \hat{A}$

## 3.3 Experiment details

### 3.3.1 Application of Label-Consistent K-SVD on MNIST

To test, we decided to use the same number of atoms as previously defined: 1024. Choosing the same number of atoms allows us to compare the results of traditional Sparse Coding with those of LC-KSVD. Besides, we decided to take a big $mu$ to promote discriminative power over reconstruction, we took $mu = 5.$.

Compared to traditional sparse coding, atoms in the Label-Consistent Sparse Coding dictionary look more like of small pencil strokes in the figure 3.3. Nevertheless, we can clearly recognize in some atoms a specific number.

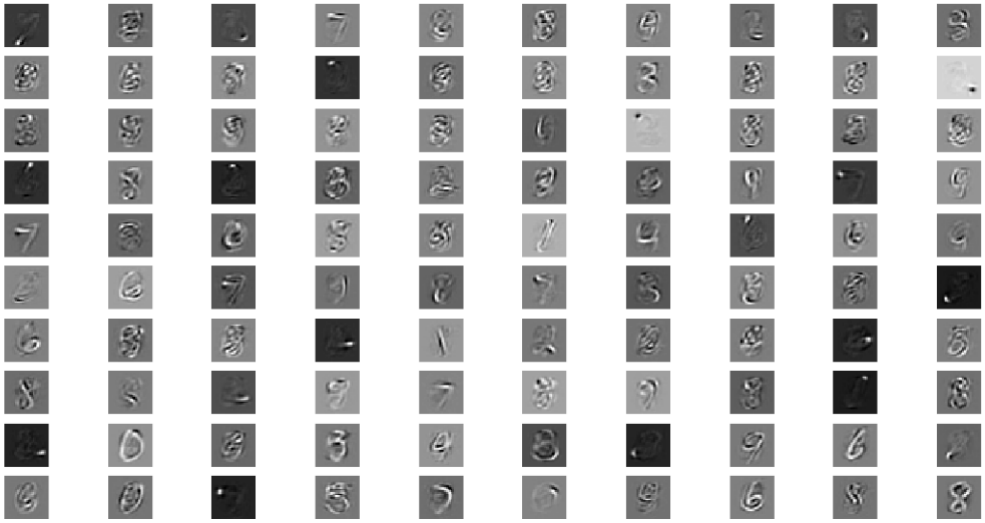

Figure 3.3: Some example of atoms from the dictionary

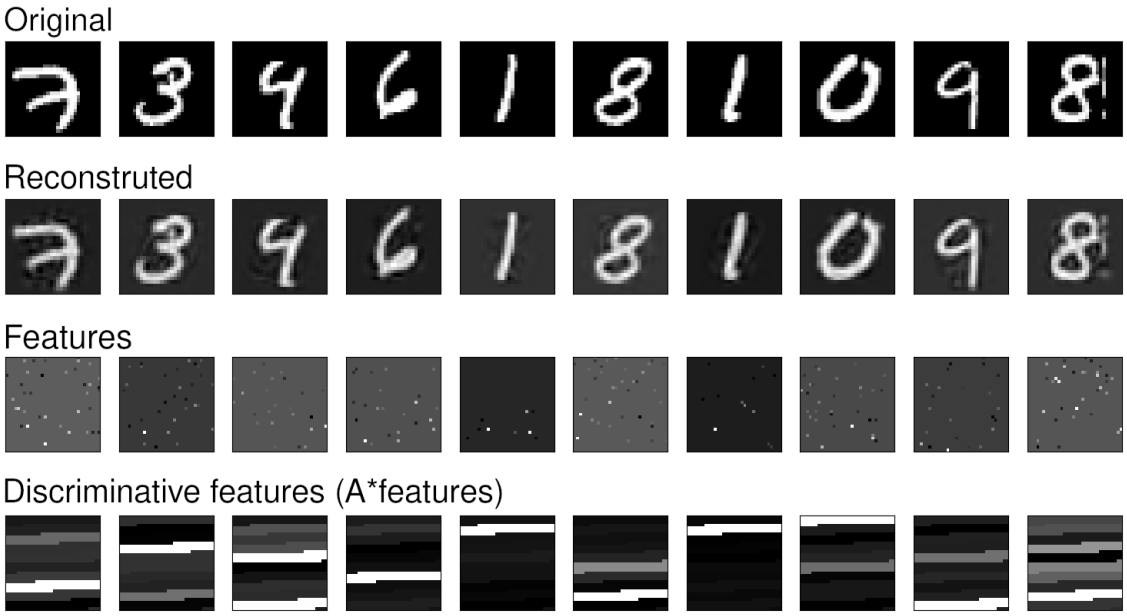


Figure 3.4: Examples of LC-KVD on MNIST data

Even if we encouraged discrimination more than reconstruction, we can see in the figure 3.4 that the reconstruction is not bad. In the same figure, we can see that $A * \gamma$ are discriminative: The strongest

white band always seem to be in the same location for the same label.
We can hope that the classification will be good, even for an unsupervised method like K-means.

### 3.3.2 Classification for Label-Consistent K-SVD

Our results of classification are the following:

- K-means accuracy: 0.78
- SVM accuracy: 0.91

Nevertheless, even if these results are good, they are applied on a non-shift-invariant dataset. If we use this method directly on speech, for example, we'll get bad results due to this shift-variance of patterns. That is why we used the convolution operation to deal with this problem. The method which merges Sparse Coding and convolution is called *Convolutional Sparse Coding*.

# Chapter 4

# Convolutional Sparse Coding

For the simplicity of reading, we will take some new notation convention during this chapter:

- Matrix: Uppercase $\mathbf{A}$
- Vector: Lower-case $\mathbf{a}$
- Scalar: Lower-case a
- 2D convolution operation : $*$
- $D \times D$ identity matrix: $I_D$
- Kronecker product : $\odot$
- $D \times D$ Fourier matrix : $\mathbf{F}$
- Hadamard product: $\otimes$
- inverse FFT : $\mathcal{F}^{-1}$
- Mapping function that preserve only $M \ll D$ active values relating to the small spatial structure of the estimated filter : $\mathcal{M}\{\}$

## 4.1 Idea

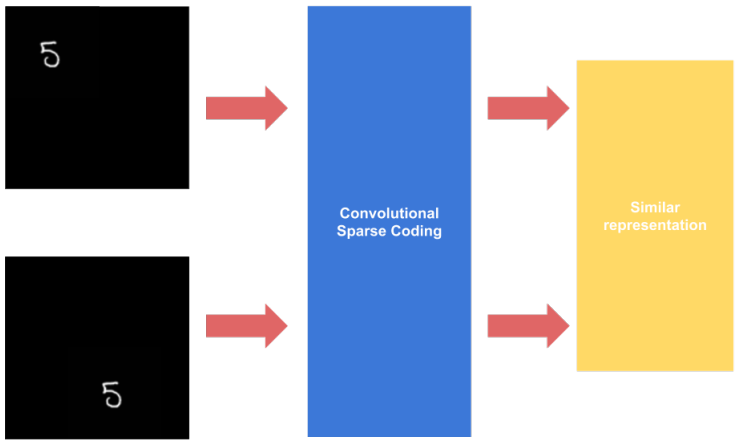


Figure 4.1: Idea behind Convolutional Sparse Coding

In signal processing, shift variance is an important factor to consider. Indeed, there are no guarantees that the searched pattern is at the same location in the signal.

