GENERATIVE NETWORKS AS INVERSE PROBLEMS WITH SCATTERING TRANSFORMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Adversarial Nets (GANs) and Variational Auto-Encoders (VAEs) provide impressive image generations from Gaussian white noise, but the underlying mathematics are not well understood. We compute deep convolutional network generators by inverting a fixed embedding operator. Therefore, they do not require to be optimized with a discriminator or an encoder. The embedding is Lipschitz continuous to deformations so that generators transform linear interpolations between input white noise vectors into deformations between output images. This embedding is computed with a wavelet scattering transform. Numerical experiments demonstrate that the resulting scattering generators have similar properties as GANs or VAEs, without learning a discriminative network or an encoder.

1 INTRODUCTION

Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs) allow training generative networks to synthesize images of remarkable quality and complexity from Gaussian white noise. This work shows that one can train generative networks having similar properties to those obtained with GANs or VAEs without learning a discriminator or an encoder. The generator is a deep convolutional network that inverts a predefined embedding operator which is Lipschitz continuous to deformations in order to reproduce relevant properties of GAN image synthesis. This embedding operator is implemented with a wavelet scattering transform. Defining image generators as the solution of an inverse problem provides a mathematical framework, which is closer to standard probabilistic models such as Gaussian autoregressive models.

GANs were introduced by Goodfellow et al. (2014) as an unsupervised learning framework to estimate implicit generative models of complex data (such as natural images) by training a generative model (the generator) and a discriminative model (the discriminator) simultaneously. A generative model of the random vector X is an operator $\widehat{G}$ which transforms a Gaussian white noise random vector Z into a model $\widehat{X}$ of X, i.e., $\widehat{X} = \widehat{G}(Z)$. When $\widehat{G}$ is a deep convolutional network it is called a generative network. Radford et al. (2016) have introduced deep convolutional architectures for the generator and the discriminator, which result in high-quality image synthesis. When the latent variable z is linearly modified the resulting image $\widehat{x} = \widehat{G}(z)$ is progressively deformed.

Mathematical models as in Goodfellow et al. (2014) and Arjovsky et al. (2016) argue that GANs select the generator $\widehat{G}$ by minimizing estimations of the Jensen-Shannon divergence or the Wasserstein distance calculated from empirical estimations of these distances with generated and training images. However, Arora et al. (2017) prove that this explanation fails to pass the curse of dimensionality since estimates of Jensen-Shannon or Wasserstein distances do not generalize with a number of training examples which is a polynomial of the dimension. Therefore, the reason behind the generalization capacities of generative networks remains an open problem.

VAEs, introduced by Kingma & Welling (2014), provide an alternative approach to GANs, by optimizing $\widehat{G}$ together with its inverse on the training samples, instead of using a discriminator. The inverse Φ is an embedding operator (the encoder) that is trained to transform X into a Gaussian white noise Z. A significant disadvantage of VAEs is that the resulting generative models produce blurred images compared with GANs. The loss function to train a VAE is also based on probabilistic distances which suffer from the same dimensionality curse shown in Arora et al. (2017).

It has been shown in Creswell & Bharath (2016) that GAN generators can be inverted, even if this is not imposed in their optimization. Generative Latent Optimization (GLO) was introduced in Bojanowski et al. (2017) to eliminate the need for a GAN discriminator while restoring sharper images than VAEs. GLO still uses an autoencoder computational 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 GANs, which reduces the quality of generated images.

GANs and VAEs raise many questions. What are the properties of the embedding operator Φ? Where are the deformation properties coming from? Why do these algorithms seem to generalize despite the curse of dimensionality? This paper shows that the embedding Φ does not need to be learned and can be specified by a priori properties. Indeed, the embedding should Gaussianize X and be Lipschitz continuous to translations and deformations so that modifications of the input noise result in deformations of X. Section 2.2 reviews these properties. We then define the generative model as an inversion of the embedding on training data, regularized by the deep convolutional network architecture.

Lipschitz continuity to translations and deformations together with Gaussianization are strong constraints that allow to specify suitable embedding operators. Lipschitz continuity to deformations requires separating the signal variations at different scales, which leads to the use of wavelet transforms. We concentrate on wavelet scattering transforms (Mallat, 2012), which linearize translations and provide appropriate Gaussianization, but other multiscale embedding representations may also be used. The generator $\widehat{G}$ is thus estimated by inverting the scattering operator on training data. We implement this inversion with a deep convolutional network having the same architecture as the generator of a DCGAN (Radford et al., 2016). Experiments in Section 4 show that these Generative Scattering Networks have similar properties as GAN generators.

