MUSIC GENERATION AND TRANSFORMATION WITH MOMENT MATCHING-SCATTERING INVERSE NETWORKS

Mathieu Andreux and Stéphane Mallat
Département d'informatique de l'ENS, École normale supérieure,
CNRS, PSL Research University, 75005 Paris, France

ABSTRACT

We introduce a Moment Matching-Scattering Inverse Network (MM-SIN) to generate and transform musical sounds. The MM-SIN generator is similar to a variational autoencoder or an adversarial network. However, the encoder or the discriminator are not learned, but computed with a scattering transform defined from prior information on sparse time-frequency audio properties. The generator is trained by jointly minimizing the reconstruction loss of an inverse problem, and a generation loss which computes a distance over scattering moments. It has a similar causal architecture as a WaveNet and provides a simpler mathematical model related to time-frequency decompositions. Numerical experiments demonstrate that this MM-SIN generates new realistic musical signals. It can transform low-level musical attributes such as pitch with a linear transformation in the embedding space of scattering coefficients.

1. INTRODUCTION

This paper investigates musical sound generation and transformation with a simplified algorithmic architecture, which relates generative networks to time-frequency representations. Image generation has led the way through the development of Generative Adversarial Networks (GANs) [7] and Variational Autoencoders [13] where images are generated from a Gaussian white noise vector, which defines a latent space. Arithmetic operations in this latent space lead to controlled transformations over the images such as aging of faces or transforming women in men. The problem is however different for audio signals which must take into account time causality properties. Some authors have applied image generation algorithms over spectrograms [3, 10] but it then requires to invert the spectrograms with a vocoder or a Griffin-Lim algorithm which is long and has a reduced quality.

Deep autoregressive neural networks such as WaveNet [15, 16] or SampleRNN [14] have achieved exceptional synthesis of music and speech signals. They do not take as input a Gaussian white noise but estimate a probability distribution with a Markov chain factorization, which computes conditional probabilities given from past values. The generated signals have an outstanding quality but these neural architectures are complex and lack interpretability. Several effective techniques have been introduced to modify audio generations [4–6, 9] by modifying the latent code to change the probability distribution or by using reference signals as targets, which is more complex than arithmetic operations used for images.

This paper introduces a simplified neural network architecture which synthesizes and modifies music signals from Gaussian white noise, with two key contributions. As opposed to image GANs or variational autoencoders, we do not learn a discriminator or an encoder: both are provided by prior information on audio signals, which is captured by their time-frequency regularity. This is done by adapting a result obtained in [2] for images, and introducing a moment matching technique. The second contribution is the introduction of a causal computational architecture allowing to progressively synthesize audio signals, and which can be parallelized in GPU's. The resulting Scattering Autoencoder architecture has some similarities with a Parallel WaveNet [16]. Similarly to warping properties over images, we show that arithmetic transformations in the latent space produce time-frequency deformations, which can modify the pitch of musical notes or interpolate music. Despite a lower synthesis quality than state-of-the-art generating methods, these preliminary results pave the way for a new approach to synthesize audio signals, without learning encoders or discriminators.

2. SCATTERING AUTOENCODER

This section introduces the principles of a scattering autoencoder and its computational architecture. The encoder is not learned but computed based on prior information on audio time-frequency properties. Audio and musical signals have sparse representations over time-frequency dictionaries such as audio wavelets [20]. Their perceptual properties are not much affected by small time-frequency warpings. We define a signal embedding which takes advantage of these characteristics.

The architecture of a scattering autoencoder is illustrated in Figure 1. The random input audio signal $X[t]$ is first transformed into a nearly Gaussian random vec-

Attribution: Mathieu Andreux and Stéphane Mallat. "Music Generation and Transformation with Moment Matching-Scattering Inverse Networks", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

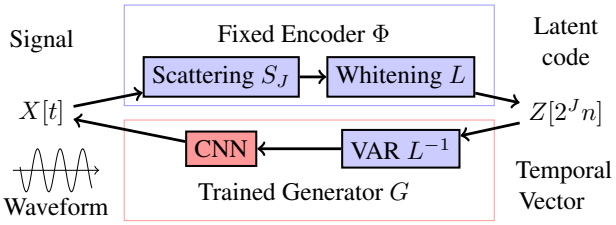

Figure 1. The audio waveforms is encoded with a time-frequency scattering S_J followed by a linear whitening operator L. The Scattering Inverse Network (SIN) G restores a signal from a Gaussian white noise by inverting $L\,S_J$ on training data.

tor $S_J(X)[2^J n]$ by applying the time-frequency scattering transform S_J, where $J \geq 0$ is a hyperparameter. Gaussianization is achieved through a time averaging over a sufficiently long time interval thanks to the central limit theorem. In order to preserve enough information on the original signal X after averaging, multiple sparse time-frequency channels are built by applying iterative wavelet transforms [1]. The Gaussian scattering $S_J(X)$ is then mapped into a Gaussian white noise with a whitening operator L, which outputs the linear prediction errors of a future scattering vector from its neighboring past. Section 4 details the whitened scattering transformation LS_J.

