ANALYZING GANS WITH GENERATIVE SCATTERING NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

GAN provide spectacular image generations from Gaussian white noise, which interpolate images through deformations, with little mathematical justifications. We show that such generators do not require to learn a discriminator or learn an embedding space. Deformation and gaussianization properties provide strong constraints allowing to specify the embedding space operator. It is implemented with a multiscale scattering transform. The resulting generator is computed by inverting the embedding operator, with a deep convolutional network which implements a sparse inversion. This provides a statistical framework to understand estimations of generators. The resulting generative scattering networks produce images of good quality at a fraction of the computational cost, and can learn from much smaller training sets. They define new classes of high-dimensional stochastic models for non-stationary and non-Gaussian processes.

1 INTRODUCTION

Generative Adversarial Networks (GAN) generate images of remarkable quality and complexity. This paper aims at explaining the mathematical principles behind GAN estimations, by avoiding to use a discriminator and specifying the properties of the embedding space operator. This embedding is not learned but predefined by a scattering transform. Learning a deep convolutional generator appears to be a natural solution to invert the embedding, with sparsity priors. It defines new classes of non-Gaussian and non-stationary processes, for one-dimensional as well as multidimensional signals.

A GAN is an unsupervised learning algorithm Goodfellow et al. (2014). It approximates a random vector X of dimension p by computing a stochastic model GZ, where Z is Gaussian white noise in a space of "latent variables" of dimension $d < p$. The generator G is a deep convolutional neural network Radford et al. (2016), selected by optimizing a cost, which involves an adversarial network which discriminates the new images generated by G from the original ones in the training data basis. A remarkable property of GAN synthesis is that $x = G\,z$ is progressively deformed when the latent vector z is linearly modified. Mathematical models Goodfellow et al. (2014); Arjovsky et al. (2016) argue that GAN's select the generator G by minimizing estimations of a Jensen-Shannon divergence or a Wasserstein distance, on the probability distributions of the generated images or of the noise Gaussian Z. However this explanation fails to pass the curse of dimensionality, since estimations of Jensen-Shannon and Wasserstein distances don't generalize with a number of examples which is a polynomial of the dimension Arora et al. (2017).

Variational autoencoder provide an alternative approach to GAN, by optimizing G together with its inverse $\Phi = G^{-1}$ instead of a discriminator. Th inverse Φ is the embedding operator which transforms X into a Gaussian white noise Z. Variational autoencoders generate images similar to GAN, but more blurred. Their cost functions are also based on probabilistic distances which suffer from the same dimensionality curse Arora et al. (2017). The inversion of G has also been computed for GAN generators Creswell & A. (2016), but a GAN does not use it in the optimization of G. Generative Latent space optimization was introduced in Bojanowski et al. (2017) to eliminate the need for a GAN discriminator while restoring sharper images than variational autoencoders. It still uses an autoencoder computation structure, where the latent space variables z are optimized together with the generator G. Despite good results, linear variations of the embedding space variables are not mapped as clearly into image deformations as in GAN's, which reduces image qualities.

GAN and autoencoders raise many questions. What are the properties of the embedding operator Φ and the latent variables ? Where are deformation properties coming from ? Why does it seems to generalize despite the curse of dimensionality ?

This paper shows that the embedding Φ does not need to be learned and can be specified from a priori properties. This embedding must Gaussianize X and it must be Lipschitz continuous to deformations so that linear modifications of the embedding space result in a deformation of X. Section 2.1 explains that the curse of dimensionality prevents finding an invertible Φ, but one can limit the information loss by controlling the mutual information between X and ΦX. However, an inverse generator G which minimize the mean-square loss $\mathbb{E}(\|X - G\Phi X|^2)$ can be estimated with sparsity priors. The statistical properties of the generator thus results from the resolution of an ill-posed inverse problem.