2 Computing a Generator from an Embedding

2.1 Generator Calculation as an Inverse Problem

Unsupervised learning consists on estimating a random model $\widehat{X}$ of a random vector X from n realizations $\{x_i\}_{i\leq n}$ of X. Autoregressive Gaussian processes are simple models $\widehat{X} = \widehat{G}(Z)$, computed from an input Gaussian white noise Z by estimating a parametrized linear operator $\widehat{G}$. This operator is obtained by inverting a linear operator, whose coefficients are calculated from the realizations $\{x_i\}_{i\leq n}$ of X. We shall similarly compute image models $\widehat{X} = \widehat{G}(Z)$ from a Gaussian white noise Z, by estimating a parametrized operator $\widehat{G}$, but which is a deep convolutional network instead of a linear operator. It is calculated by inverting an embedding $\{\Phi(x_i)\}_{i\leq n}$ of the realizations $\{x_i\}_{i\leq n}$ of X, with a predefined operator $\Phi(x)$.

Let us denote by $\mathcal{G}$ the set of all parametrized convolutional network generators defined by a particular architecture. We impose that $\widehat{G}(\Phi(x_i)) \approx x_i$ by minimizing a mean-square error in the convolutional network class $\mathcal{G}$:

$$\widehat{G} = \operatorname*{argmin}_{G\in\mathcal{G}} n^{-1} \sum_{i=1}^{n} \|x_i - G(\Phi(x_i))\|^2 \,. \tag{1}$$

The resulting generator $\widehat{G}$ depends upon the training examples $\{x_i\}_{i\leq n}$ and on the regularization imposed by the network class $\mathcal{G}$. We shall say that the network generalizes over $\Phi(X)$ if $\mathbb{E}(\|X - \widehat{G}(\Phi(X))\|^2)$ is small and comparable to the empirical error $n^{-1}\sum_{i=1}^{n}\|x_i - G(\Phi(x_i))\|^2$. We say that the network generalizes over a Gaussian white noise Z if realizations of $\widehat{X} = \widehat{G}(Z)$ are realizations of X. If $\widehat{G}$ generalizes over $\Phi(X)$ then a sufficient condition to generalize over Z is that Φ transforms X into a Gaussian white noise. Besides this condition, the role of the embedding Φ is to specify the properties of $x = \widehat{G}(z)$ that should result from modifications of z.

We shall define $\Phi = A\overline{\Phi}$, where $\overline{\Phi}$ is a fixed normalized operator and A is an affine operator which performs a whitening and a projection to a lower dimensional space. We impose that $\{\overline{\Phi}(x_i)\}_{i\leq n}$

are realizations of a general Gaussian process and that A transforms this process into a lower dimensional Gaussian white noise. We normalize $\overline{\Phi}$ by imposing that $\overline{\Phi}(0) = 0$ and that it is contractive, this is:

$$\forall (x, x') \in \mathbb{R}^{2D} \quad \|\overline{\Phi}(x) - \overline{\Phi}(x')\| \leq \|x - x'\| \,, \tag{2}$$

where D is the dimension of the space where X takes its values. To guarantee that the embedding of the training samples $\{\Phi(x_i)\}_{i \leq n}$ has a stable inverse, the normalized operator $\overline{\Phi}$ should not contract too much the distances between the $\{x_i\}_{i \leq n}$. This means that $\overline{\Phi}$ is a bi-Lipschitz embedding of these samples, and hence that there exists a constant $\alpha > 0$ so that for all samples x_i and $x_{i'}$ in $\{x_i\}_{i \leq n}$ we have:

$$\frac{1}{\alpha}\|x_i - x_{i'}\| \leq \|\overline{\Phi}(x_i) - \overline{\Phi}(x_{i'})\| \leq \|x_i - x_{i'}\|. \tag{3}$$

