A review of Compressed Sensing with Generative Models

Guillo Clément
ENSAE & ENS Paris Saclay
clement_guillo@ensae.fr

Meunier Dimitri
ENSAE & ENS Paris Saclay
dimitri.meunier@ensae.fr

Abstract

Compressed Sensing aims at reconstructing a sparse vector living in a high dimensional space from few measurements. [1] introduced a way to perform Compressed Sensing with Generative Models. Those methods have the advantages to generate accurate reconstructions with very few measurements. They use pre-trained generative models and the method is therefore data-adaptive. Later [3] introduced a new framework to overcome the saturation of [1] in the high measurement setting. More recently [7] suggested a way to learn a Generative Model directly for the reconstruction task leading to a fully-adaptative algorithm for Compressed Sensing. It improves speed of signal recovery and performances. In this report, we will detail the three methods and explain the links with the standard Compressed Sensing theory. We illustrate the results with experiments on the Fashion MNIST dataset [6].

Introduction

In this report, we review, [1], [3] and [7]. We explain their contributions and illustrate them on the Fashion MNIST [6].

The remainder is organized as follow. In section 1, we review standard Compressed Sensing as we saw in class. In section 2, we detail every generative approach, and finally, in section 3, we illustrate the results with our experiments.

1 Sparse Compressed Sensing

The idea of Compressed Sensing is to recover a vector x living in $\mathbb{R}^n$ using m measurements such that $m << n$. It can be formalized as:

$$y = Ax$$

With A a $m \times n$ measurement matrix. Since $m << n$ the system is usually under-determined and we need additional assumptions on x. The key assumption in Compressed Sensing, is the sparsity of the signal. If this assumption is satisfied on the data, CS provides algorithms and theoretical guarantees which allow to reconstruct the original signal with high probability. This is the case if A is a Gaussian random matrix for instance, and we only need roughly $s \log(en/s)$ measures, where s is the required sparsity.

The sparsity assumption can be relaxed in two ways. First, only *approximate sparsity* is necessary, i.e., the s-largest entries in the signal dominate the rest. Secondly, *approximate sparsity* in another basis is sufficient. For instance, it is common to use wavelet basis for audio signals and Fourier basis

for images. If we call Φ the change-of-basis matrix, and $x = \Phi\tilde{x}$, only $\tilde{x}$ has to be sparse.

$$\begin{aligned} y &= Ax + \eta \\ &= A\Phi\tilde{x} + \eta \\ &= \tilde{F}\tilde{x} + \eta \end{aligned}$$

And we are back to the initial problem. We saw in class that the problem can be solved efficiently using the **Basis Pursuit** method. Given the measurement matrix A and the measurement vector y, we solve:

$$\underset{x \in \mathbb{R}^n : Ax = y}{\arg\min} \|x\|_1$$

This problem can be obtained as a convex relaxation of the l_0 version (which is a NP-hard problem) but has the advantage of being a linear program. One can for example use the simplex algorithm to solve it.

It is worth noting that in the literature the noisy case arises frequently. If we assume a noise η on the measurements, the model is rewritten $y = Ax + \eta$. We connect the problem to the high dimensional sparse regression framework. In fact, the Basis Pursuit optimization problem in that case is called **Basis Pursuit Denoising** and is modified as,

$$\underset{x \in \mathbb{R}^n : \|Ax - y\|_2^2 \leq \delta}{\arg\min} \|x\|_1$$

Using convex optimization theory [2], we can show that there is a Langrange multiplier λ such that the following problem is equivalent:

$$\underset{x \in \mathbb{R}^n}{\arg\min} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|x\|_1$$

In that form, we see tight connection to the LASSO formulation arising in high dimensional statistics. It is unconstrained and quadratic which is computationally suitable. One can for example use the interior point method in order to solve it.

2 Generative Compressed Sensing

The main problem with the traditional approach is the strong assumption of sparsity. Finding a basis such that sparsity holds requires a deep understanding of the signal and feature engineering, and therefore do not generalize to all tasks. [1] launched a line of works with methods where we get ride of the sparsity assumption using generative models. We detail this method in the next section.