Lipschitz continuity to deformations, and Gaussianizing a random vector without loosing too much information are very conditions which strongly specify the class of operators. Lipschitz continuity to deformations requires to separate the signal variations at different scales which leads to wavelet scattering transforms. These operators are Lipschitz continuous to deformations and include a scale parameter which adjusts the trade-off between Gaussianization and maximization of mutual information. The generator G can thus be computed by inverting the scattering operator. We shall explain why this inversion is implemented with a deep convolutional networks, as in GAN and autoencoders. As opposed to GAN, we can compute the generalization errors on test samples. Numerical results demonstrate that these deep convolution network perform sparse expansions in a redundant dictionary, which is a well-known strategy to regularize ill-posed inverse problems Hastie et al. (2015). Generative Scattering networks can also be interpreted as an associative memory since it can recover close approximations of training images. The embedding space Z plays the role of an address allowing to associate close images. Experiments in Section 3.3 show that these Generative Scattering Networks generate images having similar qualities and properties as GAN generators, with much less computations. Generators are well estimated with fewer examples than GAN because the embedding is not learned.

2 Computing a Generator from an Embedding

2.1 Embedding with a Maximum Entropy Gaussianization

Images $x = Gz$ generated by a GAN appear to be deformed when the latent variable z is linearly modified. It implies that the inverse Φ of G must locally linearize small translations and deformations, and hence be Lipschitz continuous to actions of diffeomorphisms. It must also Gaussianize the random process X without loosing too much information. The mutual information between ΦX and X is maximum if Φ is invertible. Finding invertible Lipschitz functions which Gaussianize a random vector X is remarkably difficulty in more than one dimension. In high dimension, we must relax the invertibility condition. It leads to a trade-off between Gaussianization and mutual information, that we shall review.

In one dimension $x \in \mathbb{R}$, an invertible differentiable operator Φ which Gaussianize a random variable X can be computed as the solution of a differential equation Friedman (1987), which transports the histogram into a Gaussian. In higher dimension, there is no such equation and this problem is addressed through independent component analysis. Indeed, if ΦX is Gaussian then its projections over the eigenvectors of its covariance matrix are independent Gaussian random variables. Gaussianization algorithms can thus be defined as a search for independent variables Chen & Gopinath (2000). Invertible Gaussianizations are then computed with greedy algorithms which identify least dependant one-dimensional variables, which are Gaussianized one after the other. However, the identification of nearly independent variables faces the curse of dimensionality when X includes many non-Gaussian dependant variables, which is the case for images.

Finding Lipschitz functions which Gaussianize X becomes much easier if we do not impose that Φ is invertible. Gaussianization results from the central limit theorems if we average enough nearly independent random variables having variances of same order of magnitude. The average $p^{-1} \sum_u X(u)$ is thus likely to be nearly Gaussian. The difficulty is to avoid loosing too much information, and hence control the mutual information between X and $Y = \Phi X$. Since Y is a function of X the mutual information is equal to the entropy $H(\Phi X)$. To avoid having infinite entropy with continu-

ous valued random vectors, this entropy must be replaced by Kolmogorov ϵ-entropy rate $H_\epsilon(\Phi X)$ Cover & Thomas (2006). It computes the entropy of coverings by sets of radius ϵ, where ϵ is an approximation error. Let Y be a Gaussian random vector having a covariance whose eigenvalues are $\{\sigma_k^2\}_{k\leq p}$. The ϵ entropy rate is asymptotically given by Posner et al. (1969)

$$H_\epsilon(Y) \sim \frac{1}{2p}\sum_{k=1}^{p}\log\left(1+\frac{\sigma_k^2}{T_\epsilon}\right) \text{ where } \sum_{k=1}^{p}\frac{\sigma_k^2}{1+\frac{\sigma_k^2}{T_\epsilon}} = \epsilon^2. \tag{1}$$

The parameter T_ϵ plays the role of a threshold which eliminates the smallest eigenvalues σ_k^2 whose sums are bounded by ϵ^2.

The maximum entropy of $H_\epsilon(\Phi X)$ is $H_\epsilon(X)$ if Φ is invertible so we know that maximizing the entropy of $H(\Phi_\epsilon X)$ is computationally not feasible. To compare ϵ-entropies, the Lipschitz embedding Φ is normalized by imposing that it is contractive: $\|\Phi x - \Phi x'\| \leq \|x - x'\|$. Measuring accurately the trade-off between Gaussianization and reduction of mutual entropy is still not possible, because we can not compute a reliable distance to a high-dimensional Gaussian distribution with a number of samples which is polynomial in the dimension. We shall thus predefine parametrized operators which adjust the trade-off between Gaussianization and mutual entropy. This parameter can be optimized as a trade off between entropy and some weak Gaussianization metric Arora et al. (2017), or by having an adversary user which chooses the best parameter.