The affine operator A performs a whitening of the distribution of the $\{\overline{\Phi}(x_i)\}_{i \leq n}$ by subtracting the empirical mean μ and normalizing the largest eigenvalues of the empirical covariance matrix Σ. For this purpose, we calculate the eigendecomposition of the covariance $\Sigma = QSQ^T$. Let $P_{V_d} = Q_d Q_d^T$ be the orthogonal projection in the space generated by the d eigenvectors with eigenvalues above a small threshold $\epsilon > 0$. We choose ϵ and hence d so that $\overline{\Phi}_d(x) = P_{V_d}\overline{\Phi}(x)$ still satisfies the bi-Lipschitz stability bound (3) up to a minor modification of α. We choose $A = \Sigma_d^{-1/2}(Id - \mu)$ with $\Sigma_d^{1/2} = Q_d S_d^{1/2}$ hence $\Sigma_d^{-1/2} = S_d^{-1/2} Q_d^T$, and Id denoting the identity. We obtain:

$$\Phi(x) = \Sigma_d^{-1/2}(\overline{\Phi}(x) - \mu) \,, \tag{4}$$

which computes a d-dimensional whitening of the $\{\overline{\Phi}(x_i)\}_{i \leq n}$. The generator $\widehat{G}$ which inverts Φ can be factorized by inverting the eq. (4):

$$\widehat{G}(z) = \overline{G}\left(\Sigma_d^{1/2} z + \mu_d\right) , \tag{5}$$

where $\mu_d = P_{V_d}\mu$ and $\overline{G}$ is a deep convolutional network which inverts $\overline{\Phi}_d$ over the training samples:

$$\overline{G} = \operatorname*{argmin}_{G \in \mathcal{G}'} n^{-1} \sum_{i=1}^{n} \|x_i - G(\overline{\Phi}_d(x_i))\|^2 \,. \tag{6}$$

The deep convolutional networks in $\mathcal{G}'$ have one layer less than in $\mathcal{G}$. Indeed, the first layer is computed in (5) with $\Sigma_d^{1/2}(Id + \mu_d)$, it reshapes and adjusts the mean and covariance of the input white noise Z of dimension d. If d is sufficiently large, we can compute $\overline{G}$ independently from the dimensionality reduction by replacing $\overline{\Phi}_d$ by $\overline{\Phi}$ in (6).

If $\overline{\Phi}$ is invertible, the operator $\overline{G}$ should not be confused with the inverse $\overline{\Phi}^{-1}$. The model $\widehat{X} = \widehat{G}(Z)$ computed from $\overline{G}$ has typically a much smaller entropy than $\widetilde{X} = \overline{\Phi}^{-1}(\Sigma_d^{1/2} Z + \mu_d)$ because $\overline{G}$ is regularized by the convolutional network architecture.

Since $\overline{G}$ is convolutional, it transforms a stationary process into a stationary process. Therefore, the convolutional non-linear operator $\overline{G}$ is capturing stationary components, factorized at multiple scales by the deep convolutional network. The non-stationarity part of the model $\widehat{X} = \widehat{G}(Z)$ is captured by the affine transformation $\Sigma_d^{1/2} Z + \mu_d$. We shall see that this is sufficient to model highly non-stationary processes such as centered faces.

Associative Memory: The existence of a known embedding operator Φ allows one to use these generative networks as associative or content addressable memories. The input Z can be interpreted as an address, of lower dimension d than the generated image $\widehat{X}$. Any training image x_i is associated to the address $z_i = \Sigma_d^{-1/2}(\overline{\Phi}x_i - \mu)$. The network is trained to generate the training images $\{x_i\}_{i \leq n}$ from these lower dimensional addresses. The inner network coefficients thus include a form of distributed memory of these training images. If the network generalizes then one can approximately reconstruct a realization x of the random process X from its embedding address $z = \Sigma_d^{-1/2}(\overline{\Phi}x - \mu)$. In that sense, the memory is content addressable.

2.2 GAUSSIANIZATION AND CONTINUITY TO DEFORMATIONS

We now describe the properties of the normalized embedding operator $\overline{\Phi}$ to build a generator having similar properties as GANs or VAEs. We mentioned that we would like $\Phi(X)$ to be nearly a Gaussian white noise and since $\Phi = A\,\overline{\Phi}$ where A is affine then $\overline{\Phi}(X)$ should be Gaussian. Therefore, $\{\overline{\Phi}(x_i)\}_{i \leq n}$ should be realizations of a Gaussian process and hence be concentrated over an ellipsoid. However, this is not the only constraint.