Based on Sparse Coding, other algorithms like Efficient Shift-Invariant Dictionary learning [17] or Convolutional Sparse Coding [2] refers to the problem of finding a set of latent basis vectors that capture informative *local patterns* at different locations in the input sequence and not in the entire input sequence (example in figure 4.1).
Among these two methods, we choose to use Convolutional Sparse coding because of the Shift-invariant power of convolution.

### 4.1.1 Problem formulation

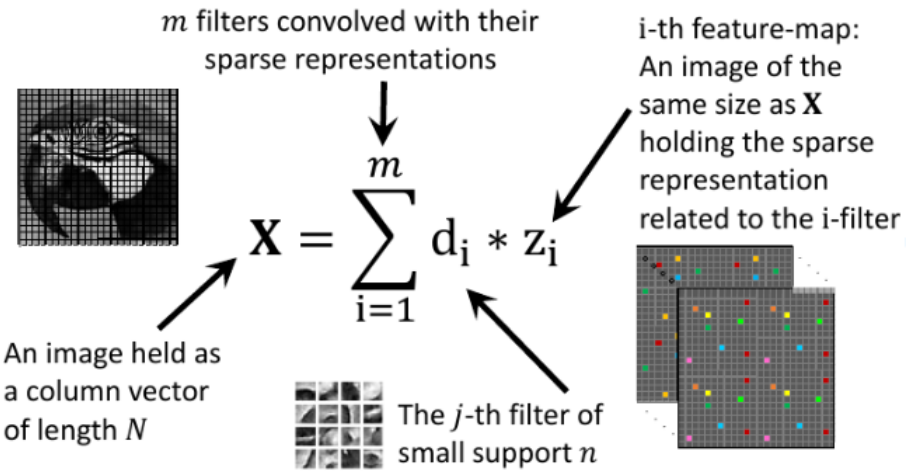


Figure 4.2: CSC principle by Michael Elad [1]

Instead of decomposing a signal as $x = D\gamma$, Convolutional Sparse Coding is the summation of convolutions between feature map $z$ (i.e. which refer to $\gamma$ in the traditional sparse coding) and the corresponding filter $d$ (i.e. our atoms in the dictionary in traditional Sparse Coding) in the dictionary $D$ (figures 4.2 and 4.3). So we have:

$$X = \sum_{i=1}^{m} d_i * Z_i$$

$$\arg\min_{d,z} \tfrac{1}{2}\|x - \sum_{k=1}^{K} d_k * z_k\|_2^2 + \beta\|z_k\|_1$$

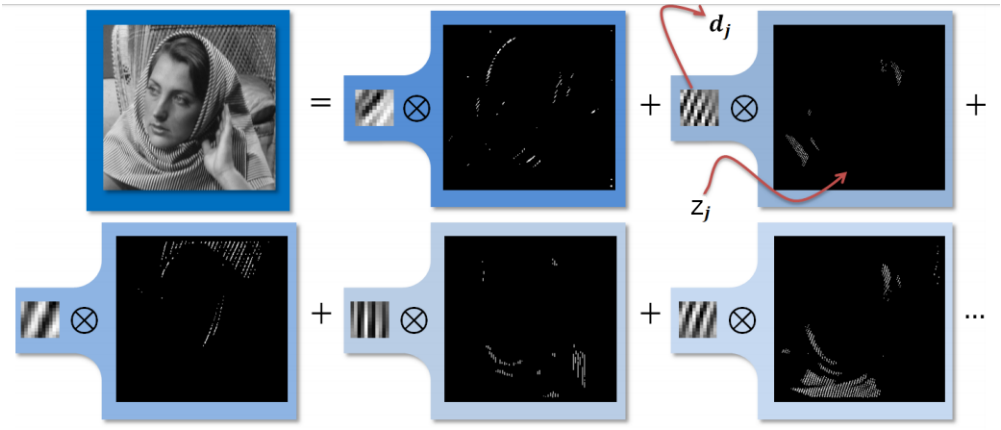


Figure 4.3: Example of reconstruction using atoms $d_j$ and features map $z_j$ [2]

[1] https://elad.cs.technion.ac.il/wp-content/uploads/2018/07/ICML_Talk_July_2018.pdf
[2] http://jsulam.cswp.cs.technion.ac.il/wp-content/uploads/sites/27/2017/12/Presentation_Handout.pdf

This new approach assumes that all input vectors X are independent of each other. But the complexity of convergence dramatically increases. Bristow [2] proposed an Augmented Lagrangian and the use of Alternative Direction Method of Multipliers (ADMM) to solve this problem. His approach is based on three key points:

- The use of ADMM
- The convolution subproblem can be solved efficiently (and explicitly) in the Fourier domain (instead of temporal domain)
- The use of quad-decomposition of the objective into four subproblems (which are convex)

In this approach, to solved a Convolutional Sparse Coding problem they introduced two auxiliary variables: **t** and **s** and posing the objective in the Fourier domain:

$$\begin{aligned} \underset{d,s,z,t}{\arg\min} \quad & \tfrac{1}{2D}\|\hat{x} - \textstyle\sum_{k=1}^{K} \hat{d}_k \odot \hat{z}_k\|_2^2 + \beta \textstyle\sum_{k=1}^{K} \|t_k\|_1 \\ \text{subject to} \quad & \|s_k\|_2^2 \leq 1 \text{ for k} = 1..\text{ K} \\ & s_k = \Phi^T \hat{d}_k \text{ for k} = 1..\text{K} \\ & z_k = t_k \text{ for k} = 1..\text{K} \end{aligned}$$

With $\Phi$ a D × M submatrix of the Fourier matrix $F = [\Phi, \Phi_\perp]$. In this formulation $\hat{d}_k$ is a D-dimensional vector whereas in the original formulation $d_k \in \mathbb{R}^M$ is of a significantly smaller dimensionality to $M \ll D$ corresponding to its smaller spatial support.