2.2 Generator Calculation as an Inverse Problem

Computing the generator G by inverting the embedding Φ is an ill-posed inverse problem since Φ is not invertible. The goal is to recover a close approximation of X from ΦX. Following standard ill-posed problem approaches, G is estimated by minimizing a regularized mean-square error over n training examples $\{x_i\}_{i\leq n}$ Hastie et al. (2015). The regularization may be implemented with a parametrized class of operators $\mathcal{G}$:

$$G = \min_{\overline{G}\in\mathcal{G}} n^{-1}\sum_{i=1}^{n}\|x_i - \overline{G}\Phi x_i\|^2 . \tag{2}$$

Since Φ is imposed to be contracting, the covariance of ΦX is typically not the identity. For synthesis, we first adjust the mean and covariance of the white noise Z to the ones of ΦX. Let $\widehat{\Sigma} = n^{-1}\sum_{i=1}^{N}\Phi x_i\,\Phi x_i^T$ be an empirical estimation of the covariance Σ of ΦX. Let $\{\hat{\sigma}_k^2\}_k$ be its eigenvalues in decreasing sorted order. To control the estimation error by ϵ^2 we associate a threshold T_ϵ such that $\sum_{\hat{\sigma}_k^2\geq T_\epsilon}\sigma_k^2 \leq \epsilon^2$. An upper bound on operator norm error

$$\|\Sigma - \widehat{\Sigma}\| \leq T_\epsilon$$

is obtained with an arbitrarily small probability if $n = \alpha\, T_\epsilon^{-2}$ with α is sufficiently large Vershynin (2011). A regularized empirical covariance matrix $\widehat{\Sigma}_\epsilon$ is defined by putting to zero the smallest $n-d$ eigenvalues $\sigma_k^2 \leq T_\epsilon$. Let $\widehat{\Sigma}_\epsilon^{1/2}$ be its square root, whose eigenvalues are the d non-zero σ_k in a basis which diagonalizes $\widehat{\Sigma}$. Let $\hat{\mu}_\epsilon$ be the projection of $\hat{\mu} = n^{-1}\sum_{i=1}^{N}\Phi x_i$ over the same d eigenvectors. For a Gaussian white noise Z of dimension d, the difference between the covariance of $\widehat{\Sigma}_\epsilon^{1/2}\,Z$ and ΦX in operator norm and Froebenius norm are controlled by T_ϵ and ϵ^2.

The resulting estimated model of X is defined from the generator G by adjusting the mean and covariance of Z:

$$\widehat{X} = G\,\widehat{\Sigma}_\epsilon^{1/2}\,(Z + \widehat{\mu}_\epsilon) . \tag{3}$$

The embedding of an image x is

$$z = \widehat{\Sigma}_\epsilon^{-1/2}\,(\Phi x - \widehat{\mu}_\epsilon) . \tag{4}$$

The ability easily compute this embedding, which is not the case for a GAN, allows to use this network as an associative memory.

3 GENERATIVE SCATTERING NETWORK

3.1 GAUSSIANIZATION BY MULTISCALE SCATTERING

To define an operator which is Lipschitz continuous to deformations, it is explained in Mallat (2012) that one must separate the variations of x at different scales, which is done by a wavelet transform. The linearization of large translations is obtained by averaging over scales 2^j which may be large. Non-linearities appear as a necessity to create interactions across multiple scale coefficients, which reduce the information loss due to averaging. In the context of this paper, the averaging at scale 2^j is used to Gaussianize the random vector X. We shall see that the scale 2^j adjusts the trade-off between Gaussianization and loss of information. A scattering transform thus appears to be a good candidate to construct an image embedding for image generation. Multiscale scattering operators are implemented in deep convolutional network architectures, with predefined wavelet filters Mallat (2016). We briefly review their properties.

A scattering operator S_j transforms $x(u)$ into a network layer $x_j(u,k)$ at the depth j, where the spatial parameter u is sampled at intervals 2^j. The number K_j of channels indexed by k is increasing with j, to partially compensate for the loss of spatial resolution. These K_j channels are computed by a non-linear covariant transformation $\Psi_j x(u,k)$, calculated with iterated wavelet transforms.

