4 Using a deeper architecture of the encoder

4.1 Motivations and presentation of the architecture

Despite encouraging results, we note that our shallow architecture does not enables to reach as a compelling part-based representation as the sparse Non-Negative Matrix Factorization, which keeps a low reconstruction error (even-though 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 [17] 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 along the domain of the data arbitrarily well. 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 unit.

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 ingoGAN architecture by Xi Chen, Yan Duan, *et al.* (2016) [2], that aims at performing interpretable representation learning based on generative adversarial networks (GANs from I. Goodfellow *et al.* [4]) 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 [11], which provides stability of the network training, as it provides normalization of the features inside the network.

TODO: explain batch-normalization ?

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 9-13 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.

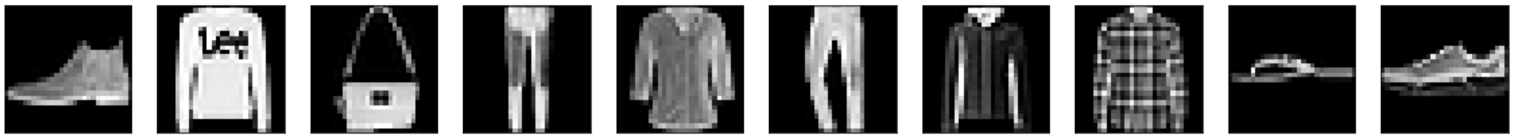
Comparing the reconstructions from Figures 9 and 4 we can note that a deeper encoder allows for stronger sparsity constraints while keeping acceptable reconstructions, usually leading to higher sparsity, with a lower reconstruction error and max-approximation error than when using a shallow encoder. Indeed, as we can see by comparing Figure 13b and Figure 8b, 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.

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 11e, 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.

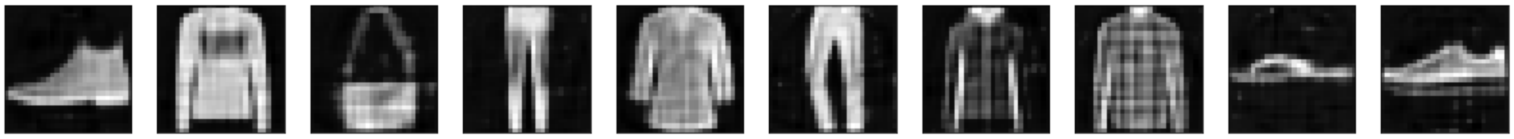
A major improvement brought by the deep encoder can be seen by comparing the atom images from both the shallow architecture and the asymmetric architecture (Figure 6 and Figure 11 respectively), where one could notice that all atoms are used when the encoder is deep, while many of them were noisy or 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 11b, 11c and 11d 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 dilatation by a disk of radius one (showed in Figure 12).

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

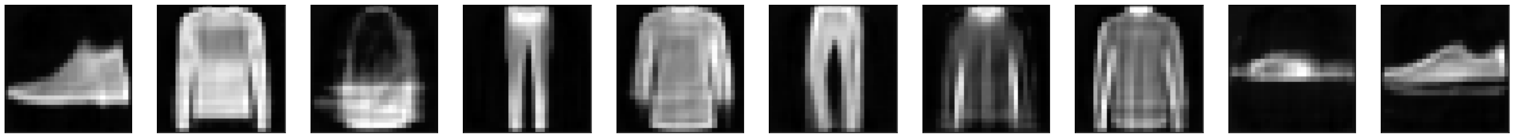
Finally, Figure 14 shows how the quality of the max-approximation to dilatation increases with the sparsity of the encoding of the images. This validates our approach of enforcing the sparsity of the encoding to improve the quality of the max-approximation.



