2 Non-Negative Matrix Factorization

2.1 General Presentation and interests

The emergence of part-based representation in human cognition can be conceptually tied to the non-negativity constraints. (Lee and Seung 1999 [9])

Why NMF is a part based decomposition [2].

2.2 Addition of sparsity constraints (Hoyer 2004)

TODO: why sparsity ? Conceptual motivations behind sparsity (V1/V2 visual cortex, [Mairal, Bach, Ponce 2012], [Lee, Ekanadham, Ng 2008], [Olshausen, Field 1997], etc.)

2.3 Implementation and application to the data

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 $h_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = \hat{\mathbf{x}} \approx \mathbf{x}$ 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** $f_{\mathbf{W}_e,\mathbf{b}_e} : \mathbf{x} \longmapsto \mathbf{h} \in \mathbb{R}^k$, 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 f 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** $L(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(i)}) = L(h(\mathbf{x}^{(i)}), \mathbf{x}^{(i)}) = L(g(f(\mathbf{x}^{(i)})), \mathbf{x}^{(i)})$ 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:

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

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 model.

The 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.)

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 Fig 1.

We hence have an auto-encoder of the form:

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

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:

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

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.

3.2 Enforcing sparsity of the encoding

Although the subject 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 a the work of H.Lee, C.Enkanadham and A. Ng [10]. In this 2008 paper, the authors presented a sparse variant of the deep belief networks proposed by Hinton *et al.* [6] 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

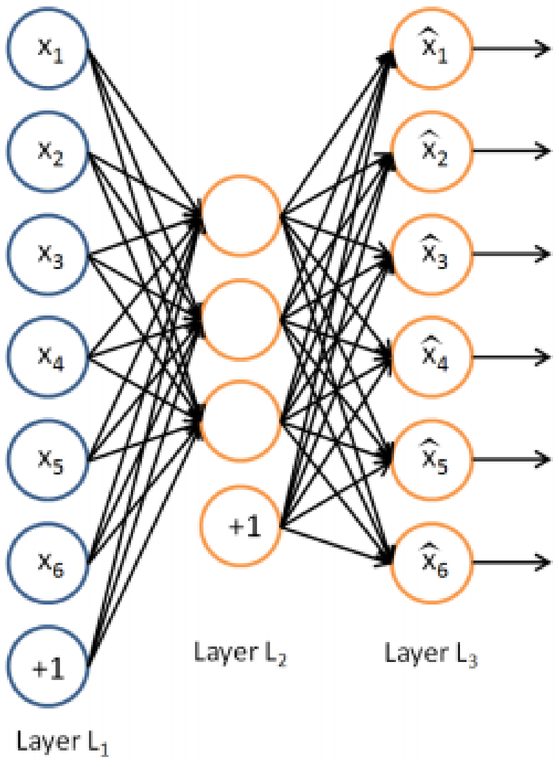


Figure 1: Three layers architecture, from [12]

a distance to the expected activation p. With our notations, the introduced regularization term can thus be written:

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

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 [10]:

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

Several works proposed their variants of the regularization function S. A paper from L.Zhang and Y.Lu (2015) [14] proposed an empirical survey on the auto-encoders with different sparsity regularizers:

- 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 controls the similarity with the L_1 norm:

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

- A non-parametric regularizer, proposed by B.A. Olshausen and D.J.Field in [13] called the Log regularizer:

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

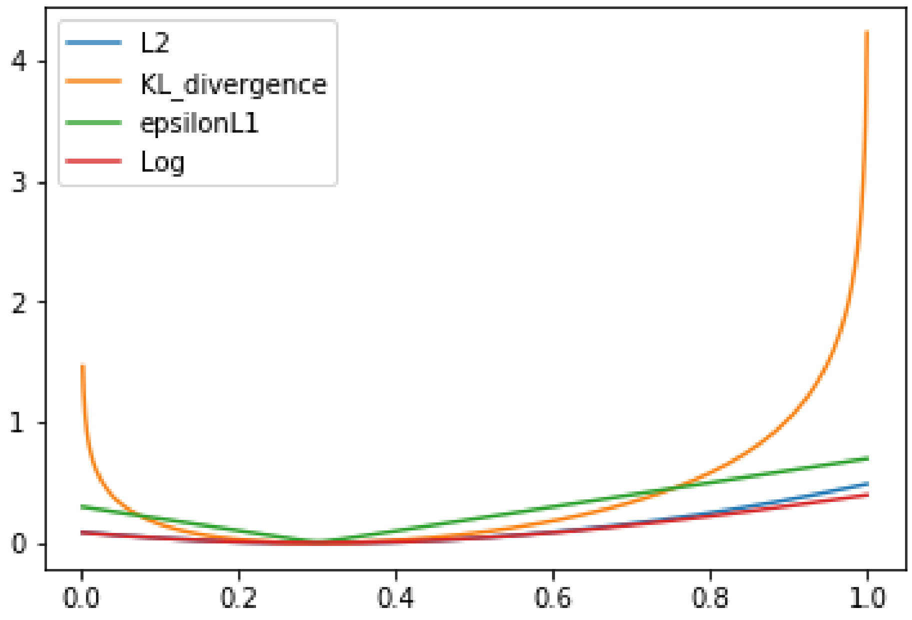


Figure 2: The four presented penalty function for $p = 0.3$.