In this paper, Ψ_j is computed as successive convolutions with complex two-dimensional wavelets and modulus, with no channel interactions. Following Bruna & Mallat (2013), the mother wavelet ψ is a Gabor function, scaled by 2^ℓ and rotated along Q angles $\theta = q\pi/Q$:

$$\psi_{\ell,q} = 2^{-2\ell}\,\psi(2^{-\ell} r_\theta u) \;\text{ for }\; 0 \le q < Q.$$

At the order 2 the multichannel scattering operator Ψ_j computes up to 2 convolutions with wavelets followed by a modulus:

$$\Psi_j x = \Big[x\,,\, |x \star \psi_{\ell,q}|\,,\, ||x \star \psi_{\ell,q}| \star \psi_{\ell',q'}|\Big]_{1\le \ell<\ell'\le j, 1\le q,q'\le Q}\,.$$

There are $K_j = 1 + Qj + Q^2 j(j-1)/2$ channels. A scattering transform averages along these channel signals with a low-pass filter ϕ_j whose spatial width proportional to 2^j:

$$S_j x = \Psi_j x \star \phi_j = \Big[x \star \phi_j\,,\, |x \star \psi_{\ell,q}| \star \phi_j\,,\, ||x \star \psi_{\ell,q}| \star \psi_{\ell',q'}| \star \phi_j\Big]_{1\le \ell<\ell'\le j, 1\le q,q'\le Q}\,. \quad (5)$$

Convolutions by ϕ_j are followed by a subsampling by 2^j. As a result, if $x \in \mathbb{R}^p$ then S_j is of dimension $\alpha_j p$ where

$$\alpha_j = 2^{-2j}(1 + Qj + Q^2 j(j-1)/2).$$

The maximum scale 2^j is limited by the image width $2^j \le p^{1/2}$. In numerical applications $Q = 8$. For images x of size $p = 64^2$, $\alpha_j > 1$ and so $S_j x$ has more coefficients than x for $j \ge 3$ but when $j > 3$ we get $\alpha_4 = 0.88$, $\alpha_5 = 0.66$ and $\alpha_6 = 0.25$. Based on this coefficient counting, we can hope that S_j is invertible for $j \ge 3$, nearly for $j = 4$ but not for $j = 5$ and $j = 6$.

The wavelets separate the variations of x at different scales ℓ along different directions $q\pi/Q$ and second order coefficients compute interactions across scales. Because of this scale separation, one can prove that S_j is Lipschitz continuous to translations and the action of diffeomorphisms Mallat (2012). Wavelets are also defined with a Littlewood-Paley condition which guaranties that the wavelet transforms are contractive operators and that S_j are also contracting Mallat (2012).

The wavelets $\psi_{\ell,q}$ and the low-pass filter ϕ_j can be factorized as a cascade of convolutions and subsamplings with small size filters, followed by a modulus non-linearity. As a result Mallat (2016), a scattering transform can be implemented by applying convolution matrices V_j and applying a modulus non-linearity:

$$S_j = |V_j S_{j-1}|.$$

It is thus implemented by a convolutional neural networks whose filters are specified by wavelets and where the non-linearity is chosen to be a modulus, as illustrated in Figure **??**.

Because of the spatial averaging by ϕ_j, the operators S_j becomes progressively more Gaussian as the scale 2^j increases if wavelet coefficients become nearly independent when they are sufficiently

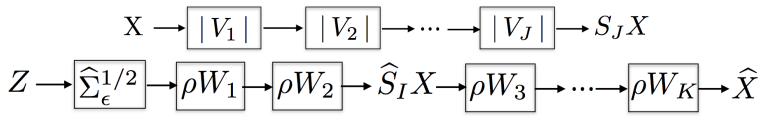

Figure 1: Top: the embedding $S_J X$ is computed by cascading J contracting convolutional operators V_j followed by a modulus. Bottom: the generation adjusts the covariance and a deep convolutional network computes a regularized inversion of the scattering operator S_J with a Relu ρ. An intermediate estimation of $S_I X$ is computed after the first two layers.

far away. If $X(u)$ is independent from $X(v)$ for $|x - v| \geq \Delta$ then the central limit theorem proves that $S_j X$ converges to a Gaussian distribution when $2^j/\Delta$ increases. Unless X is originally nearly Gaussian, this Gaussianization usually happens at a large 2^J where S_J is not invertible. Inverting a scattering embedding $\Phi = S_J$ then amount to solving an ill-posed inverse problem.

