Morphological Multi-scale Decompositions and efficient representations with Auto-Encoders

Internship Report, April-September 2018

Bastien PONCHON

Supervised by: ***Jesus Angulo, Santiago Velasco-Forero, Samy Blusseau and Isabelle Bloch***

Contents

1 Introduction **2**
1.1 Context . . . 2
1.2 Representation Learning and Part based representation . . . 2
1.3 Some reminders on Mathematical Morphology . . . 3
1.4 Max-Approximation to morphological operators . . . 4
1.5 Interests of Deep Learning . . . 6
1.6 Evaluation and Data . . . 7

2 Non-Negative Matrix Factorization **8**
2.1 General Presentation and interests . . . 8
2.2 Addition of sparsity constraints (Hoyer 2004) . . . 9
2.3 Implementation and application to the data . . . 10

3 Part-Based representation using Auto-Encoders **10**
3.1 Introduction using a shallow architecture . . . 12
3.2 Enforcing sparsity of the encoding . . . 14
3.3 Enforcing Non-Negativity of the atom images . . . 16
3.4 Implementation and results . . . 17

4 Using a deeper architecture for the encoder **24**
4.1 Motivations and presentation of the architecture . . . 24
4.2 Implementation and results . . . 25

5 Conclusion and possible continuation **33**
5.1 Conclusion . . . 33
5.2 Suggested improvements and continuation . . . 33
5.3 Acknowledgments . . . 34

1 Introduction

1.1 Context

The aim of this internship was to study the interactions between two widely used but very different non-linear methods for image processing: deep-learning and mathematical morphology. We chose to focus our work on learning efficient representations of images, and more specifically representations mimicking human cognition of natural images, and that could be used to approximate the application of morphological operators to a large set of images at a lesser cost. Finally we tried to study the impact of feeding our neural network with morphological multi-scaled decompositions of images.

1.2 Representation Learning and Part based representation

Representation Learning [3] stems from the need to find an underlying structure/process explaining the data, that can somehow be represented as a set of latent features. In the case of probabilistic models, a good representation is often one that captures the posterior distribution of the underlying explanatory factors for the observed input. The assumptions usually being that the data points live on a manifold of lesser dimension than the original space, these explanatory factors are hence points in a space of smaller dimension than the input data. Finding a good representation therefore usually comes with reduced storage and computation costs, along with a solution to the curse of dimensionality, that causes most learning models to over-fit when their input data are points in a space of too great dimension. It thus comes with no surprise that representation learning has become in the past year a major field of study in the machine learning community.

Sparse coding and dictionary learning [18] are branches of representation learning where data is assumed to be well represented as a linear combination of a few elements from a dictionary, called the atom images. It is an active research field that leads to state-of-the-art results in image processing applications, such as image denoising, inpainting, demosaicking or compression. If we represent by $\mathbf{X} \in E^{M \times N}$ our data set of M images of N pixels, it boils down to the matrix factorization:

$$\mathbf{X} \approx \mathbf{HW}$$

where $\mathbf{H} = (h_{i,j})_{i,j} \in \mathbb{R}^{M \times k}$ is the matrix holding the encoding of each of the input images, that is the latent features and each row of $\mathbf{W} \in \mathcal{R}^{k \times N}$ is a dictionary images. In other words each image $\mathbf{x^{(i)}} = \mathbf{X}_{i,:}$ of the set can be written as a weighted linear combination of the atoms $\mathbf{w}_j = \mathbf{W}_{j,:}$ of the dictionary:

$$\begin{aligned} \forall i \in [1, M], \mathbf{x}^{(i)} &= \mathbf{H}_{i,:}\mathbf{W} \\ &= \mathbf{h}^{(i)}\mathbf{W} \\ &= \sum_{j=1}^{k} h_{i,j}\mathbf{w}_j \end{aligned}$$

In terms of representation it means that each image $\mathbf{x}^{(i)}$ of $\mathbf{X}$, of N pixels, can be represented as a point $\mathbf{h}^{(i)}$ in a space of dimension k. In addition, the sparsity of the encoding in enforced, which means that we enforce most of coefficients of $\mathbf{h}^{(i)}$ to be zero (or close to zero), so that only a few of the atoms of the image are really needed to approximate $\mathbf{x}^{(i)}$. The dictionary of images is fixed for all the M images of our set. The assumption behind this decomposition is that the more similar the images of the set, the smaller the required dimension to accurately approximate it. Note that only $k(N + M)$ values need to be stored or handled when using the previous approximation to represent the data, against the NM values composing the original data.

Finally, the notion of "part" based representation [21] was introduced by Daniel D.Lee and H. Sebastian Seung in a 1999 *Nature* article ([15]) where they proposed the first Non-Negative Matrix

Factorization algorithm (NMF). The aim was to perform dictionary learning in such a way that the learned atom images would represent localized features corresponding with intuitive notions of the parts of the input image family, such as parts of faces, in the case of the face database that the authors used for the first experiment of their algorithm. We see more about NMF and one of its variants in Section 2.

The aim of my internship was to use deep auto-encoder models to learn similar part-based representations of our input images, and to see how this representation could be used to approximate morphological operators, by applying them on the basis images, as we will see in the next subsections.

1.3 Some reminders on Mathematical Morphology

Mathematical morphology (Serra, 1982, 1988; Heijmans, 1994) [4] is a nonlinear image processing methodology based on the application of lattice theory to spatial structures. It can be applied to spaces with a complete lattice structure (that is partially ordered with definition of infimum and supremum): sets (example: binary images), functions (example: grey-scale images). We will here only focus on the definition of morphological operators on space of functions, and more precisely on grey-scale images. Note that we also could have worked only with binary images, as numerical grey-scale images (whose pixel intensities are quantified in a number of grey-scales) can also be decomposed into binary images representing their level sets. Morphological operators are defined using very intuitive geometrical notions which allows the perceptual development and interpretation of complex algorithms by combination of various operators.

The two fundamental morphological operators are the **dilation** and the **erosion**. In the theory of complete lattices, a dilation is any function that commutes with the supremum and erosion is any function that commutes with the infimum. We note $\mathbf{x}$ a grey-scale image, which can be seen as a function from a subset E of the discrete space $\mathbb{Z}^2$ (a grid of pixels, considered as the support space of the 2D image) to $\mathbb{R}^+$. Note that in practice, the value of the intensity of the pixels of a numerical images is not a continuous space but is quantified into a $[l_0, ..., l_L]$ discrete space, where L is the number of grey scales l_i in the numerical encoding of images, however both types of function spaces are complete lattices and assuming the pixel intensities to be in $\mathbb{R}^+$ does not change what follows. In this work, we will only consider the framework called **flat** mathematical morphology, which means that structuring elements are chosen to be subsets of E, that can be seen as well as binary images. A more general setting exists, in which structuring elements may be any functions. Let $SE \subset E \subset \mathbb{Z}^2$ bu a subset symmetric with respect to the origin, that is to say $z \in SE \iff -z \in SE$. Then the dilation δ_{SE} of the image $\mathbf{x}$ by SE can be defined as the image:

$$\delta_{SE}(\mathbf{x}) : i \in E \longmapsto \bigvee_{z \in SE(i)} \mathbf{x}(z)$$

where $SE_i = \{z + i, z \in SE\}$ is the set SE translated by i (centered at pixel i if SE is centered on the original). In this operation, SE is called the **structuring element**. $\bigvee$ and $\bigwedge$ are respectively the supremum and infimum operations. In the general case of unflat structuring functions, dilation is the counterpart of convolution in max-plus algebra.

Similarly, we define the erosion ε_{SE} of the image $\mathbf{x}$ by the structuring element SE as the image:

$$\varepsilon_{SE}(\mathbf{x}) : i \in E \longmapsto \bigwedge_{z \in SE(i)} \mathbf{x}(z)$$

Erosion and dilation by a same symmetric structural elements are dual by inversion, that is:

$$\delta_{SE}(\mathbf{x}) = n\left(\varepsilon_{SE}(n(\mathbf{x}))\right)$$

where n is an inversion: an involution (a bijection from the image space to the image space, whose inverse function is itself) such that if $\mathbf{x}$ and $\mathbf{y}$ are two images, $\mathbf{x} < \mathbf{y} \iff n(\mathbf{x}) > n(\mathbf{y})$. This property can be used to some results applying to the dilation also are true for the erosion. The most common example of inversion when the image is encoded in L grey levels $l_0, ..., l_L$ is the function defined by $\forall i \in [1, N], n(\mathbf{x})(i) = l_L - (\mathbf{x}(i) - l_0)$

Together, the dilation and the erosion form an **adjunction**, that is they verify the following property:

$$\delta(\mathbf{x}) \leq \mathbf{y} \iff \mathbf{x} \leq \varepsilon(\mathbf{y})$$

Now let us define the **opening** and **closing** by a structuring element SE, which are defined as compositions of dilation and erosion, respectively by:

$$\gamma_{SE}(\mathbf{x}) = \delta_{SE}\left(\varepsilon_{SE}(\mathbf{x})\right)$$
$$\phi_{SE}(\mathbf{x}) = \varepsilon_{SE}\left(\delta_{SE}(\mathbf{x})\right)$$

These two idempotent operators are dual by inversion just like (dilation and erosion). Qualitatively, opening darkens narrow bright zones of the image (like small object protuberance), while closing brightens narrow dark zones (like "holes" and thin cavities).

Morphological filters and transformations are useful for various image processing tasks, such as denoising, contrast enhancement, multi-scale decomposition, feature extraction and object segmentation.

There exists many other morphological operators, some of them have been considered within this study (skeleton), but are not included in this report, others (reconstructions by dilation/by erosion, and opening/closing by reconstruction) will be presented later.

1.4 Max-Approximation to morphological operators

One of the core ideas of this internship rises from two works from J.Angulo and S.Velasco-Forero ([23] and [1]), whose aim was to explore how image sparse representations can be useful to efficiently calculate approximations to morphological operators. The main motivation is to be able to apply morphological operators to massive sets of images by applying them only to the reduced set of images of the representation, in such a way that the original processed images are approximately obtained by projecting back to the initial space. This framework gets more efficiency and interest if the number of images to process is larger than the number of atoms and if many operators are to be applied.

In this context, the authors based their work on a sparse variant of the NMF decomposition introduced by Hoyer in 2004 [13], which we will explain in more depths in Section 2.2. Let us consider a family of M images (binary or gray-scale) $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ..., $\mathbf{x}^{(M)}$, which are reshaped (linearized) into vectors and are aggregated in a $M \times N$ data matrix $\mathbf{X}^T = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(M)})$ (the i^{th} row of $\mathbf{X}$ is the transpose of the linearized version of $\mathbf{x}^{(i)}$, and $|\mathbf{x}^{(i)}| = N$ is the number of pixels in all the images). The sparse NMF algorithm gives a linear non-negative approximate data decomposition of $\mathbf{X}$ into two matrices $\mathbf{H} \in \mathbb{R}_+^{M \times k}$ and $\mathbf{W} \in \mathbb{R}_+^{k \times N}$:

$$\mathbf{X} \approx \mathbf{HW} = \hat{\mathbf{X}}$$

Each of the lines of $\mathbf{W}$ contains a basis vector $\mathbf{w}_j$ (which is a linearized image of N pixels), and each row of $\mathbf{H}$ contains the coefficient vector $\mathbf{h}^{(i)}$ corresponding to image $\mathbf{x}^{(i)}$. We say that the matrix $\mathbf{W}$ contains the dictionary, and $\mathbf{H}$ the encoding of the decomposition.

Recall from Section 1.2 that, each image $\mathbf{x^{(i)}}$ of the set can be written as a weighted linear combination of the atoms $\mathbf{w}_j$ of the dictionary:

$$\forall i \in [1, M], \mathbf{x}^{(i)} = \sum_{j=1}^{k} h_{i,j} \mathbf{w}_j$$

The authors then define the **Sparse Max-Approximation to gray-level dilation and erosion** respectively as:

$$
\begin{aligned}
D_{SE}(\mathbf{x}^{(i)}) &= \sum_{j=1}^{k} h_{i,j}\delta_{SE}(\mathbf{w}_j) \\
E_{SE}(\mathbf{x}^{(i)}) &= \sum_{j=1}^{k} h_{i,j}\varepsilon_{SE}(\mathbf{w}_j) \\
&= \sum_{j=1}^{k} h_{i,j} n\left(\delta_{SE}(n(\mathbf{w}_j))\right)
\end{aligned}
$$

where the erosion and dilation are applied to the atom gray-scaled images (in 2D, the same notation is kept for the images and their linearized column-vector versions, for the sake of simplicity).

Note that these approximate non-linear operators do not satisfy the standard properties of gray-level dilation and erosion.

Similarly, the authors define the **Sparse Max-Approximation to gray-level opening and closing**, respectively as:

$$
\begin{aligned}
G_{SE}(\mathbf{x}^{(i)}) &= \sum_{j=1}^{k} h_{i,j}\gamma_{SE}(\mathbf{w}_j) \\
F_{SE}(\mathbf{x}^{(i)}) &= \sum_{j=1}^{k} h_{i,j}\phi_{SE}(\mathbf{w}_j) \\
&= \sum_{j=1}^{k} h_{i,j} n\left(\gamma_{SE}(n(\mathbf{w}_j))\right)
\end{aligned}
$$

As the dilation commutes with the supremum of functions, and as the support of our positive functions being pairwise-disjoint is a sufficient condition for the commutation of opening with the supremum, the authors state and empirically show that a good non-redundant sparse-NMF decomposition yields a good max-approximation to the dilation and to the opening. Indeed, this is when the part-based decomposition, along with the sparsity and non-negativity of the learned representation comes into play. As we will see in Section 2, the non-negativity prevents mutual cancellations between parts of the atom images, leading to part-based representations. Moreover the sparsity constraint enforces the decomposition of each image to use fewer atoms, and therefore enforces the atoms not to overlap in their supports. As a result, weighted atoms of the decomposition of an image can be considered almost pair-wise disjoint, that is:

$$
\forall i \in [1, M], \forall (j, l) \in [1, k]^2, h_{i,j}\mathbf{w}_j \bigwedge h_{i,l}\mathbf{w}_l \approx 0 \tag{1}
$$

as sparsity is enforced on the encoding, and as NMF ensures a part-based decomposition. The supremum of pair-wise disjoint functions is equal to their sum, which implies that the decomposition by sum performed by the NMF is close to a decomposition by max (all the atoms does not need to be mutually disjoint as long as the weighted atoms of the decomposition of an image are):

$$
\forall i \in [1, M], \bigvee_{j \in [1,k]} h_{i,j}\mathbf{w}_j \approx \sum_{j=1}^{k} h_{i,j}\mathbf{w}_j \tag{2}
$$

This motivates the framework proposed by the authors:

$$\begin{aligned}
\delta_{SE}(\mathbf{x}^{(i)}) &= \delta_{SE}\left(\sum_{j=1}^{k} h_{i,j}\mathbf{w}_j\right) \\
&\approx \delta_{SE}\left(\bigvee_{j\in[1,k]} h_{i,j}\mathbf{w}_j\right) \\
&= \bigvee_{j\in[1,k]} \delta_{SE}(h_{i,j}\mathbf{w}_j) \\
&= \bigvee_{j\in[1,k]} h_{i,j}\delta_{SE}(\mathbf{w}_j) \text{ as } h_{i,j} \geq 0, \forall (i,j) \in [1,M]\times[1,k] \\
&\approx \sum_{j=1}^{k} h_{i,j}\delta_{SE}(\mathbf{w}_j) \\
&= D_{SE}(\mathbf{x}^{(i)})
\end{aligned}$$

With a similar result for the max-approximation to openings, as well as for the max-approximation to erosions and closings by duality.

The framework is shown to provide empirically encouraging results by the authors when experimenting on data sets of gray-scale images.

1.5 Interests of Deep Learning

Even though NMF have the very nice properties that will be listed in Section 2, it also has one major drawback. Indeed, the dictionary of the representation is learned for a specific data set of images, and the algorithm provides no way to represent, using the same atom images, images that were not included in the training set on which the NMF algorithm was run. Unlike in PCA decomposition, where the orthogonality of the components is enforced, and therefore a projection on the learned space is tractable, there is no closed form giving the weights $\mathbf{h}^{(M+1)}$ of the linear combination of the atom images $\mathbf{w}_1, ..., \mathbf{w}_k$ approximating a new data point $x^{(M+1)}$ with the proper sparsity constraint, when the atom images $\mathbf{W}$ have been learned on the images $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ...\mathbf{x}^{(M)}$. If we could find a neural network performing the same kind of sparse and non-negative part-based representation, then the framework proposed in Section 1.4 using this decomposition would enable to compute max-approximation to morphological operators on new similar but previously unseen images, without the need to re-train the model and alter the atom images. Applying the morphological operator to a limited number of k fixed images would hence theoretically enable to compute the max-approximation of it on an unlimited number of images showing similar content (as variance of the content showed by the images increases, so would do the size of the latent representation space).