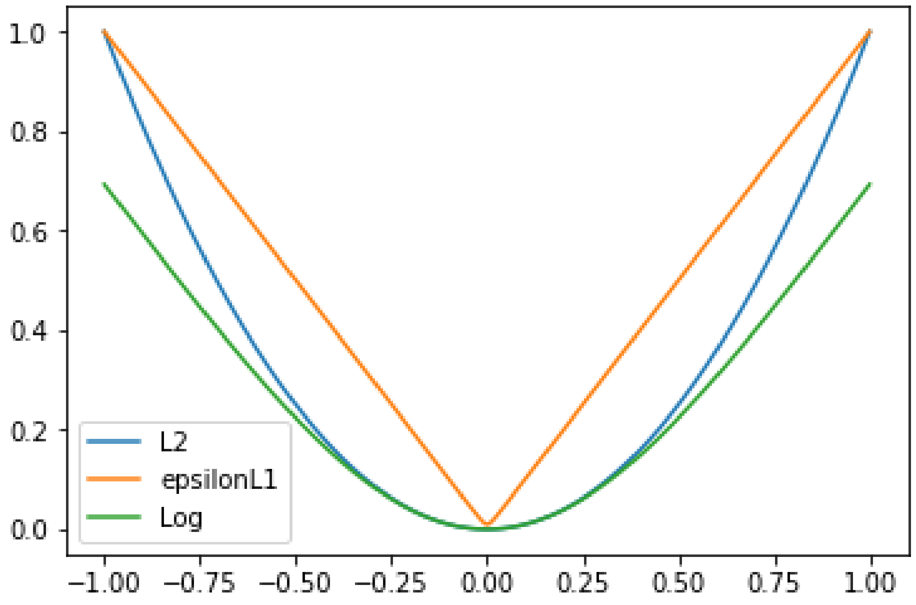


Figure 3: When the code can be negative, the KL divergence regularization may no longer be defined, and one of the three others has to be used, with $p = 0$

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 2 shows the four sparsity penalty functions we just introduced, for a given expected activation $p = 0.3$. Figure 3 shows the three penalty functions that are defined outside of]0,1]. In [14], the authors decided to consider only the ϵL_1 and the log regularizer with $p = 0$ which comes with the loss of the ability to control the degree of sparseness we want. Even-though we implemented also the ϵL_1 and the *Log* regularization functions, we decided to used the KL divergence in most of our experiments, as [14] did not enable to state whether the *Log* regularizer performed better the KL divergence one or not and as the latter is currently the most used in the literature we could read on the subject. In particular it was the one used in the related works [7] and [1].

Along my internship, I came across some other ways of enforcing sparsity of the encoding, for instance the *intrinsic plasticity* introduced in [11] 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, Pool, Narang *et al.* [5]. This paper from 2017 presents a more robust to over-fitting way of training a deep neural network by 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.

TODO: Explain the effects of the regularization using L_1 et L_2 normalization (selecting coefficients, and decreasing their magnitude respectively).

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 network, 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:

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

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

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. This approach, used in [7], does however not assure that all weights will be positive, depending on the chosen value for the parameters α and λ, as noted in [1]. Indeed, the original L_2 norm used here penalizes the weights with bigger magnitudes stronger than the those with smaller magnitudes. This forces a good number of the weights to take on small negative values, but not necessarily to be positive. [11] and [1] therefore 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 quadrant 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 [3], 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 criterion, 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 [8] (2004) and presented in section 2.2.
- The Max-Approximation error to dilatation 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 dilatation 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 dilatation of its input. That is:

$$\begin{aligned} E_{\delta_1}(h) &= \frac{1}{mN}\sum_{i=1}^{m} ||h_{\delta_1}(\mathbf{x}^{(i)}) - \delta_1(\mathbf{x}^{(i)})||_2^2 \\ \text{with } h_{\delta_1}(\mathbf{x}^{(i)}) &= g_{\delta_1}(f(\mathbf{x}^{(i)})) \\ &= \sigma_d\left(\delta_1(\mathbf{W}_d)^T\mathbf{h}^{(i)} + \mathbf{b}_d\right) \end{aligned}$$

