# Sparsity in Variational Autoencoders

Andrea Asperti

***Abstract*— Working in high-dimensional latent spaces, the internal encoding of data in Variational Autoencoders becomes - unexpectedly - *sparse*. We highlight and investigate this phenomenon, that seems to suggest that, at least for a given architecture, there exists an *intrinsic* internal dimension of data. This can be used both to understand if the network has sufficient internal capacity, augmenting it to attain sparsity, or conversely to reduce the dimension of the network removing links to zeroed out neurons. Sparsity does also explain the reduced variability in random generative sampling from the latent space one may sometimes observe with variational autoencoders.**

## I. Introduction

Variational Autoencoders (VAE) ([6], [8]) are a fascinating facet of autoencoders, supporting, among other things, random generation of new data samples. Many interesting researches have been recently devoted to this subject, aiming either to extend the paradigm, such as conditional VAE ([9], [10]), or to improve some aspects it, as in the case of importance weighted autoencoders (IWAE) and their variants ([1], [7]). From the point of view of applications, variational autoencoders proved to be successful for generating many kinds of complex data such as natural images [5] or facial expressions [12], or also, more interestingly, for making probabilistic predictions about the future from static images [11]. In particular, variational autoencoders are a key component of the impressive work by DeepMind on representation and rendering of three-dimensional scenes, recently published on Science [3].

Variational Autoencoders have a very nice mathematical theory, that we shall briefly survey in Section II. One important component of the objective function neatly resulting from this theory is the Kullback-Leibler divergence $KL(Q(z|X)||P(z))$, where $Q(z|X)$ is the distribution of latent variables $z$ given the data $X$ guessed by the network, and $P(z)$ is a prior distribution of latent variables (typically, a Normal distribution). This component is acting as a regularizer, inducing a better distribution of latent variables, essential for generative sampling.

The contribution of our work is to highlight an additional benefit of the Kullback-Leibler component: working in latent spaces of sufficiently high-dimension, the latent representation becomes *sparse*. Many latent variables are zeroed-out (independently from the input), the associated variance computed by the network is around one (while the *real* variance is close to 0), and in any case those variables are *neglected* by the decoder.

As far as we have been able to check, this phenomenon has been never previously documented in the literature, in spite of its evident relevance. It was conjecture before that the Kullback-Leibler divergence was acting as a regularizer, but the regularization effect was never associated with sparsity (that is typically achieved in neural networks by means of weight-decay L1 regularizers, see e.g. [4]). The relation between Kullback-Leibler divergence and sparsity will be informally discussed in Section VI.

The most interesting consequence is that, at least for a given architecture, there seems to exist an *intrinsic* internal dimension of data. This property cab be exploited both to understand if the network has sufficient internal capacity, augmenting it to attain sparsity, or conversely to reduce the dimension of the network removing links to unused neurons. Sparsity also help to explain a loss of variability in random generative sampling from the latent space one may sometimes observe with variational autoencoders.

## II. Variational Autoencoders

In this section we briefly recall the theory behind variational autoencoders; see e.g. [2] for a more thoroughly introduction.

In latent variable models we express the probability of a data point $X$ through marginalization over a vector of latent variables:

$$P(X) = \int P(X|z,\theta)P(z)dz \approx \mathbb{E}_{z\sim P(z)}\, P(X|z,\theta) \quad (1)$$

where $\theta$ are parameters of the model (we shall omit them in the sequel).