2.1 Pre-trained Generative Models

Let p_{data} denote the unknown data-generating distribution. The goal of generative modeling is to train a model that induces a distribution p_G on the data space such that p_G is similar to p_{data}. Formally, G is a deterministic function $G : \mathbb{R}^k \to \mathbb{R}^n$ where $\mathbb{R}^k$ is the *latent space* and $\mathbb{R}^n$ is the sample space, usually $k << n$. Let p_l be a distribution over the latent space, if z is sampled from p_l we would like the distribution of $G(z)$ to match p_{data}.

The two most common generative models are Generative Adversarial Networks (GAN) [4] and Variational AutoEncoders (VAE) [5]. In [1], the generator G is pre-trained and instead of optimizing in the sample space as with Basis Pursuit, we optimize in the latent space given the measurements y and the measurement matrix A. Thus we solve,

$$\hat{z} \in \underset{z \in \mathbb{R}^k}{\arg\min} \|AG(z) - y\|_2^2$$

Several remarks can be made. First, usual generative models such as GAN or VAE are highly non convex. Optimization methods such as simplex methods or interior point methods no longer work and we have to turn to gradient descents methods. Backpropagation through G can be done easily using

modern deep learning libraries and at the end, we can use $\hat{z}$ in order to have a good reconstruction of the signal. Secondly, the norm in the problem is not the l_1 norm anymore but the l_2 norm. We therefore speak about l_2-reconstruction. Finally, since the latent space is smaller than the sample space it should be easier to work with z, instead of x. However, since the process is not convex anymore it is unstable and in practice it necessitates several re-start to find a good solution.

In practice, [1] suggested the regularized following problem,

$$\hat{z} \in \arg\min_{z \in \mathbb{R}^k} \|AG(z) - y\|_2^2 + \lambda \|z\|_1$$

[1] brings theoretical guarantees when the generative model is a neural network or when the generator is Lipschitz. We will now discuss the first one with the following theorem.

Theorem 2.1 *Let $G : \mathbb{R}^k \to \mathbb{R}^n$ be a generative model from a d-layer neural network using ReLU activations. Let $A \in \mathbb{R}^{m \times n}$ be a random Gaussian matrix for $m = O(kd\log(n))$, scaled so that $\forall i, j, A_{ij} \sim N(0, \frac{1}{m})$. For any $x^* \in \mathbb{R}^n$ and any observation $y = Ax^* + \eta$, let $\hat{z}$ minimize $\|y - AG(z)\|_2$ to within additive ε of the optimum. Then with $1 - e^{-\Omega(m)}$ probability,*

$$\|G(\hat{z}) - x^*\|_2 \leq 6 \min_{z^* \in \mathbb{R}^k} \|G(z^*) - x^*\|_2 + 3\|\eta\|_2 + 2\varepsilon$$

The number of measurements is now $m = O(kd\log(n))$. In the BP setting it was $m = O(s\log(en/s))$. Instead of the sparsity level s we now have the dimension of the latent space k and the number of layers d from the generator. The first term in the bound is linked to the power of the generative model. If the signal does not lie in the range of G this error term is non-zero. When comparing to standard Compressed Sensing we do not assume sparsity over the signal but we now require that the signal lie in the range of the generator such that this error is small. We therefore would like to improve G either by extending the latent space (therefore increasing k) or improving the neural architecture (therefore increasing d). In both case, this will increase the number of required measurements. We thus have to find a trade-off between the number of measurements and a good l_2-reconstruction error. The second term is the stochastic error and cannot be improve. The last term is linked to the optimization process, we can make it small by sacrificing computation time.

One thing that does not change is the use of Gaussian matrices. In class we have seen that the **Null Space Property** (NSP) is a necessary and sufficient condition for exact recovery with sparse vectors. A similar property used in high-dimensional statistics is the **Restricted Eigenvalue Condition** (REC) property.

Definition 2.1 *A satisfies the REC for a constant $\gamma > 0$ if for all approximately sparse vectors $x, \|Ax\| \geq \gamma\|x\|$. Where the approximately sparse vectors satisfy $\|x_J\| \leq \|x_{J^c}\|, \forall J \subset [n], |J| = s$, s the sparsity level.*

We saw previously that the noisy Compressed Setting is close to the LASSO setting and it can be shown that the REC condition is sufficient for sparse recovery with LASSO (see for example the High-Dimensional class of Professor Tsybakov at ENSAE).