3.2 Inverse Scattering with Deep Convolutional Networks

We compute $S_J x$ from $S_0 x = x$ by iteratively calculating $S_j x$ from $S_{j-1} x$ which amounts to reduce the spatial resolution while increasing the number of channels. One may thus invert $S_J x$ by progressively inverting each $S_j x$ for $J \leq j \leq 0$ Andén & Mallat (2014). We shall see that such cascaded inversions can be efficiently implemented by deep convolutional generators used in GAN. When J is too large then S_J is not invertible, but we shall see that its inversion is stabilized by taking advantage of sparse expansions of scattering vectors.

The inversion of S_J have been computed with an iterative algorithm to test the inversion and model stationary stochastic processes **?**. We recover an approximation of such an $\tilde{x}$ by minimizing the loss

$$\|S_J \tilde{x} - S_J x\|^2 \tag{6}$$

with a gradient descent which is initialized with a Gaussian white noise. If S_J is not invertible then $\tilde{x} \neq x$ and depends upon the initialization. Such reconstructions can be interpreted as sampling a maximum entropy microcanonical model conditioned by the observation $S_J x$ Bruna & Mallat (2013). It defines stochastic models of stationary ergodic processes but does not apply to non-stationary processes such as images of faces. We shall denote 2^I the maximum invertibility scale for which the inversion of S_I is stable. As expected from the ratio between the number of image pixels and the number of scattering coefficients, Figure 2 shows that for images of $p = 64^2$ pixels, the minimization of (6) nearly recovers x from $S_J x$ for $J = 4$, because has nearly as much coefficients as x. Because of the information loss for $J = 5$ the reconstructed image $\tilde{x}$ is very different and depends upon the initialization.

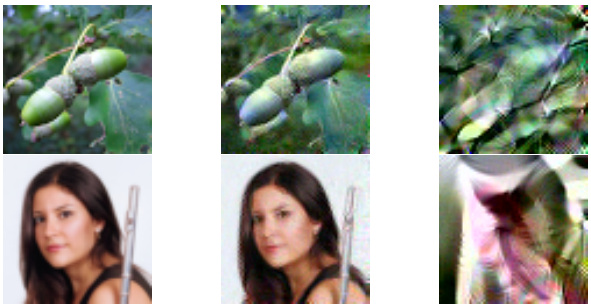

Figure 2: Left: original images. Middle: reconstructed from $S_4 X$. Right: reconstructed from $S_5 X$.

Even-though S_J may not be invertible, according to (2) a generator G may be optimized by minimizing empirical mean-square error:

$$G_J = \min_{\overline{G} \in \mathcal{G}} n^{-1} \sum_{i=1}^{n} \|x_i - \overline{G} S_J x_i\|^2 \,. \tag{7}$$

We define the operator class $\overline{G}$ as a parametrized convolutional neural network whose architecture is adapted to a progressive reconstruction of the $S_j x$ for $J \geq j \leq 0$. It computed with a cascade of linear operators W_j with bias b_j and a Relu non-linearities, illustrated in Figure **??**. These parameters are optimized with a stochastic gradient descent, by minimizing the regression loss (7).

Analysis of GAN generators The scattering inversion is implemented with a convolutional network generator which has the same architecture as a GAN generator. It is optimized according to (2), to reconstruct an approximation of the training samples x_i from $S_J x_i$. Notice that the first layer of a GAN generator takes in input the noise vector Z and computes

$$Z_1(u,k) = \rho(W_1 Z(u,k) + b_1(u,k)) \tag{8}$$

where the bias $b_1(u,k)$ can be adjusted at each location u. At all other layers, the bias $b_j(u,k) = b_j(k)$ is invariant by translation. It results that if $Z_1(u,k)$ is translated along u then the reconstructed signal $\tilde{X}(u)$ is also translated. The non-stationary properties of $\tilde{X}(u)$ are thus entirely captured by the first layer (8). Moreover, more than $3/4$ of the parameters of a GAN generator are stored in the second matrix W_2 because the first two layers are the biggest ones. We thus concentrate on the analysis of these two first layers.

