Disentangled latent representations of images with atomic autoencoders

Alasdair Newson, Yann Traonmilin

Disentangled latent representations of images with atomic autoencoders

A. Newson[1] and Yann Traonmilin[2]

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris, France.
anewson@telecom-paris.fr
[2]CNRS,Univ. Bordeaux, Bordeaux INP, IMB, UMR 5251, F-33400 Talence, France.

Abstract. We present the atomic autoencoder architecture, which decomposes an image as the sum of elementary parts that are parametrized by simple separate blocks of latent codes. We show that this simple architecture is induced by the definition of a general low-dimensional model of the considered data. We also highlight the fact that the atomic autoencoder achieves disentangled low-dimensional representations under minimal hypotheses. Experiments show that their implementation with deep neural networks is successful at learning disentangled representations on two different examples: images constructed with simple parametric curves and images of filtered off-the-grid spikes.

Keywords: low dimensional models · autoencoders · disentanglement.

1 Introduction

The autoencoder is a well-known neural network architecture whose goal is to project data to and from a *latent space*, which is usually of much smaller dimensionality than the original data space, this feature being useful for a wide variety of tasks. Given a collection of samples $X = (x_1, \dots, x_N)$ in $\mathbb{R}^n$, autoencoders aim at providing a low dimensional representation of the x_i by using an encoder/decoder pair: the autoencoder f is the composition of an encoder f_E and a decoder f_D and is typically trained by minimizing the quadratic reconstruction loss $\mathcal{L}_X(f) := \sum_{i=1}^{N} \|f(x_i) - x_i\|_2^2$, i.e.

$$f^* = f_D^* \circ f_E^* \in \arg\min_{f \in \mathcal{F}} \mathcal{L}_X(f) \tag{1}$$

where $\mathcal{F}$ is the set of autoencoders having a given architecture. Typically, the encoder f_E projects samples to a low-dimensional "latent" space $\mathbb{R}^d$, and the decoder f_D performs the opposite operation, producing data in $\mathbb{R}^n$ from a latent code. In practice, estimating f^* as a deep neural network parametrized by its weights and biases has shown a huge number of applications from solving inverse problems [18] to photo realistic image rendering which employs this architecture in recent diffusion models.

The intermediate latent representation can be used to create a *generative* model of the considered data, i.e. data points can be generated as the image of

low dimensional points by the decoder in the latent space. However, we can not only generate but also edit or manipulate data by moving in the latent space. Towards this goal, several fundamental questions arise: how can one navigate the latent space with the guarantee that the chosen path does not leave the latent set? Is it possible to manipulate latent codes in a meaningful way? Such questions are linked to what is known as the "disentanglement" problem. Indeed, the ultimate goal of generative models is to establish a latent space where each coordinate corresponds to a given feature of the data, thus disentangling these features. For example, for facial images, this could be high-level attributes such as an expression. Thus, editing an image would correspond simply to moving along a coordinate in the latent space.

Moreover, we remark that autoencoders are heavily linked to the theory of low-dimensional representation of data that has seen much development in the past twenty years with the theory of sparse recovery and compressed sensing and its extensions (see [10] for an overview). Thus, it appears natural to consider the disentanglement problem from the viewpoint of this theory.

Contributions In this paper, we consider a widely-used class of low-dimensional models (in particular for signal and image processing) and then design an autoencoder architecture tailored for the learning of such models. The resulting design leads to disentangled interpretable low-dimensional representations that can be manipulated in a meaningful way. We consider signals that can be represented as the sum of a few atoms from a (possibly continuous) dictionary. Examples of such signals include sketch images and off-the-grid sums of spikes models (used in microscopy, astronomy, echo retrieval etc.).

We define in Section 2 the autoencoder architecture that we call the *atomic autoencoder*. We show how this architecture is induced by the definition of the general atomic model of the data and discuss its practical implementation with deep neural networks (DNN). We show two properties of such autoencoders: under a perfect learning hypothesis, they provide a naturally disentangled representation. Secondly, it should not be possible to decompose individual atoms of the dictionary as the sum of simpler atoms if one is to define meaningful atoms of the dictionary.

In Section 3, we apply the architecture to two examples: the decomposition of images composed of parametric curves and the estimation of off-the-grid spikes (filtered by a Gaussian kernel). In these different contexts, we access the disentangled representation and manipulate this representation in a meaningful way: we modify the local shape of the image in the first and access the position and amplitudes of the spikes in the second example. These properties confirm that we have access to a disentangled latent representation with *no more constraints* in the training than the specific shape of the atomic autoencoder.