The generator $\overline{G}$ is a deep convolutional network which is covariant to translations, and we compute $\overline{G}$ such that $\overline{G}(\overline{\Phi}(x_i)) \approx x_i$. As previously explained, $\overline{G}$ will not be an exact inverse of $\overline{\Phi}$ because the generator is regularized by its architecture. However, we want to ensure that $\overline{G}$ is as close as possible to a left inverse of $\overline{\Phi}$. A convolutional network $\overline{G}$ maps a multichannel image z' defined over a subsampled spatial grid into a full resolution image x. It is covariant to translations in the sense that if z' is translated then x is also translated. The inverse of such a network should also be covariant to translations. Consequently, $\overline{\Phi}$ is required to be covariant to translations.

Another important property is that images $x = G(z)$ appear to be translated and deformed when z is modified. To enforce this property, we require $\overline{\Phi}$ to be Lipschitz continuous to translations and deformations. A translation and deformation of an image $x(u)$ can be written as $x_\tau(u) = x(u - \tau(u))$, where u denotes the spatial coordinates. Let $|\tau|_\infty = \max_u |\tau(u)|$ be the maximum translation amplitude. Let $\nabla\tau(u)$ be the Jacobian of τ at u and $|\nabla\tau(u)|$ be the norm of this Jacobian matrix. The deformation induced by τ is measured by $|\nabla\tau|_\infty = \max_u |\nabla\tau(u)|$, which specifies the maximum scaling factor induced by the deformation. It defines a metric on diffeomorphisms (Mallat, 2012) and thus specifies the deformation "size". The operator $\overline{\Phi}$ is said to be Lipschitz continuous to translations and deformations over domains of scale 2^J if there exists a constant C such that for all images x we have:

$$\|\overline{\Phi}(x) - \overline{\Phi}(x_\tau)\| \leq C \left(2^{-J}|\tau|_\infty + |\nabla\tau|_\infty\right) \tag{7}$$

This implies that translations of x that are small relative to 2^J and small deformations are mapped by $\overline{\Phi}$ into small quasi-linear variations of $z = \overline{\Phi}(x)$.

We also mentioned that $\{\overline{\Phi}(x_i)\}_{i \leq n}$ should be concentrated on an ellipsoid. This Gaussianization property is always true if $n \leq d$, but can be difficult to achieve if $n \gg d$. In one dimension $x \in \mathbb{R}$, an invertible differentiable operator Φ which Gaussianizes a random variable X can be computed as the solution of a differential equation that transports the histogram into a Gaussian (Friedman, 1987). In higher dimension, this strategy has been extended by iteratively performing a Gaussianization of one-dimensional variables, through independent component analysis (Chen & Gopinath, 2000) or with random rotations (Laparra et al., 2011). However, such strategies might not apply here directly because they do not necessarily lead to operators that are translation covariant and Lipschitz continuous to translations and deformations.

Another Gaussianization strategy comes from the central limit theorem by averaging enough nearly independent random variables having variances of the same order of magnitude. The advantage of this approach is that an averaging can be covariant to translations if implemented with convolutions with a low-pass filter. Moreover, this averaging leads to Lipschitz continuity to translations and deformations. An averaging operator is contractive; therefore, it loses all high-frequency information, and hence confuses images. To define an operator $\overline{\Phi}$ that discriminates large collections of images and is bi-Lipschitz (3), we must preserve high-frequency information despite the averaging. The next section explains how to do so with a Scattering transform.

3 GENERATIVE SCATTERING NETWORKS

3.1 GAUSSIANIZATION BY MULTISCALE SCATTERING

We will show that scattering transforms (Mallat, 2012; Bruna & Mallat, 2013) are good candidates to construct an image embedding for image generation. It is explained in Mallat (2012) that to define an operator that is Lipschitz continuous to deformations and hence satisfy (7), one must separate variations of x at different scales. Indeed, the first order term of the non-rigid part of a deformation

is a scaling; controlling the impact of this scaling term over signal frequencies requires decomposing the signal at different scales, which is done by a wavelet transform. The linearization of translations at a scale 2^J is obtained by an averaging at this scale. Non-linearities appear as a necessity to create interactions across multiple scale coefficients; these interactions reduce the information loss due to averaging, and thus help to avoid modifying too much the Euclidean distances (3). The averaging at the scale 2^J also Gaussianizes the random vector X. Therefore, the scale 2^J adjusts the trade-off between Gaussianization and contraction of distances.