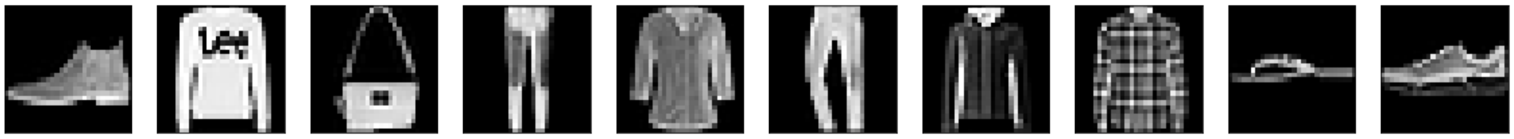
where $\delta_1(\mathbf{W}_d)$ represent the matrix whose rows are the flatten versions (in a row vector of size $(1, N)$) of the application of the dilatation δ_1 by a disk of radius 1 to each of the atom images of the decoder $\mathbf{W}_{d,j}$ (seen as $d \times d$ images, where in our case $d = 28$ pixels). h_{δ_1} is an auto-encoder whose encoder is the same as the encoder f of the trained auto-encoder h, and whose decoder g_{δ_1} is obtained by applying the dilatation δ_1 to each atom image of the decoder g of the trained auto-encoder. Note that the dilatation 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 (by the encoding coefficients) sum of atom images.

Figures 4-8 shows 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.

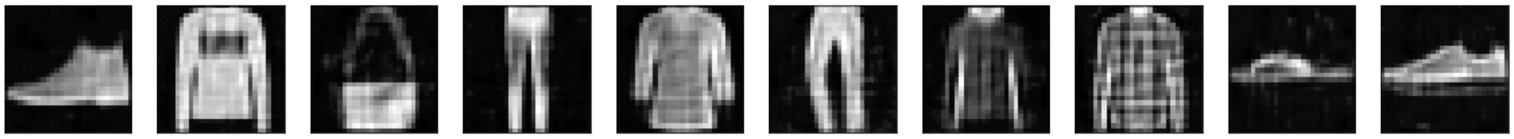
Figure 4 shows the reconstruction by our shallow auto-encoder, with various non-negativity and sparsity constraints, of 10 images from a test set not used to train the model. As we can see in Figure 5, the results are ordered by increasing sparsity constraint, the second row representing the results 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 8a. Figure 6 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 where 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 no setting gives an as efficient decomposition as the sparsity constrained NMF introduced by Hoyer in 2004 ([8]).

It is worth noting in Figure 8b that the sparsity metrics 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 equals, 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 blacks but a few pixels, resulting in blurry reconstructions dictated by the bias image, and hence independent of the input.

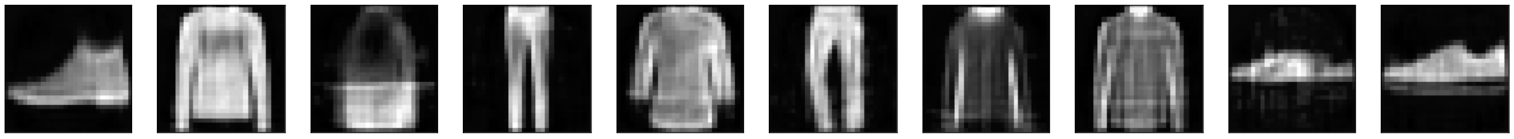
Finally, Figure 7 shows the max-approximation to dilatation by a disk of radius 1 of the various shallow auto-encoders. We can see from Figure 8c that the sparsity of the encoding improves the quality of the max-approximation, except when it reaches the points where all coefficients of the encoding average zero. Figure 8d shows the mse between the max-approximation to dilatation and the dilatation of the reconstructed images (instead of the dilatation of the original ones), this measure is less informative than its Figure 8c counterpart as it can get pretty low when the reconstruction is very blurry.