Theorem 2.1 holds thanks to an equivalent property in the generative case. In the generative case we do not work with approximately sparse vectors but with vectors in the image of G. In the theoretical proof for the sparse setting they use the REC property and the fact that if two vectors are approximately sparse then their difference is also approximately sparse. In the generative case, if G is not linear there is no guarantee that the difference of two elements of $Im(G)$ is still in $G(\mathbb{R}^k)$. This is how [1] motivates the following S-REC property where S is taken as $G(\mathbb{R}^k)$.

Definition 2.2 *(Set-Restricted Eigenvalue Condition) Let $S \subset \mathbb{R}^n$. For some parameters $\gamma > 0$, $\delta \geq 0$ a matrix $A \in \mathbb{R}^{m \times n}$ is said to satisfy the S-REC conditions $S - REC(S, \gamma, \delta)$ if $\forall x_1, x_2 \in S$, $\|A(x_1 - x_2)\| \geq \|x_1 - x_2\| - \delta$.*

Gaussian matrices that are known to satisfy the REC property also satisfied the S-REC property when $S = G(\mathbb{R}^k)$ and G is Lipschitz or a neural network as in 2.1. [1] shows that this approach gives better results than LASSO recovery when the number of measurements is weak but it quickly saturates as

the performances are limited by the power of the generator. We will see in the next section how to eliminate this limitation in theory. The second flaw is the computation time. The method is unstable and it requires to repeat the process with different initialization z before finding a suitable candidate. [1] suggests at leat 10 restart and 1000 gradient descent iterations for each candidate.

2.2 Bypassing the generative support limitation with sparse deviations

We saw in the last section with theorem 2.1 that we can only guarantee exact recovery over the range of the generator. [3] introduced a variation that allows to recover signal with a sparse deviation from the range of the generator:

$$(\hat{z}, \hat{v}) \in \underset{z \in \mathbb{R}^k, v \in \mathbb{R}^n}{\arg\min} \|A(G(z) + v) - y\|_2^2 + \|v\|_1 \tag{1}$$

The reconstructed signal is $G(\hat{z}) + \hat{v}$. We see that $\hat{v}$ is a deviation from the range of G and minimizing $\|v\|_1$ ensures that this deviation is sparse. This can be seen as an hybrid version between the Generative setting of the last section and the traditional Basis Pursuit.

Despite the attractive theoretical properties derived in [3] we found this approach even more unstable. It eliminates the saturation of the last approach but the need for restarts is stronger. Furthermore, with the last approach we were only optimizing over the latent space which has a low dimension comparing to the sample space. Here, the added v term is in the same space as the samples, it makes the optimization process longer.

2.3 Adaptive Generative Compressed Sensing with fixed measurements matrix

In the last section we saw that we can use a pre trained generative model to perform Compressed Sensing without the sparsity assumption. The process is however limited by the power of the generative models and unstable leading to high computation times. [7] introduced a way to directly learn a generative model for the reconstruction process. Once trained, the generative model is fitted for the CS task and the reconstruction is order of magnitudes faster and more stable since we do not need any restart anymore.

Here the algorithm starts from a non trained generative network G_θ. For small batches of images and for each image x_i of a batch, an optimal $\hat{z}_i$ from the latent space is calculated such that:

$$\hat{z}_i \in \underset{z \in \mathbb{R}^k}{\arg\min} \|AG_\theta(z) - y_i\|_2^2$$

Introducing $\mathbb{E}_\theta(y_i, \hat{z}_i) = \|AG_\theta(z) - y_i\|_2^2$ the algorithm adapts the parameter θ with minimizing the empirical loss $\mathcal{L}_G$ on the batch, with $\mathcal{L}_G = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_\theta(y_i, \hat{z}_i)$.

In order to avoid that the generator finds a solution closed to the null space of A (which leads to a degenerate solution), a pseudo loss $\mathcal{L}_A$, inspired from the RIP property is minimized. This loss is written as:

$$\mathcal{L}_A = \frac{1}{N_s} \sum_{j=1}^{N_s} \left[\left(\left\| A(x_1^j - x_2^j) \right\|_2 - \left\| x_1^j - x_2^j \right\|_2 \right)^2 \right]$$

where $(x_1^j, x_2^j)_{j=1}^{N_s}$ have to be drawn randomly from the data and the Generator G_θ for a more efficient traininig. In class, we have seen that sparse signals can be recovered exactly with high probability if the measurement matrix satisfies the RIP condition. Since we do not work with sparse vectors

anymore, the strategy with $\mathcal{L}_A$ is to enforce G_θ to generate images such that the RIP condition is satisfied with A. The whole algorithm is described below.