The generator G inverts the linear whitening operator and the joint scattering transform by synthesizing an approximation of $X[t]$ from $Z[2^J n]$. It first applies a vector autoregressive (VAR) filter L^{-1}, which can be deduced from L. This operation is followed by a causal convolutional neural network (CNN), with the convolutions acting along time. The network has J layers to invert a scattering at scale 2^J. Section 3 describes the architecture which has similarities with a Parallel WaveNet [16]. The optimization of the generator G amounts to inverting the scattering transform S_J in an adapted metric. The statistics of synthesized signals are constrained by ensuring that they have the same moments in the scattering space as the input signal $X[t]$. This network is thus called a Moment Matching-Scattering Inverse Network (MM-SIN).

3. MOMENT MATCHING-SCATTERING INVERSE GENERATOR

A Moment Matching-Scattering Inverse Network G is computed by inverting a scattering embedding computed at a scale 2^J. It is a causal network which takes as input a nearly Gaussian white noise vector Z computed with a scattering transform, to recover an estimation of the signal X. This network is trained with a loss which incorporates the inverse problem loss computed with a scattering metric, regularized by a moment matching term that can be interpreted as a discriminative metric.

The MM-SIN generator is a linear recurrent neural network L^{-1} followed by a causal convolutional network implemented with a cascade of J convolutions and pointwise non-linearities, illustrated in Figure 2. Each intermediate

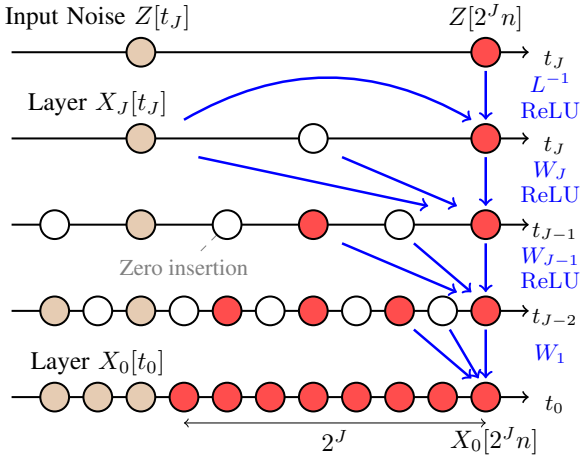

Figure 2. An MM-SIN is a linear recurrent network followed by a causal deep convolutional network with J layers. It takes as input a vector of Gaussian white noise $Z[2^J n]$ (top right, red), and computes the corresponding scattering vector $X_J[2^J n]$ by applying L^{-1}. Intermediate layers $X_j[t_j]$ are then computed with causal convolutions denoted by blue arrows and zero insertions (white points). The single vector $Z[2^J n]$ outputs 2^J values for $X_0[t_0]$, marked with red points.

network layer is composed of vectors $X_j[t_j]$ having k_j channels and sampled at intervals 2^j, for $0 \leq j \leq J$. All layers in Figure 2 appear to be aligned but to understand the causality structure one must realize that each layer is indexed by a time index t_j which is shifted by 2^j relatively to the absolute time variable t of the original input signal $X[t]$: $t_j = t - 2^j$. Using the absolute time t, we thus use the noise vector at a time $t = 2^J(n+1)$ to generate 2^J new output signal values at times $2^J n - 2^J + 1 < t \leq 2^J n + 1$.

The first layer maps $Z[t_J]$ to $X_J[t_J]$ with a vector autoregressive filter L^{-1} which inverts the whitening operator L, followed by a ReLU non-linearity $\rho(u) = \max(u, 0)$

$$X_J = \rho\left(L^{-1} Z\right). \tag{1}$$

A new noise vector $Z[2^J n]$ outputs a new vector $X_J[2^J n]$. At depth j, this initial vector gives rise to 2^{J-j} temporal vectors $X_j[t_j]$ for $2^J n - 2^{J-j} < t_j \leq 2^J n$.