Sampling in the latent space may be problematic for several reasons. The variational approach exploits sampling from an auxiliary distribution $Q(z)$. In order to understand the relation between $P(X)$ and $\mathbb{E}_{z\sim Q(z)}\, P(X|z,\theta)$ it is convenient to start from the Kullback-Leibler divergence of $Q(z)$ from $P(z|X)$:

$$KL(Q(z)||P(z|X)) = \mathbb{E}_{z\sim Q(z)}\, log\frac{Q(z)}{P(z|X)} \quad (2)$$

or also, exploiting Bayes rule,

$$KL(Q(z)||P(z|X)) = \mathbb{E}_{z\sim Q(z)}\, log\frac{Q(z)P(X)}{P(X|z)P(z)} \quad (3)$$

$P(X)$ does not depend on $z$ and may come out of the expectation; rephrasing part of the right hand side in terms of

A. Asperti is Professor at the University of Bologna, Department of Informatics: Science and Engineering (DISI).

An Extended Abstract of this article has been accepted for presentation at the 1st International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2019), 20-22 March 2019, Barcelona, Spain.

the KL divergence of $Q(z)$ from $P(z)$ we obtain, by simple manipulations:

$$\begin{array}{l} log(P(X)) - KL(Q(z)||P(z|X)) = \\ \qquad \mathbb{E}_{z\sim Q(z)}\, log(P(X|z)) - KL(Q(z)||P(z)) \end{array} \tag{4}$$

Since the Kullback-Leibler divergence is always positive, the term on the right is a lower bound to the loglikelihood $P(X)$, known as Evidence Lower Bound (ELBO).

In Equation 4, $Q(z)$ can be any distribution; in particular, we may take one depending on $X$, hopefully resembling $P(z|X)$ so that the quantity $KL(Q(z)||P(z|X))$ is small; in this case the loglikelihood $P(X)$ is close to the Evidence Lower Bound; our learning objective is its maximization:

$$\begin{array}{l} log(P(X)) \approx \\ \qquad \mathbb{E}_{z\sim Q(z|X)}\, log(P(X|z) - KL(Q(z|X)||P(z)) \end{array} \tag{5}$$

The term on the right has a form resembling an autoencoder, where the term $Q(z|X)$ maps the input $X$ to the latent representation $z$, and $P(X|z)$ decodes $z$ back to $X$.

The common assumption in variational autoencoders is that $Q(z|X)$ is normally distributed around an encoding function $\mu_\theta(X)$, with variance $\sigma_\theta(X)$; similarly $P(X|z)$ is normally distributed around a decoder function $d_\theta(z)$. All functions $\mu_\theta$, $\sigma_\theta$ and $d_\theta$ are computed by neural networks.

Provided the decoder function $d_\theta(z)$ has enough power, the shape of the prior distribution $P(z)$ for latent variables can be arbitrary, and for simplicity we may assume it is a normal distribution

$$P(z) = G(0,1)$$

The term $KL(Q(z|X)||P(z)$ is hence the KL-divergence between two Gaussian distributions $G(\mu_\theta(X), \sigma^2_\theta(X))$ and $G(1,0)$ which can be computed in closed form:

$$\begin{array}{l} KL(G(\mu_\theta(X),\sigma_\theta(X)), G(0,1)) = \\ \qquad \frac{1}{2}(\mu_\theta(X)^2 + \sigma^2_\theta(X) - log(\sigma^2_\theta(X)) - 1) \end{array} \tag{6}$$

As for the term $\mathbb{E}_{z\sim Q(z|X)}\, log(P(X|z)$, under the Gaussian assumption the logarithm of $P(X|z)$ is just the quadratic distance between $X$ and its reconstruction $d_\theta(z)$. Sampling, according to $Q(z|X))$ is easy, since we know the moments $\mu_\theta(X)$ and $\sigma^2_\theta(X)$ of this Gaussian distribution. The only remaining problem is to integrate sampling with backpropagation, that is solved by the well known reparametrization trick ([6], [8]).

Let us finally observe that, for generating new samples, the mean and variance of latent variables *is not used*: we simply sample a vector of latent variables from the normal distribution $G(0,1)$ and pass it as input to the decoder.