We will show that the GAN generators regularizes the ill-posed inversion with sparse representations. For this purpose we also implemented a reconstruction decomposed in two stages. Let 2^I be the invertibility scale of the scattering transform. In Figure 2, $2^I = 16$. The first step computes an estimation $\widehat{S}_I X$ of $S_I X$ from $S_J X$. It solves the most ill-posed part of the scattering inversion. It is implemented with the first two layers of a GAN. The second network estimates X from $\widehat{S}_I X$. It is implement with the next layers of a GAN.

The estimation of $S_I X$ from $S_J X$ is thus implemented with a a one-hidden layer network

$$\widehat{S}_I X = W_2 \rho(W_1 Z + b),$$

where W_2 is a multichannel operator which is convolutional in space. The parameters are optimized by minimizing the empirical loss of this first inverse problem:

$$n^{-1} \sum_{i=1}^{n} \Big(S_I x_i - W_2 \rho(W_1 S_J x_i + b) \Big)^2 . \tag{9}$$

Because W_2 is a convolution, $S_I x(u,k)$ is computed from $\rho(W_1 S_J x + b)$ with the same matrix at each position u. All scattering vectors $S_I x(u,.)$ for u fixed are therefore computed as linear combinations of columns vectors of W_2. The decomposition coefficients is given by $\rho(W_1 S_J X + b)$. Sparse expansions of solutions is a standard approach to regularize ill-posed inverse problems. On average over all the image data bases described in the next section, the proportion of non-zero coefficients of $\rho(W_1 S_J X + b)$ is 98%. This means that the network optimization computes a very sparse expansion by choosing large amplitude bias which act as a threshold, and produce a sparse activation layer $\rho(W_1 S_J X + b)$. This two hidden-layer network thus perform a sparse estimation of $S_I X$ from $S_J X$. The second stage is implemented with the second part of a GAN generator, to recover an estimation of $S_J X$ from the estimation $\widehat{S}_I X$ of $S_I X$.

For the end to end convolutional neural network, we do not observe this strong sparsity effect in the first layer. The proportion of zero coefficients of the first layer $\rho(W_1 S_J X + b)$ is about 50% on average over all images of the test data bases. The sparse regularization seems to be diluted across the whole network which makes it more difficult to analyze. As expected, results obtained by the end to end convolutional network are slightly better than the one obtained by the two stage neural network so next section will give results obtained with the full convolutional network.

Associative Memory and Associative Fields We shall see in the next section that such a generator recover complex images such as faces from a Gaussian white noise Z, and stores with a high precision the images x_i of the training data basis. The network can thus also be interpreted as an associative memory storage, where Z has the role of an address. Indeed, we can compute the address $z = S_J x$ of an image x, choose z' which is close to z and recover an image x' with the generator. Next section will show that x' is essentially a deformation of x and hence a perceptually close image.

In the two stage reconstruction network, the memory is essentially stored in the first two layer network. Indeed, little memory is needed to recover x from $S_I x$ since $S_I x$ is nearly invertible. The memory of the first network is stored in the dictionary W_2 whose columns capture important structures of $S_I x(u, .)$ to build sparse approximations. However it does not control the relative positions of these vectors as a function of u. This is encoded by the first matrix W_1 and the bias b which can be interpreted as an associative field. Indeed, it encodes the relative spatial positions of structures in $S_I x(u, .)$. The next layers are convolutional and are therefore not able to encode non-stationary spatial positioning.

3.3 Numerical Experiments

We consider three datasets of color RGB images of 64^2 pixels: : CelebA, ImageNet64x64 and Polygon5. The last dataset are images of random polygons of at most 5 vertex, with random colors. Each dataset has 8192 training images and 2048 test images. All results are obtained with a scattering generator generator G_5 which inverts S_5 at the scale $2^5 = 32$. The architecture of the generator is identical to the architecture in Radford et al. (2016) modulo the addition of two more layers at the finest scale. Nearly no optimization was performed on this architecture.

Figure 3 shows images reconstructed from the generator G_5 by interpolating the latent variables z_i and $z_{i'}$ of two images x_i and $x_{i'}$: $z = \alpha z_i + (1-\alpha) z_{i'}$ with $\alpha \in [0, 1]$. Observe that this interpolation result in a continuous deformation from one image to the other while colors and image intensities are also adjusted. It reproduces similar results as the ones obtained by a GAN. This is a consequence of the Lipschitz continuity of a scattering transform relatively to actions of diffeomorphisms.