Related work The goal of automatically discovering a hidden parametrisation of a class of objects is the objective of representation learning. Autoencoders are an ideal setting for this problem, and have existed for a long time, starting in the 1980's [1,4,9]. The work of Cheung et al. [8], Kumar et al. [15] and

Lezama [17] attempt to achieve disentanglement in autoencoders by incorporating a covariance loss function into the training process. Lample et al. [16] proposed Fader networks, which try to isolate a single image characteristic in a single latent component, with an innovative use of a discriminator network. This produces a network where the characteristic can be effectively controlled with a slider. Finally, the authors of β-VAE B [11], β-VAE H [6], FactorVAE [13] and β-TCVAE [7], propose frameworks or regularisation to disentangle variational autoencoders by weighting the Kullback-Leibler divergence term to encourage factorised representations in the latent space.

An area of learning that is closely related to our approach is dictionary learning (see e.g. [20] for an overview), where signals of interest are decomposed on a finite family of fixed vectors (atoms). Atomic autoencoders can be seen as a continuous version of dictionary learning. Extension to coninuous dictionary with DNN has been proposed using composition of autoencoders in [19]. However, this architecture does not permit to isolate simple features in a given block of latent code like atomic autoencoders.

Finally, recent work [21] indicates that trained DNNs can learn the simplest model, i.e. this model can be learned with minimal distortion, even if worst case bounds suggest that it is difficult to achieve in practice [3]. We also note that the general architecture proposed in this paper shares some similarities with "branching" architectures [12], however our architecture is geared towards disentangled latent representations, which is novel.

2 Atomic autoencoders: from sparse models to disentangled representations with deep neural networks

In this section, we introduce the *atomic autoencoder* architecture as a consequence of considering a class of generalized sparse models, we discuss some of its elementary properties and their implementation with DNNs. A now almost canonical sparsity model is the following:

$$\Sigma_{\mathcal{A},k} = \{x : x = \sum_{i=1}^{k} a_i, a_i \in \mathcal{A}\} \tag{2}$$

where $\mathcal{A}$ is a set of *atoms* (often called a dictionnary) and k is the number of atoms required to represent the data x. For example, in classical sparsity atoms are weighted vectors of sparsity 1. Such models are ubiquitous in signal and image processing and have been proven powerful for the resolution of ill-posed inverse problems.

In the following, we consider first an ideal data model and its induced *ideal* atomic autoencoder. We make the hypothesis that the x_i belong to a model defined by Equation (2). Of course, a more realistic modeling would consider x_i which approximately belong to $\Sigma_{\mathcal{A},k}$ (e.g. by considering that some distance $d(x_i, \Sigma)$ between the x_i and Σ is bounded), however the impact of this approximation goes beyond the scope of this paper and is left for future work.

Afterwards, we consider a *learned* neural network architecture which approximates this ideal model. Obviously, in a real dataset we have no guarantee that learned architecture perfectly recovers the ideal model. However, recent work indicates [21] that given a large enough dataset the ideal model can be learned up to a given distortion.

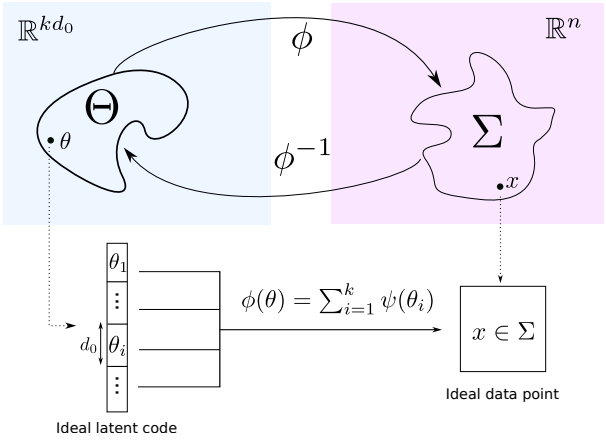

Fig. 1. Atomic data model. In this data model, we consider a ideal latent space Θ. Starting from an ideal latent code $\theta \in \Theta$, an ideal data point is generated by the sum of *the same* function ψ applied to k sub-blocks of θ: $\phi(\theta) = \sum_{i=1}^{k} \psi(\theta_i)$, thus defining an *atomic decoder* architecture.

2.1 Ideal atomic autoencoders

We consider data from the ideal model set $\Sigma_{\mathcal{A},k} \in \mathbb{R}^n$. We suppose that elements $x \in \Sigma_{\mathcal{A},k}$ are generated from a vector of parameters θ, via a function ϕ, i.e. $x = \phi(\theta)$. We refer to θ as the *ideal latent code*.