## III. Discussion

The mathematical theory behind Variational Autoencoders is very neat; nevertheless, there are several aspects whose practical relevance is difficult to grasp and look almost counter-intuitive. For instance, in his Keras blog on Variational Autoencoders[1], F.Chollet writes - talking about the Kullback-Leibler component in the objective function - that "you can actually get rid of this latter term entirely". In fact, getting rid of it, the Gaussian distribution of latent variables would tend to collapse to a Dirac distribution around its mean value, making sampling pointless: the variational autoencoder would resemble a traditional autoencoder, preventing any sensible generation of new samples from the latent space.

Still, the relevance of sampling during the training phase, apart from reducing overfitting and improving the robustness of the autoencoder, is not so evident. The variance $\sigma^2_\theta(X)$ around the encoding $\mu_\theta(X)$ is typically very small, reflecting the fact that only a small region of the latent space will be able to produce a reconstruction close to $X$. Experimentally, we see that the variance decreases very quickly during the first stages of training; since we rapidly reduce to sample in a small area around $\mu_\theta(X)$, it is natural to wonder about the actual purpose of this operation.

Moreover, the quadratic penalty on $\mu_\theta(X)$ in Equation 6 is already sufficient to induce a Gaussian-like distribution of latent variables: so why we try to keep variance close to 1 if we reasonably expect it to be much smaller?

In the following sections we shall try to give some empirical answers to these questions, investigating encodings for different datasets, using different neural architectures, with latent spaces of growing dimension. This will allow us to uncover the interesting sparsity phenomenon, that we shall discuss in Section VI.

## IV. MNIST

In a video available on line[2] we describe the trajectories in a binary latent space followed by ten random digits of the MNIST dataset (one for each class) during the first epoch of training. The animation is summarized in Figure1, where we use fading to describe the evolution in time.

Each digit is depicted by a circle with an area proportional to its variance. Intuitively, you can think of this area as the portion of the latent space producing a reconstruction similar to the original. At start time, the variance is close to 1, but it rapidly gets much smaller. This is not surprising, since we need to find a place for 60000 *different* digits. Note also that the ten digits are initially very sparse, but progressively dispose around the origin in a Gaussian-like shape.

The real purpose of sampling during training is to induce the generator to exploit *as much as possible* the latent space surrounding each encoding of data points. How much space can we occupy? In principle, *all the available space*, that is why we try to keep the distribution $P(z|y)$ close to a normal distribution (the entire latent space). The hope is that this should induce a better coverage of the latent space, resulting in a better generation of new samples. In the case of MNIST, we start getting some significant empirical evidence of the previous fact when considering a sufficiently deep architecture in a latent space of dimension 3 (with 2 dimensions it is difficult to appreciate the difference).

[1] https://blog.keras.io/building-autoencoders-in-keras.html

[2] http://www.cs.unibo.it/~asperti/variational.html

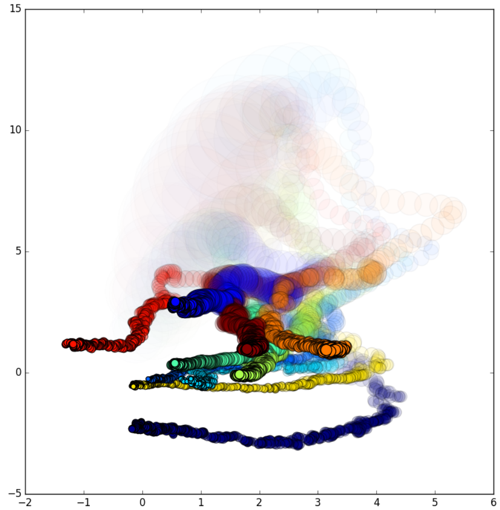



Fig. 1. Trajectories of ten MNIST digist in a binary latent space during the first epoch of training; pictures fade away with time. The area of the circle is proportional to the variance: it is initally close to 1, but it rapidly decrease to very small values.