Moreover, deep neural networks are believed to have the ability to capture relationships much more complex than linear ones, the universal approximator theorem guaranteeing that a feed-forward neural network with at least one hidden layer can represent an approximation of any function (within a broad class) to an arbitrary degree of accuracy, provided that it has enough hidden units. The last few years indeed have seen significant interest in deep learning algorithms that learn layered, hierarchical representations of high-dimensional data. Much of this work appears to have been motivated by the hierarchical organization of the human's brain visual cortex, and indeed authors frequently compare their algorithms' output to the oriented simple cell receptive fields found in visual area V1 or V2. Indeed, some of these models are often viewed as first attempts to elucidate what learning algorithm (if any) the cortex may be using to model natural image statistics [16]. Given the nature of the

decomposition we want to learn, it seems that auto-encoders are the most intuitive architectures answering to problem at stake, as we will see in Section 3.

Throughout my research work, I encountered several attempts at applying "deep" methods with sparsity and non-negativity constraints to perform part-based decompositions, similar to the very efficient one presented by the NMF algorithms, without considering the interest of using the learned features to approximate non-linear transformations, but rather focusing on reconstruction quality and/or classification accuracy. This works are namely:

- Lemme *et al.* (2011) [17], who presented a shallow auto-encoder architecture, with tied weights between the encoder and the decoder (that is the decoder weight matrix being the transpose of the encoder one) yielding higher sparsity and lower reconstruction errors than related algorithms based on matrix factorization. The authors qualified their algorithm as *online* as it generalizes to new inputs both accurately and without costly computations, which is fundamentally different from the classical matrix factorization approaches, that they relate to as *offline* algorithms.

- Hosseini-Asl *et al.* (2016) [12], that introduces a deep learning autoencoder network, trained by a non-negativity and sparsity constrained algorithm that appears to learn features which show part-based representation of data. The authors assess the performance of their network using classification and reconstruction error.

- Guo and Zang (2017) [9] which introduces a deep Non-Negative Factorization framework with a specifically designed optimization algorithm, not really using tools from the Deep Learning paradigm such as back-propagation.

- Ayinde and Zurada (2018) [2], which is an improvement proposal to the algorithms proposed in [12] to train an understandable classifier initialized using auto-encoders with non-negativity and sparsity constraints.

1.6 Evaluation and Data

Many different kinds of image families could be used in the context of part-based representation: binary shapes, database of registered gray images, patches of large images, time series, Hyper-spectral images, etc. While, we were at first planning to use multi-modal PET/MRI images of the brain, we finally opted for the use of the fashion MNIST data set for the sake of interpretability and simplicity. Fashion-MNIST ([24]) is a dataset of Zalando's article (clothes) images consisting of a balanced training set of 60,000 examples and a balanced test set of 10,000 examples. Just like the regular MNIST data set of handwritten digits, the images are 28x28 grayscale images, each associated with a label from 10 classes. It is intended to serve as a drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. The latter is indeed considered by the authors of [24] as too easy (as many models easily achieve a 97% accuracy in the classification task), overused (even suspecting an over-fitting of the machine learning community to this data set) and not suitable to modern day computer vision tasks. For our concerns, we also chose Fashion-MNIST as it represented shapes that can easily be interpreted as union of easily identifiable parts (such as sleeves in shirts), unlike medical images.

I implemented my models and experiments in Python 3.6, with the Keras v.2.2.0 deep learning framework on top of Tensorflow 1.9.

In order to evaluate and compare my various approaches to part, the following metrics will be used:

- The reconstruction error, that is the Mean-Squared Error (L_2, eventually normalized by the number of pixels in the images) between the original input images and their approximation by the learned representation.

- The best Support-Vector Machine (SVM) classification accuracy obtained using the encoding of the test images. A SVM with a linear kernel is trained and evaluated using the encodings

of the images obtained with the various methods. 30 values of the C penalty parameter, in a logarithmic range from 10^{-2} to 10^3 were tested, and the model with the highest classification accuracy, on a 15 folds cross-validation, was selected. A model with this value of the parameter was then re-trained and evaluated on a 25 folds cross-validation on left out data.

- The sparsity of the encoding, measured using the mean on all test images of the sparsity metric introduced by P.O. Hoyer in [13] (2004) and presented in section 2.2, based on the relationship between the L_1 and the L_2 norm:

$$S(\mathbf{h}^{(i)}) = \frac{\sqrt{k} - \frac{||\mathbf{h}^{(i)}||_1}{||\mathbf{h}^{(i)}||_2}}{\sqrt{k} - 1}$$

- The Max-Approximation error to dilation by a disk of radius 1, obtained by computing the mean squared error between the dilation by a disk of radius 1 of the original image and the max-approximation error to the same dilation, using the learned representation.

The three first metrics are quite commonly used in the field of representation learning. In particular they were used by the related works mentioned in Section 1.5.

In the following sections, we will first expose in further details the Non-Negative Matrix Factorization, as well as its sparse variant introduced by P.O.Hoyer in [13] (2004), used by J.Angulo and S.Velasco-Forero in [23] and [1], and its performance on our data set. We will then present how we tried to implement a similar sparse part-based representation using auto-encoders, using firstly a shallow architecture, and then an asymmetric architecture with a deep encoder, along with the performance of the implemented architectures.

2 Non-Negative Matrix Factorization

2.1 General Presentation and interests

The first Non-Negative Matrix Factorization algorithm was introduced by D.D. Lee and H.S. Seung in 1999 [15] as a way to find a set of basis functions for representing non-negative data. They claimed that the notion is particularly applicable to image articulation libraries made up of images showing a composite object in many articulations and poses. They suggested (in the very title of the article) that when used in the analysis of such data, NMF would find the intrinsic 'parts' going to make up the object being pictured. They based their methods on the observation of psychological and physiological evidence for part-based representations in the brain, which can be conceptually tied to the non-negativity constraints.

The NMF is a factoriation matrix algorithm, just like Principal Component Analysis (PCA). But the latter, whose atoms are the eigen vectors associated with the largest eigenvalues of the covariance matrix of the data, and therefore are the directions along which the variance of the data is maximal, produces distorted versions of the input images. NMF on the opposite produces atoms corresponding to localized features that correspond better with the intuitive notion of parts, only additive combinations of the atoms being allowed. The variability of the data is generated by combining these different parts, without using all of them for each image, even though all atoms are used by at least one image. Hence this representation respects a latent sparsity constraint, which is explicitly imposed in the variant we used as a baseline in our study [13], as we will see in the following section.

The decomposition of the data set of images into $\hat{\mathbf{X}} = \mathbf{HW} \approx \mathbf{X}$ is obtained though very simple update rules of both the encoding and the atom matrices, that preserve their non-negativity and also constrain the norm of $\mathbf{W}$ as the decomposition into $\mathbf{HW}$ is invariant to scaling of both $\mathbf{H}$ and $\mathbf{W}$. This iterative algorithm is proven to monotonically converge to a local maximum of the objective function:

$$\sum_{i=1}^{M}\sum_{n=1}^{N}(\mathbf{X}_{i,n}\log(\mathbf{HW})_{i,n} - (\mathbf{HW})_{i,n})$$

A work from D.Donoho and V.Stodden in 2003 [6] explains how and when this Non-Negative Matrix Factorization gives a correct decomposition into parts. The paper shows that the NMF algorithm actually recovers the parts of the images if the data set of images is a *separable factorial articulation family*, that is follows the three following properties:

- Generative Model: each image is actually generated by a linear combination of positive atom images associated with non-negative weights.
- Separability: for a given pixel, there is only one atom image whose value is not null at this pixel, that is the atom images all have separated supports.
- Complete Factorial Sampling: all different combinations of parts are exhaustively sampled.

[6] theoretically and empirically shows that in such Separable Factorial Articulation Families, non-negative factorization is effectively unique (up to scaling and permutation of the atom images). In such libraries, NMF will indeed successfully 'find the parts'.

Note however that our Fashion-MNIST data probably does not verify the properties of a Separable Factorial Articulation Families, as a set of $k = 100$ separable atom images cannot easily be explicitly constructed.

Many variants were built on top of the original algorithm, usually producing additional constraints and properties: local NMF (enforcing atom images containing localized features), sparse NMF (enforcing sparse encoding or atom images), non linear variants, binary variants, etc. In the following section, we will present one of those variants, that was the one used by the authors of [23] when they presented the framework of max-approximation to morphological operators.

2.2 Addition of sparsity constraints (Hoyer 2004)

The idea of incorporating sparse constraints to NMF, is to construct a succinct representation of the image data as a combination of a few typical patterns (few atoms of the dictionary) learned from the data itself. By enforcing each image to be represented by a weighted combination of a small number of atoms, we expect those atoms to be closer to being pair-wise disjoint, and to verify equation (1), and as consequence to verify equation (2) which is sufficient for our max-approximations to be palatable approximations of the morphological operators. Moreover, there exist conceptual motivations behind sparsity, as stated in [20], [18] and [16], as attempts to reproduce the receptive fields found in the visual cortex of human brain.

In the case of NMF, the sparse variant we will use as a baseline is the one introduced by P.O.Hoyer in 2004 [13], and stems from the observation that NMF not always produced an intuitive decomposition into parts that would correspond to the human idea of the 'building blocks' of the data and is based on a new sparseness measure involving the relationship between the L_1 norm and the L_2 norm, defined, for some vector $\mathbf{v}$, as:

$$S(\mathbf{v}) = \frac{\sqrt{|\mathbf{v}|} - \frac{||\mathbf{v}||_1}{||\mathbf{v}||_2}}{\sqrt{|\mathbf{v}|} - 1} \tag{3}$$

where $|\mathbf{v}|$ is the dimensionality of the vector $\mathbf{v}$. This function evaluates to unity if and only if $\mathbf{v}$ contains only a single non-zero entry, and takes a value of zero if and only if all components are equal (up to signs), interpolating smoothly between the two extremes.

The algorithm proposed by [13] constrains the sparsity measure of each image encoding $S(\mathbf{h}^{(i)}), i \in [1, M]$ and/or of each atom $S(\mathbf{w}_j), j \in [1, k]$ to be equal to fixed values S_h and S_w respectively. This is done by reprojecting the weights (atoms) and encoding matrices on the corresponding space after each update.

2.3 Implementation and application to the data

I used the same Matlab implementation provided by [13] as the one used by the authors of [23]. I opted for not applying any sparsity constraint on the atom images (note that this is not equivalent to setting $S_w = 0$) , after trying several values, I chose to show the results for $S_h = 0.6$, for the fixed latent dimension $k = 100$ I will use throughout my study, in Figures 1 and 2. We note that despite sparsity and non-negativity constraints, leading to a really nice part-based decomposition, with each atom showing localized features, even though the support of the atoms are not pair-wise separable, the approximation is quite close to the original images, in spite of some loss in the texture information.

As we can see in Figure 1c, most coefficients in the encoding of each image are close to 0 or very small. This leads to the quite accurate max-approximation to dilation by a disk of radius 1 that can be seen in Figure 2.

3 Part-Based representation using Auto-Encoders

An **Auto-Encoder** is a Neural Network that performs representation learning by approximating the identity function, that is learning a function $f_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = \hat{\mathbf{x}} \approx \mathbf{x}$ where $\mathbf{x} \in \mathcal{R}^N$ is an input of dimension N, $\hat{\mathbf{x}}$ its image by the auto-encoder, and $\mathbf{W}$ and $\mathbf{b}$ respectively the **weights** and **biases** of the network. In our case, the dimension of the input data points N will be equal $d \times d \times c$, as we will deal with images of size d by d with c channels (more precisely in the case of the Fashion-MNIST data set: $d = 28$, $c = 1$). Internally, the network has a hidden layer $\mathbf{h} \in \mathcal{R}^k$ that describes a code of fixed size k used to represent the input. It is composed of two sub-networks:

- An **encoder** $h_{\mathbf{W}_e,\mathbf{b}_e} : \mathbf{x} \longmapsto \mathbf{h} \in \mathbb{R}^k$, that learns a representation of the input of dimension k.
- A **decoder** $g_{\mathbf{W}_d,\mathbf{b}_d} : \mathbf{h} \longmapsto \hat{\mathbf{x}}$ that learns to reconstruct the input from an encoding of it.

We therefore aim at finding the parameters of the functions h and g, $\mathbf{W} = (\mathbf{W}_e, \mathbf{W}_d)$ and $\mathbf{b} = (\mathbf{b}_e, \mathbf{b}_d)$, that minimize the mean of the **Reconstruction Error** $L(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(i)}) = L(f(\mathbf{x}^{(i)}), \mathbf{x}^{(i)}) = L(g(h(\mathbf{x}^{(i)})), \mathbf{x}^{(i)})$ over all the M points $\mathbf{x}^{(i)}$ of the data set (we omit the indexation of the function by their parameters for the sake of readability). The **objective function** (that we will call the **loss** of the auto-encoder) we try to minimize is thus:

$$L_{AE} = \frac{1}{M} \sum_{i=1}^{M} L(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)})$$

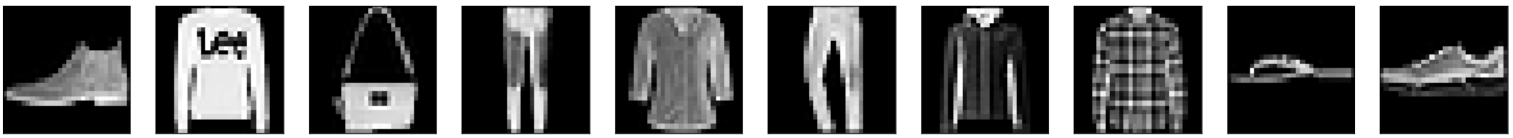
The reconstruction error function L can be of various types:

- **Mean-Squared Error**: $L(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(i)}) = \frac{1}{N} ||\hat{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}||_2^2$
- **Binary-Cross Entropy**: $L(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(i)}) = \frac{1}{N} \sum_{l=1}^{N} \left(x_l^{(i)} \log \hat{x}_l^{(i)} + (1 - x_l^{(i)}) \log(1 - \hat{x}_l^{(i)}) \right)$
- etc.

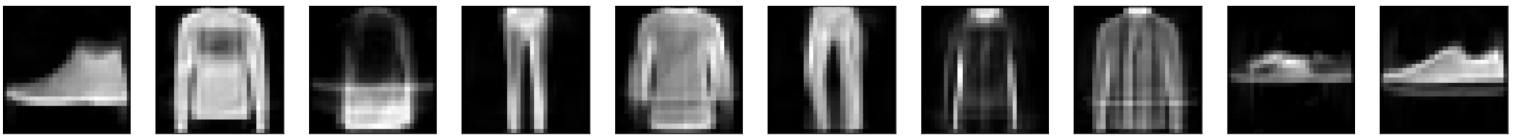
The Mean-Squared Error was our implementation choice for the design of our models, as it is both a commonly used metric between images and a commonly used loss function in deep learning. However, one could argue that a better loss function could be chosen, as blurry images may not be as discouraged by this metric as we would like to (we can get a quite low mean-squared error between blurry reconstructions and the original image).

The representation capacity of the model increases with the dimension of the encoding and with the depth of the auto-encoder, but when the size of the encoding is chosen to be smaller than the dimension of the input, the so-called **under-complete** auto-encoder performs a form of dimensionality reduction that can capture hidden structure in the input.

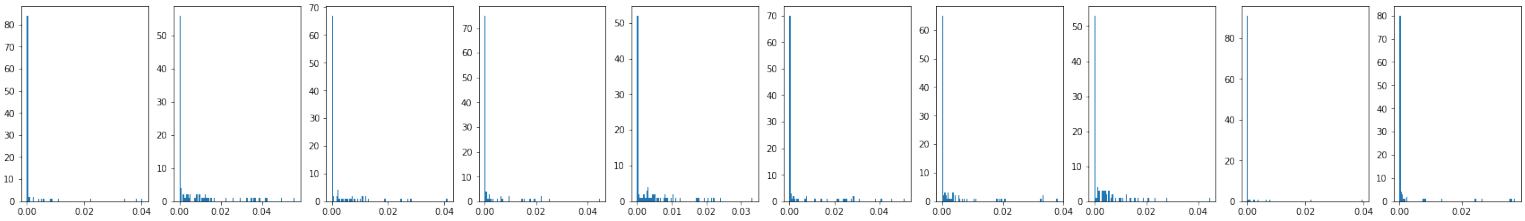
Another way to perform an efficient representation learning is to add other types of constraints on the model (sparsity of the representation, smallness of the derivatives of the representation, robustness to noise, robustness to adversarial examples, etc.)