Let us emphasize that the scattering transform is not a unique solution to this problem. Multiscale scattering operators can be implemented with a deep convolutional network architecture, using pre-defined wavelet filters (Mallat, 2016); however, other deep convolutional networks might have the same properties. Also, other multiscale representations such as the ones in Portilla & Simoncelli (2000); Lyu & Simoncelli (2009); Malo & Laparra (2010) may also be used. We concentrate here on scattering operators because their mathematical properties are relatively well understood.

A scattering operator S_j transforms $x(u)$ into a tensor $x_j(u,k)$ where the spatial parameter u is sampled at intervals 2^j and the channels are indexed by k. The number K_j of channels increases with j to partially compensate for the loss of spatial resolution. These K_j channels are computed by a non-linear translation covariant transformation Ψ_j. 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), we choose a Morlet wavelet ψ, scaled by 2^ℓ and rotated along Q angles $\theta = q\pi/Q$:

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

To obtain an order two scattering operator S_j, the operator Ψ_j computes sequences of up to two wavelet convolutions and complex modulus:

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

Therefore, there are $K_j = 1 + Qj + Q^2 j(j-1)/2$ channels. A scattering transform is then obtained by averaging each channel with a Gaussian low-pass filter $\phi_j(u) = 2^{-2j}\phi(2^{-j}u)$, whose spatial width is proportional to 2^j:

$$S_j(x) = \Psi_j(x) \star \phi_j = \Big[x \star \phi_j\,,\, |x \star \psi_{\ell,q}| \star \phi_j\,,\, ||x \star \psi_{\ell,q}| \star \psi_{\ell',q'}| \star \phi_j\Big]_{1\leq \ell<\ell'\leq j, 1\leq q,q'\leq Q}\,. \quad (8)$$

Convolutions by ϕ_j are followed by a subsampling by 2^j; as a result, if x has p pixels then S_j is of dimension $p\,\alpha_j$ where:

$$\alpha_j = 2^{-2j}(1 + Qj + Q^2 j(j-1)/2). \quad (9)$$

The maximum scale 2^j is limited by the image width $2^j \leq p^{1/2}$. For our experiments we used $Q = 8$ and images x of size $p = 64^2$. In this case we have $\alpha_4 = 1.63$ and $\alpha_5 = 0.66$. Moreover, $\alpha_j > 1$ for $j \leq 4$ and so $S_j(x)$ has more coefficients than x. Based on this coefficient counting, we expect S_j to be invertible for $j \leq 4$, but not for $j \geq 5$.

The wavelets separate the variations of x at different scales 2^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 deformations (Mallat, 2012) in the sense of eq. (7). Wavelets are also defined with a Littlewood-Paley condition which guarantees that the wavelet transforms and also S_j are contractive operators (Mallat, 2012).

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

$$S_j = |V_j S_{j-1}|, \quad (10)$$

it is thus an instance of a deep convolutional network whose filters are specified by wavelets and where the non-linearity is chosen to be a modulus.

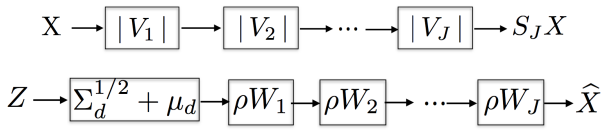


Figure 1: Top: the scattering embedding $S_J X$ is computed by cascading J convolutional wavelet operators V_j followed by a modulus. Bottom: A generative scattering network is a deep convolutional network which first adjusts the mean and covariance of the input white noise Z and then cascades J ReLUs ρ with multichannel convolution matrices W_j.

If wavelet coefficients become nearly independent when they are sufficiently far away then $S_j(X)$ becomes progressively more Gaussian as the scale 2^j increases, because of the spatial averaging by ϕ_j. So, 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. However, as the scale increases there are instabilities created by the contractions (as in eq. (3)) induced by S_j. This trade-off between Gaussianization and stability defines the optimal choice of the scale 2^j.

3.2 Inverse Scattering Generator with Deep Convolutional Networks

A scattering generator network is computed by choosing the normalized embedding operator $\overline{\Phi}$ to be a scattering operator S_J at some predefined scale 2^J. As explained in Section 2.1, we compute in (6) a scattering generator by finding a deep convolutional network $\overline{G}$ which minimizes the MSE over all training samples. The resulting network $\overline{G}$ is a regularized approximate inversion of S_J.