| no. | variance | mean($\sigma_\theta^2(X)$) |
|---|---|---|
| 0 | 8.847272e-05 | 0.9802212 |
| 1 | 0.00011756 | 0.99551463 |
| 2 | 6.665453e-05 | 0.98517334 |
| 3 | 0.97417927 | 0.008741336 |
| 4 | 0.99131817 | 0.006186147 |
| 5 | 1.0012343 | 0.010142518 |
| 6 | 0.94563377 | 0.057169348 |
| 7 | 0.00015841 | 0.98205334 |
| 8 | 0.94694275 | 0.033207607 |
| 9 | 0.00014789 | 0.98505586 |
| 10 | 1.0040375 | 0.018151345 |
| 11 | 0.98543876 | 0.023995731 |
| 12 | 0.000107441 | 0.9829797 |
| 13 | 4.5068125e-05 | 0.998983 |
| 14 | 0.00010853 | 0.9604088 |
| 15 | 0.9886378 | 0.044405878 |

TABLE I
INACTIVE VARIABLES IN THE VAE FOR GENERATING MNIST DIGITS (DENSE CASE)

In Figure 2, we show the final distribution of 5000 MNIST digits in the 3-dimensional latent space with and without sampling during training (in the case without sampling we keep the quadratic penalty on $\mu_\theta(X)$). We also show the result of generative sampling from the latent space, organized in five horizontal slices of 25 points each.

We may observe that sampling during training induces a much more regular disposition of points in the latent space. In turn, this results in a drastic improvement in the quality of randomly generated images.

### A. *Towards higher dimensions*

One could easily wonder how the variational approach scales to higher dimensions of the latent space. The fear is as usual related to the curse of dimensionality: in a high dimensional space encoding of data in the latent space will eventually be scattered away, and it is not evident that sampling during training will be enough to guarantee a good coverage in view of generation.

In Figures 3 we show the result of generative sampling from a latent space of dimension 16, using a 784-256-32-24-16 dense VAE architecture. The generated digits are not bad, confirming a substantial stability of Variational Autoencoders to a growing dimension of the latent space. However, we observe a different and much more intriguing phenomenon. In Table I we show, for each latent variable, the mean of its variance (that we expect to be around 1, being normally distributed), and the mean of the computed variance $\sigma_\theta^2(X)$ (that we expect to be a small value, close to 0). The mean value is around 0 as expected, and we do not report it.

All variables highlighted in red have an anomalous behavior: their variance is very low (in practice, they always have value 0), while the variance $\sigma_\theta^2(X)$ computed by the network is around 1 for each $X$. In other words, the representation is getting *sparse*! Only 8 latent variables out of 16 are in use: the other ones (we call them *inactive*) are completely ignored by the generator. For instance, in Figure 4 we show a few digits randomly generated from Gaussian sampling in the latent space (upper line) and the result of generation when inactive latent variables have been zeroed-out (lower line): they are indistinguishable.

### B. *Convolutional Case*

With convolutional networks, sparsity is less evident. We tested a relatively sophisticated network, whose structure is summarized in Figure 5.

The previous network is able to produce excellent generative results (see Figure 6).