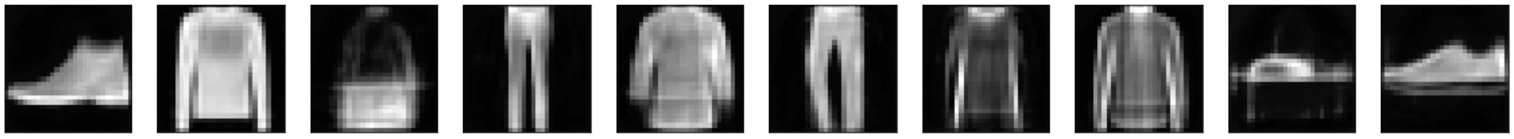
(a) Original Images



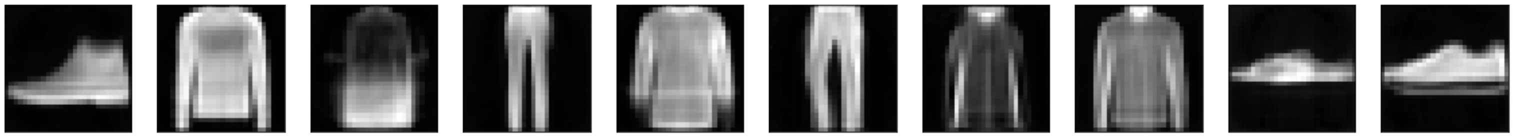
(b) Asymmetric Auto-encoder with no constraints - *reconstruction error: 0.00646*



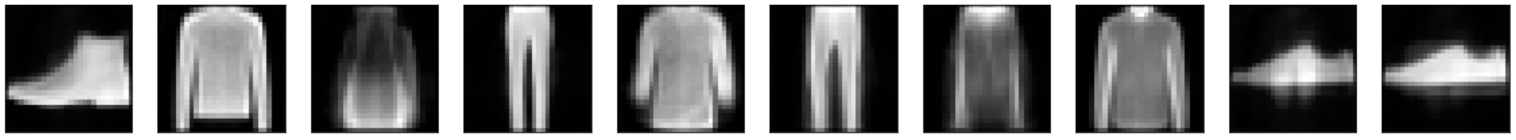
(c) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *reconstruction error: 0.0115*



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

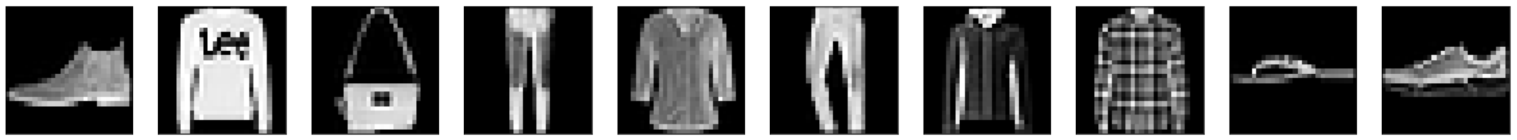


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$ - *reconstruction error: 0.0150*

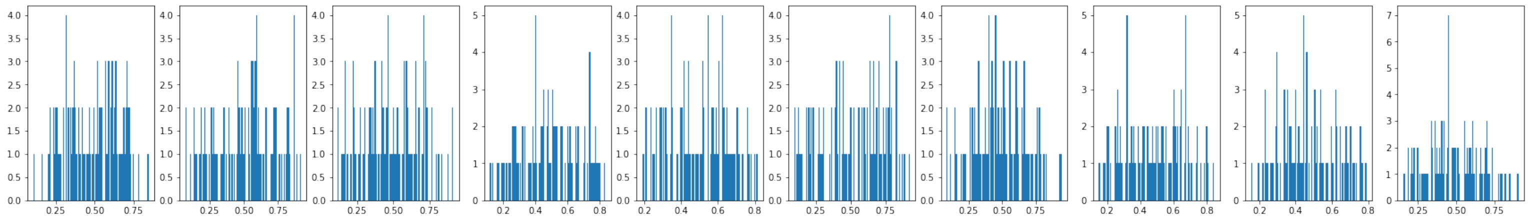


(f) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$ - *reconstruction error: 0.0212*