The deep convolutional network architecture that defines the family $\mathcal{G}'$ has depth J and follows the shapes of only the convolutional layers of a DCGAN (Radford et al., 2016) generator:

$$\overline{G} = \rho\, \widetilde{W}_1 \circ \rho\, \widetilde{W}_2 \circ \ldots \circ \rho\, \widetilde{W}_J \,. \tag{11}$$

The non-linearity ρ is a ReLU, and each $\widetilde{W}_j$ consists of multiplying by W_j, which is an upsampling followed by a multichannel convolutional operator along the spatial variables, and then adding per-channel biases. The weights of this network are optimized by minimizing a mean-square error loss:

$$n^{-1} \sum_{i=1}^{n} \|\overline{G}(S_J(x_i)) - x_i\|^2 \tag{12}$$

with stochastic gradient descent. We explained that $S_J(x)$ is computed from $x = S_0(x)$ by iteratively calculating $S_j(x)$ from $S_{j-1}(x)$ with (10), which reduces the spatial resolution while increasing the number of channels. We can thus interpret (11) as a progressive inversion of this decomposition.

4 Numerical experiments

In this section, we test the ability to reconstruct training images from their scattering coefficients and the generalization properties of the obtained generative networks. We shall also verify that modifications of the input noise produce a morphing of the generated images as in GANs.

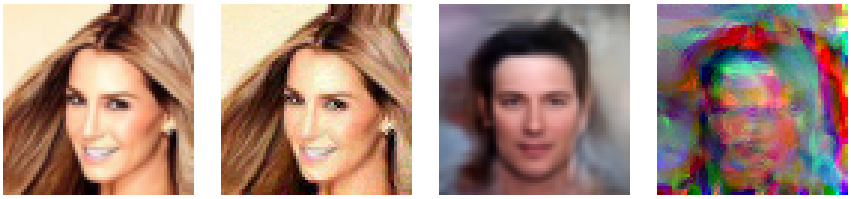
We consider three datasets that have different levels of variabilities: CelebA (Liu et al., 2015), a 64^2 downsampled version of ImageNet (Russakovsky et al., 2015) and Polygon5. The last dataset consists of images of random convex polygons of at most five vertices, with random colors. All datasets consist of RGB color images with shape $64^2 \times 3$. For each dataset, we consider only 65536 training images and 16384 test images.

In all experiments, we chose the scattering averaging scale to be $2^J = 2^4 = 16$, which is 4 times smaller than the image width 64. This operator is stable to deformations over big domains. $S_4(x)$ is computed over the 3 color channels of each image x. Because of the subsampling by 2^{-4}, $S_4(x)$ has a spatial resolution of 4 by 4, with $417 \times 3 = 1251$ channels, and is thus of dimension $\approx 2 \times 10^4$. It has more coefficients than the input image x and numerical experiments indicate that S_4 is invertible.

This dimension is reduced to $d = 100$ by the whitening operator. All scattering transforms are computed with PyScatWave (Oyallon et al., 2017). The operator S_4 is a bi-Lipschitz embedding of the training images of the three datasets in the sense of (3), with a Lipschitz constant $\alpha = 5$ and over 99.5% of the distances between image pairs $(x_i, x_{i'})$ are preserved with a Lipschitz constant smaller than 3.

The architecture of the generator is shown in Fig. 1. The input Gaussian white noise Z is of dimension $d = 100$. All the convolutional layers have filters of size 5, use symmetric padding at the boundaries and are preceded by a bilinear upsampling of their inputs. The number of channels of each layer is $512, 256, 128$ and 3. As in Radford et al. (2016), we use a hyperbolic tangent instead of a ReLU for the final non-linearity. We minimize the loss in eq. (12) using the Adam optimizer (Kingma & Ba, 2014) with default hyperparameters.