In the literature of sparse representations, it is often supposed that the set of atoms $\mathcal{A}$ has some additional structure. It can be parametrized using a function $\psi : \mathbb{R}^{d_0} \to \mathbb{R}^n$. Hence, each atom can be written $a_i = \psi(\theta_i)$ and for any $x \in \Sigma_{\mathcal{A},k}$,

$$x = \phi(\theta) = \sum_{i=1}^{k} \psi(\theta_i) \tag{3}$$

where $\theta_i \in \mathbb{R}^{d_0}$ represents a sub-block of size d_0 of the ideal latent code[1]. We refer to a coordinate of a sub-block θ_i as a latent *coordinate*, and note $\theta_{i,j}$ the j^{th} coordinate of the i^{th} block. We have $\theta = (\theta_1, \dots, \theta_k) \in \mathbb{R}^{kd_0}$, as in Equation (3). We refer to ϕ as the *ideal decoder*: each data point $x \in \Sigma_{\mathcal{A},k}$ can be coded with kd_0 parameters, using ϕ (and ψ). The ideal atomic data model is illustrated in Figure 1 and can be rewritten as:

[1] We also refer to a sub-block θ_i simply as a latent block

$$\Sigma_{\mathcal{A},k} = \Sigma_{\psi,k} := \{x : x = \sum_{i=1}^{k} \psi(\theta_i), \theta_i \in \Theta_0\}, \tag{4}$$

where $\Theta_0 \subset \mathbb{R}^{d_0}$ is the set where blocks of ideal latent codes live. We suppose that the ideal latent set is $\Theta = \phi^{-1}(\Sigma_{\psi,k}) = \Theta_0^k \subset \mathbb{R}^{kd_0}$ (we start out with the ideal data, and find the ideal latent set through ϕ^{-1}). Note that low-dimensional representations, i.e. $kd_0 << n$, are often sought after; for example, to serve as a prior model in inverse imaging problems.

To take a concrete example, for sums of off-the-grid spikes convolved with a Gaussian kernel, we can set $\psi(\theta) = \psi(a,t) = a_i G(t)$ where $\theta = (a,t)$, G is a Gaussian function centered at t, and a_i is the amplitude of the spike. Hence the function ψ parametrizes individual spikes with their amplitude and position. In another case, with images of non-overlapping disks, ψ is a function which produces an image of a disk from the position and size of the disk.

From the definition of $\Sigma_{\psi,k}$, we see that there exists $\zeta : \mathbb{R}^n \to \Theta$, such that $\phi \circ \zeta(x) = x$ for any $x \in \Sigma$: just define ζ that arbitrarily choses one of the representations of x in Θ (e.g. using lexicographical order). Indeed, due to the sum in Equation (3), the ordering of the latent blocks does not matter. We call ζ the *ideal atomic encoder*. We have just defined the *ideal atomic autoencoder* $\phi \circ \zeta$ induced by the model with the following constraints: the latent set has a block structure, ie $\theta = (\theta_1, \dots, \theta_k) \in \mathbb{R}^{kd_0}$, and the decoder is the sum of the same function ψ of different blocks of latent codes (a generating sum of atoms).

To provide useful disentangled representations, the main ingredient for this architecture is to perform encoding with the smallest dimension possible. We formalize this by supposing that the ideal latent set is in a (smooth) bijection with the cube $[0,1]^{kd_0}$ (see Section 2.3 for a brief discussion on why the latent set must be bounded). In other words, there is no space left around Θ for codes that do not produce an element of Σ. This relies on the fact that there is no smooth bijection from $[0,1]^{d'}$ to $[0,1]^d$ if $d' < d$. Given a sets U, V, let $\mathcal{C}_1(U,V)$ be the set of continuously differentiable functions from U to V.

Assumption 1 (Filling latent set). *We say an atomic autoencoder $\phi \circ \zeta$ of Σ yields a filling latent set $\Theta = \phi^{-1}(\Sigma)$ if there is a bijection $h \in \mathcal{C}^1([0,1]^{kd_0}, \Theta)$ with $\mathcal{C}^1$ inverse between $[0,1]^{kd_0}$ and Θ.*

Now that we have defined the atomic autoencoder, we come back to one of the initial questions posed in this paper: disentanglement. We argue that an ideal atomic autoencoder that verifies Assumption 1 does indeed achieve disentanglement in the following sense: as Θ_k is in a smooth bijection with $[0,1]^{kd_0}$, given a non extremal point θ in Θ there is an open set containing θ such that all points of this set parametrize an element of Σ: we can generate elements of Σ freely and navigate safely in all directions of the latent set, without "falling outside", which is a direct consequence of Assumption 1. Of course we would need to know h to do this fully, without this knowledge we can still do it locally. Moreover, thanks to the intrinsic atomic structure, we can modify

individually each code to change only one "simple" feature of x at a same time. If $\theta \in \Theta$ and we want to modify one latent block i by a (sufficiently) small change Δ_i, the previous property ensures that $\theta + (0, \dots, 0, \Delta_i, 0, \dots, 0) \in \Theta$.