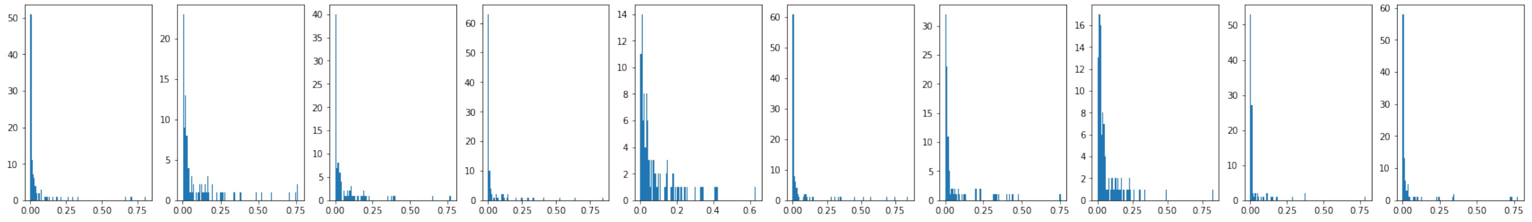
Figure 9: Reconstruction of 10 images by an Asymmetric auto-encoder with various regularizations and constraints



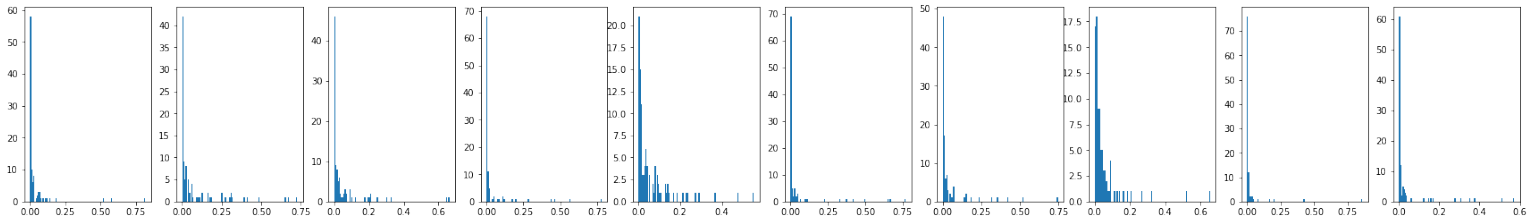
(a) Original Images



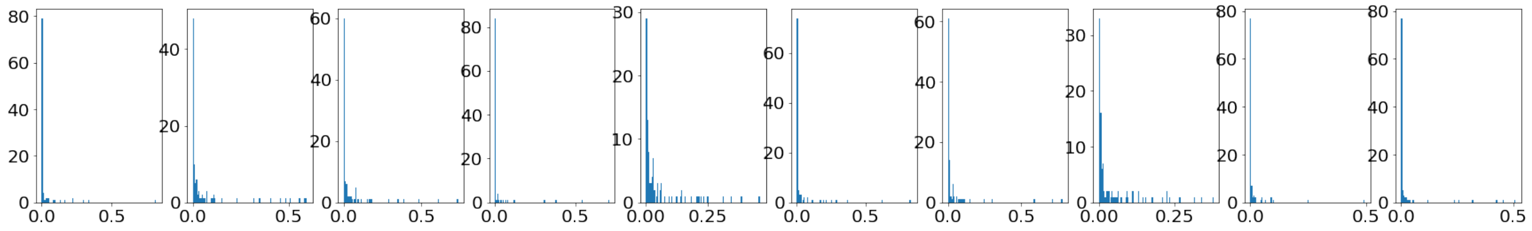
(b) Asymmetric Auto-encoder with no constraints - *Sparsity (Hoyer 2004): 0.0956*