Algorithm 1: DCS: Adaptive Generative compressed sensing Algorithm

Data: Input minibatchs of data $\{x_i\}_{i=1}^N$
initialize A, G_θ, learning rates α_z, α_θ initialization;
for $i = 1$ *to* N **do**
  $y_i = A(x_i)$;
  $\hat{z_i} \sim p_z(z)$;
  for *for* $t = 1$ *to* T **do**
    Optimize $\hat{z}_i = \hat{z}_i - \alpha_z \frac{\partial}{\partial z}\mathbb{E}(y_i, \hat{z}_i)$

$\mathcal{L}_G = \frac{1}{N}\sum_{i=1}^N \mathbb{E}_\theta(y_i, \hat{z}_i)$
Sample $(x_1^j, x_2^j)_{j=1}^{N_s}$ with (x_1^j, x_2^j) coming from the data or the Generator G_θ
$\mathcal{L}_A = \frac{1}{N_s}\sum_{j=1}^{N_s}\left[\left(\left\|A(x_1^j - x_2^j)\right\|_2 - \left\|x_1^j - x_2^j\right\|_2\right)^2\right]$
$\theta = \theta - \alpha_\theta \frac{\partial}{\partial\theta}(\mathcal{L}_G + \mathcal{L}_A)$

The following setting presents a way to learn the measurement function instead of using the random matrix A.

2.4 Fully adaptive setting: learning the measurements matrix

The idea here is to learn the parameters of the transformation A during the training. We write A_ϕ this learnable measurement function which is now a neural network.

The final algorithm is described as follow:

Algorithm 2: DCS: Fully adaptive compressed sensing Algorithm

Data: Input minibatchs of data $\{x_i\}_{i=1}^N$
initialize F_ϕ, G_θ, learning rates α_z, α_θ,α_ϕ initialization;
for $i = 1$ *to* N **do**
  $y_i = F_\phi(x_i)$;
  $\hat{z_i} \sim p_z(z)$;
  for *for* $t = 1$ *to* T **do**
    Optimize $\hat{z}_i = \hat{z}_i - \alpha_z \frac{\partial}{\partial z}\mathbb{E}(y_i, \hat{z}_i)$

$\mathcal{L}_G = \frac{1}{N}\sum_{i=1}^N \mathbb{E}_\theta(y_i, \hat{z}_i)$
Sample $(x_1^j, x_2^j)_{j=1}^{N_s}$ with (x_1^j, x_2^j) coming from the data or the Generator G_θ
$\mathcal{L}_A = \frac{1}{N_s}\sum_{j=1}^{N_s}\left[\left(\left\|A_\phi(x_1^j - x_2^j)\right\|_2 - \left\|x_1^j - x_2^j\right\|_2\right)^2\right]$
$\theta = \theta - \alpha_\theta \frac{\partial}{\partial\theta}\mathcal{L}_G$
$\phi = \phi - \alpha_\phi \frac{\partial}{\partial\phi}\mathcal{L}_A$

Link with Generative adversarial training: CSGAN The approach we just described generalize the generative adversarial network where a generative network called the Generator(G_θ in our case), is trained to fool a network call the Discriminator (A_ϕ here).

The following numerical experiments present comparative results from the classic framework and the Generative compressed sensing with pretrain generative network. However we do not present any results based on the adaptative and fully adaptative settings, since the obtained results are not qualitatively and quantitatively relevant.