At layer $j > 0$, the layer X_j is mapped to X_{j-1} with an *à trous* convolution followed by a ReLU non-linearity plus bias. We first double the size of X_j with a zero insertion:

$$\tilde{X}_j[n2^j] = X_j[n2^j] \text{ and } \tilde{X}_j[n2^j + 2^{j-1}] = 0. \tag{2}$$

Each X_{j-1} is then calculated with a causal convolution along time and a linear operator along channels with a bias, W_j, followed by a ReLU:

$$X_{j-1} \quad = \quad \rho(W_j \tilde{X}_j). \tag{3}$$

Except for the autoregressive layer, the ReLU is preceded by a batch normalization [11]. The last convolution is not followed by a ReLU so that we can output a

signal with negative values. The final output is $X_0[t_0]$ for $2^J n - 2^J < t_0 \leq 2^J n$ which corresponds to $X[t]$ for $2^J n - 2^J + 1 < t \leq 2^J n + 1$. As in standard generative networks [18], the number of the channels k_j decreases with j according to a geometric law fitted such that X_0 has only one channel and X_J has k_J channels, which is the dimensionality of each vector $Z[2^J n]$.

The parameters of the network G are optimized by inverting the scattering transform followed by the causal whitening operator L. The loss $\mathcal{L}$ minimized by G is a sum of two terms, weighted by a hyperparameter $\lambda > 0$. The first term $\mathcal{L}_{\text{inv}}$ measures the accuracy of the inversion. The second discriminative term $\mathcal{L}_{\text{MM}}$ measures the distance between the scattering moments of the synthesized signals and the scattering moments of the original signals.

$$\min_G \mathcal{L} = \mathcal{L}_{\text{inv}} + \lambda \mathcal{L}_{\text{MM}}. \tag{4}$$

The inverse problem loss $\mathcal{L}_{\text{inv}}$ computes the reconstruction error on each training example x_i from its embedding z_i computed with the scattering transform S_J followed by the linear whitening operator L. The reconstruction error is calculated over scattering coefficients computed at a scale $2^K < 2^J$:

$$\mathcal{L}_{\text{inv}} = \frac{1}{N} \sum_{i=1}^{N} \|S_K(x_i) - S_K G(z_i)\|_1 \text{ with } z_i = L S_J(x_i). \tag{5}$$

The l^1 norm promotes the sparsity of the responses, while the scattering S_K allows to generate signals which may be locally deformed, but are perceptually similar to the original ones. In order to avoid useless computations, we do not apply $L^{-1} z_i$ but directly input the vectors $S_J(x_i)$ to the convolutional part of G for the computation of $\mathcal{L}_{\text{inv}}$.

The loss $\mathcal{L}_{\text{inv}}$ does not control the quality of the generated samples $G(z)$ when z is sampled as a Gaussian white noise. Similarly to GANs which have a discriminator, the quality is controlled by introducing another loss term $\mathcal{L}_{\text{MM}}$, which controls the distance between the generated distribution and the distribution of the original signals.

The moment matching term $\mathcal{L}_{\text{MM}}$ computes the distance between scattering coefficients of generated signals averaged over time t and batch index i, $\overline{S_K G(z_i)[t]}$, and scattering coefficients of the training signals averaged over time t and training examples i, $\overline{S_K x_i[t]}$:

$$\mathcal{L}_{\text{MM}} = \left\| \overline{S_K x_i[t]} - \overline{S_K G(z_i)[t]} \right\|^2 \tag{6}$$

The codes $\{z_i\}$ correspond to a batch of random vectors which is renewed at each iteration of the gradient descent algorithm.

The loss $\mathcal{L}_{\text{MM}}$ is similar to the Maximum Mean Discrepancy regularization introduced in [19]. The moment matching term can be interpreted as a distance with a scattering transform kernel [8]. However, in this case it can directly be implemented as a difference of moments.

4. WHITENED TIME-FREQUENCY SCATTERING

This section details the time-frequency scattering transform $S_J(X)$ originally introduced in [1] and its whitening L, which results in the embedding $Z[2^J n]$. Figure 3 sketches the different computational steps. This transform relies on priors on musical signals in order to build a time-dependent vector representation $S_J X[2^J n]$ which is approximately Gaussian and linearizes small time-frequency deformations.