Figure 2 shows that we can indeed recover a precise approximation of an image x from $S_4(x)$. This reconstruction was also computed with PyScatWave (Oyallon et al., 2017), which recovers $\hat{x}$ by iteratively minimizing $\|S_4(\hat{x}) - S_4(x)\|$ with a gradient descent, initialized with white noise. The convolutional network operator $\overline{G}$ is very different from the inverse scattering S_4^{-1} despite the fact that they both recover the same training images. Figures 2(c) and 2(d) are respectively realizations of $\widehat{X} = \widehat{G}(Z) = \overline{G}(\Sigma_d^{1/2} Z + \mu_d)$ and of $\widetilde{X} = S_4^{-1}(\Sigma_d^{1/2} Z + \mu_d)$, for the same realization of the Gaussian white noise Z. The convolutional network generator $\overline{G}$ restores a face image, which is barely the case for S_4^{-1}. However, the image in 2(d) shows that the non-stationary component provided by Σ_d and μ_d does provide some global shape information on faces.



(a) (b) (c) (d)

Figure 2: (a) Original image x. (b) Iterative reconstruction $\hat{x}$ of x from $S_4(x)$. (c) $\widehat{X} = \widehat{G}(Z)$ for a realization of Z. (d) $\widetilde{X} = S_4^{-1}(\Sigma_d^{1/2} Z + \mu_d)$ for the same realization.

We now evaluate the training and testing errors of the generator $\widehat{G}$ to evaluate its generalization properties on $\Phi(X) = \Sigma^{-1/2}(S_4(X) + \mu)$. This is done by comparing x with the image $\widehat{G}(z)$ generated from $z = \Phi(x)$, where x may be a training or a test image. Figures 3 and 4 show such reconstructions on training and test images. Table 1 gives the average training and test errors for each dataset. The training error is between 1.5db and 3db above the test error, which is a sign of overfitting. However, this overfitting is not large compared to the variability of errors from one dataset to the next. Overfitting is not good for unsupervised learning where the intent is to model a probability density, but if we consider this network as an associative memory, it is not a bad property. Indeed, it means that the network performs a better recovery of known images used in training than unknown images in the test, which is needed for high precision storage of particular images.

Polygons are simple images that are much better recovered than faces in CelebA, which are much simpler images than the ones in ImageNet. This simplicity is related to the sparsity of their wavelet coefficients, which is higher for polygons than for faces or ImageNet images. Wavelet sparsity drives the properties of scattering coefficients, which are localized $\mathbf{l^1}$ norms of wavelet coefficients. The network regularizes the inversion by storing information on training images, which is a form of memorization. ImageNet images require more memorization because their wavelet coefficients are less sparse than polygons or faces; this might explain the difference of accuracies over datasets, but the link between sparsity and the network memory capacity is not fully understood. The generative network has itself sparse activations with about 70% of them being equal to zero on average over all images of the three datasets. Sparsity thus seems to be an important element of the regularization given by the network.

Similarly to VAEs, scattering image generations can be slightly blurred, even on training samples. This blurriness is probably due to a memorization issue because we observe that the blur is smaller

	CelebA	Polygon5	Imagenet64x64
Train	24.95	41.41	18.87
Test	22.06	39.67	17.21

Table 1: Train and test errors in dB

on images of polygons which are simpler. Also, if the dataset has only 16 images, the generator can fully recover all high frequencies, but learning is very slow and requires many iterations. For 65536 images, fitting the high frequencies does not seem to be possible with Adam even after 2000 epochs. Therefore, this difficulty can be due to the memory capacity of the network which is not sufficient to capture all the training images, but it may also be partly due to the optimizer.

Figure 3: Top: original images in the training set. Bottom: reconstructions from $S_4(x)$ using $\overline{G}$.

Figure 4: Top: original images in the test set. Bottom: reconstructions from $S_4(x)$ using $\overline{G}$.

To evaluate the network generalization on Gaussian white noise Z, Figure 5 shows images $\widehat{X} = \widehat{G}(Z)$ generated from a random sampling of the input Gaussian white noise Z. Generated images have strong similarities with the ones in the training set for polygons and faces. The network recovers colored geometric shapes for the polygon images even tough they are not exactly polygons and it recovers faces for CelebA with a blurred background. For ImageNet, it recovers blurred images, showing that in this case the model is not good.

Figure 6 shows images reconstructed from the generator $\overline{G}$ by interpolating the latent variables $z_i = S_4(x_i)$ and $z_{i'} = S_4(x_{i'})$ of two images x_i and $x_{i'}$: $z = \alpha z_i + (1-\alpha) z_{i'}$ with $\alpha \in [0,1]$. This linear interpolation results in a continuous deformation from one image to the other while colors and image intensities are also adjusted. This behavior is similar to the one obtained by GANs, in our case it is a consequence of the Lipschitz continuity to deformations of the scattering transform.