3 Numerical Experiments — Compressed Sensing for Fashion MNIST

To illustrate the suggested methods from [1], [3] and [7] we conduct our experiments on the Fashion MNIST dataset [6]. Fashion MNIST shares the same image size (28*28 gray pixel images) and

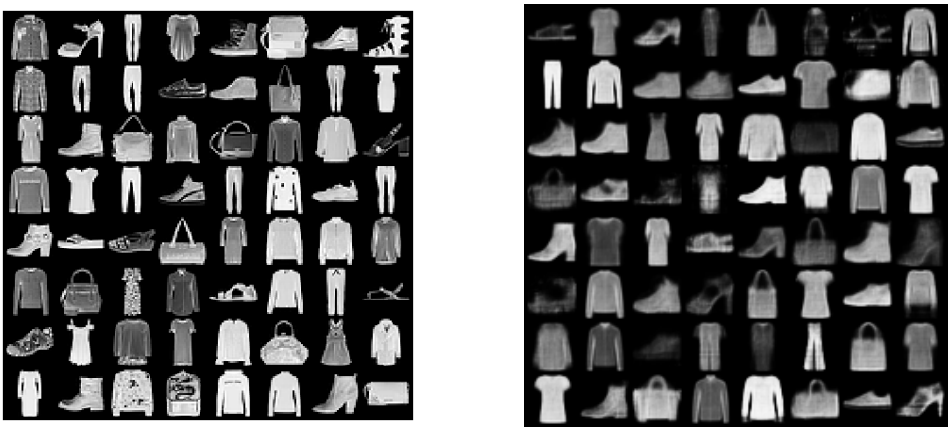

Figure 1: (*left*) 64 real images extracted from the Fashion MNIST dataset (*right*) 64 fake generations from a fully-connected VAE trained on the Fashion MNIST dataset. The generations are convincing although the details on the clothes are limited.

testing splits as MNIST. Recently, Fashion MNIST has been suggested as a drop-in replacement for the MNIST dataset as MNIST is too easy (a simple shallow Machine Learning algorithms can achieve 99.7% of accuracy), overused and does not represent modern Computer Vision tasks anymore. Figure 1 (left) gives a glimpse on the dataset.

Experimentation details. For all our experiments we trained the generative models on the full train set and evaluate the methods on a subset of the test set. The recovery process is computationally heavy and we did not have access to a multi-gpu hardware, therefore, we could only evaluate on 20 images (2 images per classes). For this reason, results are to be taken with a grain of salt as we expect high variances.

We choose the noiseless setting ($\eta = 0$) for our experiments as the generative methods are really long to train and adding noise would require spending long hours tuning the networks looking for the right architectures. That is also for this reason that we only use VAE and not GAN. On the Fashion MNIST, VAE were a lot easier to manipulate with good convergence speed.

The official GitHub repository of [1], [3] and [7] contains Tensosorflow codes that are not up-to-date. For this reason, we implemented every methods from scratch using PyTorch and CVXPY and scikit-learn. We will see that [7] was the most challenging as we did not obtain any convincing results.

Compressed Sensing without learning. Similarly to [1] and [3] we started by benchmarking the Compressed Sensing task on the Fashion MNIST using standard tools that do not require learning. Since images are already relatively sparse in the pixel space, it did not requires the use of wavelet or Fourier transforms.

As shown earlier, since we are in the noiseless case we can directly use the Basis Pursuit algorithm with CVXPY. For comparison, we also treated the problem as a sparse regression and use the LASSO algorithm with scikt-learn. Figure 2 presents the reconstruction on a subset of 10 images for 50, 300 and 700 measurements for the Basis Pursuit. Scores on the test set are in rows 1 and 2 in 5

BP performs poorly in the low and medium measurement setting, however in the high measurement setting most of the images are reconstructed accurately.

Compressed Sensing with pre-trained generative models As in [1], in a first step we trained a VAE on the dataset. For comparison we used a fully connected VAE and a convolutional VAE with latent dimension equal to 100. Architecture details can be found in the file **vae.py** in our repository at `https://github.com/DimSum2k20/DeepCompressedSensing`.

Surprisingly, the regularization mentioned in [1] was always harmful (see rows 3 and 6). The best model was the fully-connected VAE with no regularization. Reconstructions are presented in 3. We