## 4.2 Inference of CSC

### 4.2.1 Augmented Lagrangian

We handle the introduction of new equality constraints through an augmented Lagrangian approach [2]:

$$\begin{aligned} \mathcal{L}(d, s, z, t, \lambda_s, \lambda_t) = & \\ & \tfrac{1}{2D}\|\hat{x} - \textstyle\sum_{k=1}^{K} \hat{d}_k \odot \hat{z}_k\|_2^2 + \beta\|t\|_1 \\ & + \lambda_s^T(s - [\Phi^T \otimes I_K]\hat{d}) + \lambda_t^T(z - t) \\ & + \tfrac{\mu_s}{2}\|s - [\Phi^T \otimes I_K]\hat{d}\|_2^2 \\ & + \tfrac{\mu_s}{2}\|z - t\|_2^2 \end{aligned}$$

### 4.2.2 Quad-decomposition of the objective

We decompose our objective into four convex subproblems:

**Subproblem z**

$$\begin{aligned} & z^* = \underset{z}{\arg\min}\ \mathcal{L}(z, d, s, t, \lambda_s, \lambda_t) \\ & = \mathcal{F}^{-1}\{\underset{z}{\arg\min} \tfrac{1}{2}\|\hat{x} - \hat{D}\hat{z}\| + \hat{\lambda}_t^T(\hat{z} - \hat{t}) + \tfrac{\mu_t}{2}\|\hat{z} - \hat{t}\|_2^2\} \\ & = \mathcal{F}^{-1}\{(\hat{D}^T\hat{D} + \mu_t I)^{-1}(\hat{D}^T\hat{x} + \mu\hat{t} - \hat{\lambda}_t)\} \end{aligned}$$

Where $\hat{D} = [diag(\hat{d}_1), ..., diag(\hat{d}_K)]$

**Subproblem t**

$$\begin{aligned} & t^* = \underset{t}{\arg\min}\ \mathcal{L}(t, d, s, z, \lambda_s, \lambda_t) \\ & = \underset{t}{\arg\min} \tfrac{\mu_t}{2}\|z - t\|_2^2 + \lambda_t^T(z - t) + \beta\|t\|_1 \end{aligned}$$

Unlike subproblem z, the solution to $t$ cannot be efficiently computed in the Fourier domain (since $L_1$ norm is not rotation invariant). Solving $t$ requires projecting $\hat{z}$ and $\hat{\lambda}_t$ back into the spatial domain. If this equation does not contain any rotations of the data, each element of $t$ can be solved independently:

$$t^* = \underset{t}{\arg\min} \beta|t| + \lambda_t(z - t) + \tfrac{\mu}{2}(z - t)^2$$

Where the optimal value of each $t$ can be found using shrinkage function:

$t^* = sign(z + \frac{\lambda_t}{\mu_t}).max\{|z + \frac{\lambda_t}{\mu_t}| - t, 0\}$

**Subproblem d**

$d^* = \arg\min_s \mathcal{L}(d, s, z, t, \lambda_s, \lambda_t)$
$= \mathcal{F}^{-1}\{\arg\min_{\hat{d}} \frac{1}{2}\|\hat{x} - \hat{Z}\hat{d}\|_2^2 + \hat{\lambda}_s^T(\hat{d} - \hat{s}) + \frac{\mu}{2}\|\hat{d} - \hat{s}\|_2^2\}$
$= \mathcal{F}^{-1}\{(\hat{Z}^T\hat{Z} + \mu_s I)^{-1}(\hat{Z}^T\hat{x} + \mu_s\hat{s} - \hat{\lambda}_s)\}$

**Subproblem s**

$s^* = \arg\min_s \mathcal{L}(d, s, z, t, \lambda_s, \lambda_t)$
$= \arg\min_s \frac{\mu_s}{2}\|\hat{d} - [\Phi^T \otimes I_K]s\|_2^2 + \hat{\lambda}_s^T(\hat{d} - [\Phi^T \otimes I_K]s)$
Solving this equation as it is a quadratically constrained quadratic programming problem (QCQP). Due to the Kronecker product with the identity matrix $I_K$ it can be broken down into K independent problems:
$s_k^* = \arg\min_{s_k} \frac{\mu_s}{2}\|\hat{d}_k - \Phi^T s_k\|_2^2 + \hat{\lambda}_{sk}^T(\hat{d}_k - \Phi^T s_k)$
Further, since $\Phi$ is orthonormal projecting the optimal solution to the unconstrained problem cab be found efficiently through:

$$s^* = \begin{cases} \|\tilde{s}\|_2^{-1}\tilde{s}_k \text{ , if } \|\tilde{s}\|_2^{-1} \geq 1 \\ \tilde{s}_k \text{ otherwise} \end{cases}$$

where,

$$\tilde{s}_k = (\mu_s \Phi\Phi^T)^{-1}(\Phi\hat{d}_k + \Phi\hat{\lambda}_{sk})$$

Finally, the solution to this equation can be found using :
$\tilde{s}_k = \mathcal{M}\{\frac{1}{\mu_s}\sqrt{D}^{-1}(\mathcal{F}^{-1}\{\hat{d}_k\} + \mathcal{F}^{-1}\{\hat{\lambda}_{sk}\})\}$

### 4.2.3 Lagrange Multiplier Update

$$\lambda_t^{(i+1)} \leftarrow \lambda_t^{(i)} + \mu_t(z^{(i+1)-t^{(i+1)}})$$
$$\lambda_s^{(i+1)} \leftarrow \lambda_s^{(i+1)} + \mu_s(d^{(i+1)-s^{(i+1)}})$$

### 4.2.4 Penalty update

Convergence may be reached if $\mu^{(i)} \to \infty$:

$$\mu^{(i+1)} = \begin{cases} \tau\mu^{(i)} \text{ if } \mu^{(i)} < \mu_{max} \\ \mu^{(i)} \text{ otherwise} \end{cases}$$

## 4.3 SPORCO

SPORCO [3] (**SP**arse **O**ptimization **R**esearch **CO**de) is a Python package developed by B. Wohlberg for solving optimization problem with sparsity problem [15, 14]. These consist primarily of sparse coding and dictionary learning problems but there is also support for other problems such as:

- Convolutional sparse coding
- ADMM
- Iterative shrinkage

[3] https://sporco.readthedocs.io/en/latest/index.html

- Total variation regularisation
- Robust PCA
- ...

We will use Convolutional Sparse coding algorithm from this toolbox to confirm that CSC is shift-invariant.

## 4.4 Experimentation details

### 4.4.1 Application of Convolutional Sparse Coding on MNIST

In this section, we show the result of applying unsupervised Convolutional Sparse Coding (denoted as CSC) previously presented in MNIST dataset.
Here, it's useless to have more than $2 * Signal\_size$ not represent the number of atoms but its represent the number of filters, and we need fewer filters than atoms in the traditional Sparse Coding. Arbitrationally, we chose 64 filters of ($8 \times 8$), this is not recommended to use filters larger than ($7 \times 7$) for MNIST dataset [12] however in our case we want to get more discriminant filters, so using larger filters is the solution.