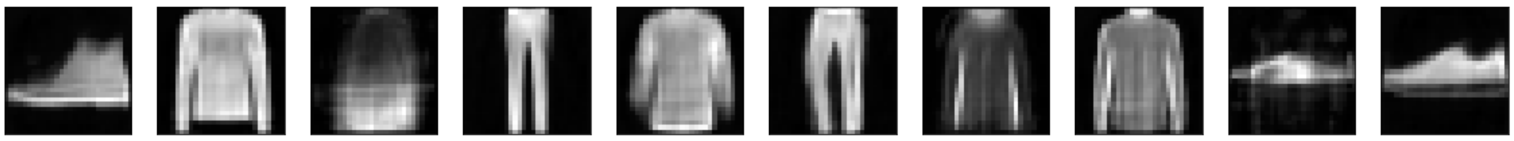
(a) Original Images



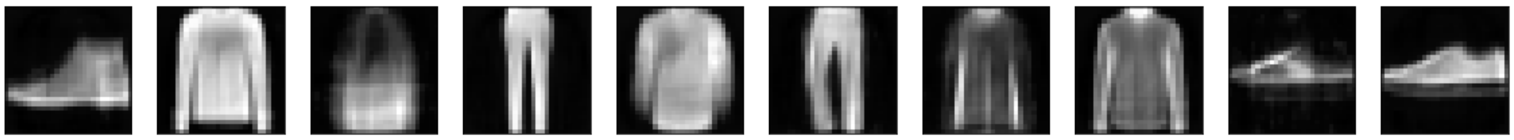
(b) Shallow Auto-encoder with no constraints - *reconstruction error: 0.00697*



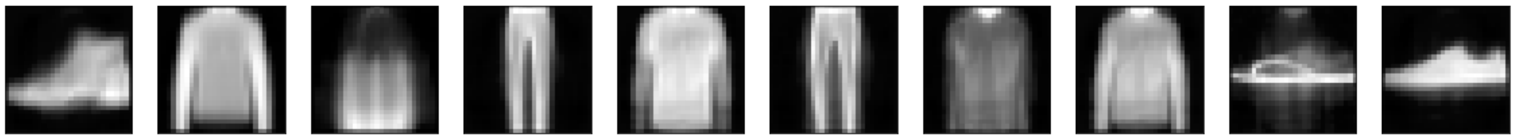
(c) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.2$, $\beta = 0.001$ - *reconstruction error: 0.0103*



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

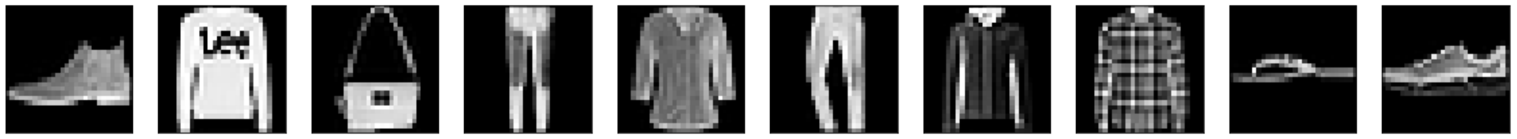


(e) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *reconstruction error: 0.0164*

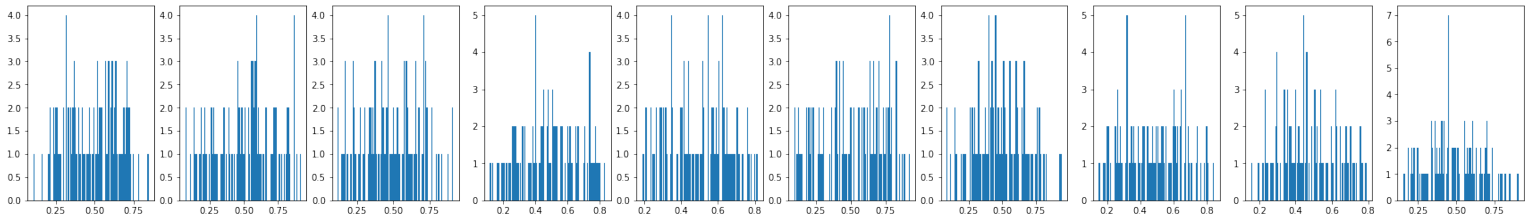


(f) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$ - *reconstruction error: 0.0288*

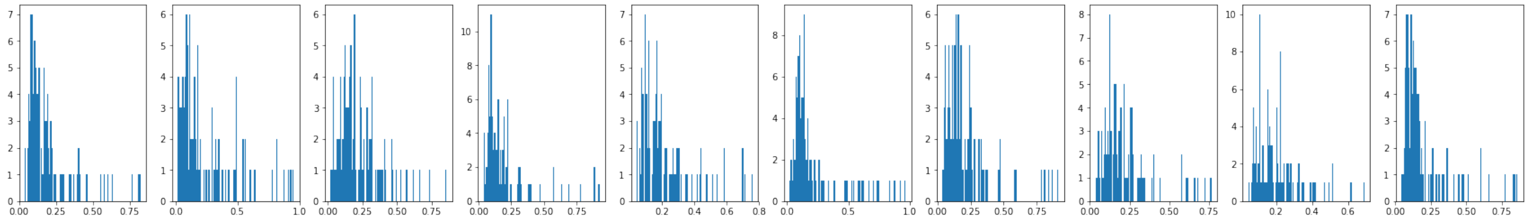
Figure 4: Reconstruction of 10 images by a shallow auto-encoder with various regularizations and constraints