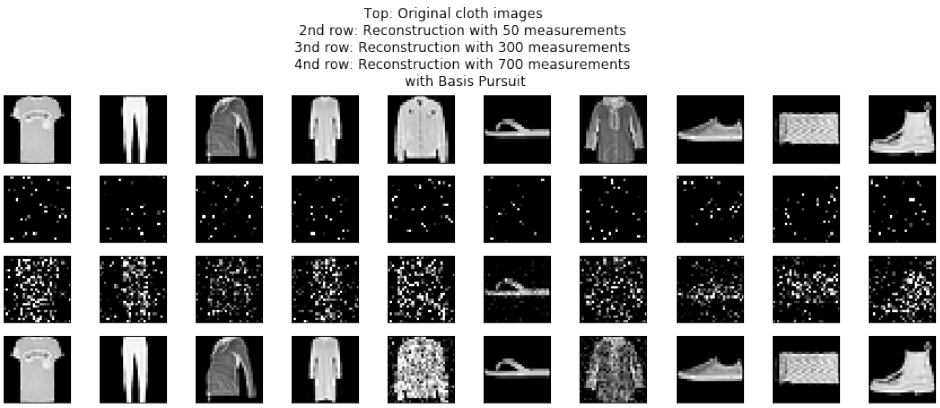

Figure 2: Reconstruction for 50, 300 and 700 measurements using Basis Pursuit.

observe accurate reconstruction with only 50 measurements. However, increasing the number of measurements does not seem to improve the results. This is the "saturation" phenomenon explains in the previous sections, as shown in 1 (right), the generator is not powerful enough to generate details in the images, therefore, in cannot reconstruct the images in great details. The reconstruction with VAE is limited by the range of the network.

Sparse deviation from the generator range. We implemented the alternative suggested by [3] to avoid the saturating effect. We kept the VAE architectures and used the optimization process from equation (1). We called those models Sparse-VAE. Reconstruction results for the fully connected Sparse-VAE can be found in figure 4. The results are disappointing as we do not see any improvements. We would have expected more details, for example the logo on the t-shirt.

Deep Compressed Sensing. Code for the approach of [7] can be found in our repository. However, the optimization process is tricky as the inner/outer loop scheme requires a careful use of back-propagation. The obtained results during our experiments are not relevant, G does not train correctly. The model requires a lot of computation times and a new training for each number of measurements. As mentioned in the paper, training the algorithm for a fixed number of measurement m requires 40,000,000 steps. This is equivalent to 70 epochs on the Fashion MNIST, assuming we have the right implementation, we approximate the training time to 6 hours on a high-end GPU. For this reason we focused on the method of [6] and [3] that are more accessible.

The method is still interesting, since the generator is learned on-the-fly, it can achieve better results with no saturating effect. For researchers having access to great computation powers, it can still be beneficial. Even if the training is heavy, once trained the recovery process is way faster that using the previous methods.

Final results. The scores are the per-pixel squared errors between the reconstructions and the true images in the test set. Overall, the best model is a fully connected VAE using the original Generative Compressed Sensing algorithm [1].

The dominance of generative methods comparing to Basis Pursuit methods is striking but the difference decreases when increasing m. Generative model saturates in the high measurement setting but still gives better results than Basis Pursuit. However, when looking at the reconstruction error images by images, the generative models reached a minimum score of 0.001968 whereas Basis Pursuit reaches 10^{-14}. This enlightens the saturating effect.

The sparse-generative framework [3] is not helpful with the fully-connected VAE but surprisingly it is helps with the convolutions version. This emphasizes that the generative methods are quite unstable and architecture-dependents. It requires a great expertise with deep learning architectures to master them.

Finally, regularization never helps.

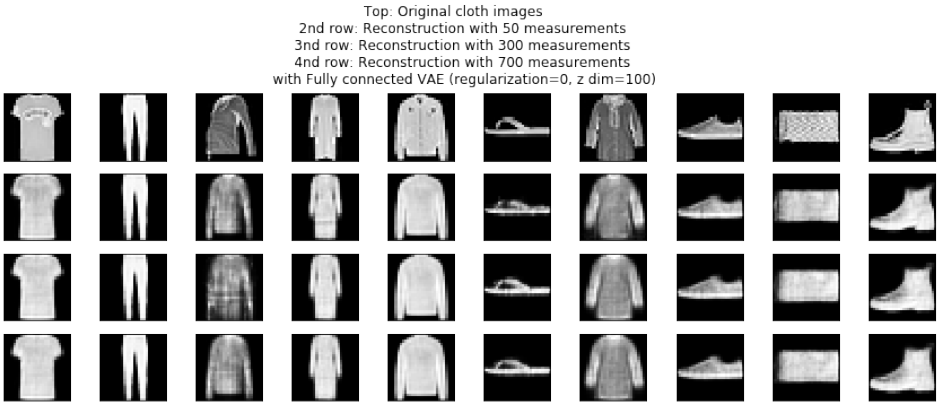

Figure 3: Reconstruction for 50, 300 and 700 measurements using a fully-connected VAE with no regularisation and latent dimension 100.