We call this notion of disentanglement *atomic disentanglement*. Given a low-dimensional model and an autoencoder, it is verified if Assumptions 1 is verified.

Lemma 1. *Suppose $\Sigma, \Theta, \phi \circ \zeta, h$ verify Assumption 1 and $\mathrm{int}(h^{-1}(\Theta)) \neq \emptyset$ (where int denotes the interior). Let $\theta \in \Theta$ such that $h^{-1}(\theta) \in \mathrm{int}(h^{-1}(\Theta))$. Then there exists an open set O of $\mathbb{R}^{kd_0}$ such that $\theta + O \subset \Theta$.*

Proof. Let $u \in \mathrm{int}(h^{-1}(\Theta))$ such that $h(u) \in \Theta$. With Assumption 1, as $h \in C^1$ and is a bijection, by continuity, there is an open set $\tilde{O} \in \mathbb{R}^{kd_0}$ such that $h(u + \tilde{O}) \in \Theta$. The image of $u + \tilde{O}$ is an open set $Q \subset \Theta$, hence $O = Q - h(u)$ is an open set and $\theta + O = h(u) - h(u) + Q = Q \subset \Theta$.

We show in Section 2.2 that Assumption 1 is verified for perfectly trained atomic autoencoders.

Note that injectivity of ϕ (discussion on injectivity is out of the scope of this paper and left for future work) is not necessarily required to provide atomic disentanglement. Having equivalent representations does not affect the ability to navigate the latent space in our definition. An example of an ideal autoencoder that achieves atomic disentanglement is one which addresses the off-the-grid sparse spike estimation problem. Indeed, in this case, we can model the data as:

$$\Sigma = \left\{ aG(t_1) + bG(t_2), \quad a, b \in [a_{min}, a_{max}]; t_1, t_2 \in [0,1]^d, t_1 \neq t_2; \right\}, \tag{5}$$

Note that in many cases, if Σ contains an element composed of k atoms, then any "simpler" signal composed of less atoms is still in the model (as is usually done in classical sparsity model), i.e. for all $(\theta_i)_{i \in I}$, with $I \subset \{1, \dots, k\}$, we have that $\sum_{i \in I} \psi(\theta_i) \in \Sigma$. We observe in experiments that trained atomic DNN autoencoders seem to have this property without any explicit constraint for this in the training, rather it is simply induced by the training data.

2.2 Some elementary properties of learned atomic autoencoders

We consider f_E the trained encoder, f_D the trained decoder (we drop the star exponent to keep notations light), and call the latent code $z = f_E(x)$ for $x \in \Sigma$. Note that z is generally not the same as the ideal latent code θ. We place ourselves in the case where an ideal atomic autoencoder $\zeta \circ \phi$ that achieves atomic disentanglement is induced by $\Sigma_{\psi,k}$. We also suppose that we are able to train an autoencoder $f_D \circ f_E$ up to an arbitrary precision: in the case of a perfectly trained autoencoder, we have $f_E : \mathbb{R}^n \to \mathbb{R}^{kd_0}$ and for any $z \in \mathbb{R}^{kd_0}$, $f_D(z) = \sum_{j=1}^{k} g(z_j)$ for some $g : \mathbb{R}^{d_0} \to \mathbb{R}^n$ and for any $x \in \Sigma$, $f_D \circ f_E(x) = x$.

If ϕ, ζ, f_D, f_E are smooth, $f_D \circ f_E$ achieves atomic disentanglement.

Proposition 1. *Suppose $\phi, \zeta, f_D, f_E \in \mathcal{C}^1$ and $\phi \circ \zeta$ verifies Assumption 1 on Σ. Then $f_E \circ f_D$ verifies Assumption 1 on Σ.*

Proof. Define $T = f_D^{-1}(\Sigma)$. Also, since $\phi \circ \zeta$ verifies Assumption 1, we have that there exists a bijection $h \in \mathcal{C}^1$ between $[0,1]^{kd_0}$ and Θ.

Let $\tilde{h} = f_E \circ \phi \circ h$, the function $\tilde{h}$ is $\mathcal{C}^1$ by composition of $\mathcal{C}^1$ functions and any element of T is the image of an element of $[0,1]^{kd_0}$. Reciprocally, by defining $\tilde{h}^{-1} = h^{-1} \circ \zeta \circ f_D$ that is also $\mathcal{C}^1$, as $f_D(T) = \Sigma$, any element of $[0,1]^{kd_0}$ is the image of an element of T. This proves Assumption 1.