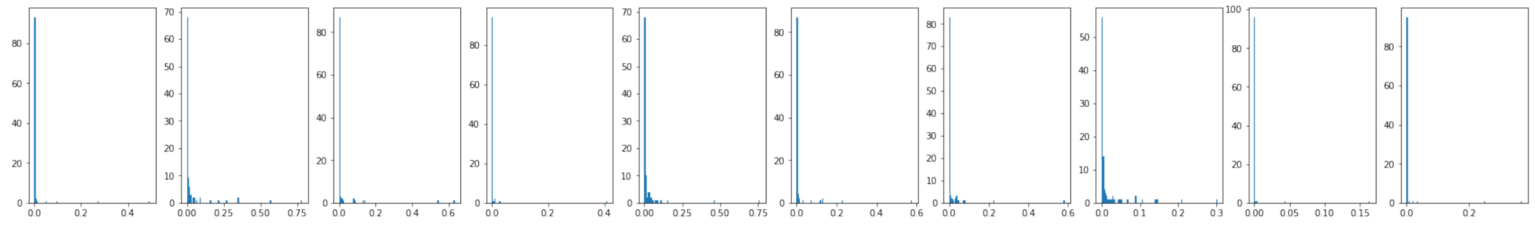
(c) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.565*



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

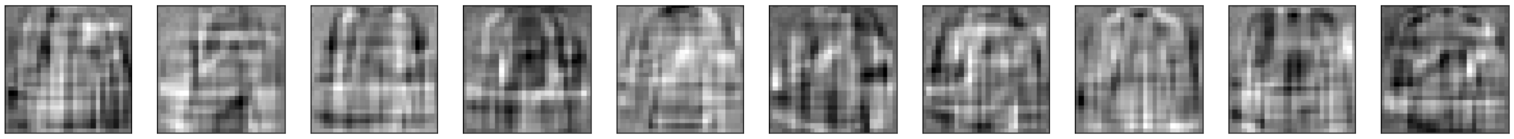


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.676*

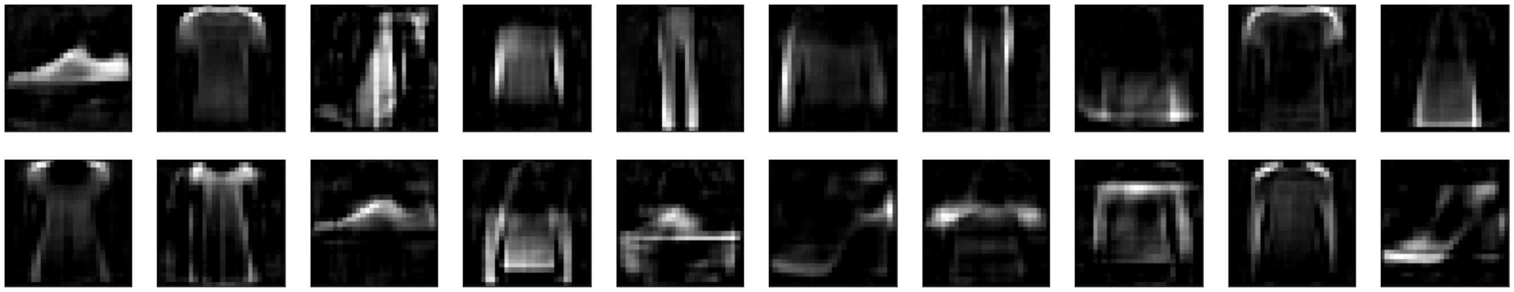


(f) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$ - *Sparsity (Hoyer 2004): 0.826*