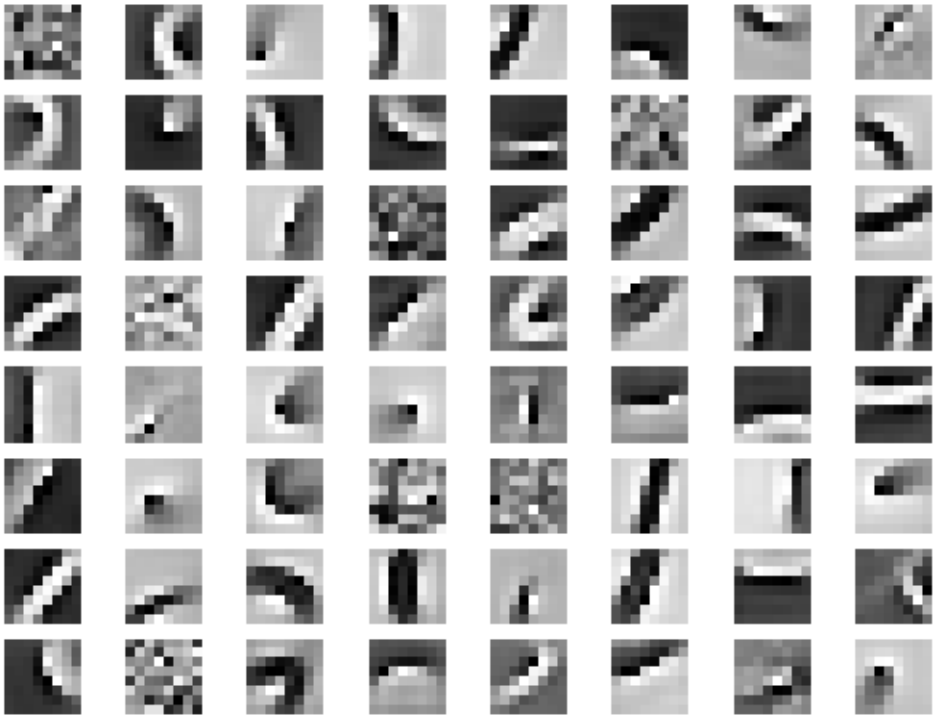

Figure 4.4: Filters from the convolutional dictionary

As explained above,larger is the filter's size, more complex the pattern is. In our filters, we learned different types of curves (see figure 4.4) that are more adapted to MNIST semantics. When we tried on smaller filter dimension we got Gabor-like filters.
In order to use the activation maps obtained to make the classification, we want more complex models as possible.
In case of figure 4.4, these curves are easy to find in some of our Latin/Western/Arabic numbers and are discriminative. For example, the second filter in figure 4.4 cannot be present for a handwritten 1.

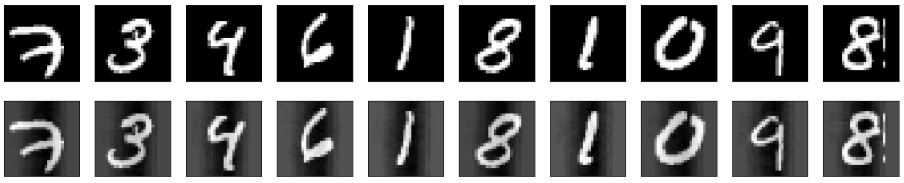

Figure 4.5: Examples of reconstruction using CSC

As expected, Convolutional Sparse Coding detects the patterns in the input signal using the Sparse activation maps and the convolutional dictionary, which allows a good reconstruction(figure 4.5). But, as we can see in figure 4.6 and 4.7 in the activation maps, we obtained new patterns which are shift variant.

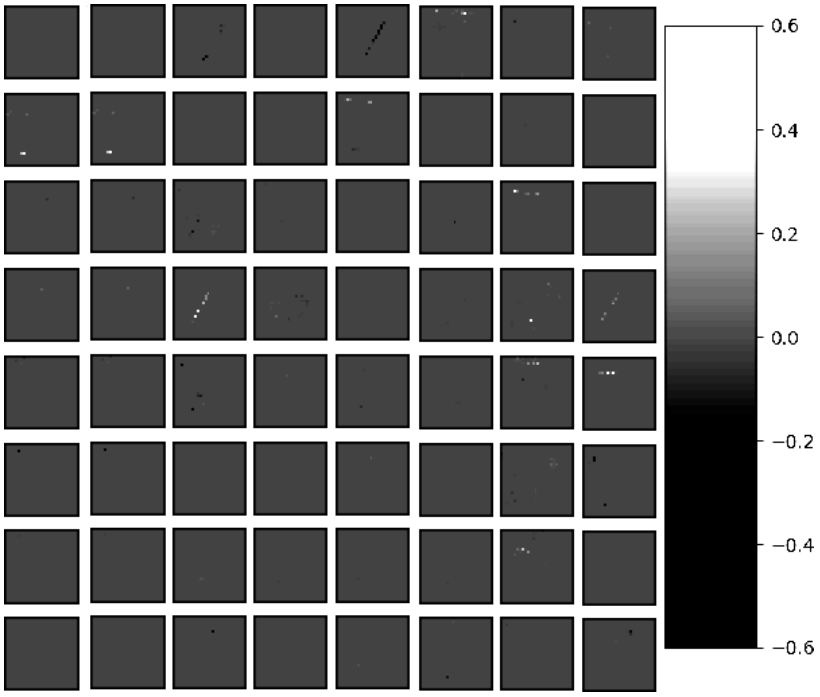


Figure 4.6: Activation maps for a handwritten "7"

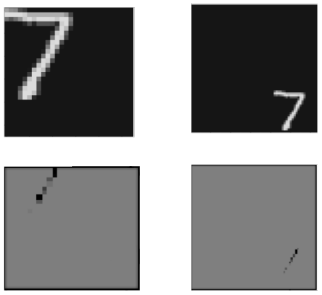

Figure 4.7: Example of shift-invariance power in a characteristic features map ($5^{th}$ atom in the CSC dictionary)

Then we cannot apply the classification directly to these activation maps. To solve this problem, we tried different methods:

- Max or Mean global pooling for each activation map:
  The objective was to create a unique activation card in which each pixel corresponds to the Max pooling of one of the 64 activation maps. But with these methods, we lose the pattern available in the activation maps, and consequently, discriminative pieces of information.

- Convolutional Sparse Coding on all activation maps:
  Apply on the activation maps a new convolutional sparse Coding. This method called Multi-Layer Convolutional Sparse Coding is described in the next chapter.

# Chapter 5

# Discriminative Convolutional Sparse Coding

## 5.1 ML-CSC

In order to get a discriminative Convolutional Sparse Coding, [13, 9] proposed to extend the CSC model to a Multi-Layers version (called ML-CSC). Indeed, adding a new CSC layer allows the algorithm to capture more complex patterns.
Add a new layer corresponds to get the features maps $Z$ obtained by CSC in the previous layers and use them like the input signal with a new dictionary, in other word $D_1$ (layer 1's dictionary) contain simple patterns, the convolution product $D_1 * D_2$ contains a linear combination of atoms from $D_1$, merging them to create more complex patterns, called molecules. Mathematically,for $L$ layers:

$$\begin{aligned} X &= D_1 * Z_1 && \text{Layer 1} \\ Z_1 &= D_2 * Z_2 && \text{Layer 2} \\ Z_2 &= D_3 * Z_3 && \text{Layer 3} \\ &\dots \\ Z_{L-1} &= D_L * z_L && \text{Layer L} \end{aligned}$$

Then X can be rewriten by:

$$X = D_1 * D_2 * D_3 * \cdots * D_L * z_L$$

Then it is possible to rewrite the dictionary learning algorithm to extend it to a multi-layer version:

**Algorithm 9** Multi-Layer Convolutional Dictionary Learning

**Require:** Training sample $\{x_1, x_2, \dots, x_k\}$, initial convolutional dictionaries $\{D_1, D_2, \dots, D_L\}$, the number of iterations for gradient descent T
  **for** $k = 1, \dots, k$ **do**
    Sparse Coding: $z_L \leftarrow \arg\min_{z_L} \|x_k - D_L z_L\|_2^2 + \lambda \|z_L\|_1$
    */* Update dictionaries*/*
    **for** $i = L, \dots, 1$ **do**
      **for** $t = 1, \dots, T$ **do**
        $D_i \leftarrow \mathcal{H}[D_i - \alpha((y_k - D_i Z_i) Z_i^T)]$
      **end for**
    **end for**
    **for** $t = 1, \dots, T$ **do**
      $D_1 \leftarrow D_i - \alpha((y_k - D_i Z_1) Z_1^T)$
    **end for**
  **end for**

Note that $D_1$ is not updated with the others dictionaries $D_i$, this is because its imposed that the other dictionaries $D_i$ are sparse to be sure that the feature maps $Z_i$ are also sparse. This is possible by applying the hard-thresholding operation $\mathcal{H}$ on them.

## 5.2 Tests details

For all our experimentations on ML-CSC we used a matlab demo writen by Jeremias Sulam [1] (writer of [13, 9]) that he provided us. We want to thank him for that.

### 5.2.1 Application of ML-CSC on MNIST

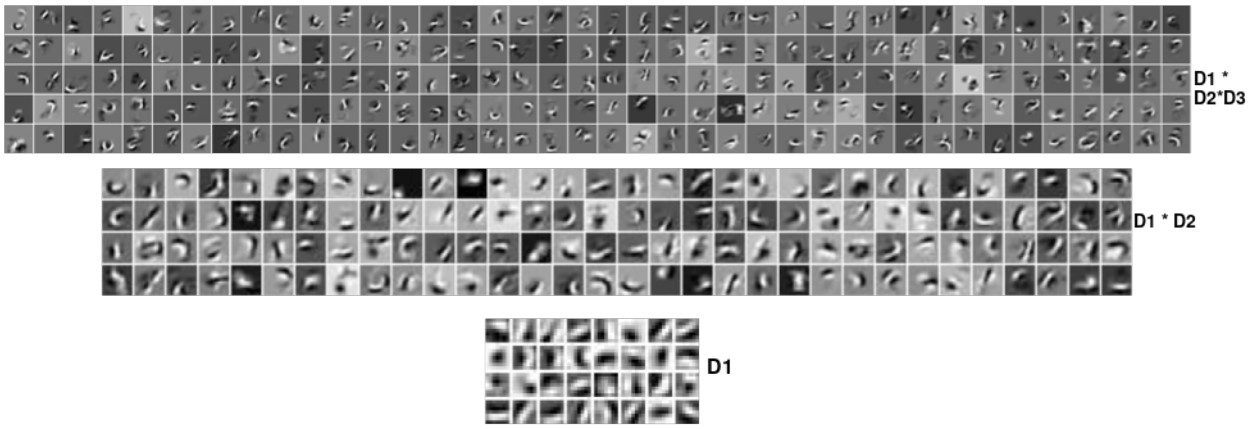


Figure 5.1: Dictionary's atoms at each layer

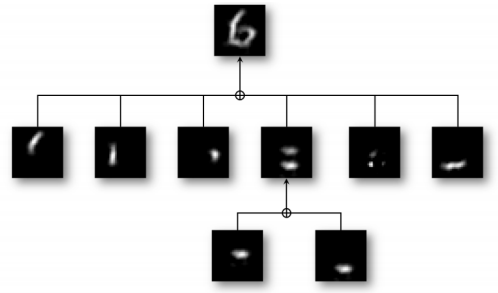

Figure 5.2: From atoms to molecules: Illustration of the ML-CSC model for a number 6. Two local convolutional atoms (bottom row) are combined to create slightly more complex structures – molecules – at the second level, which are then combined to create the global atom representing, in this case, a digit [13]

Here we see in figure 5.1the 3 layers of dictionaries with the learned forms. The deeper the dictionary got the more complex forms and therefore discriminative. In the figure 5.2 a reconstruction of a 6 is displayed.

### 5.2.2 Classification for ML-CSC

With the Multi-layer Convolutional Sparse Coding, if we try to classify the obtained features we got:

- SVM Accuracy = 93,67%
- K-means Accuracy = 11,46%

The problem here is that once again, we get bad results for K-means because the representations for the same classes can be different and the only way to solve it is to use the information of the labels that we have in the supervised setting. To address this problem we proposed a new method based on LC-KSVD.

[1] `https://sites.google.com/view/jsulam/`

## 5.3 LC-ML-CSC

SAs we used Label-Consistent Sparse Coding instead of traditional Sparse Coding to get more discriminating Sparse coefficients, we want to create a new convolutional Multi-Layer Sparse Coding model that adds a consistent Label term: We called it *Label Consistent Multi-Layer Convolutional Sparse coding* (LC-ML CSC).
We already explained what is the label consistent term and the multi-layers convolutional Sparse Coding, thus we can already present the LC-ML CSC algorithm:

**Algorithm 10** Labels Consistent Multi Layers Convolutional Sparse Coding (LC-MLCSC )

**Require:** Y: input data, Q: discriminative matrix
  **for** $k = 1, \ldots, K$ **do**
    $y_k \leftarrow$ The $k^{th}$ batch of Y
    /* *Sparse Coding* */
    $z_{kL} = \arg\min_{z_k} \underbrace{\|y_k - D^L z_k\|_2^2 + \alpha\|z_k\|_1}_{CSC\ normal} + \underbrace{\beta\|Q - A\check{a}z_k\|}_{LCcomposant}$
    $= z_k - \alpha(D^T(Dz_k - y_k) + \lambda(sign(z_k)) + \beta(A^T(Az_k - Q)))$
    /* *Dictionary Update* */
    **for** $i = L, \ldots, L$ **do**
      $D_i \leftarrow \mathcal{H}[D_i - \alpha((y_k - D_i z_k)z_k^T)]$
    **end for**
    /* *Update Matrix A* */
    $A \leftarrow A + \alpha(\beta(Q - Az_k)z_k^T)$
  **end for**
  **return** $z_k$, D, A

However, at the moment when we write these internship report the LC-ML CSC still a work in progress method. And we don't have a good result in classifying even with SVM. That why we will not present the results of this method here.