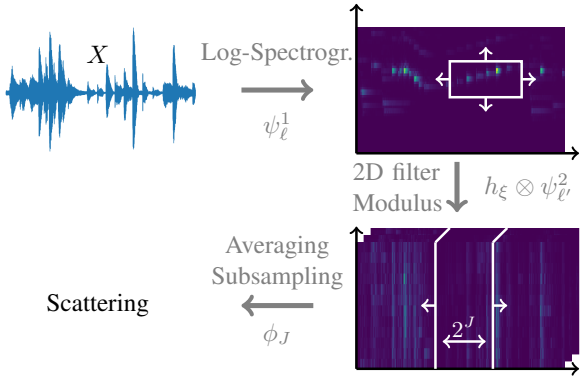

Figure 3. Time-Frequency Scattering transform. The log-spectrogram is obtained with a first wavelet transform ψ^1_ℓ followed by a modulus. A joint time-frequency filtering of this log-spectrogram with the filters $h_\xi \otimes \psi^2_{\ell'}$ regularizes the time-frequency deformations of the signal. The low-pass convolution with ϕ_J Gaussianizes the resulting tensor.

Musical signals admit a sparse decomposition in time-frequency representations with a spectrogram. Here, we first compute a spectrogram with frequencies sampled on a logarithmic scale thanks to a wavelet filterbank $\{\psi^1_\ell\}_{0 \leq \ell < J}$ followed by a modulus non-linearity. The wavelets ψ^1_ℓ are defined by dilations of a single mother wavelet:

$$\psi^1_\ell[t] = 2^{-\ell/Q} \psi^1[2^{-\ell/Q} t] \tag{7}$$

We use causal analytic Gammatone wavelets [20], which are good perceptual models of auditory filters, with $Q = 12$ wavelets per octave in order to separate high-frequency partials.

On this sparse spectrogram, small time-frequency deformations of the input signal X produce small local translations in the time-frequency plane. These deformations result in smooth perceptual variations. As a consequence, the embedding should be regular with respect to these deformations This is obtained with a joint 2D filtering of the spectrogram in the time and log-frequency axis. One can prove that the resulting representation is Lipschitz-continuous to these deformations, while preserving invertibility thanks to the use of filters spanning all the energy of the signal [1]. This will imply a form a linearization of these deformations, paving the way for meaningful arithmetic in the latent space.

The time-frequency filters are built as a separable product $h_\xi \otimes \psi^2_{\ell'}$ of frequential filters h_ξ and temporal filters $\psi^2_{\ell'}$. The frequential filters h_ξ are localized Fourier atoms with a Hann window whose size P matches one octave, $P = Q = 12$. The convolution is computed in half-overlaps over the frequency axis. The temporal filters $\psi^2_{\ell'}$

are also Gammatone wavelets, with only $Q_2 = 1$ wavelet per octave. After performing the convolution, a modulus non-linearity is also applied in order to remove the local phase, thereby regularizing the representation.

The time-frequency filtering of the log-spectrogram results in a large tensor with one temporal axis. Its coefficients are sparse along the channel axis and typically decorrelate as they get farther apart in time. To obtain a variable which is more Gaussian, we use the central limit theorem which says that the averaging of a large number of independant variables converges towards a Gaussian random variable. We perform this averaging with a window size 2^J which should be larger than the typical decorrelation length in order to average over enough independent events. This averaging is carried out with a low-pass filter ϕ_J along the temporal axis. It is followed by a subsampling by 2^J in order to remove redundant information.

The time-frequency scattering transform $S_J X$ is defined as:

$$S_J X[2^J n] = \Big[x \star \phi_J[2^J n], \big||x \star \psi^1_\ell| \star h_\xi\big| \star \phi_J[2^J n], \big||x \star \psi_\ell| \star (h_\xi \otimes \psi^2_{\ell'})\big| \star \phi_J[2^J n]\Big], \quad (8)$$

for all ℓ, ξ, ℓ', where convolutions with h_ξ should be interpreted along the frequential ℓ axis and other convolutions along time. $S_J X$ is subsampled in time by a factor 2^J. Each vector $S_J X[2^J n]$ has dimension $k_J = 1 + Q(2J - 3) + Q(J - 2)^2$. For $J = 10$ and $Q = 12$, this amounts to 973 channels. The scale 2^J is chosen as a trade-off between the Gaussianization condition which improves as J increases, and the stability of the scattering invertibility which improves when J decreases.