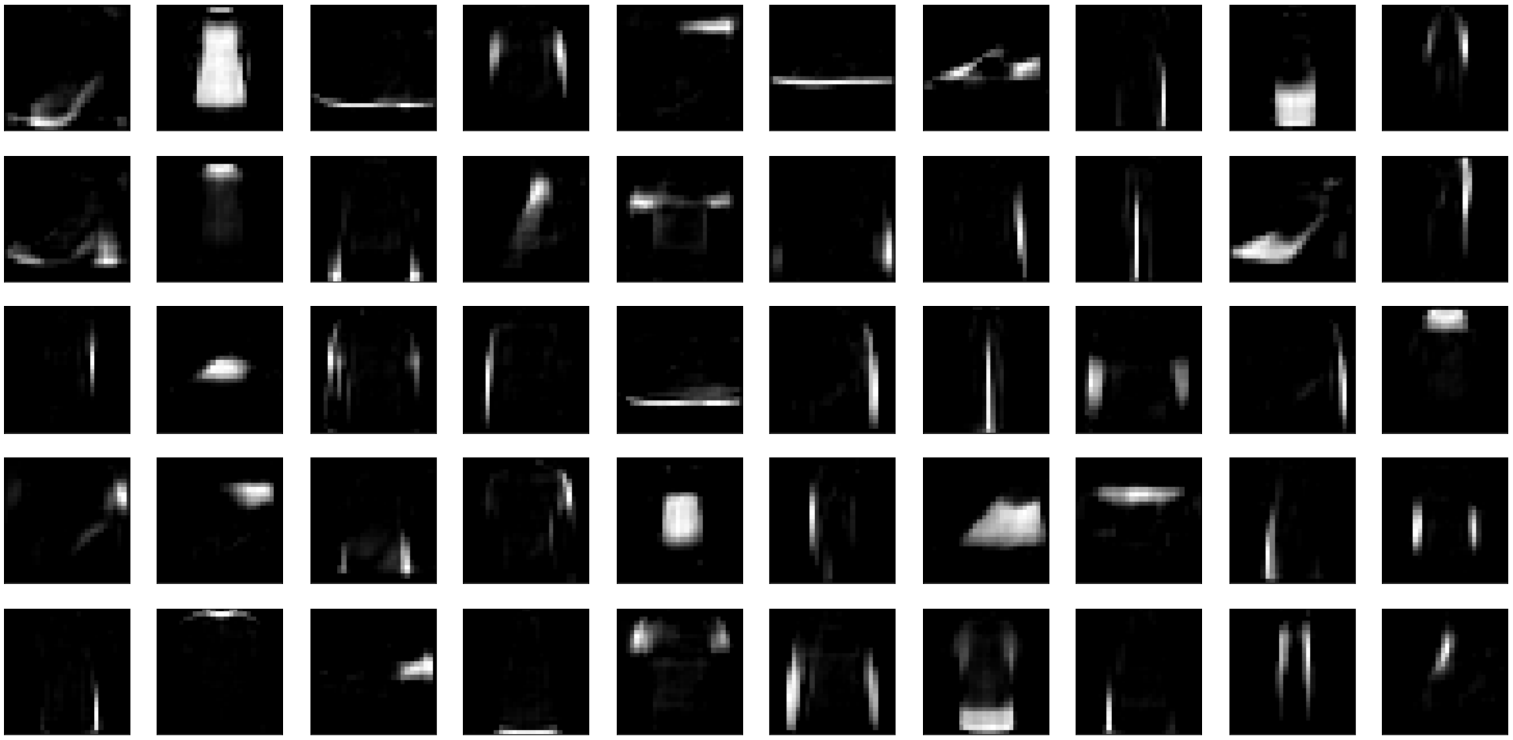
(a) Original Images



(b) Approximation (reconstruction) of the original images by the sparse NMF - *Reconstruction error: 0.0109*

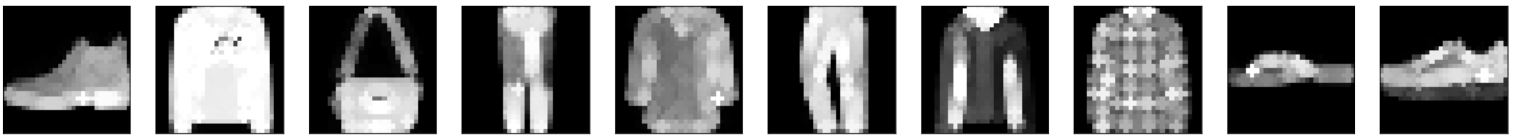


(c) Histogram of the encoding of each of the images - *Sparsity (Hoyer 2004): 0.650*

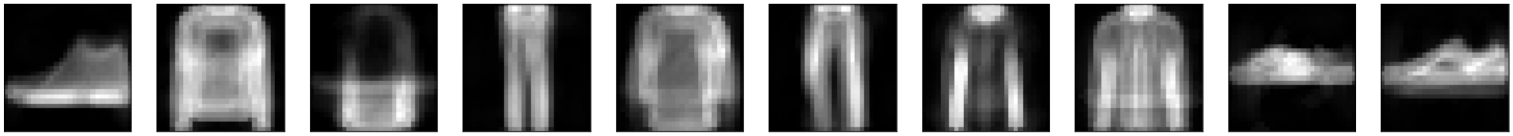


(d) 50 of the $k = 100$ learned dictionary images (atoms)

Figure 1: Part-based representation learned using the Hoyer NMF with sparsity constraint on the encoding: $S_h = 0.6$



(a) Dilation of the original images by a disk of radius 1



(b) Max-approximation to a disk of radius 1 - *Max-approximation error: 0.107*

Figure 2: Max-approximation to dilation of a disk of radius 1 using the representation learned with Hoyer NMF with sparsity constraint on the encoding: $S_h = 0.6$[13] [23]

3.1 Introduction using a shallow architecture

For the sake of simplicity and understandability of the method, we first tried to perform an efficient part-based representation using **shallow** Auto-Encoders, that is, encoders and decoders both composed of only two densely connected layers of neurons (input and output) as represented in Figure 3.

We hence have an auto-encoder of the form:

$$\begin{aligned}\hat{\mathbf{x}} &= f(\mathbf{x}) \\ &= g(h(\mathbf{x})) \\ &= g(\mathbf{h}) \\ &= \sigma_d\left(\mathbf{W}_d^T\mathbf{h} + \mathbf{b}_d\right) \\ \mathbf{h} &= h(\mathbf{x}) \\ &= \sigma_e\left(\mathbf{W}_e^T\mathbf{x} + \mathbf{b}_e\right)\end{aligned}$$

where σ_e and σ_d are two **non-linear** functions applied element-wise to their input vectors. Once our network trained, we hence can represent each image of our database as the image by σ_d of a linear combination of k **atom images** $\mathbf{W}_{d,1}, \mathbf{W}_{d,2}, ..., \mathbf{W}_{d,k}$ (each row of $\mathbf{W}_d$ is an atom image flatten in a $(1, N)$ row vector), to which a bias image $\mathbf{b}_d$ is added (presented as flatten $(N, 1)$ column vecotr). Therefore the weights of the decoder constitute a dictionary of images, similarly to the learned representation in the NMF of section 2.

When applied to a batch of M points, represented as a design matrix $\mathbf{X}$ of size $M \times N$ (N is the input dimension), the computation performed by our feed-forward network can be written:

$$\begin{aligned}\hat{\mathbf{X}} &= f(\mathbf{X}) \\ &= g(h(\mathbf{X})) \\ &= g(\mathbf{H}) \\ &= \sigma_d\left(\mathbf{H}\mathbf{W}_d + \mathbf{b}_d^T\right) \\ \mathbf{H} &= h(\mathbf{X}) \\ &= \sigma_e\left(\mathbf{X}\mathbf{W}_e + \mathbf{b}_e^T\right)\end{aligned}$$

where the addition by the bias vectors $\mathbf{b}_e$ and $\mathbf{b}_d$, of size $k \times 1$ and $N \times 1$ respectively.

Note that if no further constraints are applied, if σ_e is the identity function, and if the reconstruction error is chosen to be the mean squared-error, then the under-complete auto-encoder learns to span the same space as PCA.

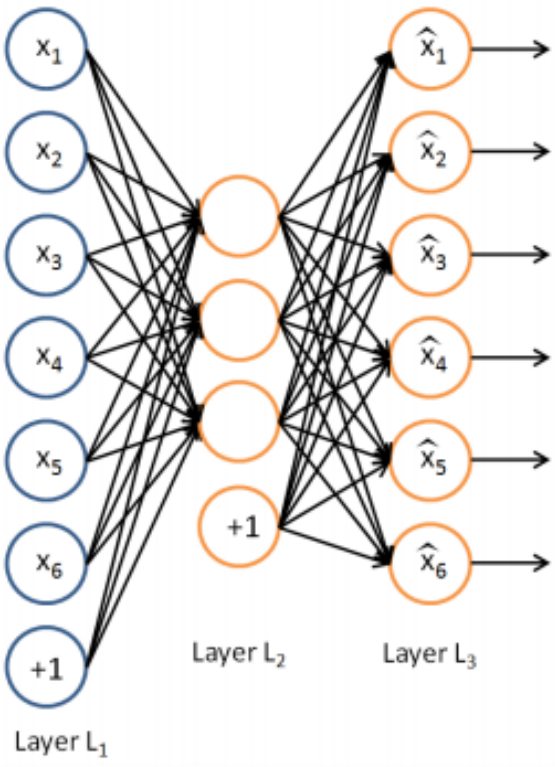


Figure 3: Three layers architecture, from [19]

3.2 Enforcing sparsity of the encoding

Although the topic of sparsity constraints is not overwhelmingly present in the Deep Learning community, it has been addressed many times in literature and many methods have been proposed to enforce either the sparsity of the learned encoding or of the weights of the network, motivated by the the objective of capturing a more robust representation of the manifold structure.

The most prevalent idea to enforce sparsity of the encoding can be traced back to the work of H.Lee *et al.* [16]. In this 2008 paper, the authors presented a sparse variant of the Deep Belief Networks proposed by Hinton *et al.* [11] which faithfully mimics certain properties of the visual area V2 in the cortical hierarchy, able to learn Gabor-like filters.

The core idea of this variant was to add, to the negative-log likelihood of the data, a regularization term that penalizes a deviation of the expected activation of each hidden unit from a (low) fixed level p. Intuitively, this regularization should ensure that the "firing rate" of the encoder neurons (corresponding to the latent random variables h_j in the Deep Belief Network model) are kept at a certain (fairly low) level, so that the encoding is sparse. The authors chose the squared L_2 norm as a distance to the expected activation p. With our notations, the introduced regularization term can thus be written:

$$||\mathbf{p} - \frac{1}{M}\sum_{i=1}^{M}\mathbf{h}^{(i)}||_2^2 = \sum_{j=1}^{k}|p - \frac{1}{M}\sum_{i=1}^{M}h_j^{(i)}|^2$$
$$\mathbf{p} = [p, p, ..., p]^T \in \mathbb{R}^k$$

Hence, the total loss of the sparse Auto-encoder can be written as a sum of the mean reconstruction error and a regularization term:

$$L_{AE} = \frac{1}{M}\sum_{i=1}^{M}L(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}) + \beta\sum_{j=1}^{k}S(p, \sum_{i=1}^{M}h_j^{(i)})$$

where the parameter p sets the expected activation objective of each of the hidden neurons, and the parameter β controls the strength of the regularization, relatively to the reconstruction error.

With, in the case of [16]:

$$S(p, \sum_{i=1}^{M}h_j^{(i)}) = |p - \frac{1}{M}\sum_{i=1}^{M}h_j^{(i)}|^2$$

Several works proposed their variants of the regularization function S. A 2015 paper from Zhang and Lu [25] proposed an empirical survey on the auto-encoders with different sparsity regularizers, namely:

- A penalty function based on the KL-divergence between two Bernouilli distributions:

$$S_{KL}(p, t_j) = p\log\frac{p}{t_j} + (1-p)\log\frac{1-p}{1-t_j}$$
$$t_j = \sum_{i=1}^{M}h_j^{(i)}$$

- A differentiable approximation of the L_1 norm, called the εL_1 norm, where the ε parameter

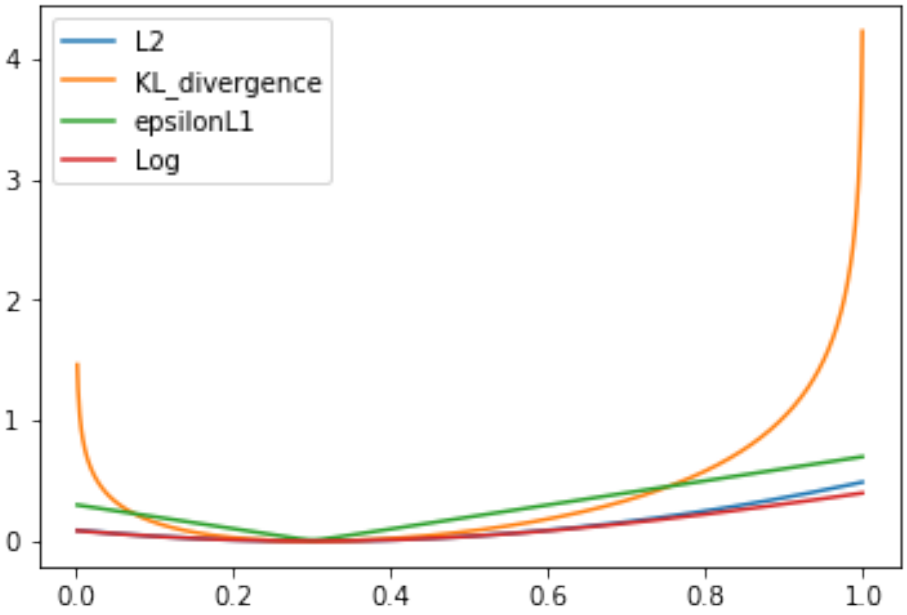


Figure 4: The four presented penalty function for $p = 0.3$.

controls the similarity with the L_1 norm:

$$S_{\varepsilon L_1}(p, t_j) = \sqrt{(p - t_j)^2 + \varepsilon}$$
$$t_j = \sum_{i=1}^{M} h_j^{(i)}$$

- A non-parametric regularizer, proposed by Olshausen and Field in [20] called the Log regularizer:

$$S_{\log}(p, t_j) = \log\left(1 + (p - t_j)^2\right)$$
$$t_j = \sum_{i=1}^{M} h_j^{(i)}$$

Note that the KL divergence regularizer requires the mean activation of unit h_j to be in $[0, 1]$, and thus the activation function of the decoder σ_e is required to output an encoding in this range of values $[0, 1]$. We hence chose the sigmoid function: $\sigma_e(z) = \frac{1}{1+e^{-z}}$. Such restrictions does not apply to the three other regularizers presented.

Figure 4 shows the four sparsity penalty functions we just introduced, for a given expected activation $p = 0.3$. Figure 5 shows the three penalty functions that are defined outside of]0,1]. In [25], the authors decided to consider only the εL_1 and the *Log* regularizer with $p = 0$ which which no longer allows to control to control the degree of sparseness we want. Even-though we implemented also the εL_1 and the *Log* regularization functions, we decided to use the KL divergence in most of our experiments, as [25] did not enable to state whether the *Log* regularizer performed better than the KL divergence one

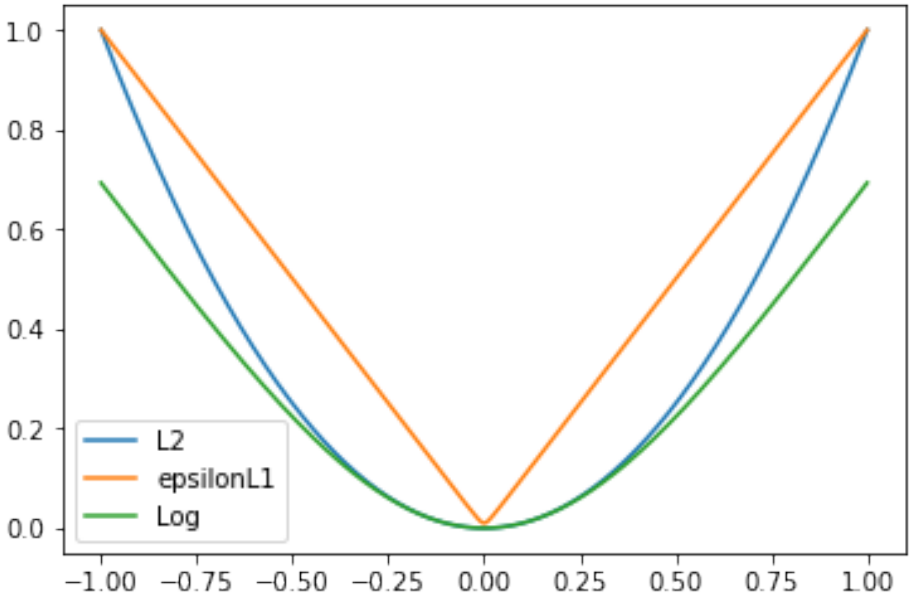


Figure 5: When the code can be negative, the KL divergence regularization may no longer be defined, and one of the three others has to be used, with $p = 0$

or not and as the latter is currently the most used in the literature. In particular it was the one used in the related works [12] and [2].