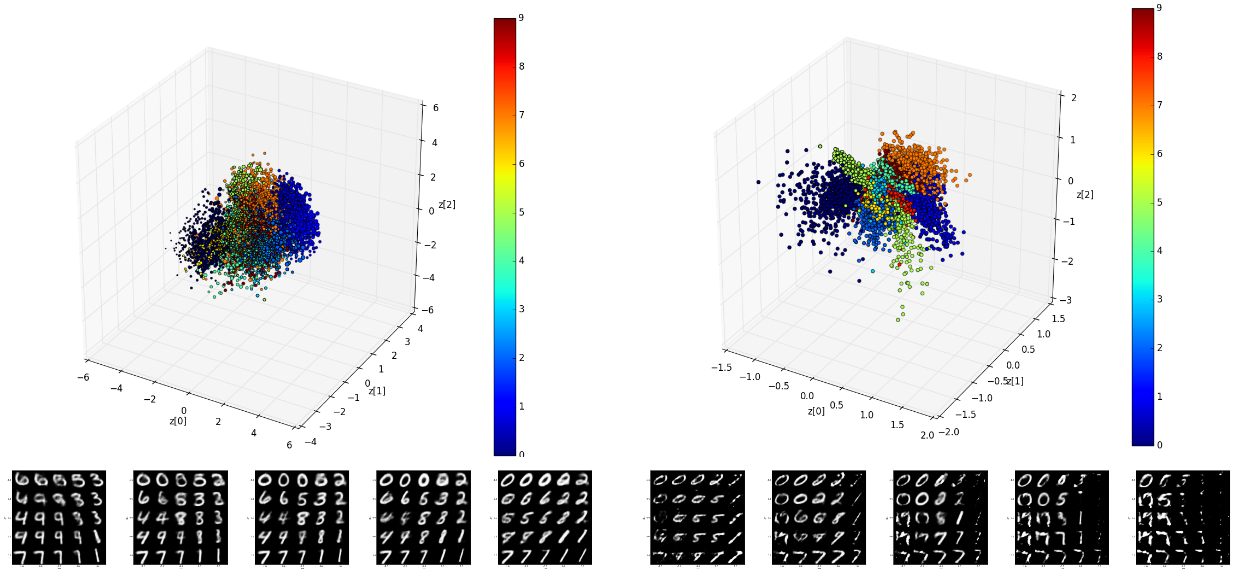
As for sparsity, 3 of the 16 latent variables are zeroed out. Having less sparsity seems to suggest that convolutional networks make a better exploitation of latent variables, typically resulting in a more precise reconstruction and improved generative sampling. This is likely due to the fact that latent variables encode information corresponding to different portions of the input space, and are less likely to become useless for the generator.

## V. GENERATING TILES

We repeated the experiment on a different generative problem, consisting in generating images containing a given number of small white tiles on a black background. The number of expected tiles should depend on the kind of training images provided to the autoencoder. In this case, we fed as training data images containing a number of tiles ranging between 1 and 3. All tiles have dimension $4 \times 4$ and the whole image is $28 \times 28$, similarly to MINST.

We start addressing the problem with the a dense network with dimensions 784-512-256-64-32-16.

In Figure 7 we give some example of input images (upper row) and their corresponding reconstructions (bottom row). The mediocre reconstruction quality testifies that this is a




(A) VAE

(B) without sampling at training time

Fig. 2. Representation of 5000 MNIST digits in the latent space, and examples of generative sampling. Figure (A) is relative to a Variational Autoencoder with a dense 784-256-64-16-3 architecture, Figure (B) is the same, but without sampling at training time (we keep the quadratic penalty on latent variables). The disposition of encodings in (A) has much more spherical shape, resulting in a drastic improvement in the quality of generated samples.

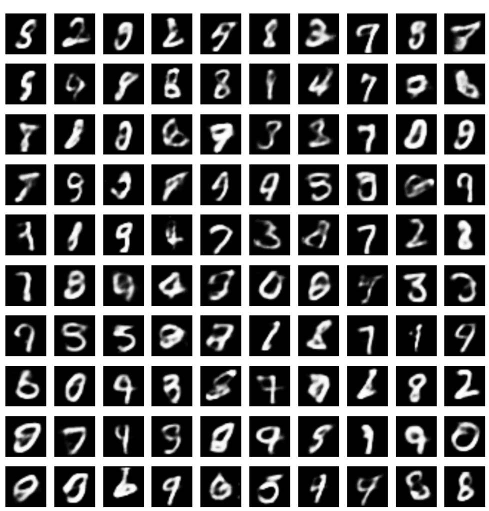


Fig. 3. Sampling from a latent spare of dimension 16 using a dense 784-256-32-24-16 VAE.

much more complex problem than MNIST. In particular, generative sampling is quite problematic (see Figure 8)

Again, we have the same sparsity phenomenon already observed in the MNIST case: a large number of latent variables is inactive (see Table II):

### A. *Convolutional Case*

We tested the same architecture of Section . In this case, reconstruction is excellent (Figure 9).