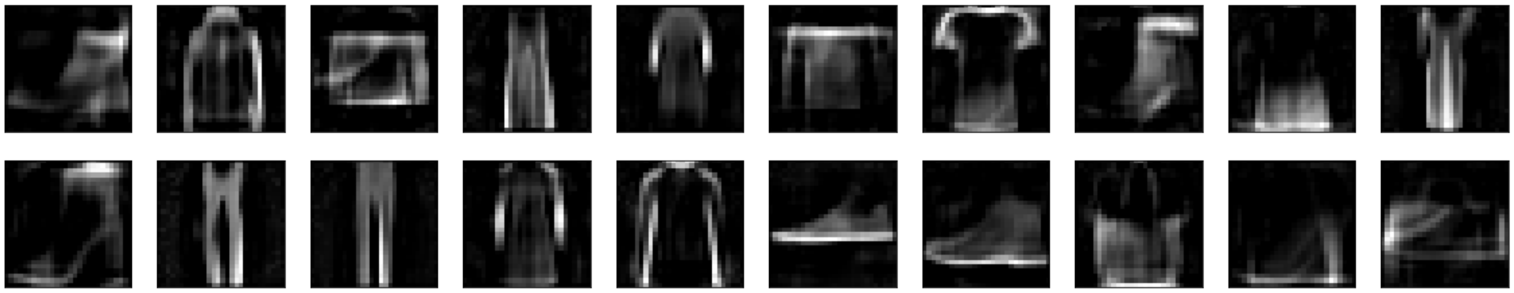
Figure 10: Histograms of the encoding of each of the 10 original images for the various versions of the Asymmetric Auto-Encoder



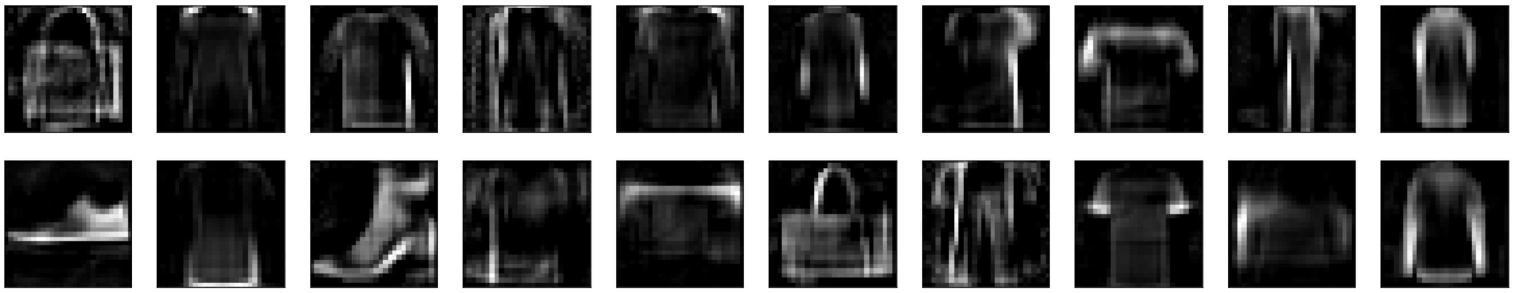
(a) Asymmetric Auto-encoder with no constraints



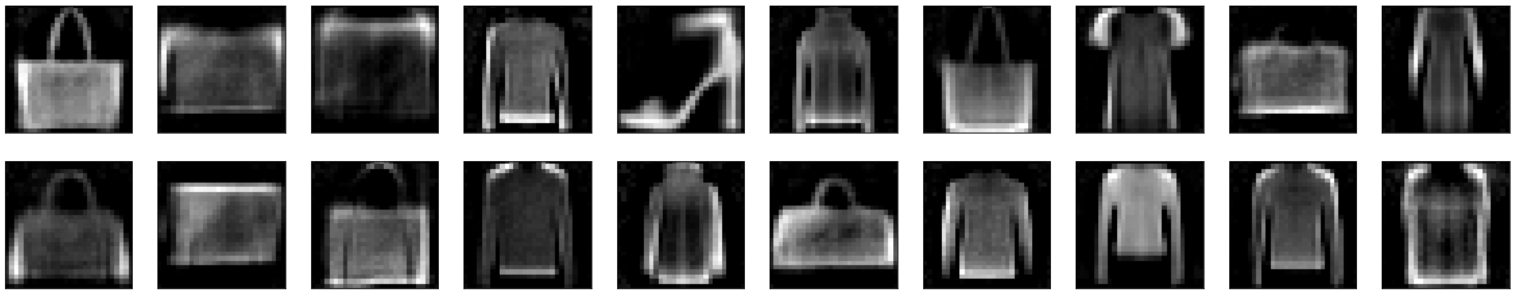
(b) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$