# Chapter 6

# Auto-encoders

## 6.1 Principle

The second feature extraction method we used during this internship was AutoEncoder [11]. Autoencoder (AE) is an unsupervised neural network algorithm that takes a signal as input and tries to reconstruct the same signal as the output. Generally, AE is composed of three parts (see figure 6.1 [1]):

- Encoder
- Latent Space Representation (Red box in figure 6.1)
- Decoder

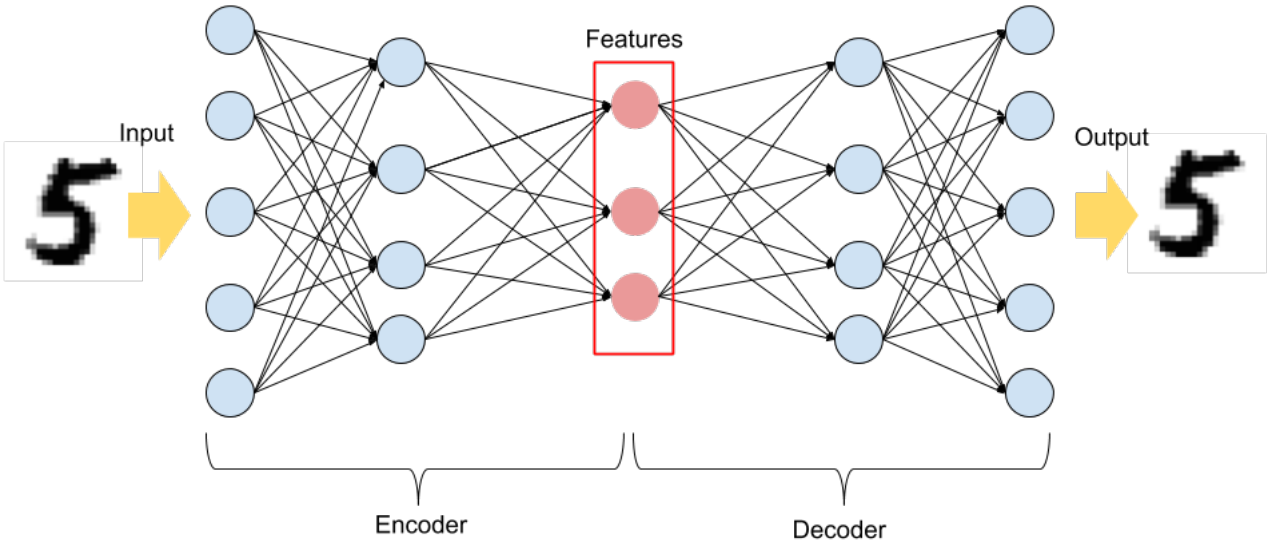


Figure 6.1: Basic architecture of AutoEncoder

Encoder and Decoder part work in the same way as a simple artificial neural network: a Feed Foward propagation algorithm to get the output from the input and Backward propagation of the error from the output to the input. The main difference with an artificial neural network is the Latent Space Representation which is typically a hidden layer located between the encoder and the decoder, with fewer neurons than any layers in Encoder or Decoder.
After the training, AutoEncoder can encode a certain set of data with less dimensionality than the input. It widely uses for feature extraction, segmentation, deconvolution, colorization, denoising,...
During the internship we tried two main architectures of AutoEncoder:

- Sparse AutoEncoder, which has approximately the same behavior the Sparse Coding.

[1] We take the convention in the figure that all circles can be anything such as Conv2D + Pooling, Dense, ...

- Label Consistent AutoEncoder, which is a model we proposed based on the idea of Label-Consistent Sparse Coding.

## 6.2 Sparse AutoEncoder

In the respect of *Sparseland* assumptions, [6] introduced the K-Sparse AutoEncoder. The goal is the same as Sparse Coding, we want to compute a sparse representation from an input signal but the method is different. Here we do not have the dictionary directly, we have an Encoder that finds the sparse representation for us using artificial neurons or convolution blocks.
To allows the Autoencoder to find this sparse representation we must add a term in the loss function. It is called a regularizer, which is applied to the last layer of our encoder. We decided to keep using the $l_1$ norm for this regularizer.
Unlike the traditional Sparse Autoencoder, we decided to use more hidden units at the end of the encoder (cf figure 6.2). With this addition, we create a discriminative representation instead of compressive representation.

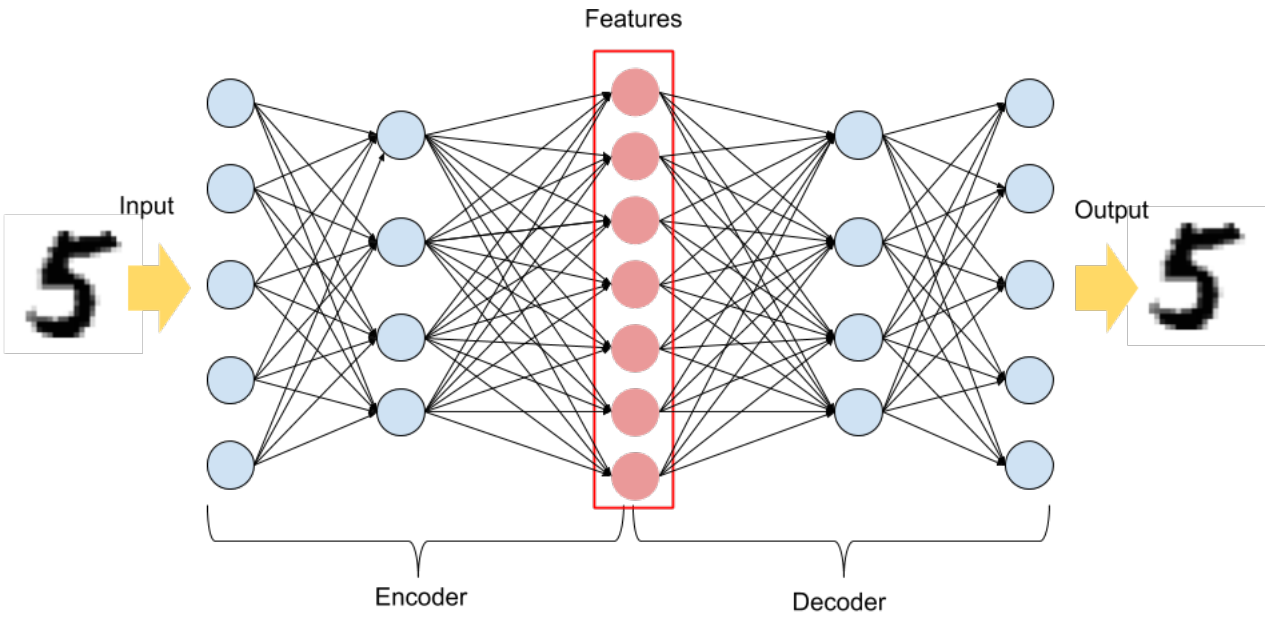


Figure 6.2: Example of Sparse AutoEncoder architecture

## 6.3 Labels Consistent AutoEncoder

We proposed a new Autoencoder model for extraction of discriminative features. For this model, we were inspired by the Label-Consistent Sparse Coding and especially the Q matrix. Our idea is to bring the information on the label by encouraging the use of certain filters for a class.
However, we also want to keep the reconstruction. To achieve these two objectives, we decided to create two parallel branches, one for each problem (we show an example of our model architecture in figure 6.3).
We called this model *Label Consistent AutoEncoder*. To fit with our previous studies (especially CSC's method) we also use convolution blocks and pooling blocks for encoding, convolution and upsampling for decoding and dense block for the label consistent term.
We also tested this model with the Sparsity constraint to see if it would improve the results. We called this model *Labels-Consistent Sparse Autoencoder*.
In addition, we train a A matrix in the same way as we did for $A_0$ in the LC-KSVD and multiply this