Fig. 4. Upper line: digits generated from a vector of 16 normally sampled latent variables. Lower line: digits generated after "red" variables have been zeroed-out: these latent variables are completely neglected by the generator.

Generative sampling has improved too (see Figure 10), but for an annoying counting problem: we expected to generate images with *at most three tiles*, while they frequently contain a much larger number of them[3].

From the point of view of sparsity, 4 latent variables out of 16 result to be inactive.

## VI. KULLBACK-LEIBLER DIVERGENCE AND SPARSITY

Let us consider again the loglikelihood for data X.

$$log(P(X)) \approx \mathbb{E}_{z\sim Q(z|X)}\, log(P(X|z) - KL(Q(z|X)||P(z))$$

If we remove the Kullback-Leibler component from previous objective function, or just keep the quadratic penalty on latent variables, the sparsity phenomenon disappears. So, sparsity must be related to that component, and in particular to the part of the term trying to keep the variance close to 1, that is

$$-\sigma^2_\theta(X) + log(\sigma^2_\theta(X)) + 1 \qquad (7)$$

[3]This counting issue was the original motivation for our interest in this generative problem.

| Layer (type) | Output Shape | Params |
|---|---|---|
| InputLayer | (None, 28, 28, 1) | 0 |
| conv2d 3x3 | (None, 14, 14, 16) | 160 |
| BatchNormalization | (None, 14, 14, 16) | 64 |
| RELU | (None, 14, 14, 16) | 0 |
| conv2d 3x3 | (None, 14, 14, 32) | 4640 |
| conv2d 3x3 | (None, 7, 7, 32) | 9248 |
| BatchNormalization | (None, 7, 7, 32) | 128 |
| RELU | (None, 7, 7, 32) | 0 |
| conv2d 3x3 | (None, 4, 4, 32) | 9248 |
| conv2d_3x3 | (None, 4, 4, 32) | 9248 |
| BatchNormalization | (None, 4, 4, 32) | 128 |
| RELU | (None, 4, 4, 32) | 0 |
| conv2d 3x3 | (None, 2, 2, 32) | 4128 |
| conv2d 3x3 | (None, 2, 2, 32) | 4128 |
| BatchNormalization | (None, 2, 2, 32) | 128 |
| RELU | (None, 2, 2, 32) | 0 |
| conv2d 3x3 | (None, 1, 1, 32) | 4128 |
| conv2d 1x1 | (None, 1, 1, 16) | 528 |
| conv2d_1x1 | (None, 1, 1, 16) | 528 |

Fig. 5. Architecture of the convolutional encoder. The two final layers compute mean and variance for 16 latent variables. The decoder is symmetric, using transposed convolutions.

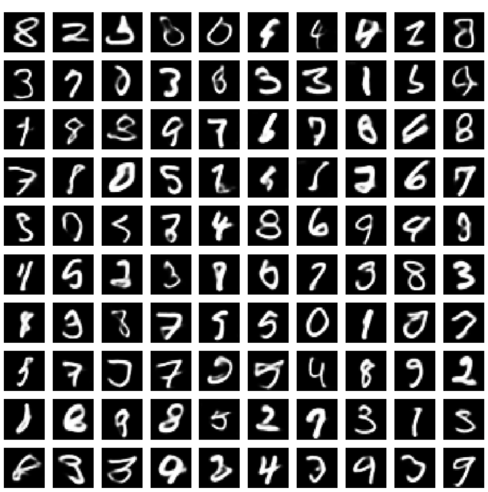


Fig. 6. Generation of MNIST digits via a convolutional network

It is also evident that if the generator ignores a latent variable, $P(X|z)$ will not depend on it and the loglikelihood is maximal when the distribution of $Q(z|X)$ is equal to the prior distribution $P(z)$, that is just a normal distribution with 0 mean and standard deviation 1. In other words, the generator is induced to learn a value $\mu_\theta(X) = 0$, ans a value $\sigma_\theta(X) = 1$; sampling has no effect, since the sampled value for $z$ will just be ignored.