Along my internship, I came across some other ways of enforcing sparsity of the encoding, for instance the *intrinsic plasticity* introduced in [17] which is based on the utilization of a parametrized logistic activation function at the end of the encoder, whose parameters are trained along with the other parameters of the network, in order to optimize information transmission of the neurons.

Another method that could be employed to achieve sparsity of the encoding was introduced in a work from Han, *et al.* [10]. This paper from 2017 presents a training method for deep learning that is more robust to over-fitting. It consists in pruning the weights of the network whose values are under a fixed threshold, during some phases of the training, enforcing the sparsity of those weights. I believe this algorithm could be adapted so as to produce a sparse encoding.

3.3 Enforcing Non-Negativity of the atom images

In the case of the NMF presented in section 2, just like in the case of the decoder, non-negativity of all the elements of the decomposition results in a part-based representation of the input images. In the case of neural networks, enforcing the non negativity of the weights of a layer eliminates cancellations of input signals, as all operations performed in the layer before the activation function are additions and multiplications between positive elements. In our case, we will only enforce the non-negativity of the decoder weights $\mathbf{W}_d$ as the encoding is already made positive by the choice of the sigmoid as the activation function σ_e of the encoder layer. Moreover, we are only interested in the dictionary of images defined by the weights of the decoder, and hence enforcing the non-negativity of the encoder weights would bring nothing but more constraints to a network with an already quite low capacity (and hence unlikely to over-fit given the amount of training data at our disposal).

In the literature, various approaches have been designed to enforce weight positivity. A popular approach is to use an **asymmetric weight decay** to enact more decay on the negative weights that on the positive ones, limiting their magnitude. The loss of the non-negative sparse Auto-Encoder thus

becomes:

$$L_{AE} = \frac{1}{M}\sum_{i=1}^{M} L(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}) + \beta \sum_{j=1}^{k} S(p, \sum_{i=1}^{M} h_j^{(i)}) + \lambda \sum_{i,j} \Phi(W_{d,(i,j)})$$

$$\text{with, } \forall (i,j) \in [1,k] \times [1,N], \Phi(W_{d,(i,j)}) = \begin{cases} \alpha |W_{d,(i,j)}|^2 & \text{if } W_{d,(i,j)} \geq 0 \\ |W_{d,(i,j)}|^2 & \text{if } W_{d,(i,j)} < 0 \end{cases}$$

where the parameter $\alpha \geq 0$ controls the regularization strength ratio between the positive and the negative weights and the parameter $\lambda \geq 0$ controls the strength of the penalty in the auto-encoder loss. A Non-Negativity constraint is applied if $\alpha < 1$, and a standard weight decay is applied if $\alpha = 1$. In the case of the shallow auto-encoder, setting $\alpha = 0$ can be a good guess as the model is already quite robust and does not need further decay on the positive weights. However this approach, used in [12], does not assure that all weights will be positive, depending on the chosen value for the parameters α and λ, as noted in [2]. Indeed, the original L_2 norm used here penalizes the weights with bigger magnitudes more strongly than those with smaller magnitudes. This forces a good number of the weights to take on small negative values, but not necessarily to be positive. Therefore works [17] and [2] both propose variants of the equation of the asymmetric weight decay, the first one by using the L_1 rather than the L_2 norm, the second by proposing a smoothed version of the decay using both the L_1 and the L_2 norms. As mentioned in the previous section the effect of the L_1 norm tends to select components, setting the other to zero, while the L_2 norm tends to reduce the magnitude of all components.

However, to enforce non-negativity of the decoder weights, we may wish to use explicit constraints rather than penalties. That is not adding any penalty term but projecting our weights on the nearest points of the positive orthant after each update of the optimization algorithm (such as the stochastic gradient descent). The first asset of this other method is that it ensures non-negativity of all weights without the need for searching a good value of parameter λ (in our implementation and experiments, we set $\alpha = 0$ to reduce the number of parameters to search).

Another reason to use explicit constraints and re-projection rather than enforcing constraints with penalties is that penalties can cause non-convex optimization procedures to get stuck in local minima corresponding to small negative values of $\mathbf{W}_{d,i,j}$, as mentioned in [8], Chapter 7 on regularization techniques.

Even-though the asymmetric weight decay was implemented and experimented during my internship, I focused most of my work on the non-negative re-projection method.

3.4 Implementation and results

We implemented and tested our shallow architecture, with and without the various regularizers and evaluated it using various criteria, as explained in section 1.6:

- The reconstruction error.
- The SVM classification accuracy obtained using the encoding of the test images.
- The sparsity of the encoding, measured using the sparsity metrics introduced by P.O. Hoyer in [13] (2004) and presented in section 2.2.
- The Max-Approximation error to dilation by a disk of radius 1, as explained in section 1.6. In the case of the shallow auto-encoder, it is computed by applying the dilation to the atom images (the weights) of the decoder and computing the mean squared error between the output of this

new feed-forward network and the dilation of its input. That is:

$$E_{\delta_1}(f) = \frac{1}{MN}\sum_{i=1}^{M} ||f_{\delta_1}(\mathbf{x}^{(i)}) - \delta_1(\mathbf{x}^{(i)})||_2^2$$
$$\text{with } f_{\delta_1}(\mathbf{x}^{(i)}) = g_{\delta_1}(h(\mathbf{x}^{(i)}))$$
$$= \sigma_d\left(\delta_1(\mathbf{W}_d)^T\mathbf{h}^{(i)} + \mathbf{b}_d\right)$$

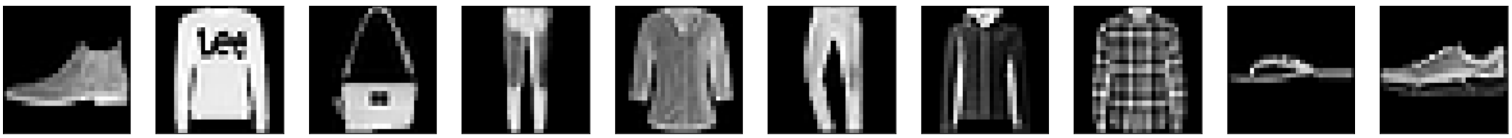
where $\delta_1(\mathbf{W}_d)$ represent the matrix whose rows are the flatten versions (in a row vector of size $(1, N)$) of the application of the dilation δ_1 by a disk of radius 1 to each of the atom images of the decoder $\mathbf{W}_{d,j}$ (seen as $d \times d$ images, where in our case $d = 28$ pixels). f_{δ_1} is an auto-encoder whose encoder is the same as the encoder h of the trained auto-encoder f, and whose decoder g_{δ_1} is obtained by applying the dilation δ_1 to each atom image of the decoder g of the trained auto-encoder. Note that the dilation could have been applied to the bias image $\mathbf{b}_d$. However, we observed better max-approximation error, when not doing so, the bias being considered more like a pixel-wise threshold on the intensity of the weighted sum of atom images (weighted by the encoding coefficients)

Figures 6-10 show the results we got with auto-encoders with various constraints, and a latent dimension fixed to $k = 100$. We chose to only show results for the non-negativity re-projection constraint and the KL-divergence sparsity regularizer.

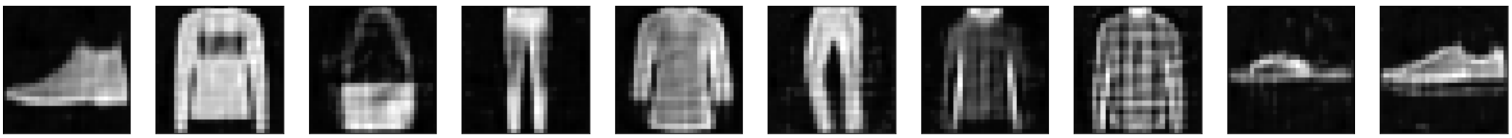
Figure 6 shows the reconstruction by our shallow auto-encoder, with various non-negativity and sparsity constraints, of 10 images from a test set not used to train the model. As we can see in Figure 7, the results are ordered by increasing sparsity constraint, the second row representing the results for the unconstrained auto-encoders. As expected, the reconstruction quality decreases with the strength of the sparsity penalty and with the expected activation objective, as it can be seen in Figure 10a. Figure 8 shows some atom images of the auto-encoders (recall that there are $k = 100$ atom images). We can see that the sparsity and non-negativity constraints unearth representations close to the part-based one. However, the values of the parameters of the sparsity regularizer has great an impact on this representation. Indeed, for a given expected activation of the hidden units, a small strength of the sparsity regularization will result in a sort of part based representation, as parts of clothes and shapes are visible on the atom images. As the strength of the sparsity regularization increases, full clothes shapes will become visible on the atom images, and not only part of it. It seems then that the model is performing some sort of K-mean, each input being averaged by one, or a combination of a few, of the atom images, that can hence be considered as representing elements of some latent clusters in the input data. Whatever the settings, we noticed that some atoms were poorly used, resulting in the blurry, almost all blacks atoms that can be seen in each representation but the unconstrained one. This phenomenon increases with the strength of the regularization (high β and low p), and we did not find any setting that achieves the efficiency of the sparsity constrained NMF introduced by Hoyer in 2004 ([13]). Specifically, the features showed by our atom images are much less localized that the one showed in Figure 1d.

It is worth noting in Figure 10b that the sparsity metric drops when the regularization becomes too strong (unlike the KL divergence regularization function, the higher the Hoyer metric, the sparser the encoding), this is because the sparsity metric is close to 0, when all coefficients are almost equal, even when they are all very close to 0, which happens when the applied sparsity regularization is too strong. The atoms then tend to be all black but a few pixels, resulting in blurry reconstructions dictated by the bias image, and hence independent of the input.

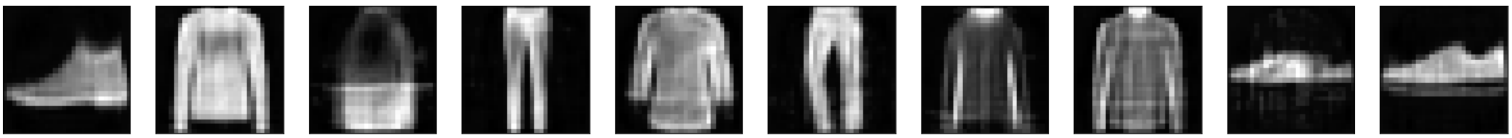
Finally, Figure 9 shows the max-approximation to dilation by a disk of radius 1 of the various shallow auto-encoders. We can see from Figure 10c that the sparsity of the encoding improves the quality of the max-approximation, except when it reaches the points where all coefficients of the encoding average zero. Figure 10d shows the MSE between the max-approximation to dilation and



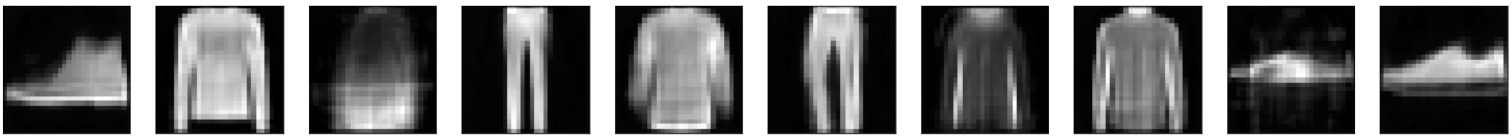
(a) Original Images



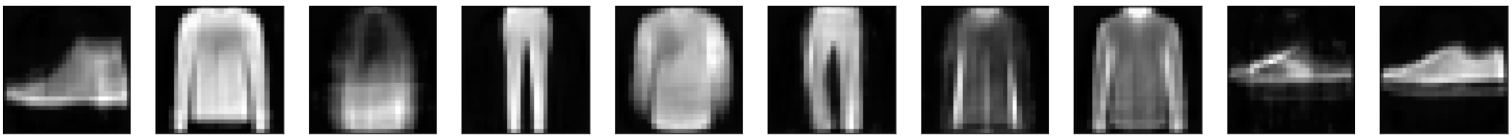
(b) Shallow Auto-encoder with no constraints - *reconstruction error: 0.00697*



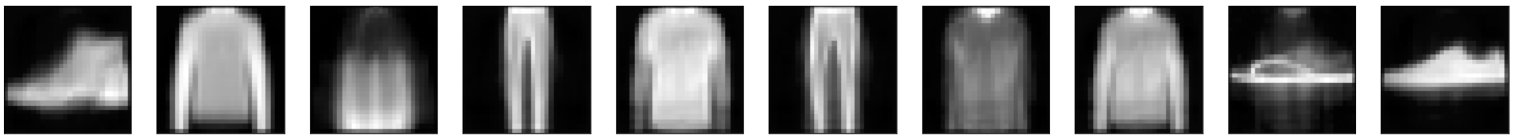
(c) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.2$, $\beta = 0.001$ - *reconstruction error: 0.0103*



(d) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.1$, $\beta = 0.01$ - *reconstruction error: 0.0139*

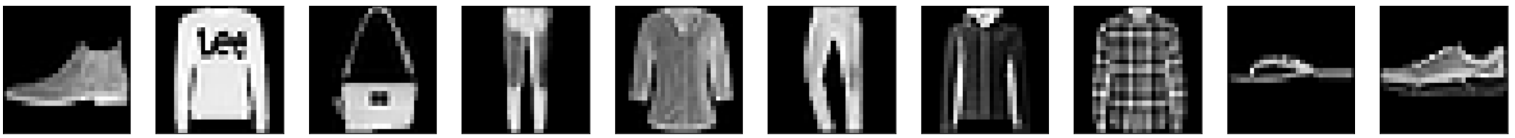


(e) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *reconstruction error: 0.0164*

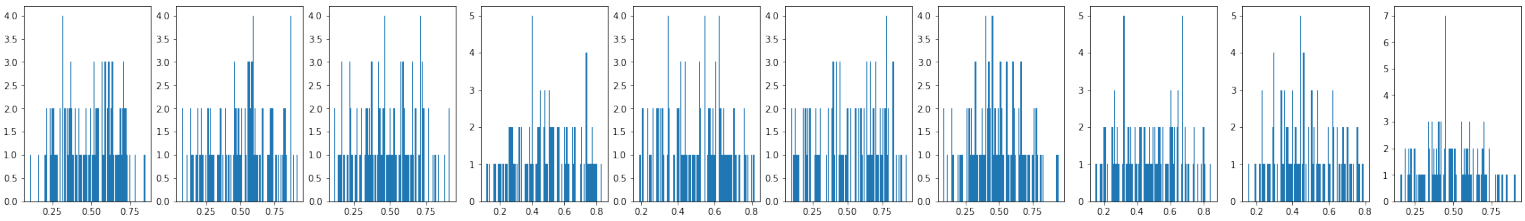


(f) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$ - *reconstruction error: 0.0288*

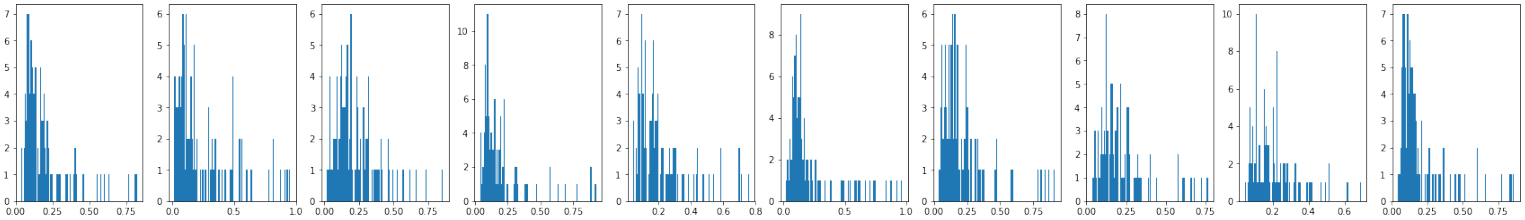
Figure 6: Reconstruction of 10 images by a shallow auto-encoder with various regularizations and constraints