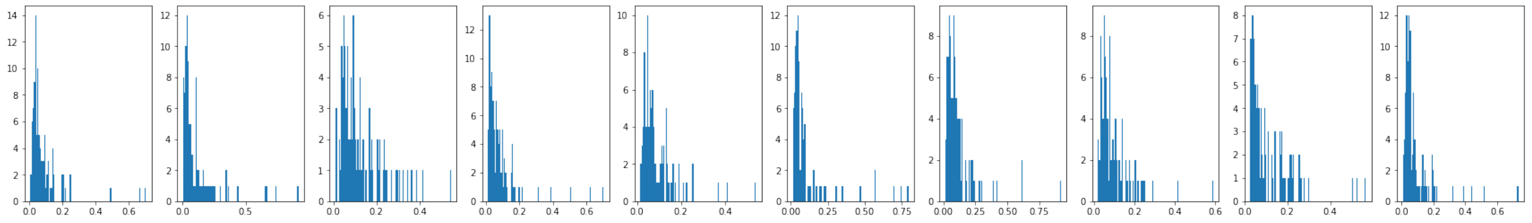
(a) Original Images



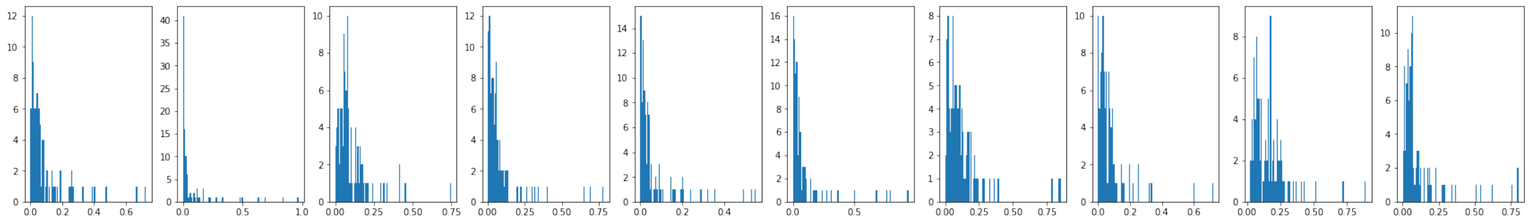
(b) Shallow Auto-encoder with no constraints - *Sparsity (Hoyer 2004): 0.0678*



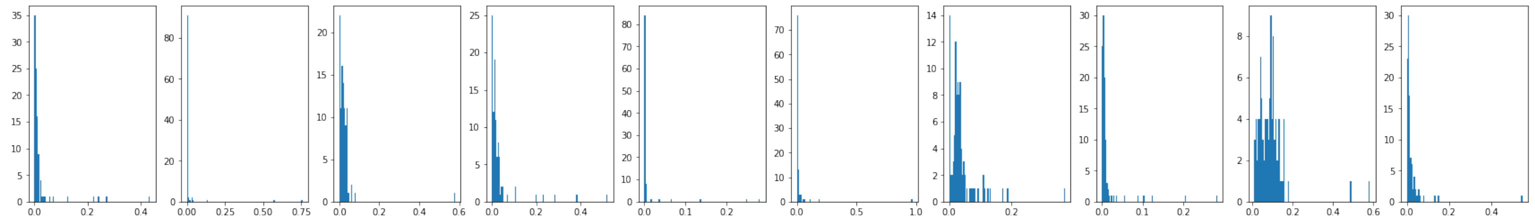
(c) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.2$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.230*



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

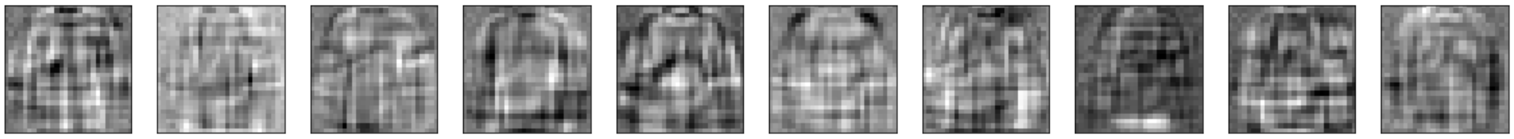


(e) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$ - *Sparsity (Hoyer 2004): 0.505*