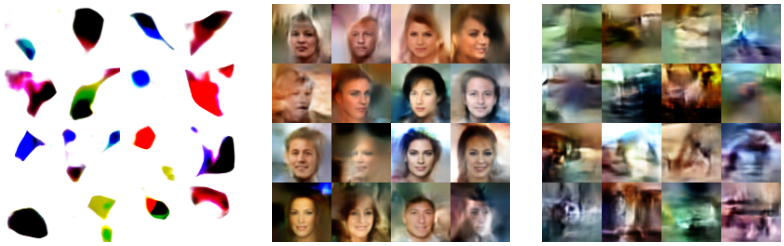


Figure 5: Sampling from Gaussian white noise Z.

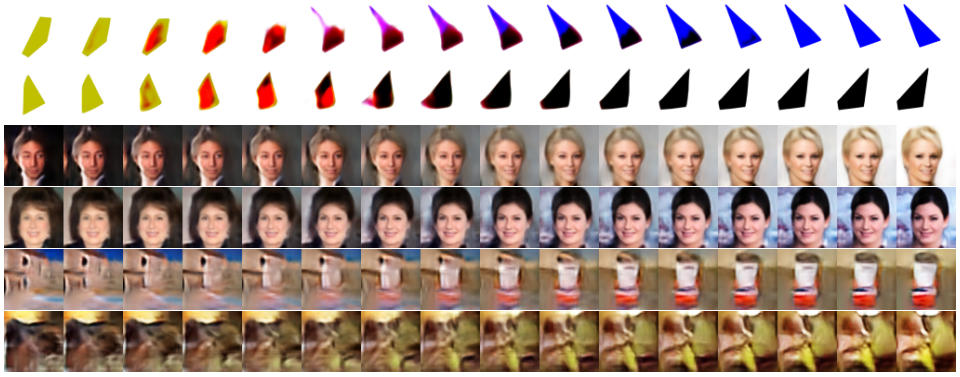


Figure 6: Interpolations through deformations given by $\overline{G}$ for training and test samples. For each dataset, the first row corresponds to training samples and the second to test ones.

5 Conclusion

This paper shows that most properties of GANs and VAEs can be reproduced with an embedding computed with a scattering transform, which avoids using a discriminator as in GANs or learning the embedding as in VAEs. It also provides a mathematical framework to analyze the statistical properties of these generators through the resolution of an inverse problem, regularized by the convolutional network architecture and the sparsity of the obtained activations. Because the embedding function is known, numerical results can be evaluated on training as well as testing samples.

We report preliminary numerical results with no hyperparameter optimization. Test on noise vector arithmetic and conditional generations have not yet been carried. Potential numerical improvements have not been explored either. In particular, the architecture of the convolutional generator can be adapted to the properties of the forward scattering operator S_j as j increases. Also, the paper uses a "plain" scattering transform which does not take into account interactions between angle and scale variables, which may also improve the representation as explained in Oyallon & Mallat (2015).

References

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.

P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam. Optimizing the latent space of generative networks. *Proc. of ICLR*, 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.

A. Creswell and A. Bharath. 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, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

D.P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proc. of ICLR*, 2014.

V. Laparra, G. Camps-Valls, and J. Malo. Iterative gaussianization: From ica to random rotations. *IEEE Transactions on Neural Networks*, 22(4):537–549, April 2011. ISSN 1045-9227. doi: 10.1109/TNN.2011.2106511.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Siwei Lyu and Eero P. Simoncelli. Reducing statistical dependencies in natural signals using radial gaussianization. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 21*, pp. 1009–1016. Curran Associates, Inc., 2009.

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.

Jess Malo and Valero Laparra. Psychophysically tuned divisive normalization approximately factorizes the pdf of natural images. *Neural Computation*, 22(12):3179–3206, 2010. doi: 10.1162/NECO_a_00046. PMID: 20858127.

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

E. Oyallon, E. Belilovsky, and S. Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proc. of ICCV*, 2017.

Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision*, 40(1):49–70, October 2000. ISSN 0920-5691. doi: 10.1023/A:1026553619983. URL `https://doi.org/10.1023/A:1026553619983`.

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

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.