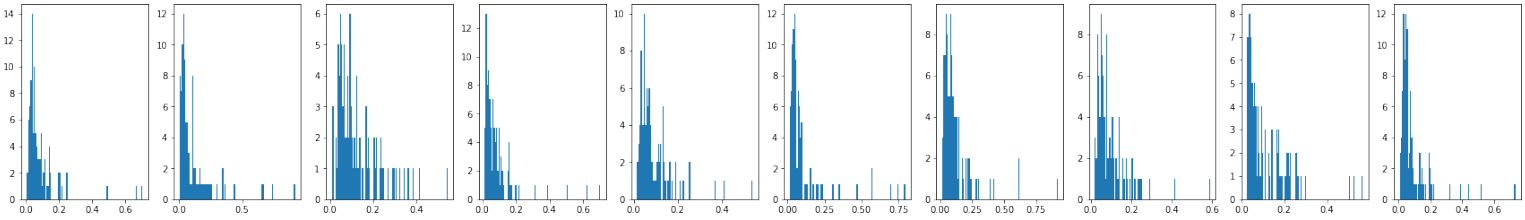
(a) Original Images



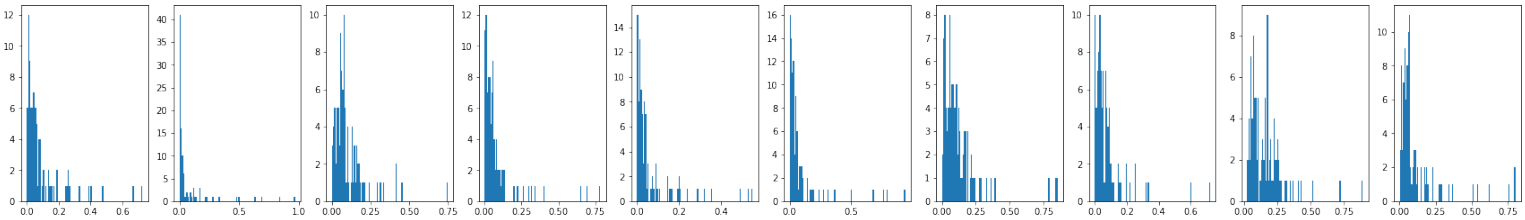
(b) Shallow Auto-encoder with no constraints - *Sparsity (Hoyer 2004): 0.0678*



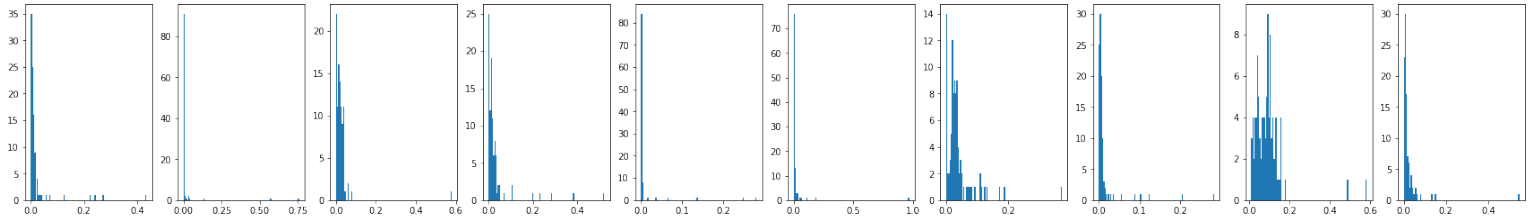
(c) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.2$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.230*



(d) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.1$, $\beta = 0.01$ - *Sparsity (Hoyer 2004): 0.363*

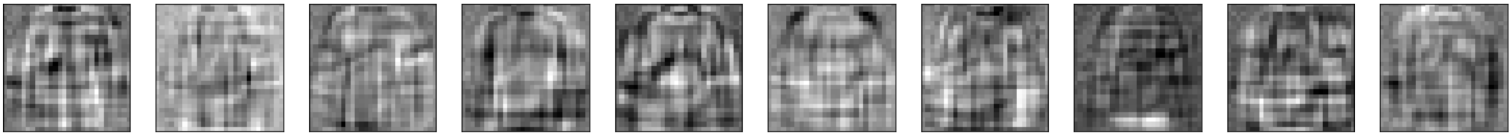


(e) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.505*

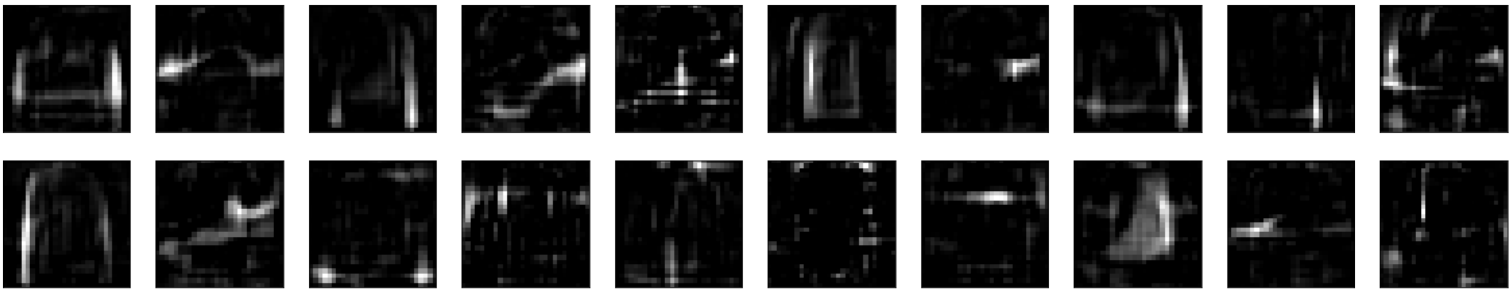


(f) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$ - *Sparsity (Hoyer 2004): 0.785*

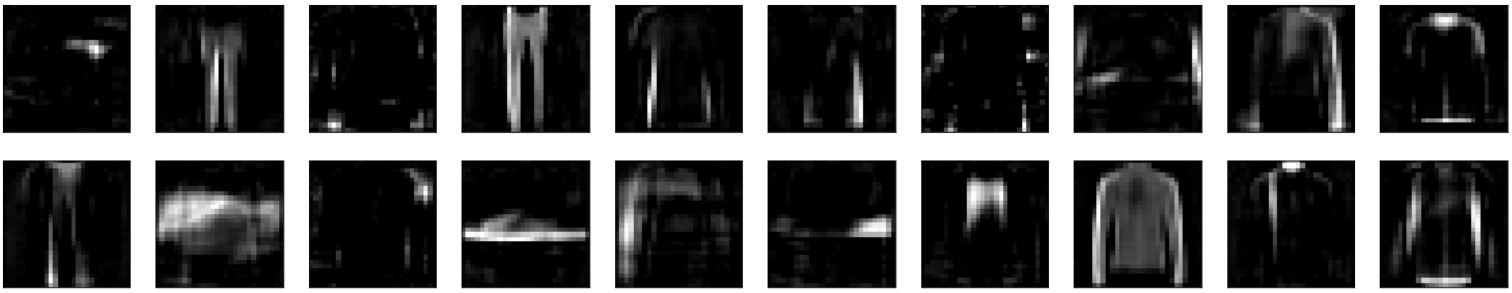
Figure 7: Histograms of the encoding of each of the 10 original images for the various versions of the shallow Auto-Encoder



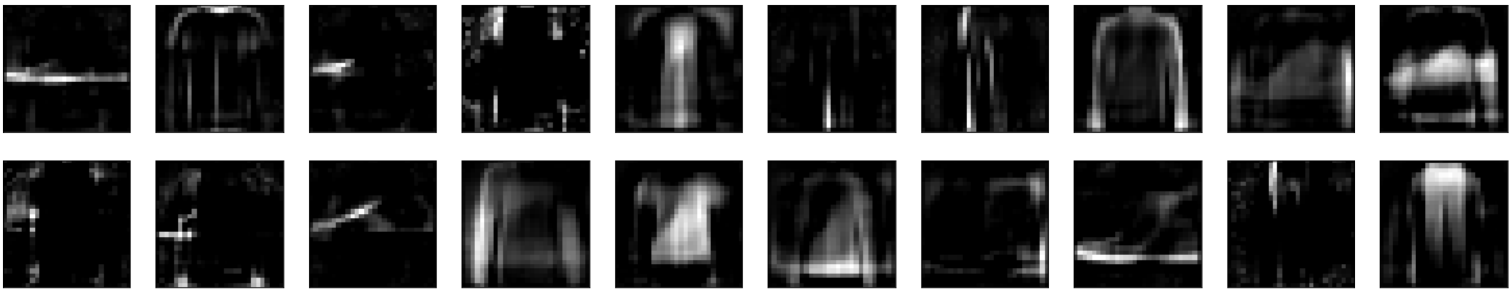
(a) Shallow Auto-encoder with no constraints



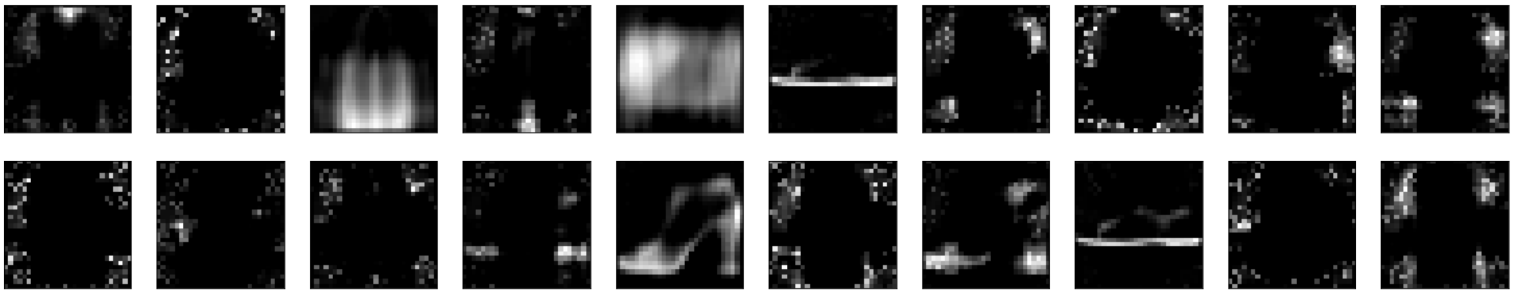
(b) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.2$, $\beta = 0.001$



(c) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.1$, $\beta = 0.01$

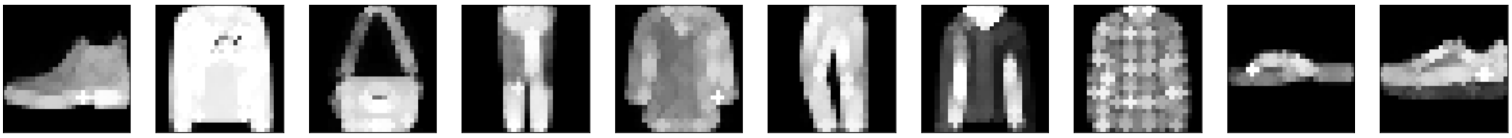


(d) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$

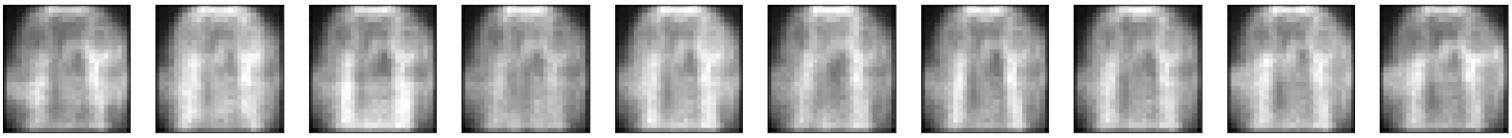


(e) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$

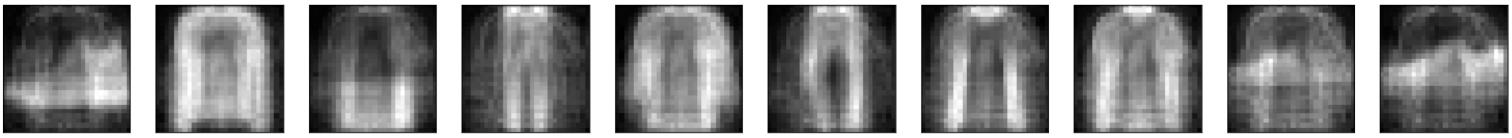
Figure 8: Some atoms (out of the 100 atoms) of the various versions of the shallow Auto-Encoder



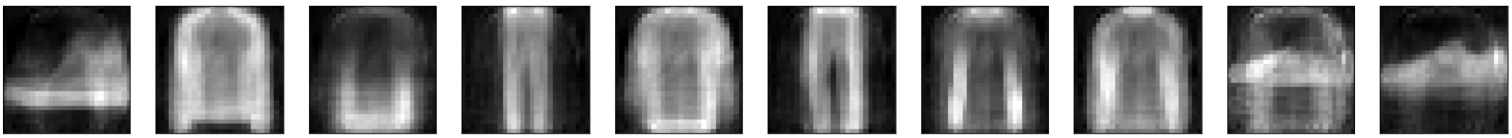
(a) Dilation of the original images by disk of radius 1



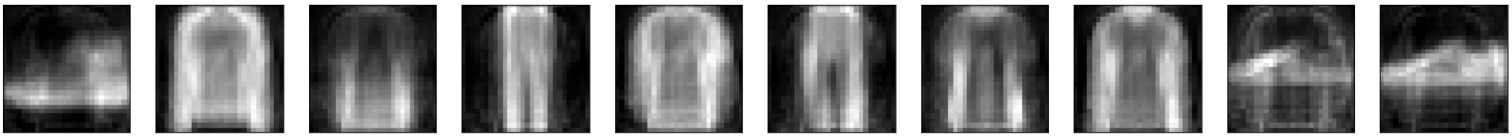
(b) Shallow Auto-encoder with no constraints - *Max-Approximation error: 18.04*



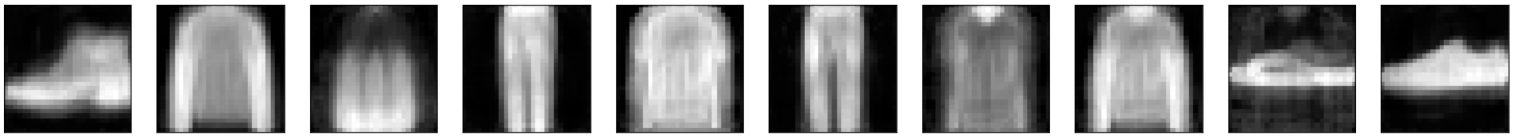
(c) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.2$, $\beta = 0.001$ - *Max-Approximation error: 1.179*



(d) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.1$, $\beta = 0.01$ - *Max-Approximation error: 0.264*

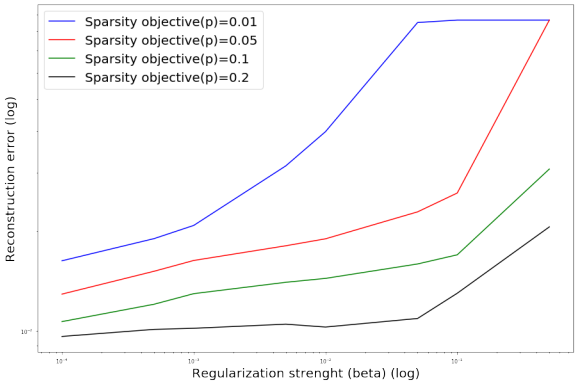


(e) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *Max-Approximation error: 0.182*

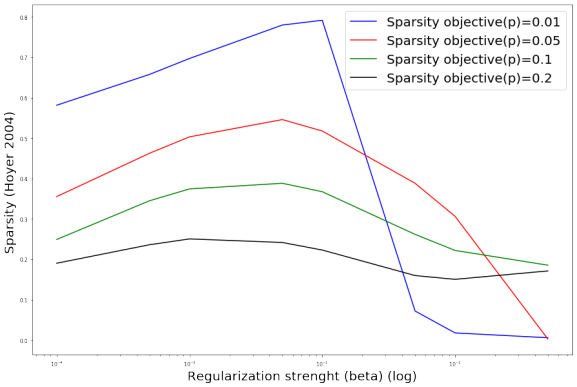


(f) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$ - *Max-Approximation error: 0.0339*

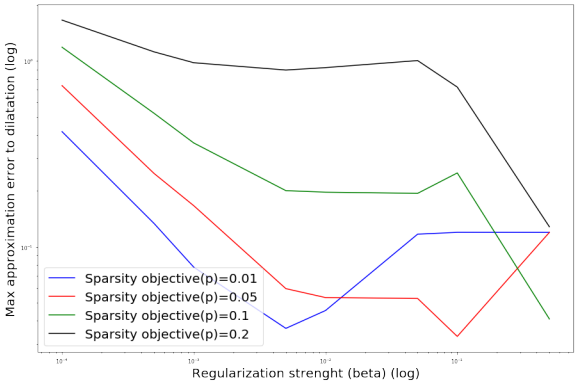
Figure 9: Max-approximation to the dilation by a disk of radius 1



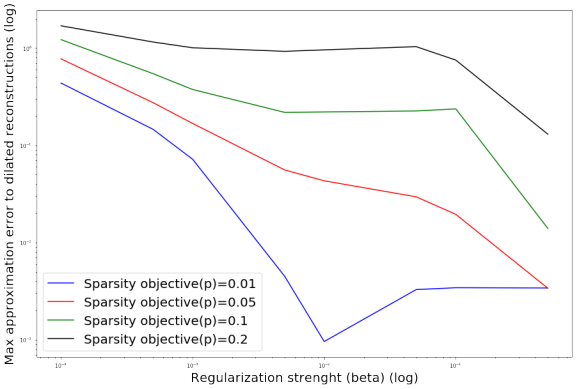
(a) Reconstruction error as a function of the parameter β (sparsity penalty strength)



(b) Sparsity metric (Hoyer 2004) as a function of the parameter β (sparsity penalty strength)



(c) Max-approximation error to dilation (of the original images) as a function of the parameter β (sparsity penalty strength)



(d) Max-approximation error to dilation (of the reconstructed images) as a function of the parameter β (sparsity penalty strength)

Figure 10: Various metrics evaluated on sparse non-negative shallow auto-encoders for various parameters of the sparsity regularization, using a test set not used to train the network

the dilation of the reconstructed images (instead of the dilation of the original ones), this measure is less informative than its Figure 10c counterpart as it can get pretty low when the reconstruction is very blurry.