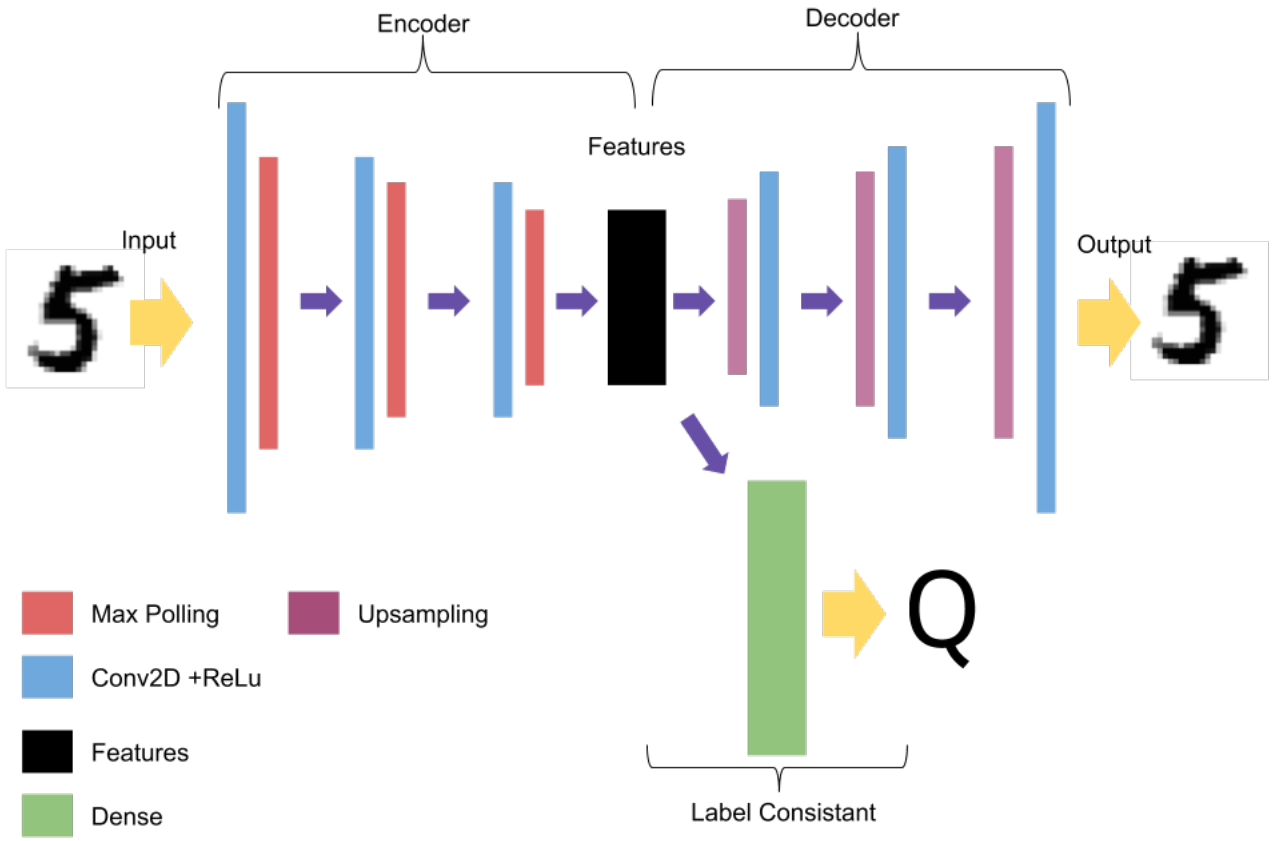


Figure 6.3: Example of Label Consistant Autoencoder architecture

A matrix with the features to get a more discriminative representation.

## 6.4 Experimentation details

### 6.4.1 Application of different Autoencoder on MNIST

**K Sparse Autoencoder**

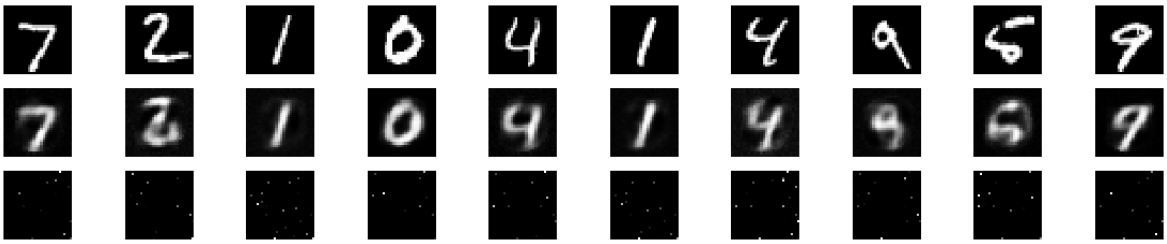

Figure 6.4: Reconstruction using decoder and encoded data

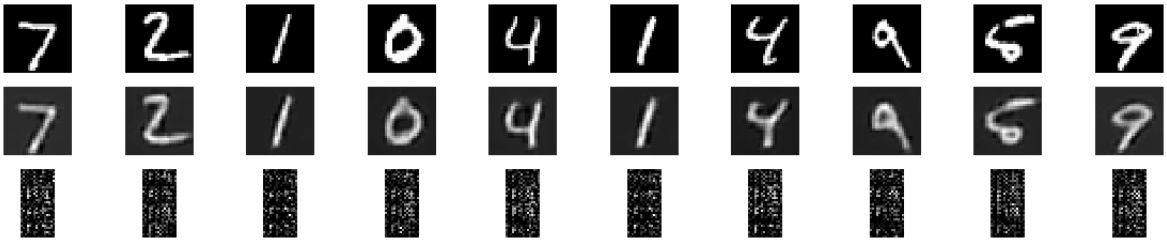

Figure 6.5: K Sparse Autoencoder using convolution, pooling, upsampling results

Here, we see that reconstruction is better in the context of convolution utilization (figure 6.5) rather than with dense (figure 6.2). This is explained by the fact that the method with Dense has only one layer whereas the method with convolution is deeper.