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

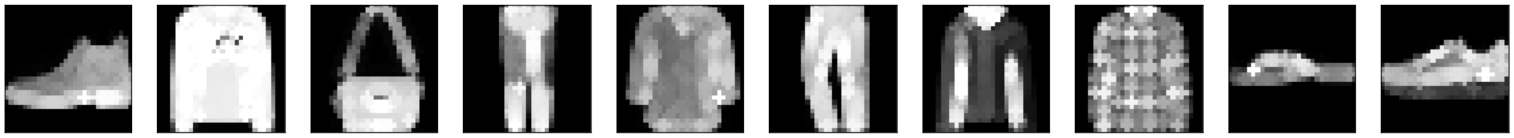


(d) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$

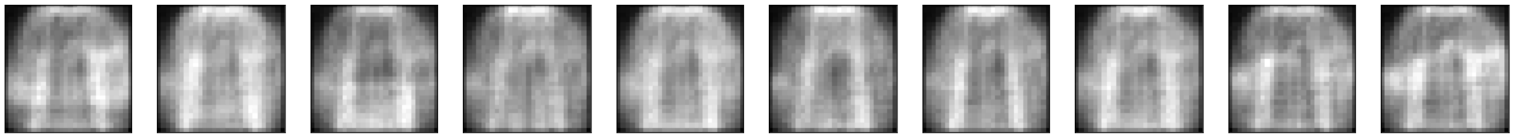


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$

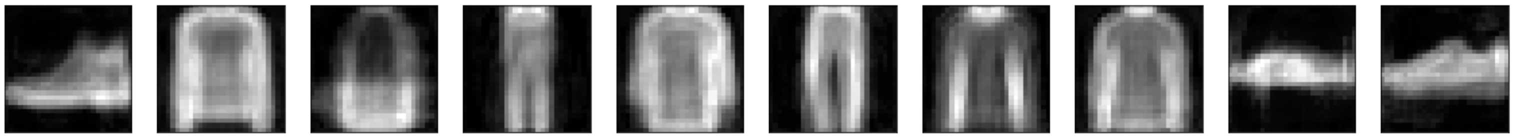
Figure 11: Some atoms (out of the 100 atoms) of the various versions of the asymmetric Auto-Encoder



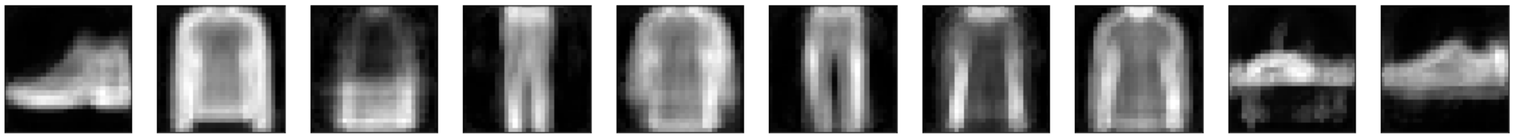
(a) Dilatation of the original images by disk of radius 1