Thanks to the local averaging, the vectors $S_J X$ tend to a Gaussian distribution. However, this distribution might not be white, *i.e.* it has temporal and channel-wise correlations. The whitening operator L is a causal vector autoregressive linear filter which removes this correlation structure. It is trained by minimizing a prediction error of $S_J X[2^J n]$ given previous vectors $S_J X[2^J(n - m)]$ for $1 \leq m \leq M$. The whitening operator L outputs the innovations of the fitted vector autoregressive process $\{Z[2^J n]\}_n$, which have an approximately Gaussian white distribution.

5. NUMERICAL EXPERIMENTS

We show that the MM-SIN is able to reconstruct waveforms from their embeddings and generate new decent waveforms from noise. Furthermore, we show that it is possible to manipulate low-level attributes of sounds such as pitch with a simple arithmetic in the embedding. In addition, a simple arithmetic in the embedding allows to merge the contents of different inputs, while preserving the musical structure of the resulting signal.

5.1 Methods

We describe the numerical details which lay out experiments. The source code supporting experiments is freely available at `http://github.com/AndreuxMath/ismir2018`, where the reader may also find the audio recordings corresponding to the figures.

The time-frequency scattering transform S_J is computed with an averaging window of size $2^J = 2^{10} = 1024$. It is implemented on GPU with a code inspired from [17]. In the case of the loss (4), we employ a first-order scattering S_K, which means that it is an averaged scalogram, with an averaging window of size $2^K = 2^5 = 32$. The filterbanks are normalized so as to have responses of average equal magnitude in each band over the training dataset.

The architecture of the SIN G is defined as follows. The whitening operator L has a past size $M = 4$. All subsequent convolutions have a kernel size equal to 7. These values were not tuned: results could likely be improved with a careful hyperparameter search. Each network is trained by Adam [12] with a learning rate of 5×10^{-4} for 1200 epochs and batches of size 128.

We use two different musical datasets: NSynth [5] and Beethoven [14]. NSynth is a dataset consisting of annotated musical notes from multiple instruments, thereby allowing to perform carefully controlled transformation experiments. All recordings begin with the onset of the note and last 4s. We restrict ourselves to two types of acoustic instruments, keyboards and flutes, totalling 40 different instruments with MIDI pitches ranging $20 - 110$, leading to a varied and well-balanced dataset. In the original dataset, instruments of the training and testing sets do not overlap. In this paper, we use an alternative split based on the velocity's attributes of the training samples: for each instrument, a random velocity is picked to define the test set. We only use the first 2s of the recordings, as they concentrate most of the energy of the signals.

The Beethoven dataset is closer to an actual musical composition than NSynth, insofar as it consists in 8s extracts of Beethoven's piano sonata. Therefore, it is a good testbed for music generation experiments. We use the train-test split provided by the authors.

For both datasets, the amplitudes of all recordings are normalized in $[-1, 1]$. The sampling rate is reduced from 16000Hz to 4096Hz so as to reduce the computational complexity. It is very likely that the quality of the synthesis could be improved by increasing this sampling rate.

5.2 Waveform generation

We first show that the MM-SIN generator is able to reconstruct and to generate realistic musical samples.

In Figure 4, we display two reconstructions of waveforms from their embeddings, along with the corresponding log-spectrograms, for each of the studied datasets. This shows that the network is able to generalize to a test set, and to adapt to the specifics of a given dataset. The reconstruction is not perfect. The network can introduce small time-frequency deformations because of the scattering encoder and the use of a scattering loss. As witnessed in the log-spectrograms, the time-frequency content of the signals is correctly retrieved, and perceptually the two signals sound similar, up to minor artifacts.

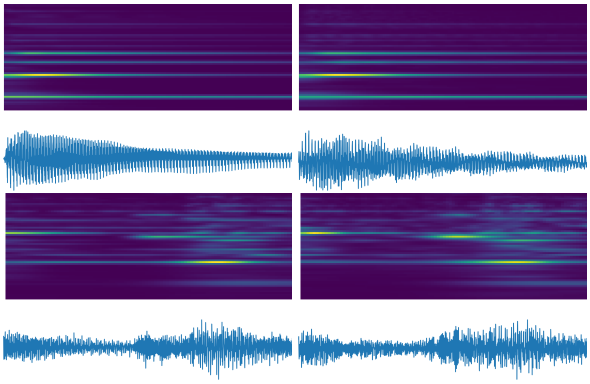

Figure 4. Reconstruction with the MM-SIN G. Left: Original test sample X. Right: Reconstruction $G(Z)$ for $Z = L\,S_J X$. Top: NSynth dataset. Bottom: Beethoven dataset. A different network was trained for each dataset.