What this simple proposition tells us is that most of the desirable disentanglement properties are guaranteed by the structure of the autoencoder itself, and having a latent set that "fills" the latent space is a byproduct of smoothness of the autoencoder given the dimensions are well chosen (i.e. small enough).

We now show that $f_D \circ f_E$ can possibly mix coordinates of the ideal latent blocks if ψ can be broken into simpler functions. We prove this for the case of two blocks of size 2 and $\Theta = [0,1]^{2\times 2}$.

Proposition 2. *Let $\phi \circ \zeta$ be an atomic autoencoder such that there exists $\tilde{\psi}$ such that $\psi(\theta_i) = \tilde{\psi}(\theta_{i,1}) + \tilde{\psi}(\theta_{i,2})$ (with $\Theta = [0,1]^{2\times 2}$). Then there exists an atomic autoencoder $f_D \circ f_E$ such that $f_D(z) = g(z_1) + g(z_2)$ and for any $\theta = (\theta_{1,1}, \theta_{1,2}, \theta_{2,1}, \theta_{2,2}) \in \Theta$, $f_E(\phi(\theta)) = (\theta_{1,1}, \theta_{2,1}, \theta_{1,2}, \theta_{2,2})$, i.e. the encoder f_E mixes the ideal latent coordinates into two different blocks.*

Proof. For the decoder, just consider $f_D = \phi$ and $g = \psi$. Now define the permutation p of $\{1,2,3,4\}$, such that $p(1,1) = (1,1)$, $p(1,2) = (2,1)$, $p(2,1) = (1,2)$, $p(2,2) = (2,2)$) and the function $\rho : [0,1]^{2\times 2} \to [0,1]^{2\times 2}$ such that $[\rho(\theta)]_{p(i,j)} = \theta_{i,j}$. Now define $f_E = \rho \circ \zeta$. We have

$$\begin{aligned} f_D \circ f_E(x) &= \phi(\rho(\zeta(x))) = \phi(\rho(\theta)) = \phi(\theta_{11}, \theta_{2,1}, \theta_{1,2}, \theta_{2,2}) \\ &= \tilde{\psi}(\theta_{1,1}) + \tilde{\psi}(\theta_{2,1}) + \tilde{\psi}(\theta_{1,2}) + \tilde{\psi}(\theta_{2,2}) = \phi(\theta) = x. \end{aligned} \tag{6}$$

In other words, if the function defining the dictionary is the sum of two elementary functions then single atoms could be coded on several blocks. This is often not desirable because each block should modify an individual "simple" feature. In terms of design of the autoencoder, this tells us that if ψ can be broken down into simpler functions, then latent blocks should be chosen to be smaller. In our two practical examples, we observe that ψ is no such function.

2.3 A tailored neural architecture

We showed that a large class of datasets can be represented with atomic autoencoders. In the experimental part of this article, we will train autoencoders using (leaky) ReLu DNN architectures. We have seen that it is natural to train autoencoders $f_D \circ f_E$ with the structure $f_D(z) = \sum_{i=1}^{k} g(z_i)$, i.e. the decoder is made of k identical blocks which approximate $\phi(z) = \sum_{i=1,k} \psi(z_i)$ and the encoder must approximate ζ. In fact, this approach is referred to in the deep learning community as "branching" [12,14,5]. However, there are two major differences between standard branching and the proposed approach. We are applying

a form of branching to the different blocks of code of the latent space of an autoencoder. In other words, the branching takes place in the decoder only. In particular, this means that we want the blocks of code to be interpretable as underlying parameters of the data. Moreover the weights of our branched decoders are the same. In the jargon of deep learning, they are said to be *tied*.

We highlight the fact that the decoder block g is indeed repeated, so that during training, its parameters are the same when applied to all z_i's. Note that the symmetrical architecture proposed by early works on autoencoders does not have any really satisfying theoretical explanation. It is known from low-dimensional recovery that low-dimensional signals can be encoded almost universally by *linear* encoders while decoders are often non-convex functions that are generally NP-hard to calculate.

Since our goal is to learn atomic parametrized representations of networks, we note that training any atomic autoencoder cannot hope to recover a given ψ up to more than a bijection (restricted to the latent set) from $\mathbb{R}^{d_0}$ to $\mathbb{R}^{d_0}$, i.e if $h : \mathbb{R}^{d_0} \to \mathbb{R}^{d_0}$ and h^{-1} are rendered possible by the NN structure, $(f_D \circ h) \circ (h^{-1} \circ f_E)$ achieves the same loss L_X as $f_D \circ f_E$.