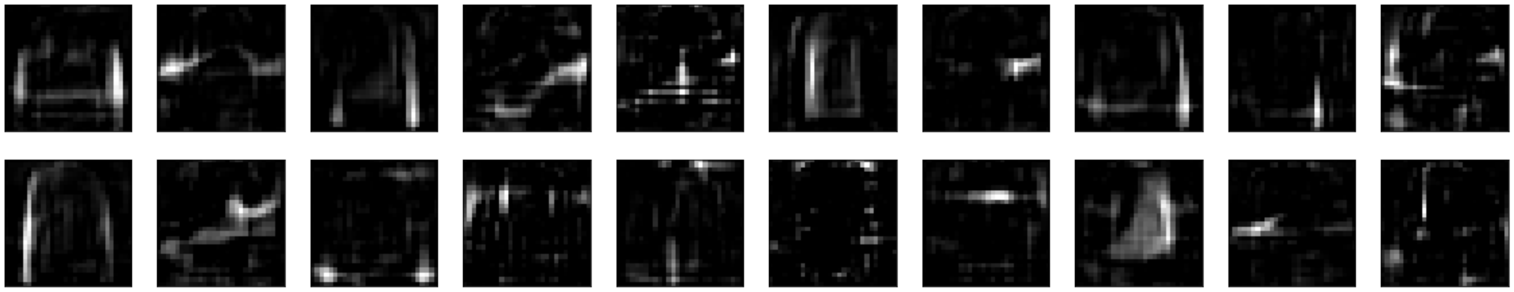


(f) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$ - *Sparsity (Hoyer 2004): 0.785*

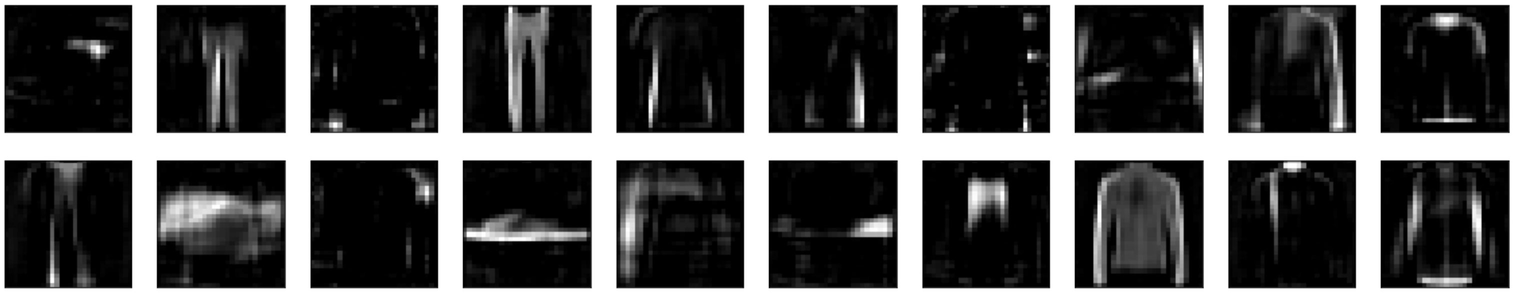
Figure 5: Histograms of the encoding of each of the 10 original images for the various versions of the shallow Auto-Encoder



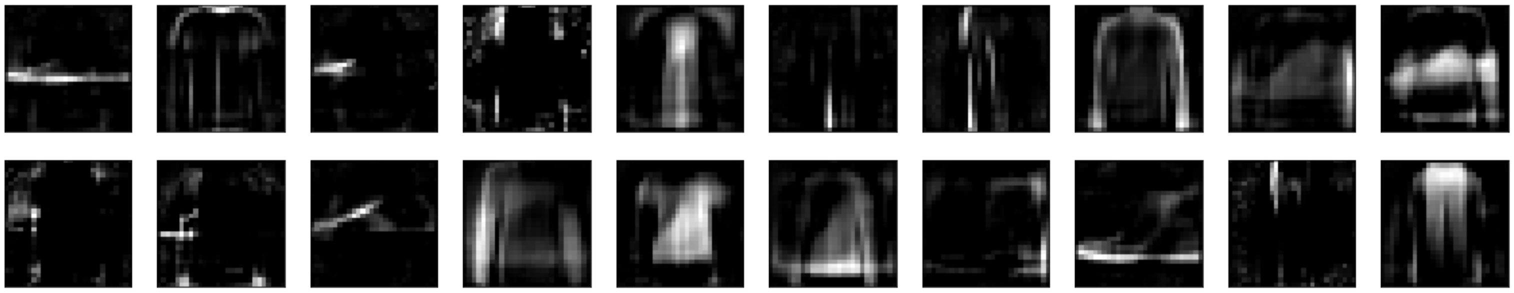
(a) Shallow Auto-encoder with no constraints