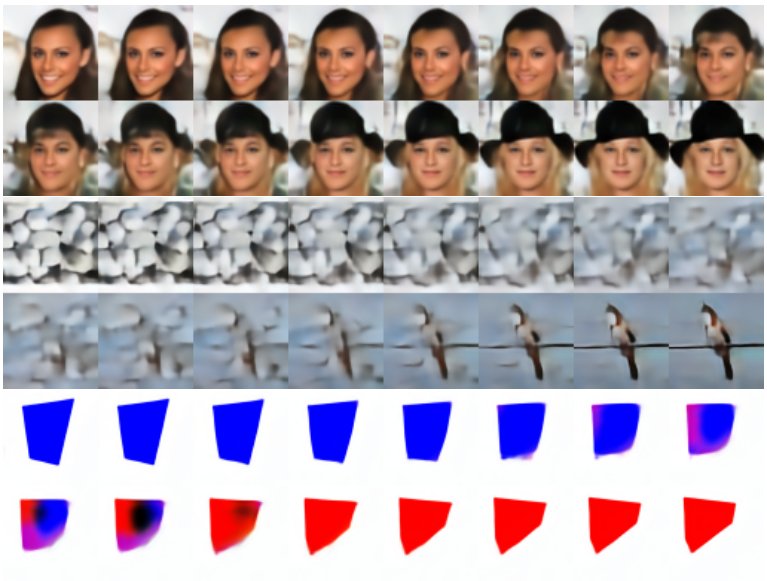

Figure 3: Interpolated images, between two training images x_i and $x_{i'}$ of each database, recovered by the scattering generator G_5.

	CelebA	**Polygon5**	**ImageNet64x64**
Train	30.80	36.43	26.44
Test	22.97	28.51	20.04

Table 1: Reconstruction error PSNR (dB) after 500 epochs, for training and testing images in each data basis.

Figure 4 shows reconstruction of training images x_i from their embedding z_i computed with (4). Figure 5 shows reconstruction of testing images x_i. Training images are much better reconstructed than test images. Table 1 gives the average training and testing errors for each data sets after 500 epoch. Training image recover is about 8db above test image recovery. This shows that the generator

Figure 4: Top: original images in the training set. Middle: scattering generation with G_5 at scale $2^J = 32$.

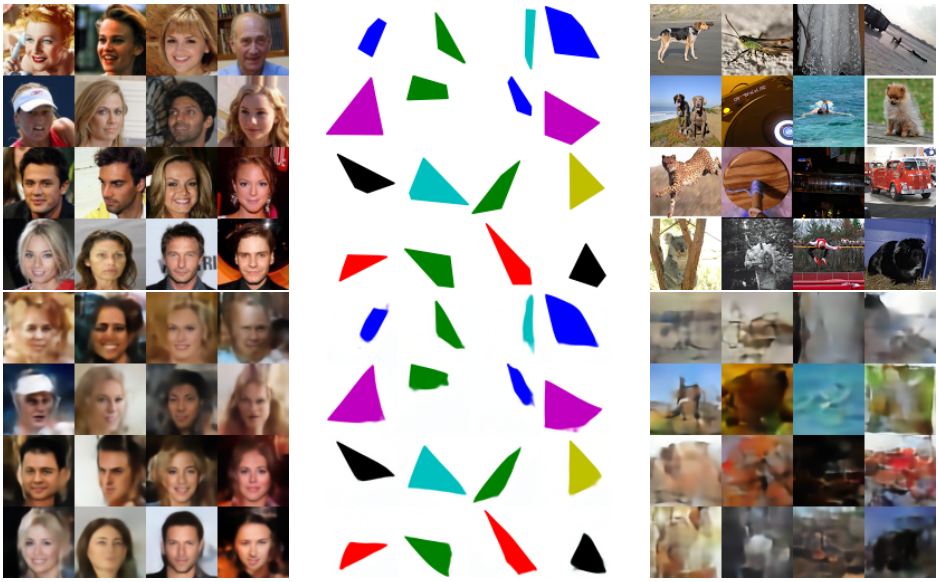

Figure 5: Original images in the test set and their reconstructions with G_5.

has clearly overfit. This overfitting futher appears in Figure 6 which shows reconstructions of images from a random sampling of the input Gaussian random variable Z. Although it does recover images that resemble to faces, color patches or complex structures, the image quality is further degrade