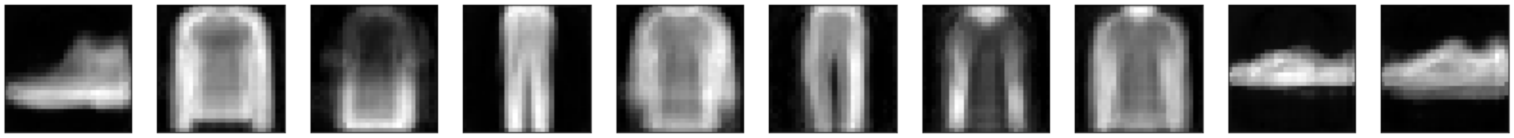
(b) Asymmetric Auto-encoder with no constraints - *Max-Approximation error: 14.31*



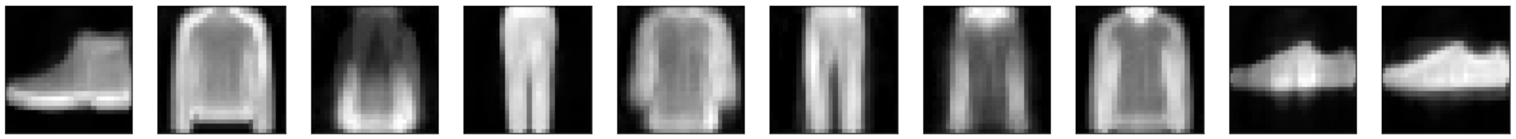
(c) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *Max-Approximation error: 0.202*



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

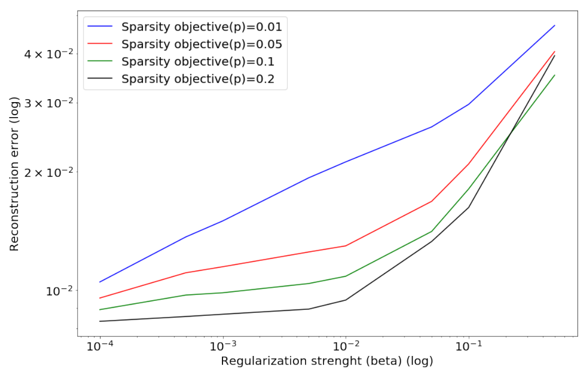


(e) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.001$ - *Max-Approximation error: 0.103*

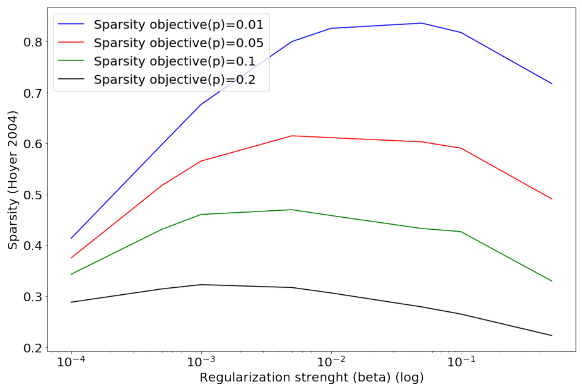


(f) Asymmetric Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.01$ - *Max-Approximation error: 0.0297*

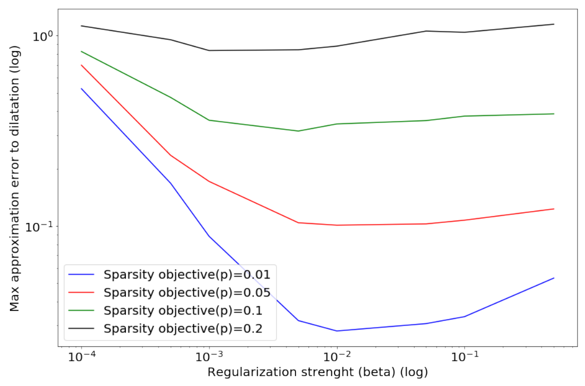
Figure 12: Max-approximation to the dilatation by a disk of radius 1