| no. | variance | mean($\sigma^2_\theta(X)$) |
|---|---|---|
| 0 | 0.89744579 | 0.08080574 |
| 1 | 5.1123271e-05 | 0.98192715 |
| 2 | 0.00013507 | 0.99159979 |
| 3 | 0.99963027 | 0.016493475 |
| 4 | 6.6830005e-05 | 1.01567184 |
| 5 | 0.96189236 | 0.053041528 |
| 6 | 1.01692736 | 0.012168014 |
| 7 | 0.99424797 | 0.037749815 |
| 8 | 0.00011436 | 0.96450048 |
| 9 | 3.2284329e-05 | 0.97153729 |
| 10 | 7.3369141e-05 | 1.01612401 |
| 11 | 0.91368156 | 0.086443416 |
| 12 | 0.79746723175 | 0.23826576 |
| 13 | 7.9485260e-05 | 0.9702732 |
| 14 | 0.92481815 | 0.089715622 |
| 15 | 4.3311214e-05 | 0.95554572 |

TABLE II

INACTIVE VARIABLES IN THE VAE FOR GENERATING TILES (DENSE CASE)

During training, if a latent variable is of moderate interest for reconstructing the input (in comparison with the other variables), the network will learn to give less importance to it; at the end, the Kullback-Leibler divergence may prevail, pushing the mean towards 0 and the standard deviation towards 1. This will make the latent variable even more noisy, in a vicious loop that will eventually induce the network to completely ignore the latent variable.

The previous interpretation seems to fit with the observed behaviour (see Figure 11): each latent variable struggles to get a small variance, but at some point some of them give up the fight, getting overwhelmed by the Kullback-Leibler component. A formal analysis of the phenomenon looks difficult, due to the peculiar shape of the loss function[4], the complex interplay between the weights of the net and the computed variance, and the difficulty of decoupling the contribution of a single latent variable in the posterior probability $P(X|z)$. We are currently working on it.

## VII. CONCLUSIONS

We highlighted the interesting phenomenon of *sparsity* in Variational Autoencoders, associating it with the Kullback-Leibler component of the objective function.

[4]The regularization component in not expressed in terms of the *weights* of the nets, as for traditional L1 or L2 regularizers, so a simple study of the shape of equation 7 is not enough.

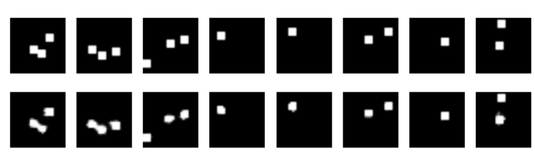


Fig. 7. Input images of the ”counting tiles” problem (upper line) and corresponding reconstructions (lower line).