Finally, Table 1 shows the values of our various metrics of our shallow auto-encoder models. It is to be noted that they all give close SVM classification scores, the comparison based on this metric may thus not be relevant. However, it seems that increasing the sparsity constraint increases the classification accuracy, if not too strong, in which case it may slightly drop.

Model	Sparsity Parameters	Reconstruction error	Sparsity (Hoyer 2004)	Max-approximation error to dilation	SVM classification accuracy (std)
Sparse NMF	$S_h = 0.6$	0.0109	0.6504	0.107	0.812 (0.036)
Unconstrained shallow AE	-	0.00697	0.0678	18.24	0.792 (0.035)
Sparse Non-Negative shallow AE	$p = 0.2$ $\beta = 0.001$	0.0103	0.230	1.179	0.781 (0.039)
Sparse Non-Negative shallow AE	$p = 0.1$ $\beta = 0.01$	0.0139	0.363	0.264	0.785 (0.033)
Sparse Non-Negative shallow AE	$p = 0.05$ $\beta = 0.001$	0.0164	0.505	0.182	0.804 (0.031)
Sparse Non-Negative shallow AE	$p = 0.01$ $\beta = 0.005$	0.0288	0.785	0.0339	0.767 (0.035)

Table 1: Comparison of some of our shallow auto-encoders and the sparse NMF model using the four metrics presented in Section 1.6, evaluated on a test set left out for the training of the auto-encoders.

4 Using a deeper architecture for the encoder

4.1 Motivations and presentation of the architecture

Despite encouraging results, we note that our shallow architecture does not enable to reach as a compelling part-based representation as the sparse Non-Negative Matrix Factorization, which keeps a low reconstruction error (eventhough higher than the shallow auto-encoder one with no constraints) while unearthing a relevant part-based representation enabling a good max-approximation error to morphological operator,as demonstrated in [23] and as shown in Table 1.

This may be due to the fact that the shallow auto-encoder architecture is one of low capacity, that does not benefit from the main assets of deep learning, that is learning highly complex and hierarchical mappings, similarly to the way humans are believed to analyze complex interactions by breaking them into isolated and understandable hierarchical concepts. Moreover one major advantage of nontrivial depth is that the universal approximator theorem guarantees that a feed-forward neural network with at least one hidden layer can represent an approximation of any function (within a broad class) to an arbitrary degree of accuracy, provided that it has enough hidden units. This means that an auto-encoder with a single hidden layer is able to represent the identity function, an obvious solution being to have as many hidden units as we have pixels in the images (N). However, the mapping from input to code is shallow. This means that we are not able to enforce arbitrary constraints, such as that the code being sparse. A deep auto-encoder, with at least one additional hidden layer inside the encoder itself, can approximate any mapping from input to code arbitrarily well, given enough hidden units.

We hence decided to use a deeper encoder to learn the encoding of our input images. However, we chose to keep a one layer decoder, so as to keep the ability to approximate the input images as a simple function of a fixed dictionary of images of the same size. This also allows us to keep the definition of Max-Approximation to morphological operators introduced in the previous section for the shallow auto-encoder.

The architecture we chose for our encoder was taken from the infoGAN architecture by Xi Chen, Yan Duan, *et al.* (2016) [5], that aims at performing interpretable representation learning based on generative adversarial networks (GANs from I. Goodfellow *et al.* [7]) and a auto-encoder-like architecture. We took one piece of the architecture (the recognition sub-netowrk of infoGAN for the MNIST dataset) and adapted it in the following **AsymAE** architecture:

- Input: 28×28 gray scale image.
- 64 **2D convolutional filters** of size 4×4 with stride 2, with a *leakyRELU activation* (with negative slope set to 0.1).
- 128 **2D convolutional filters** of size 4×4 with stride 2, with **batch-normalization** and *leakyRELU activation*.
- Flattening of the feature maps into one vector.
- **Fully connected layer** of 1024 neurons, with **batch-normalization** and *leakyRELU activation*.
- **Fully connected encoding** (output of the encoder) layer ($k = 100$ hidden units) with *sigmoid activation function* so as to be able to apply the KL divergence sparsity regularizer.

The choice of this architecture was motivated by several reasons:

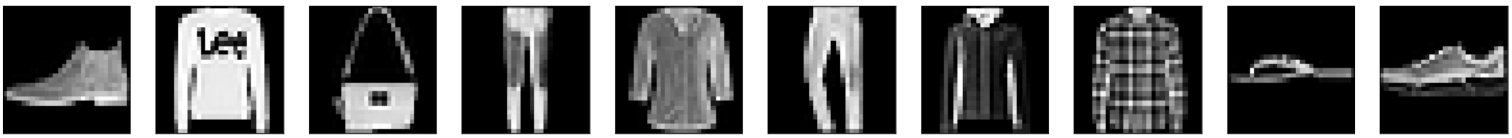
- The relatively good results of *infoGAN* obtained on representation learning tasks when applied to the MNIST data set (that have the same dimensions as the Fashion-MNIST dataset to which we will apply the encoder).
- A simple architecture, with a relatively small number of layers, when compared with other networks commonly used in computer vision tasks.
- The use of convolutional layers, well adapted to images, as it detects patterns at various positions in the input, and proven to lead to compelling performance on computer vision tasks.
- The use of widely adopted state of the art techniques in deep learning, such as leakyRELU activation function, that enables to overcome the vanishing gradient problem encountered with usual activation functions such as sigmoid and hyperbolic tangent, while still allowing a small non negative gradient when the input to the activation function is negative, and batch-normalization layers, introduced in [14], which provides stability of the network training, as it provides normalization of the features inside the network.

We left the decoder architecture untouched and applied the sparsity constraints to the output layer of the encoder (the encoding) and the Non-Negativity constraint on the weights of the decoder like in the previous section.

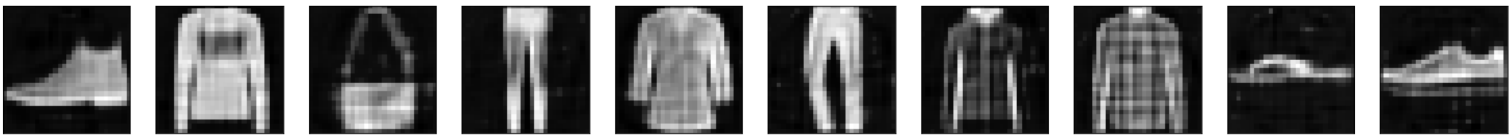
4.2 Implementation and results

Figures 11-15 show the results we got with this new architecture, using various constraints, and a latent dimension fixed to $k = 100$. The results are ordered by increasing sparsity.

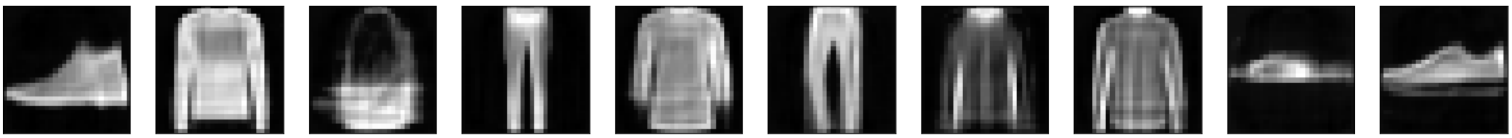
Comparing the reconstructions from Figures 11 and 6 we can note that a deeper encoder allows for stronger sparsity constraints while keeping acceptable reconstructions, usually leading to higher sparsity, with a lower reconstruction error and max-approximation error than when using a shallow encoder. Indeed, as we can see by comparing Figure 15b and Figure 10b, the deep encoder allows reaching higher values for the sparsity metrics, unreachable with the shallow auto-encoder as strengthening the sparsity constraints usually ended in the collapsing of all encoding coefficients towards zero and therefore to low values of the Hoyer metric.



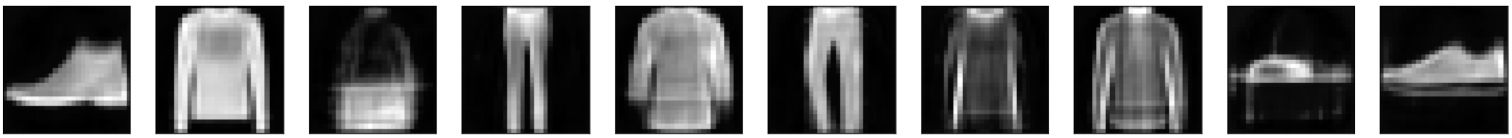
(a) Original Images



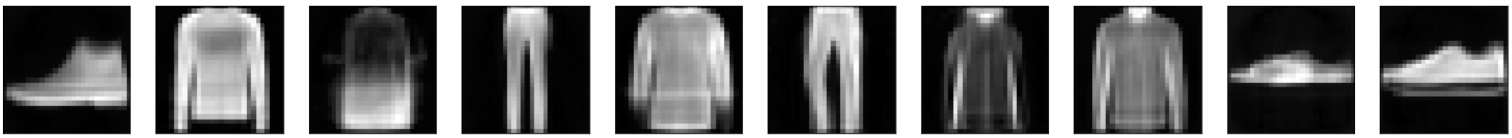
(b) Asymmetric Auto-encoder with no constraints - *reconstruction error: 0.00646*



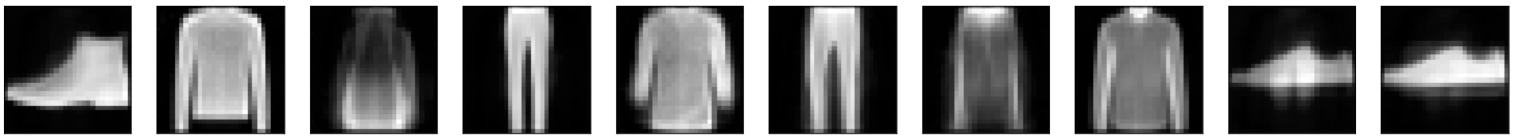
(c) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *reconstruction error: 0.0115*



(d) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.005$ - *reconstruction error: 0.0125*

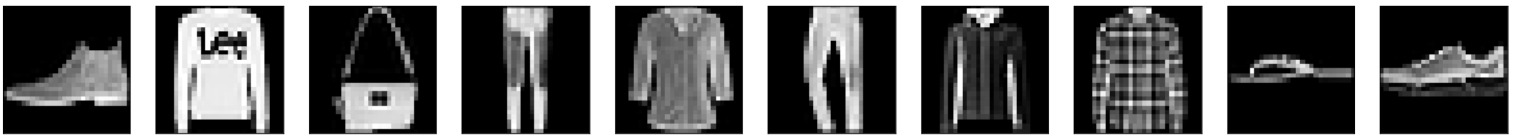


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$ - *reconstruction error: 0.0150*

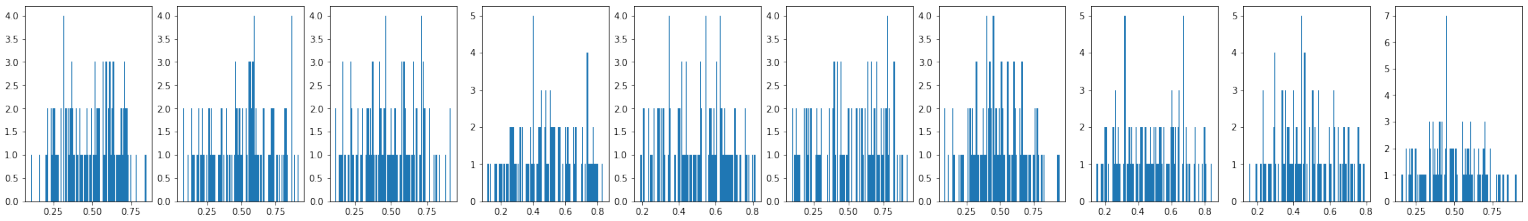


(f) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$ - *reconstruction error: 0.0212*

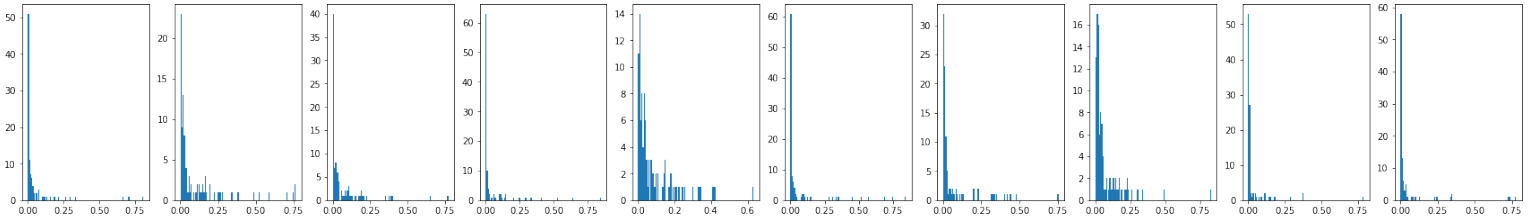
Figure 11: Reconstruction of 10 images by an Asymmetric auto-encoder with various regularizations and constraints



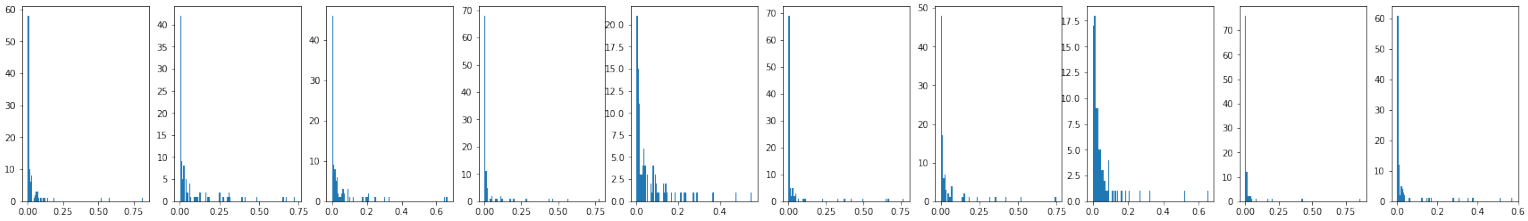
(a) Original Images



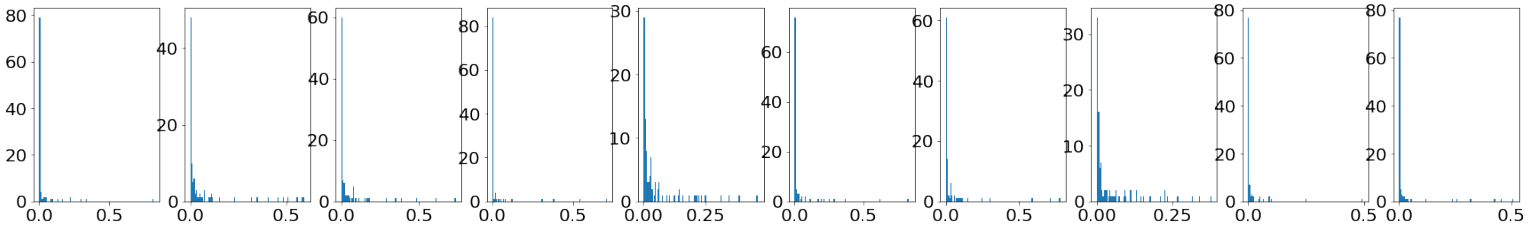
(b) Asymmetric Auto-encoder with no constraints - *Sparsity (Hoyer 2004): 0.0956*