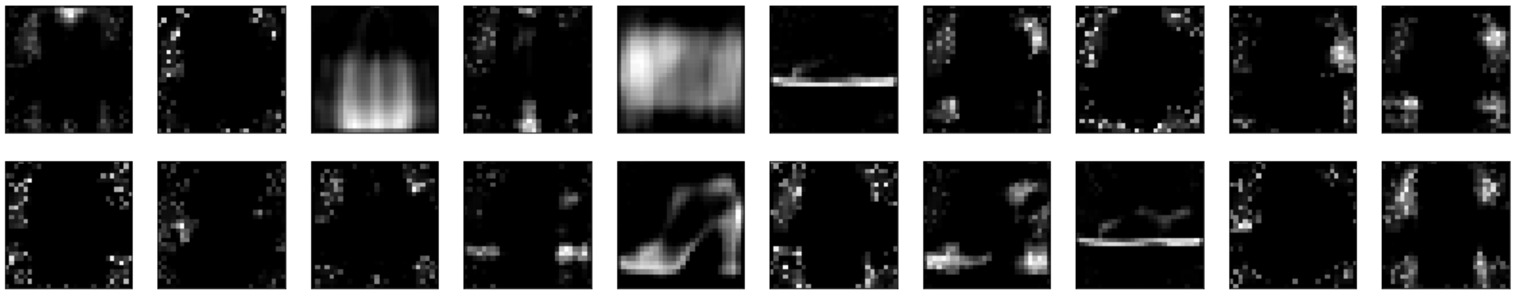
(b) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.2$, $\beta = 0.001$



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

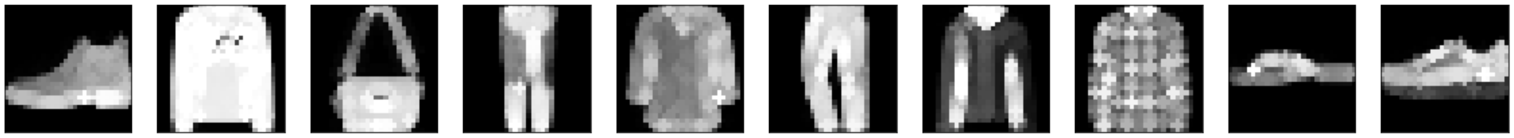


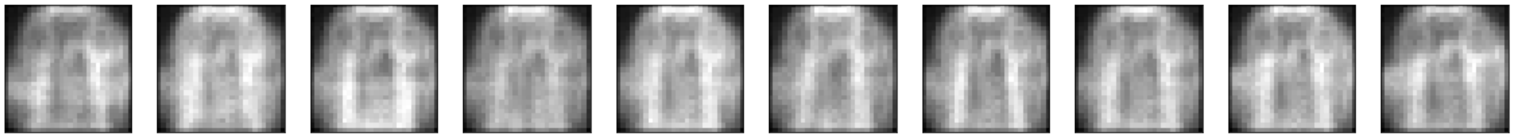
(d) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.05$, $\beta = 0.001$

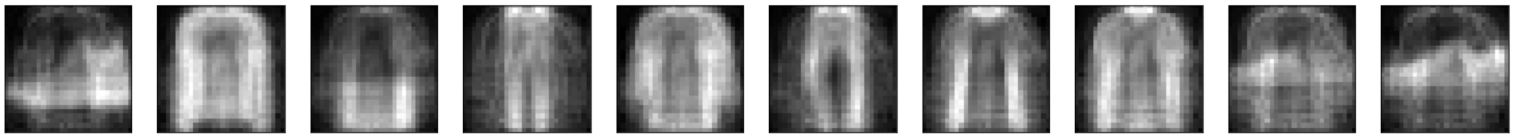


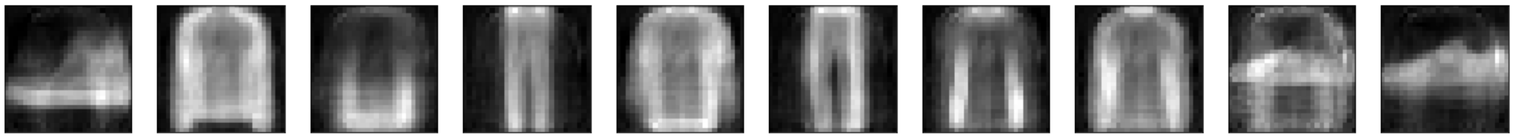
(e) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$