With a fixed autoencoder parametrized by a leaky ReLU DNN, complex data can only be represented on a bounded latent set in $\mathbb{R}^{kd_0}$. We see this by noticing that such an architecture can be seen as a piece-wise affine functions. In particular, the width of unbounded affine regions necessarily grows when the norm of the code increases to infinity. Determining the bounds of the latent set is an open question in itself. In our examples we suppose that they can be well estimated by looking at the bounds of the latent codes of the training data $f_E(x_i)$.

Of course, the question of determining the size of the latent code is important in itself. This question is outside the scope of this paper, but we can mention that ideas such as sparse regularization of the atomic latent code may be useful in estimating these parameters. Also, Proposition 2 shows that d_0 must be chosen small enough so that atoms cannot be "split" into smaller atoms.

Implementation details The DNN architecture is designed to be as simple as possible. The encoder performs iterated 3×3 convolutions, with a stride of 2×2 (subsampling by 2 at each layer), followed by a leaky ReLU non-linearity ($\alpha = 0.2$). There are four such layers.The number of convolutional channels is divided by 2 from 32 to 8, and is kept at 8 until the final convolution. Finally, two fully connected layers, with leaky ReLU ($\alpha = 0.2$) are used to project the tensors to the latent space $\mathbb{R}^{kd_0}$, with k, d_0 chosen according to application.

The decoder block g takes a latent block (in $\mathbb{R}^{d_0}$) and outputs an image. The first layers of g are two fully connected layers with leaky ReLUs to reach a size of $2 \times 2 \times 8$. This is followed by a convolutional section, with 3×3 convolutions. The convolutional section is not symmetric with respect to the encoder, since we go from a small $2 \times 2 \times 8$ tensor to a full image. Each convolutional layer contains a 3×3 convolution, followed by an upsampling by 2×2 and finally a leaky ReLU. The number of channels is chosen in a symmetric manner to the

Fig. 2. Autoencoding 128×128 mnist images on $\mathbb{R}^{30}$.

encoder: we keep 8 channels until we can increase them by multiplying by 2 until the final layer, which is 32.

Again, we draw attention to the fact that the architecture is simple, with around 300,000 parameters (depending on the exact application), which is an extremely lightweight network. Indeed, one of the core ideas is to achieve a latent space which is as small as possible, to try and fulfill Assumption 1.

3 Applications

Application to images consisting of parametric curves We first consider the autoencoding process applied to the model of images x consisting of a set of parametric curves: we suppose $x = \sum_{i=1}^{K} S(\theta_i$. $S(\theta_i)$ is a parametric curve, with parameters θ_i (e.g. a stroke parametrized by Bezier curves). We apply this modelling to the MNIST database, a good fit as each number in mnist is an amalgamation of small penstrokes, which are approximately simple splines convolved with the shape of the tip of the pen. In order to have a resolution where strokes will be meaningful, we use a high-resolution version of MNIST [2] (500×500), which we downscale to 128×128. The final latent space size is 30 (block size 3).

We verify that the autoencoder has indeed worked by showing the input and output in Figure 2 (with some distortion typical of raw autoencoder architectures). We show that the decomposition given by the model is meaningful in Figure 3. In this Figure, we show both the input and output of the network, as well as the individual decomposition images $f_{D_0}(z_i)$'s. We observe that the network successfully separates the different strokes which compose mnist. We also remark that the network does not mix up the spatial locations of the strokes: each one is connected and continuous. This is very satisfying behaviour, since at no point have we shown any such examples to the autoencoder: it learns these simple atoms on its own. In Figure 4, we perform a linear interpolation in the latent space between two numbers. We verify that each point of the interpolation does not leave the space of images that are sum of parametric curves.

Application to off-the-grid sparse spike modelling An important inverse problem with applications in microscopy, astronomy or acoustic signal processing is spike detection in images. This corresponds to the task of estimating positions and