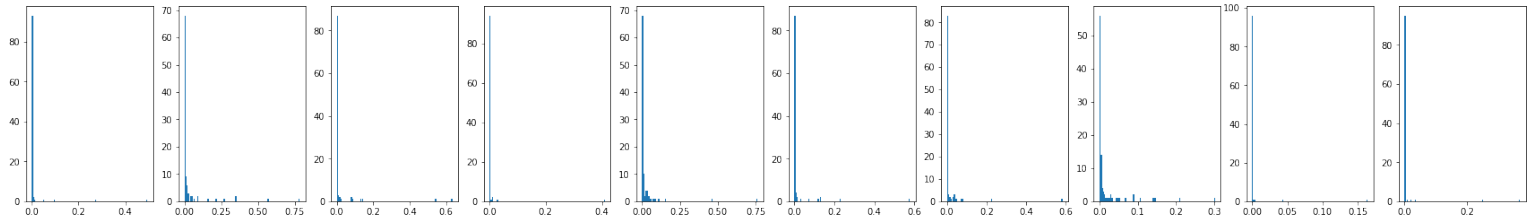
(c) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.565*



(d) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.005$ - *Sparsity (Hoyer 2004): 0.615*

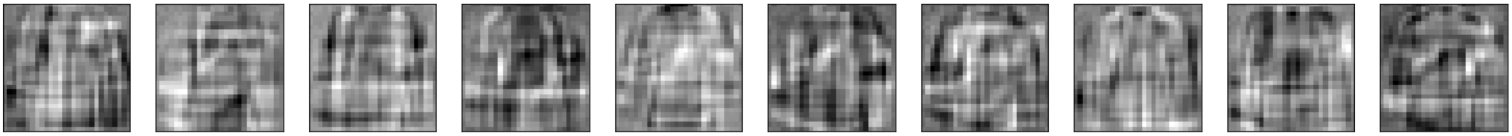


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.676*

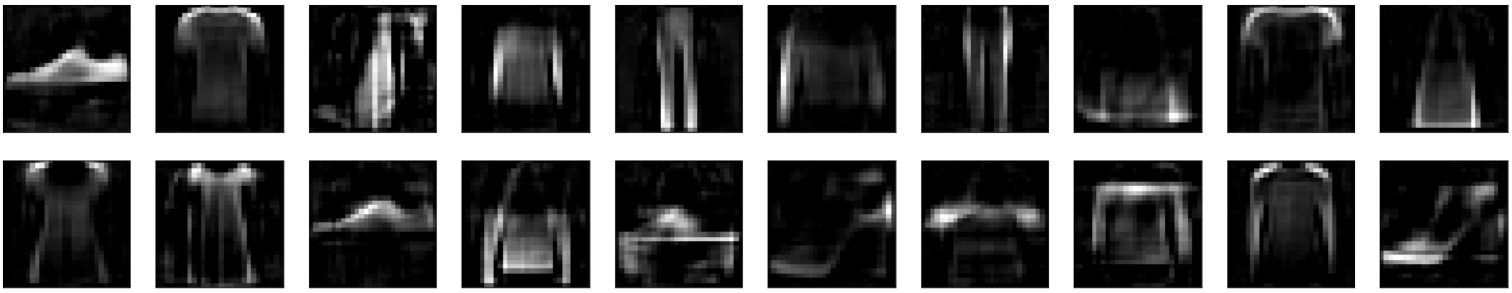


(f) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$ - *Sparsity (Hoyer 2004): 0.826*

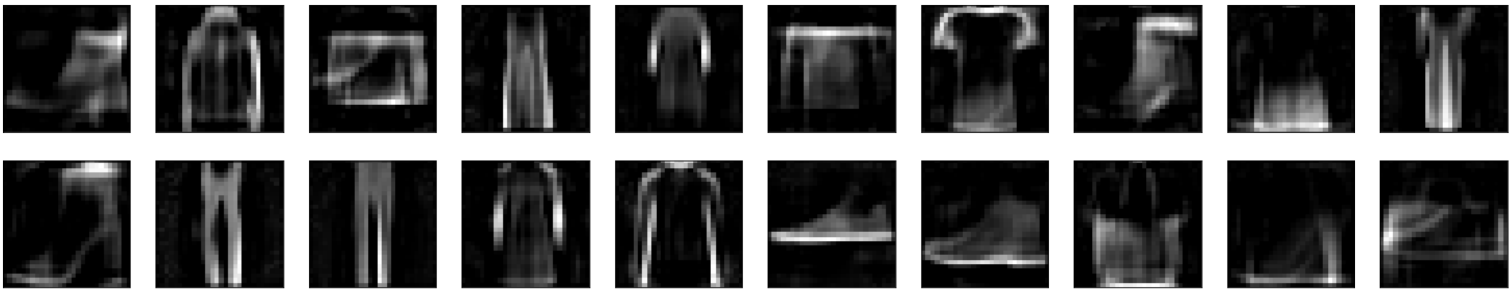
Figure 12: Histograms of the encoding of each of the 10 original images for the various versions of the Asymmetric Auto-Encoder



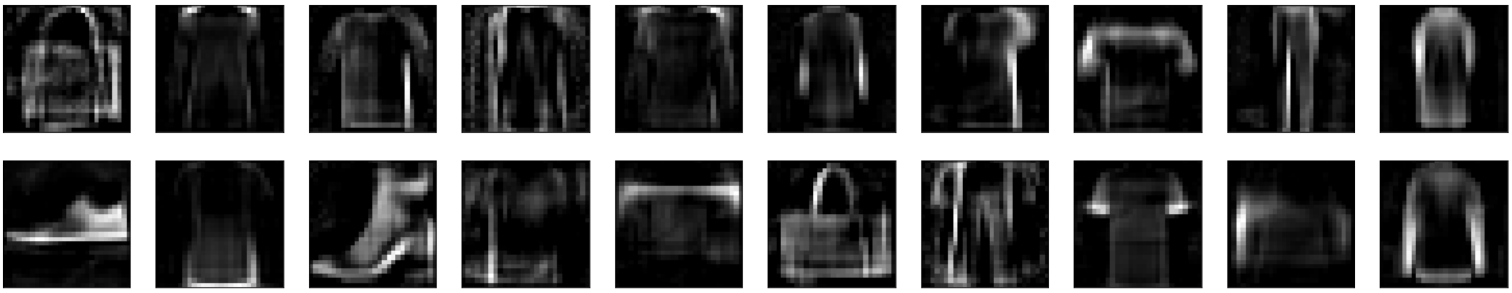
(a) Asymmetric Auto-encoder with no constraints



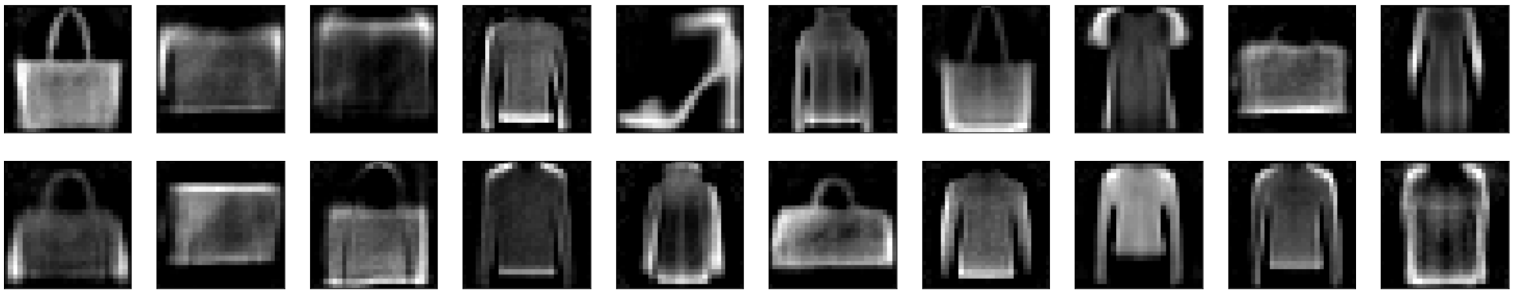
(b) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$



(c) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.005$

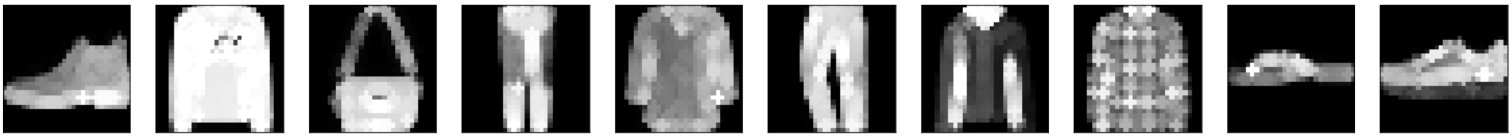


(d) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$

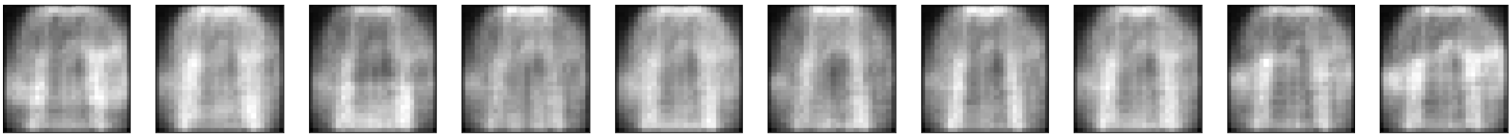


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$

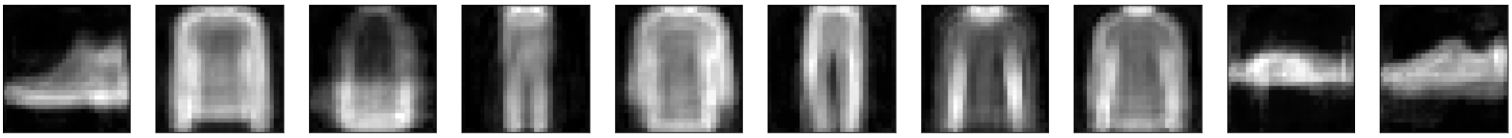
Figure 13: Some atoms (out of the 100 atoms) of the various versions of the asymmetric Auto-Encoder



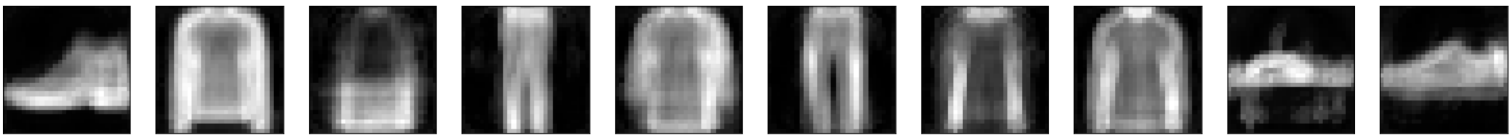
(a) Dilation of the original images by disk of radius 1



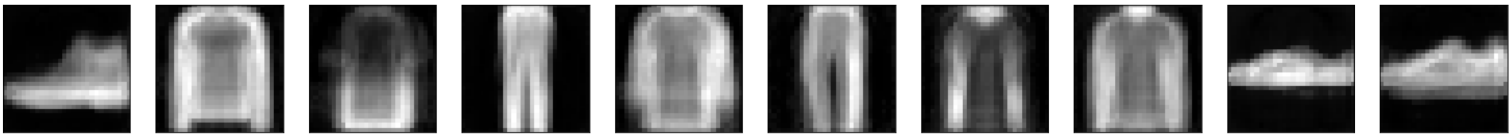
(b) Asymmetric Auto-encoder with no constraints - *Max-Approximation error: 14.31*



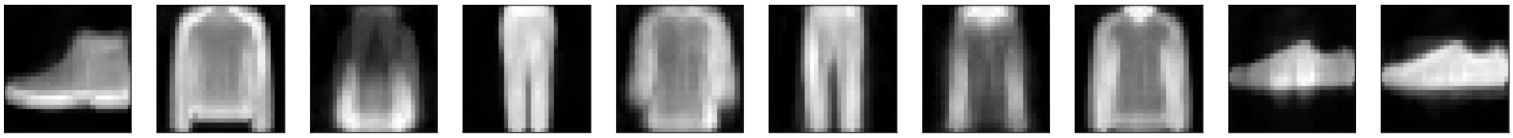
(c) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *Max-Approximation error: 0.202*



(d) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.005$ - *Max-Approximation error: 0.123*

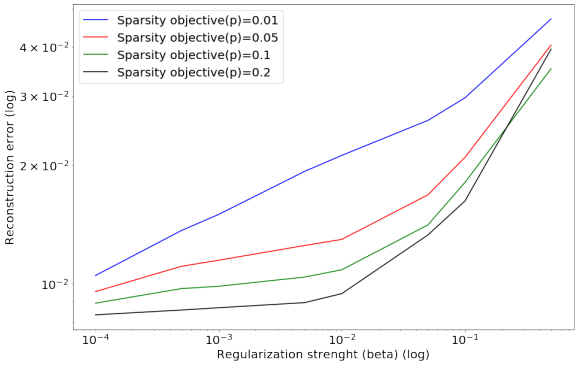


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$ - *Max-Approximation error: 0.103*

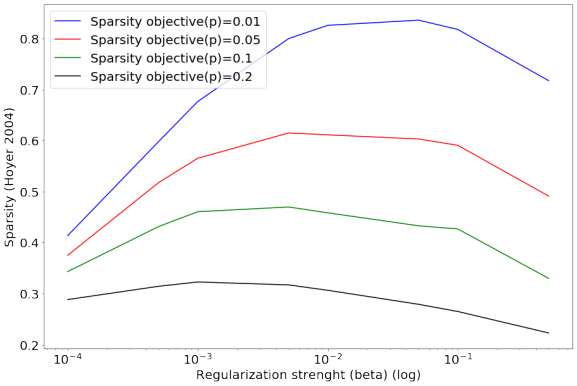


(f) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$ - *Max-Approximation error: 0.0297*

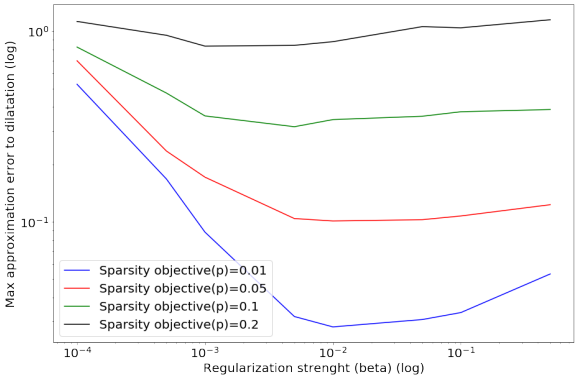
Figure 14: Max-approximation to the dilation by a disk of radius 1



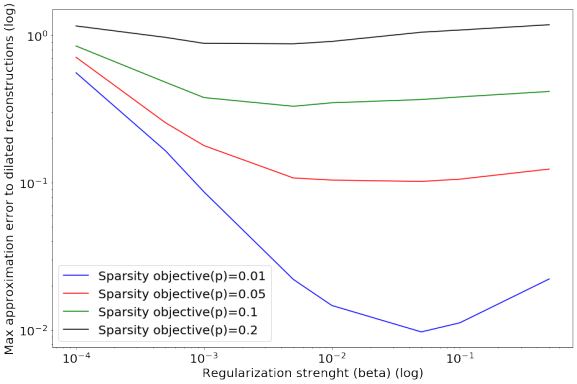
(a) Reconstruction error as a function of the parameter β (sparsity penalty strength)



(b) Sparsity metric (Hoyer 2004) as a function of the parameter β (sparsity penalty strength)



(c) Max-approximation error to dilation (of the original images) as a function of the parameter β (sparsity penalty strength)



(d) Max-approximation error to dilation (of the reconstructed images) as a function of the parameter β (sparsity penalty strength)

Figure 15: Various metrics evaluated on sparse non-negative asymmetric auto-encoders for various parameters of the sparsity regularization, using a test set not used to train the network

The maximum of the sparsity metric was reached with the sparsity parameters $p = 0.1$ and $\beta = 0.1$, whose atoms are showed in Figure 13e, where full clothes shapes can be seen, illustrating the phenomenon described in the previous section, in which the encoding process performs a kind of k-mean algorithm, the atoms being the cluster centers.