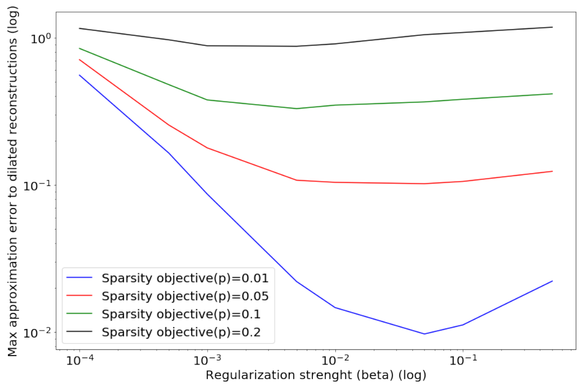
(a) Reconstruction error as a function of the parameter β (sparsity penalty strength)



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



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



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

Figure 13: 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

Model	Sparsity Parameters	Reconstruction error	Sparsity (Hoyer 2004)	Max-approximation error to dilatation	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.

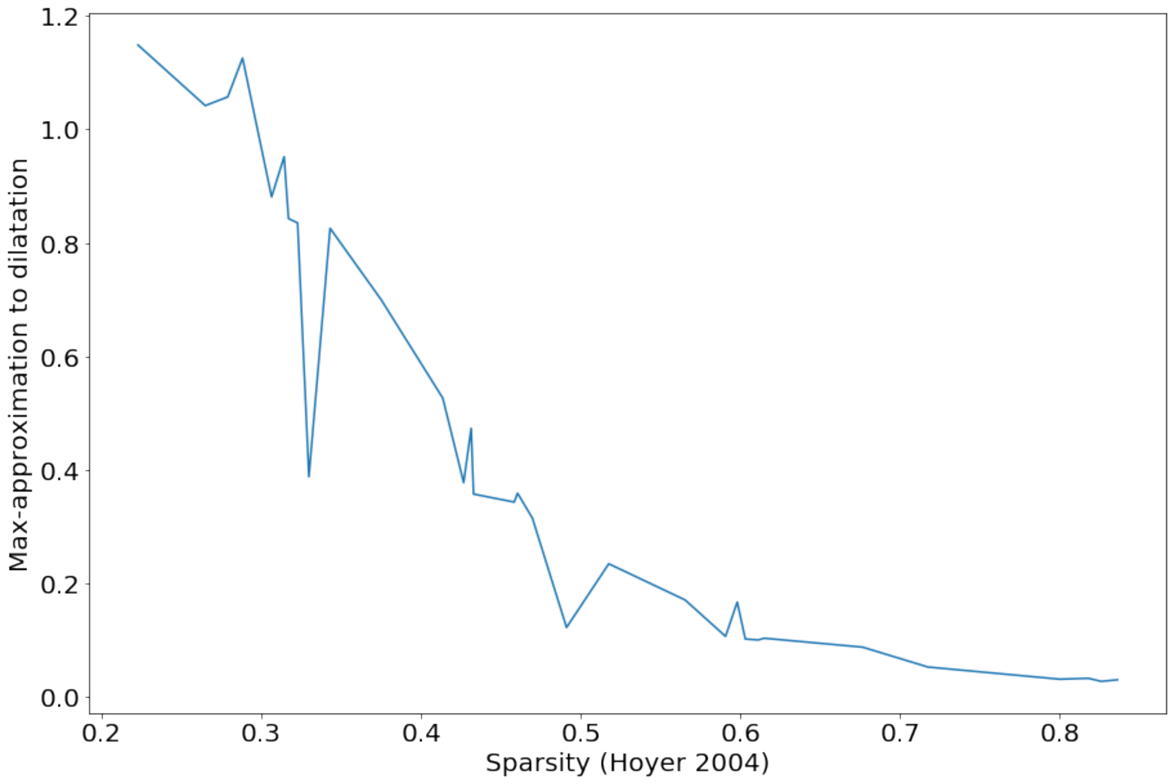


Figure 14: Max-approximation error as a function of sparsity (Hoyer 2004) of the encoding by the asymetric auto-encoder for all the tested sparsity parameters.