**Labels Consistent Autoencoder**

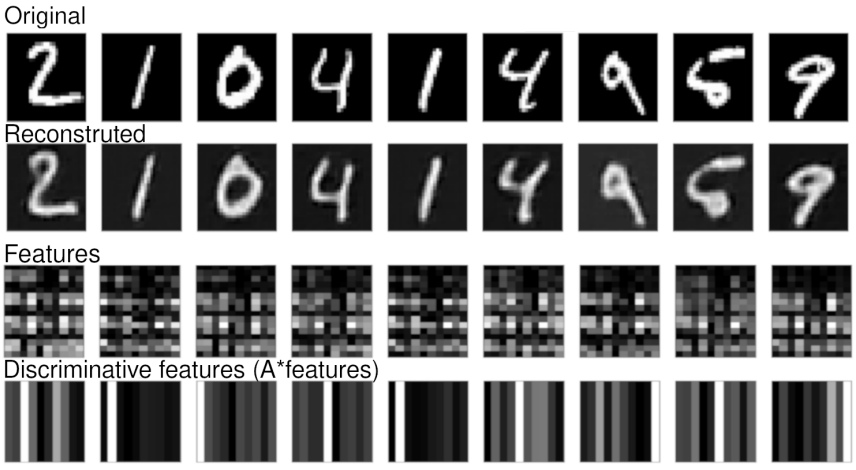


Figure 6.6: MNIST samples with the LC-Autoencoder

**Label Consistent Sparse Autoencoder**

In figure 6.6 and figure 6.7 we can see that there is a good reconstruction but especially that the discriminating features $A * Features$ are perfect for classification, indeed, there is a white band at a precise location depending on the number label.

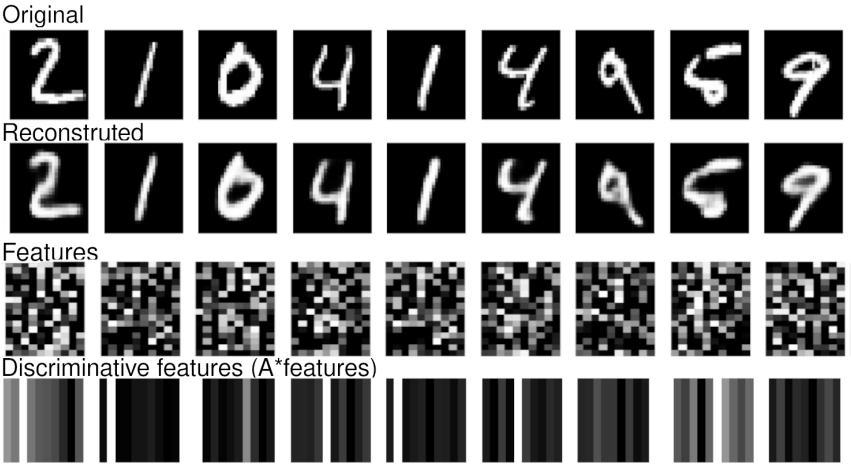


Figure 6.7: MNIST samples with the Sparse LC-Autoencoder

### 6.4.2 Classification for Autoencoder

| × | **SVM accuracy** | **K-means accuracy** |
|---|---|---|
| **K Sparse AE (Dense)** | 0.91 | 0.22 |
| **K Sparse AE (Conv with 32 filters)** | 0.96 | 0.19 |
| **K Sparse AE (Conv with 10 filters)** | 0.95 | 0.23 |
| **Label Consistent AE** | 0.98 | 0.98 |
| **LC Sparse AE** | 0.98 | 0.98 |

It is clear that the results of the different autoencoders are good when using an SVM as a classifier. Nevertheless, the addition of the Label Consistency branch allows a clear improvement both on the SVM but also on the K-means, going to classification results of over 98%.

# Chapter 7

# Conclusion

As a reminder, all results of this internship are:

| Summary table of accuracy depending on the methods | | |
|---|---|---|
| × | × | × |
| × | K-means Accuracy | SVM accuracy |
| Traditional Sparse Coding | 0.13 | 0.90 |
| LC-KSVD | 0.78 | 0.91 |
| ML-CSC | 0.11 | 0.94 |
| Sparse AE (Dense) | 0.22 | 0.91 |
| Sparse AE(Convolution) | 0.19 | 0.96 |
| LC-AutoEncoder | 0.98 | 0.98 |
| LC-Sparse AutoEncoder | 0.98 | 0.98 |

On the first hand, Statistical and deep learning methods for discriminant feature extraction is an important open problem that requires the introduction of new methods to enable the increasing of correct classification for many problems as Speech recognition. During this internship, we were able to compare different statistical methods based on the "Sparseland" hypothesis but also on the most recent deep learning methods.
All results are relatively close with the SVM classifier (approximatively 90%). However, only Label consistency term brings good result for the K-means classifier (approximatively 90% for the Label Consistency versus 10% for the others) because the information from the label is brought in during the trainning.
At the end of our research, we found that the LC-AutoEncoder and his sparse version allows good classification using both classifiers, SVM and K-means.

In general, adding label information with a label consistency constraint, or more generally directly with the label information improves the results of various methods (Sparse Coding, AutoEncoder, Sparse AutoEncoder) and we hope to get the same improvement for the Mutli-layer Convolutional Sparse Coding with the model Label-Consistent ML-CSC. It remains a job to be done, but we now know that there will already be at least two persons to take it back and improve our results.

On the other hand, this internship was an opportunity to discover in detail the world of research and all related challenges. The cross-disciplinary field between statistical approach and deep learning was very interesting, especially now, when the world is becoming more and more interested in deep learning. It is important for us to have the theoretical justifications of our models in order to be sure of the results they produce. This internship allowed us to obtain new skills, whether in optimization, machine learning (particularly deep learning) and signal processing.
It was also interesting to take part in the life of a laboratory during this internship, to discover other people's work, to offer our opinion to help and to share our personal experiences.

# Bibliography


[1] M. Aharon, M. Elad, and A. Bruckstein. lt;tex gt;*rmk* lt;/tex gt;-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, Nov 2006.

[2] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 391–398, June 2013.

[3] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, December 1998.

[4] Z. Jiang, Z. Lin, and L. S. Davis. Label consistent k-svd: Learning a discriminative dictionary for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2651–2664, Nov 2013.

[5] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM.

[6] Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013.

[7] S. G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, Dec 1993.

[8] Bruno A. Olshausen and David J. Fieldt. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[9] V. Papyan, Y. Romano, J. Sulam, and M. Elad. Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks. *IEEE Signal Processing Magazine*, 35(4):72–89, July 2018.

[10] I. Ramirez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3501–3508, June 2010.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[12] Patrice Y. Simard, Dave Steinkraus, and John Platt. Best practices for convolutional neural networks applied to visual document analysis. Institute of Electrical and Electronics Engineers, Inc., August 2003.

[13] Jeremias Sulam, Vardan Papyan, Yaniv Romano, and Michael Elad. Multi-layer convolutional sparse modeling: Pursuit and dictionary learning. *CoRR*, abs/1708.08705, 2017.

[14] Brendt Wohlberg. SParse Optimization Research COde (SPORCO). Software library available from http://purl.org/brendt/software/sporco, 2016.

[15] Brendt Wohlberg. SPORCO: A Python package for standard and convolutional sparse representations. In *Proceedings of the 15th Python in Science Conference*, pages 1–8, Austin, TX, USA, July 2017.

[16] Z. You, R. Raich, X. Z. Fern, and J. Kim. Weakly supervised dictionary learning. *IEEE Transactions on Signal Processing*, 66(10):2527–2541, May 2018.

[17] Guoqing Zheng, Yiming Yang, and Jaime Carbonell. Efficient shift-invariant dictionary learning. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 2095–2104, New York, NY, USA, 2016. ACM.