A major improvement brought by the deep encoder can be seen by comparing the atom images from both the shallow architecture and the asymmetric architecture (Figure 8 and Figure 13 respectively), where one could notice that all atoms are used when the encoder is deep, while many of them were noisy or black images (usually associated with low code coefficient for all of the images of the data set) in the shallow architecture. The representation showed in Figures 13b, 13c and 13d are quite close to a part-based representation, even though the supports of the atom images are less disjoint as they would be in an ideal part-based representation, such as the sparse NMF, whose atom images are very neat. We note that this three sets of parameters lead to both good reconstructions and good max-approximations to the dilation by a disk of radius one (showed in Figure 14).

Note that the bias of the decoder, as well as the weights and biases of the encoder may not be positive.

Finally, Figure 16 shows how the quality of the max-approximation to dilation increases with the sparsity of the encoding of the images. This validates our approach of enforcing the sparsity of the encoding to improve the quality of the max-approximation, as proposed by the framework introduced in [23], which assumes that enforcing sparsity of the encoding tends to make weighted atoms used to approximate each image almost disjoint, and therefore pull the weighted linear combination of atom images processed by the decoder close to a supremum operation between our weighted set of images, which indeed commute with the dilation.

Figure 17 shows the max-approximation error as a function of the reconstruction error. Recall from Figure 15a that the reconstruction error is an increasing function of the sparsity penalty weight β for a given p. The x axis of Figure 17 is therefore also ordered by increasing sparsity penalty weight. We can see that there is a kind of trade-off between reconstruction quality and max-approximation accuracy. The flat regions are due to the fact too strong sparsity penalties lead to increasing reconstruction errors, resulting from blurry or black atoms, leading to slightly increasing max-approximation error.

Model	Sparsity Parameters	Reconstruction error	Sparsity (Hoyer 2004)	Max-approximation error to dilation	SVM classification accuracy (std)
Sparse NMF	$S_h = 0.6$	0.0109	0.6504	0.107	0.812 (0.036)
Unconstrained Asymmetric AE	-	0.00646	0.0956	14.31	0.787(0.036)
Sparse Non-Negative Asymmetric AE	$p = 0.05$ $\beta = 0.001$	0.0115	0.565	0.202	0.834(0.036)
Sparse Non-Negative Asymmetric AE	$p = 0.05$ $\beta = 0.005$	0.0125	0.615	0.123	0.81(0.034)
Sparse Non-Negative Asymmetric AE	$p = 0.01$ $\beta = 0.001$	0.0150	0.676	0.103	0.796(0.039)
Sparse Non-Negative Asymmetric AE	$p = 0.01$ $\beta = 0.01$	0.0212	0.826	0.0297	0.78(0.033)

Table 2: Comparison of some of our asymmetric deep auto-encoders and the sparse NMF model using the four metrics presented in Section 1.6, evaluated on a test set left out for the training of the auto-encoders.

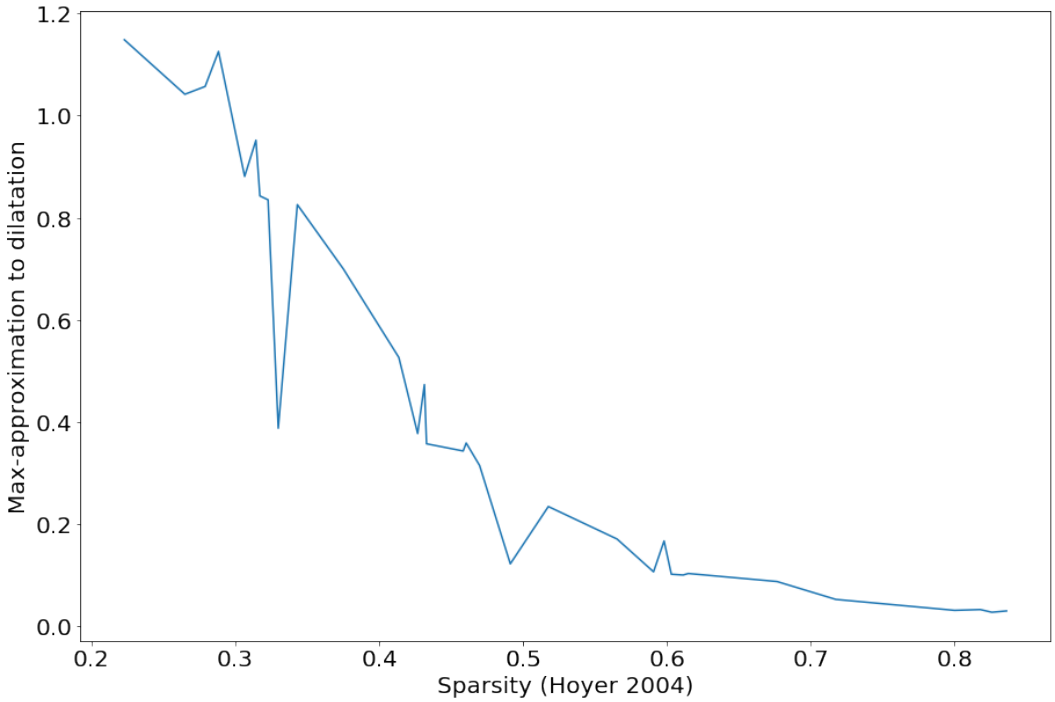


Figure 16: Max-approximation error as a function of sparsity (Hoyer 2004) of the encoding by the asymmetric auto-encoder for all the tested sparsity parameters.

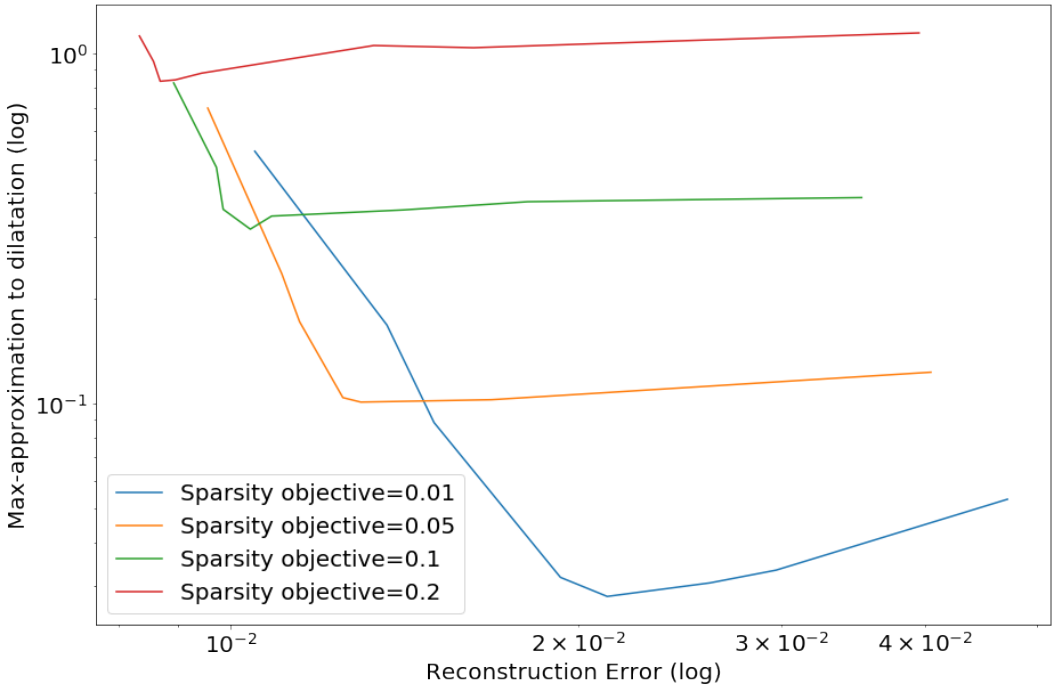


Figure 17: Max-approximation error as a function of the reconstruction error by the asymmetric auto-encoder for all the tested sparsity parameters.

5 Conclusion and possible continuation

5.1 Conclusion

During this study, we have seen that the NMF with sparsity constraints algorithm used in [23] could be interestingly replaced by auto-encoder architectures in order to approximate the effect of the dilation morphological operator. We empirically noticed that enforcing sparsity and non-negativity constraints indeed led our networks to produce quite accurate approximations of both the original images and of the dilation by a disk of radius 1, while those two criteria for an efficient encoding seems to be at the expense of one another. We empirically verified that those constraints indeed produced part-based-like representations, even though the atom images we got were less localized than those obtained using the NMF algorithm. In both cases, the part-based representation turned the weighted linear combination of atoms into an approximation of the supremum of the weighted atoms used to approximate each image. This explains why our max-approximation framework remains accurate to approximate the dilation operator.

5.2 Suggested improvements and continuation

The following line of my study was based on multi-scales morphological decompositions of the input images before learning a part-based representation with the auto-encoder, in order to explicit the texture and structure information of the images. These decompositions being highly redundant, our hope was that the network would learn to select and combine the relevant information to learn a more efficient part-based representation. Two different decompositions were considered and implemented:

- A positive decomposition based on the differences between openings by reconstruction with decreasing size of structuring elements.
- An Additive Morphological Decomposition, from [22], composed of both residuals between openings by reconstruction with decreasing size of structuring elements and closings by reconstruction with increasing size of structuring elements.

Unfortunately, to the date of the writing of this internship report, I did not manage to get compelling results using these pre-processings of the input data, and therefore chose not to include them in this report. I will however focus the end of my internship on this matter.

At the end of this internship, many ways of pursuing and improving my work seems possible:

- First of all, we fixed the dimension of the dictionary $k = 100$ at the beginning of our study, for the sake of simplicity. It is very likely that the value of this parameter of our network may have a great impact on the learned representation, and on the max-approximation error.
- Many parameters of our network were arbitrarily chosen: architecture of the encoder, activation functions, and more importantly the implementations of sparsity and non-negativity constraints. We chose the KL divergence sparsity regularizer and the non-negativity re-projection constraints after implementing and trying some other combinations of methods with our first shallow architecture, but a more extensive study of the impact of those various elements may be of great interest.
- As mentioned above, the dictionaries of images learned by our sparse and non-negative auto-encoder architectures lack localized features that correspond more closely to the intuitive conception of part-based representation, which is witnessed when using the NMF algorithm, as seen in Section 2. Maybe enforcing the sparsity of the weights of the decoder, and not only of the encoding, may lead to more localized features in the atoms of the learned dictionary. It may also enforce the atoms to be closer to being pair-wise disjoint, and thus improve the max-approximation accuracy.

- We could also modify the objective function of our auto-encoders or add another regularizer to enforce the atoms to be pair-wise disjoint, or directly train the network in order to reduce the max-approximation error.
- We noticed that good trade-offs between reconstruction and max-approximation errors usually come with a sparsity metric (Hoyer 2004) close to 0.6, whatever the methods employed (NMF with sparsity constraint, shallow auto-encoder or asymmetric auto-encoder). Maybe the sparsity regularization of the encoding could be replaced by a function of this sparsity metric and of this objective value of 0.6.
- Finally, we could directly try to learn a factorization in the max-plus algebra, in which case we could replace the decoder linear connections by a max-plus layer that could enforce the network to learn a decomposition of the images by max of its parts, which would commute with the dilation.

5.3 Acknowledgments

I would like to thank my supervisors Jesus Angulo, Santiago Velasco-Forero, Samy Blusseau and Isabelle Bloch for their availability, kindness and for sharing with me their knowledge and inspirations. Many thanks to the Center For Mathematical Morphology for welcoming me and especially to Mrs.Anne-Marie De Castro.

References

[1] Jesus Angulo and Santiago Velasco-Forero. "Sparse mathematical morphology using non-negative matrix factorization". In: *10th International Symposium on Mathematical Morphology and Its Application to Signal and Image Processing, ISMM 2011*. Ed. by Martino Pesaresi Pierre Soille and Georgios K. Ouzounis. Vol. 6671. Lecture Notes in Computer Science. ISBN: 978-364221568-1. Verbania-Intra, Italy: Springer, July 2011, pp. 1–12. DOI: 10.1007/978-3-642-21569-8_1. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-00658963.

[2] Babajide O. Ayinde and Jacek M. Zurada. "Deep Learning of Constrained Autoencoders for Enhanced Understanding of Data". In: *CoRR* abs/1802.00003 (2018). arXiv: 1802.00003. URL: http://arxiv.org/abs/1802.00003.

[3] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. "Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives". In: *CoRR* abs/1206.5538 (2012). arXiv: 1206.5538. URL: http://arxiv.org/abs/1206.5538.

[4] Heijmans H. Bloch I. and Ronse C. "Mathematical Morphology". In: *Handbook of Spatial Logics*. Springer Netherlands, 2007. Chap. 14.

[5] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *CoRR* abs/1606.03657 (2016). arXiv: 1606.03657. URL: http://arxiv.org/abs/1606.03657.

[6] David Donoho and Victoria Stodden. "When Does Non-Negative Matrix Factorization Give Correct Decomposition into Parts?" In: MIT Press, 2003, p. 2004.

[7] I. J. Goodfellow et al. "Generative Adversarial Networks". In: *ArXiv e-prints* (June 2014). arXiv: 1406.2661 [stat.ML].

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[9] Zhenxing Guo and Shihua Zhang. "Sparse Deep Nonnegative Matrix Factorization". In: *CoRR* abs/1707.09316 (2017). arXiv: 1707.09316. URL: http://arxiv.org/abs/1707.09316.

[10] Song Han et al. “DSD: Regularizing Deep Neural Networks with Dense-Sparse-Dense Training Flow”. In: *CoRR* abs/1607.04381 (2016). arXiv: 1607.04381. URL: http://arxiv.org/abs/1607.04381.

[11] G E Hinton and R R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (July 2006), pp. 504–507. DOI: 10.1126/science.1127647. URL: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google.

[12] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui. “Deep Learning of Part-Based Representation of Data Using Sparse Autoencoders With Nonnegativity Constraints”. In: *IEEE Transactions on Neural Networks and Learning Systems* 27.12 (Dec. 2016), pp. 2486–2498. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2015.2479223.

[13] Patrik O. Hoyer. “Non-negative matrix factorization with sparseness constraints”. In: *CoRR* cs.LG/0408058 (2004). URL: http://arxiv.org/abs/cs.LG/0408058.

[14] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[15] Daniel D. Lee and H. Sebastian Seung. “Learning the parts of objects by non-negative matrix factorization”. In: *Nature* 401 (Oct. 1999), 788 EP -. URL: http://dx.doi.org/10.1038/44565.

[16] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. “Sparse deep belief net model for visual area V2”. In: *Advances in Neural Information Processing Systems 20*. Ed. by J. C. Platt et al. Curran Associates, Inc., 2008, pp. 873–880. URL: http://papers.nips.cc/paper/3313-sparse-deep-belief-net-model-for-visual-area-v2.pdf.

[17] Andre Lemme, René Felix Reinhart, and Jochen J. Steil. “Online learning and generalization of parts-based image representations by non-negative sparse autoencoders”. In: *Neural networks : the official journal of the International Neural Network Society* 33 (2012), pp. 194–203.

[18] Julien Mairal, Francis R. Bach, and Jean Ponce. “Sparse Modeling for Image and Vision Processing”. In: *CoRR* abs/1411.3230 (2014). arXiv: 1411.3230. URL: http://arxiv.org/abs/1411.3230.

[19] Andrew Ng. “Sparse Auto-Encoders”. In: *CS294A Lecture notes* (2011). URL: https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

[20] Bruno A. Olshausen and David J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”. In: *Nature* 381 (June 1996), 607 EP -. URL: http://dx.doi.org/10.1038/381607a0.

[21] David Ross and David A. Ross. “Learning Parts-Based Representations of Data”. In: *JMLR* 7 (2003), pp. 2369–2397.

[22] Santiago Velasco-Forero and Jesus Angulo. “Classification of hyperspectral images by tensor modeling and additive morphological decomposition”. In: *Pattern Recognition* 46.2 (Feb. 2013), pp. 566–577. DOI: 10.1016/j.patcog.2012.08.011. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-00751338.

[23] Santiago Velasco-Forero and Jesus Angulo. “Non-Negative Sparse Mathematical Morphology”. In: *Advances in Imaging and Electron Physics*. Vol. 202. Elsevier Inc.Academic Press, 2017. Chap. 1. URL: %5Curl%7Bhttps://www.sciencedirect.com/science/article/pii/S1076567017300411%7D.

[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[25] Li Zhang and Yaping Lu. “Comparison of auto-encoders with different sparsity regularizers”. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. July 2015, pp. 1–5. DOI: 10.1109/IJCNN.2015.7280364.