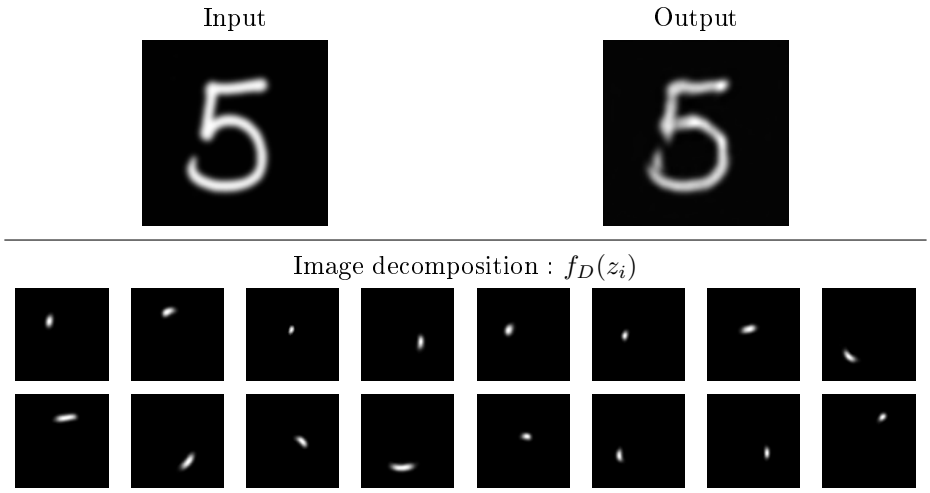

Fig. 3. Decomposition of an mnist image. We can see that the image of the number 5 has been broken down into several simple strokes, which are spatially localized.

Fig. 4. Interpolation with mnist images. We have interpolated linearly between two latent codes of images from the mnist dataset. We observe that the strokes are shortened or lengthened to go from a 6 to a 2.

amplitudes of a series of *spikes* in an image convolved with a filter. We create a synthetic database of such images. We allow for a maximum of 10 spikes, which are convolved with a Gaussian filter. The spikes are have minimum separation larger than the standard deviation of the Gaussian filter. In this situation, we know the size of latent blocks is $d_0 = 3$ (position and amplitude). We show in Figure 5 the application of the trained atomic autoencoder. Again, the network has learned, with no supervision, to separate the spikes in each image $g(z_i)$. Futhermore, in Figure 6, we see an example of navigation in the latent space. We have chosen a certain block i and then modified each latent coordinate of z_i. This modifies the behaviour of the third spike from the top of the image. The network has put the amplitude of the spike in the first coordinate, then two perpendicular motions in the next two. We note that there is no specific reason for the network to do this: it could have mixed amplitude and positions in one block. Finally, we observe that the motion is faster in one direction than the other; indeed there is nothing which imposes them to be the same.

4 Conclusion

We have introduced a simple architecture for the design of autoencoders induced by the theory of low-dimensional models. We have shown some elementary properties giving some qualitative facts about the behaviour of this architecture. We

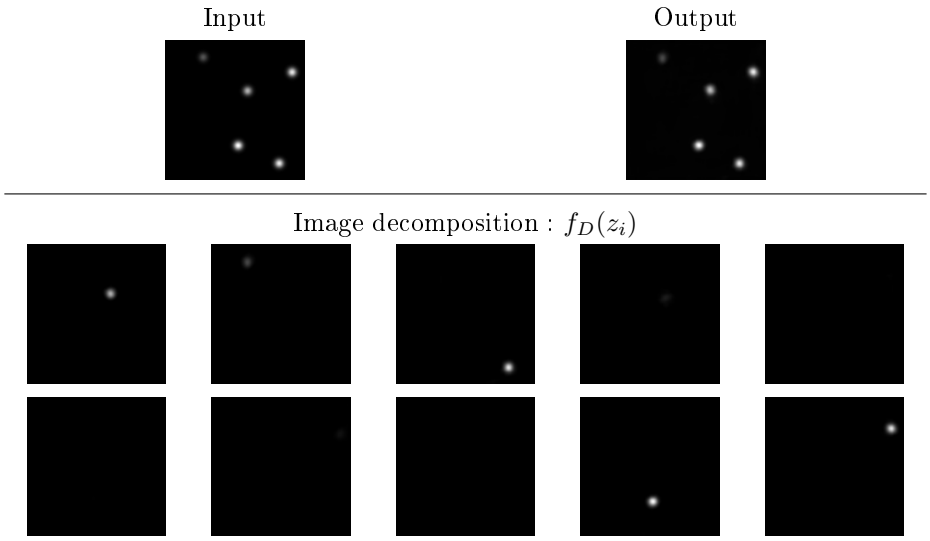

Fig. 5. Decomposition of an image of off-the-grid spikes convolved with a Gaussian using 10 **blocks of latent codes of dimension** $d_0 = 3$**.** We see that our atomic autoencoder learns to separate each spike seperately in each latent code block, even though our method is wholly unsupervised. This is due to its atomic structure.

show in two different applications how atomic autoencoders yields disentangled low-level interpretable parametetric representations.

Many questions arise from these works. From a practical perspective, does this architecture achieve high level disentangled semantic representations in images? Could we further normalize latent blocks to ease navigation in the latent space? Of course many applications could benefit from this architecture and adapting it to their specificities is an open line of work. From a theoretical point of view, what exact behaviour can we expect from atomic autoencoders when they learn the data model up to a given distortion? In order to better understand the disentanglement properties of atomic autoencoders, can we further expand the notion of functions ψ that cannot be broken down into simpler functions?