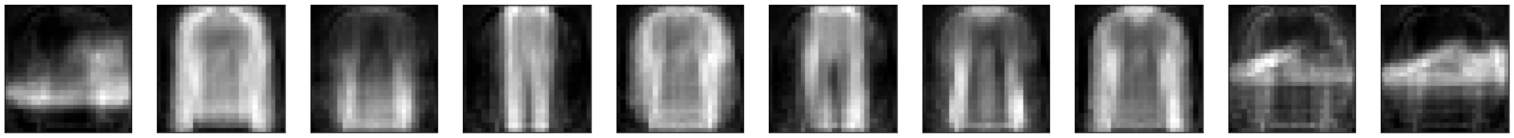
Figure 6: Some atoms (out of the 100 atoms) of the various versions of the shallow Auto-Encoder


(a) Dilatation 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*

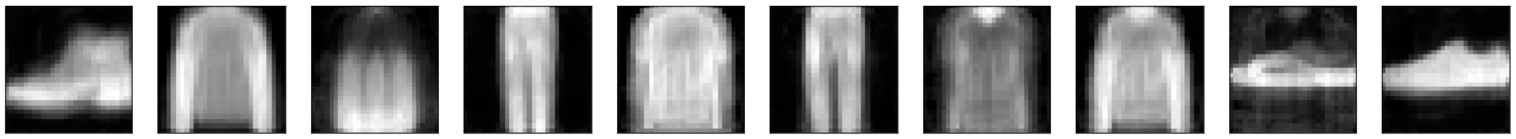
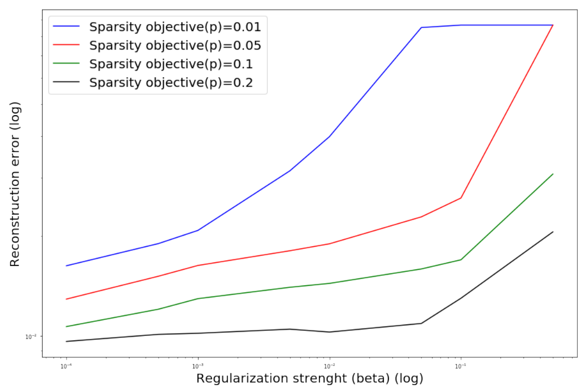
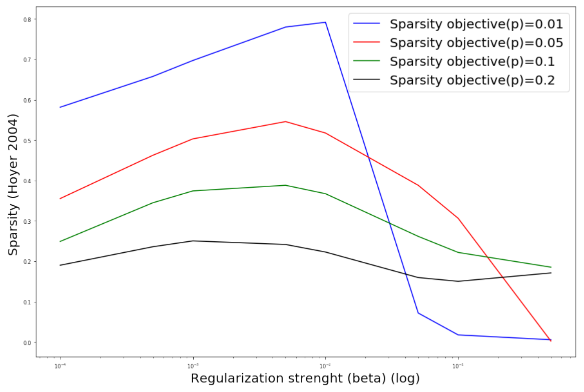

(f) Shallow Auto-encoder with Non-Negativity and Sparsity Constraints $p = 0.01$, $\beta = 0.005$ - *Max-Approximation error: 0.0339*

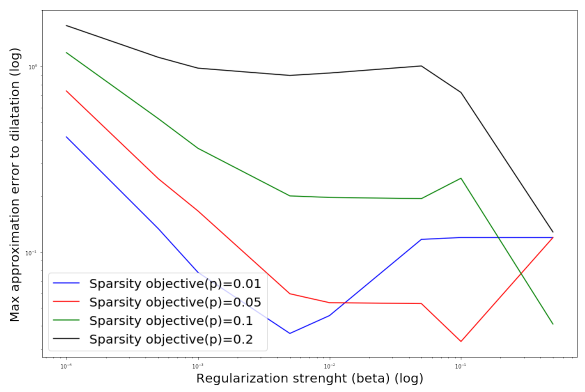
Figure 7: Max-approximation to the dilatation by a disk of radius 1



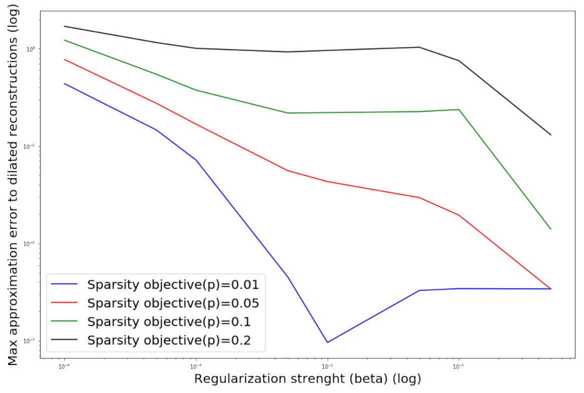
(a) Reconstruction error as a function of the parameter β (sparsity penalty strength)



(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 8: 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