Overfitting is clearly not good for unsupervised learning where the intent is to model a probability density. Avoiding to overfit requires to further regularize the reconstruction network, which has not been tried yet. However, if we consider this network as a memory storage, overfitting may not be a bad property. We may indeed want to better recover known faces than "unknown" faces. Simple images such as polygons have a training error which is much smaller than faces or more complex image net images. This can be explained by the fact that they have much more sparse wavelets representations. Scattering coefficients are averaging the modulus of wavelet coefficients and are

thus "local" $\mathbf{l}^1$ norms. They encode more precisely images having a sparse representation than non-sparse

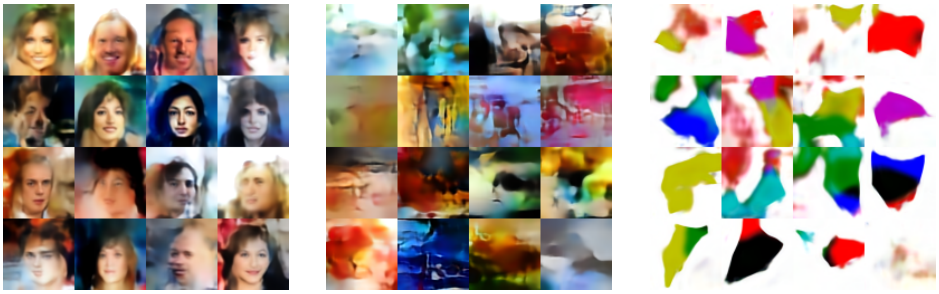

Figure 6: Sampling from a Gaussian white noise Z.

3.4 CONCLUSION

This paper shows that most properties of GAN can be reproduced with an embedding computed with a scattering transform, which avoids uses a discriminator as in GAN or learning the embedding as in autoencoders. It also provides a mathematical framework to analyze the statistical properties of these generators through the resolution of an inverse problem, regularized by sparsity.

Because the scattering embedding function is known numerical results can be evaluated on training as well as testing sets. Depending upon applications, one may want to use a generative scattering network as an associate memory and thus store more precisely the training images, in which case we are facing a coding problem which can be evaluated in terms of distortion rate functions. On the contrary, for unsupervised learning, we must avoid overfitting and regularize the generator in order to obtain similar errors on training and testing images.

This paper shows preliminary numerical results with no hyper parameter optimization. Test on noise vector arithmetic and conditional generation have not yet been carried. Potential numerical improvement have not been explored. In particular, the architecture of the CNN generator can be adapted to the properties of the forward scattering operator S_j as j increases. The paper uses a "plain"scattering transform which does not take into account interactions between angle and scale variables. This is known to improve synthesis and classification results of most scattering applications Oyallon & Mallat (2015); Anden et al. (2015).

REFERENCES

J. Andén and S. Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62: 4114–4128, 2014.

J. Anden, V. Lostanlen, and S. Mallat. Joint time-frequency scattering for audio classification. In *Proc. of IEEE MLSP*, 2015.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. In *arXiv:1701.07875*, 2016.

S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets. *arXiv:1703.00573*, 2017.

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv:1707.05776*, 07 2017.

J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1872–1886, 2013.

S.S. Chen and R. A. Gopinath. Gaussianization. In *Proc. NIPS*, 2000.

T. Cover and J. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications, 2006.

A. Creswell and Bharath A. Inverting the generator of a generative adversarial network. In *Proc. NIPS Workshop*, 2016.

J.H. Friedman. Exploratory projection pursuit. *J. American Statistical Association*, 82:249–266, 1987.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, Ozair S., A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, pp. 2672–2680, 2014.

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity*. CRC Press, 2015.

S. Mallat. Group invariant scattering. *Commun. Pure Appl. Math.*, 65(10):1331–1398, 2012.

S. Mallat. Understanding deep convolutional networks. *Philos. Trans. Royal Society A*, March 2016.

E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. In *Proc. of CVPR*, 2015.

E.C. Posner, E.R. Rodemich, and Rumsey H. Epsilon entropy of gaussian processes. *Ann. Math. Statist.*, 40(4):1272–1296, 1969.

A. Radford, L. Metz, and R. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. of ICLR*, 2016.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. In *Chapt. of Comprised Sensing, Theory and Applications*, 2011.