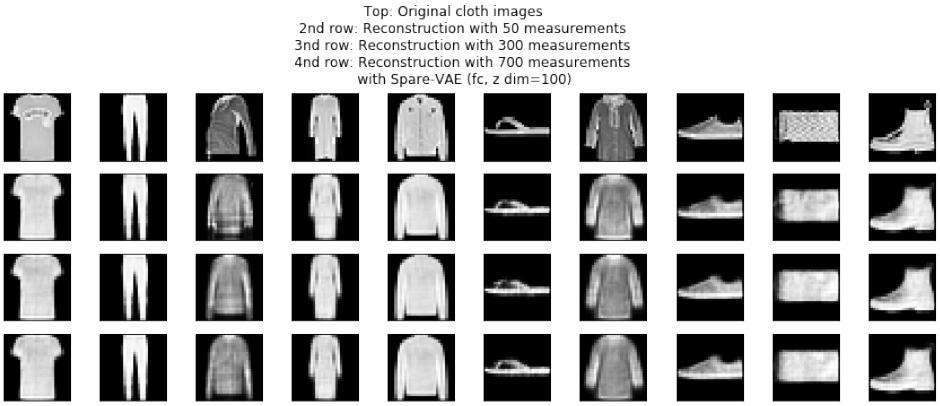

Figure 4: Reconstruction for 50, 300 and 700 measurements using a fully-connected VAE in with latent dimension 100 in the sparse-generative framework.

	m=50	m=300	m=700
Basis Pursuit	0.1973	0.1413	**0.0103**
LASSO	0.1979	0.1369	**0.0101**
VAE reg=0.1 (fc)	0.0235	0.0517	0.0757
VAE reg=0 (fc)	**0.0099**	**0.0076**	**0.0072**
VAE reg=0.1 (cnn)	0.0253	0.052	0.0769
VAE reg=0, (cnn)	0.0112	0.052	0.0769
Sparse-VAE (fc)	0.0107	**0.0075**	**0.0073**
Sparse-VAE (cnn)	0.0433	0.0094	**0.0087**

Figure 5: Results on the reconstruction task for 50, 300 and 700 measurements (input signal has 784 dimensions). Rows 1 and 2 are standard CS approaches. Rows 3 to 6 are methods from [1] where 'reg' is the regularization, 'fc' stands for fully-connected and 'cnn' for convolutional. Rows 7 and 8 are methods from [3].

Conclusion

We have seen with different methods that the generative approach consists in formulating an alternative to the sparsity hypothesis on the vector that we are trying to reconstruct. Theoretical results concerning reconstruction under the hypothesis of sparsity have been established in the course. The contribution of [1] is to suggest a generative framework that gets rid of the sparsity while keeping rigorous theoretical guarantees. [7] further extended this results using a fully adaptive framework that allows to learn the measurement function and the generator directly for compressed sensing. Whereas they show impressive results, they did not express theoretical guarantees.

To satisfy our curiosity, we experimented with every frameworks and implemented them in Pytorch on a new dataset. Our experiments highlight the fact that generative methods are unstable and that network architectures necessitate careful tuning. This is even more true with the method suggested by [7] as we failed to obtain satisfying results. Nonetheless, for the other methods, our results agree with the articles. Generative models for Compressed Sensing lead to far better results than traditional Basis Pursuit in the low and medium number of measurements setting.

References

[1] Ashish Bora et al. "Compressed sensing using generative models". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 537–546.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. USA: Cambridge University Press, 2004. ISBN: 0521833787.

[3] Manik Dhar, Aditya Grover, and Stefano Ermon. "Modeling sparse deviations for compressed sensing using generative models". In: *arXiv preprint arXiv:1807.01442* (2018).

[4] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[5] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

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

[7] Mihaela Rosca Timothy Lillicrap Yan Wu. "Deep Compressed Sensing". In: *Conference on Machine Learning PMLR* (2019).