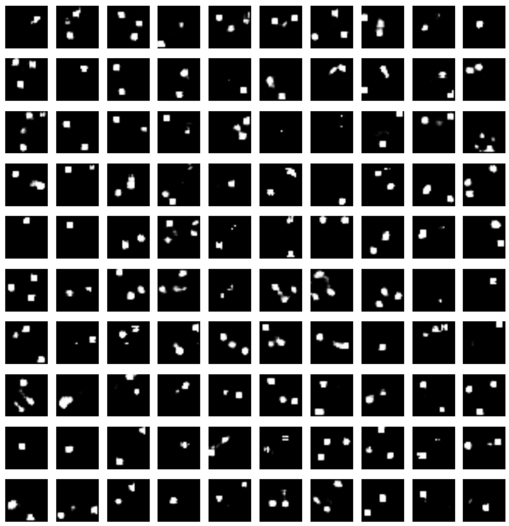


Fig. 8. Random generation of images with tiles from the latent space

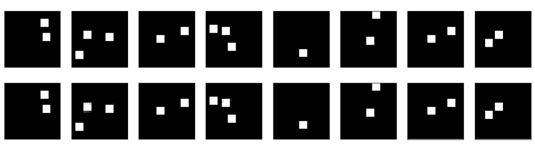


Fig. 9. Input images of the "counting tiles" problem (upper line) and corresponding reconstructions (lower line) with a convolutional network.

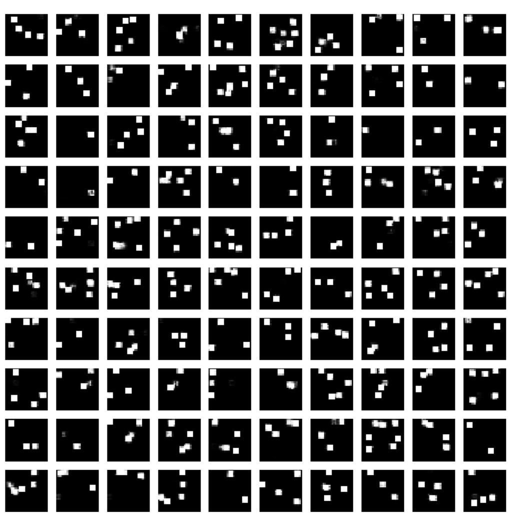


Fig. 10. Random generation of images with tiles from the latent space with a convolutional network.

Given a particular network, the degree of sparsity (the number of inactive latent variables) may slightly vary from training to training, but not in a really substantial way: in particular, for similar final values of the loss function, it is seems to be relatively stable. This seems to suggest that,

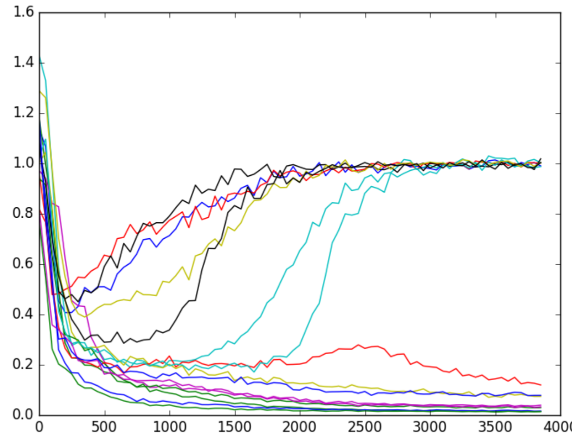



Fig. 11. Evolution of the variance along training (16 variables, MNIST case). On the x-axis we have numbers of minibatches, each one of size 128.

given a certain neural architecture, there exists an intrinsic compression of data in the latent space. If the network does not exhibit sparsity, it is probably a good idea to augment the dimension of the latent space; conversely, if the network is *sparse* we may reduce its dimension removing inactive latent variables and their connections.

## REFERENCES


[1] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2015.

[2] Carl Doersch. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908, 2016.

[3] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. 360(6394):1204–1210, 2018.

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

[5] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

[6] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[7] Tom Rainforth, Adam R. Kosiorek, Tuan Anh Le, Chris J. Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 4274–4282. JMLR.org, 2018.

[8] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org, 2014.

[9] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc., 2015.

[10] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. *CoRR*, abs/1606.07873, 2016.

[11] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*, volume 9911 of *Lecture Notes in Computer Science*, pages 835–851. Springer, 2016.

[12] Raymond A. Yeh, Ziwei Liu, Dan B. Goldman, and Aseem Agarwala. Semantic facial expression editing using autoencoded flow. *CoRR*, abs/1611.09961, 2016.