Acknowledgments

A. Newson acknowledges the support of the French Agence Nationale de la Recherche (ANR) under reference ANR-21-CE23-0024

Y. Traonmilin acknowledges the support of the French Agence Nationale de la Recherche (ANR) under reference ANR-20-CE40-0001 EFFIREG.

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for boltzmann machines. Cognitive Science **9**(1), 147–169 (1985)

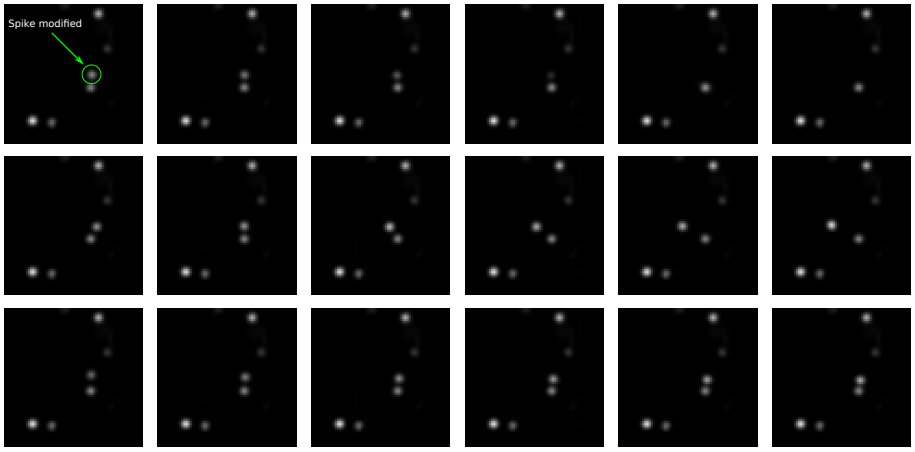

Fig. 6. Navigation in the latent space for images of spikes. Left to right: linearly modification of a latent coordinate in a block, keeping the others constant. Top to bottom: modifictation of the three different coordinates of the latent block. We observe that the first coordinate changes the amplitude, while the second and third modify the position. No supervision in the training was involved here.

2. Beaulac, C., Rosenthal, J.S.: Introducing a new high-resolution handwritten digits data set with writer characteristics. SN Computer Science **4**(1), 1–12 (2023)
3. Berner, J., Grohs, P., Voigtlaender, F.: Training relu networks to high uniform accuracy is intractable. arXiv preprint (2022)
4. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biological Cybernetics **59**(4), 291–294 (1988)
5. Brokman, J., Gilboa, G.: Analysis of branch specialization and its application in image decomposition. arXiv preprint arXiv:2206.05810 (2022)
6. Burgess, C.P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., Lerchner, A.: Understanding disentangling in β-VAE. arXiv:1804.03599 (2018)
7. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: NIPS. pp. 2180–2188. NIPS'16 (2016)
8. Cheung, B., Livezey, J.A., Bansal, A.K., Olshausen, B.A.: Discovering Hidden Factors of Variation in Deep Networks. In: 3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings (2015)
9. Elman, J.L., Zipser, D.: Learning the hidden structure of speech. The Journal of the Acoustical Society of America **83**(4), 1615–1626 (1988)
10. Foucart, S., Rauhut, H.: A mathematical introduction to compressive sensing. Springer (2013)
11. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework (2016)
12. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural computation **3**(1), 79–87 (1991)
13. Kim, H., Mnih, A.: Disentangling by Factorising. In: Proceedings of the 35th International Conference on Machine Learning. pp. 2649–2658. PMLR (2018)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NIPS **25** (2012)

15. Kumar, A., Sattigeri, P., Balakrishnan, A.: Variational Inference of Disentangled Latent Concepts from Unlabeled Observations (2018)
16. Lample, G., Zeghidour, N., Usunier, N., Bordes, A., DENOYER, L., Ranzato, M.A.: Fader Networks:Manipulating Images by Sliding Attributes. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
17. Lezama, J.: Overcoming the Disentanglement vs Reconstruction Trade-off via Jacobian Supervision. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019)
18. Peng, P., Jalali, S., Yuan, X.: Solving inverse problems via auto-encoders. IEEE Journal on Selected Areas in Information Theory **1**(1), 312–323 (2020)
19. Tariyal, S., Majumdar, A., Singh, R., Vatsa, M.: Deep dictionary learning. IEEE Access **4**, 10096–10109 (2016)
20. Tošić, I., Frossard, P.: Dictionary learning. IEEE Signal Processing Magazine **28**(2), 27–38 (2011)
21. Wang, Y., Hua, Y., Candés, E., Pilanci, M.: Overparameterized relu neural networks learn the simplest models: Neural isometry and exact recovery (2022)