Table 1 provides quantitative reconstruction results on the Beethoven dataset. Training a network with the mean-square error (MSE) metric $\|x_i - G(z_i)\|_2^2$ instead of the proposed perceptual metric within the inverse problem loss (5) negatively impacts results, both on the training and testing sets. Further, the moment-matching term has a positive effect on the reconstruction: even though it degrades reconstruction on the training set, it improves the generalization on the test set both in absolute and relative terms.

We now investigate the ability of the network to generate new waveforms from Gaussian white noise. Figure 5 displays several samples generated from white noise through a network trained on the Beethoven dataset. Despite the input being a pure white noise, the network is able to generate samples which alternate silences and more active phases. Further, the fundamental frequency which is played varies through the samples.

In order to measure the variability of the generated samples, we measure the spread σ of the distribution of the time-averaged scattering coefficients $S_K X$ of the samples. This spread corresponds to the average Euclidean distance between the time-averaged scattering of the waveforms and the average scattering coefficients of this distribution. In the case of the training distribution, we obtained

Loss	$\mathcal{L}_{\text{inv}}$	$\mathcal{L}_{\text{inv}}$	$\mathcal{L}_{\text{inv}} + \lambda\mathcal{L}_{\text{MM}}$
Metric	MSE	S_K	S_K
Train error	0.56	**0.16**	0.23
Test error	0.77	0.37	**0.31**
Gap test/train (dB)	1.36	3.53	**1.21**

Table 1. Reconstruction errors on the Beethoven dataset, expressed in terms of the perceptual loss (5). MSE denotes the mean-square error metric $\|x_i - G(z_i)\|_2^2$. Using the perceptual metric for training instead of the MSE metric reduces the error. Further, adding the moment-matching term during training improves the reconstruction results and the generalization.

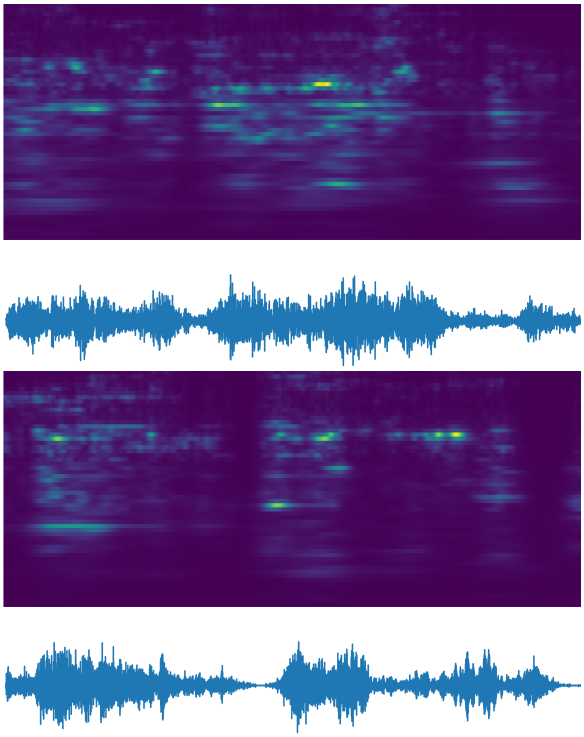

Figure 5. Musical signal $G(Z)$ generated from a white noise Z, where G is learned on the Beethoven dataset. Each line corresponds to an independent sample obtained from a different white noise realization. The resulting signals last about $4s$.

$\sigma = 6.69$, whereas $\sigma = 3.51$ for the distribution generated from white noise. This shows that the generated samples exhibit a non-negligible variability, even though it is lower than the one expressed in the training set.

The effect of the moment matching loss (6) on the generated samples is difficult to assess qualitatively, so we resort to quantitative measures. In the case of a network trained without this loss, the moment matching distance between generated samples and the training set was equal to 38.7, whereas the same distance was equal to 0.176 when also optimizing this loss. As a comparison, the testing set has a distance of 0.334 with respect to the training set. Thus, using this loss term brings generated samples much closer to the natural signals' statistics.

5.3 Pitch modification

We now study the ability of the algorithm to transform the pitch of musical signals with an arithmetic operations in the latent space. We use the NSynth dataset, whose careful construction allows to perform modifications with fixed factors of variability. In the test set, we pick two samples belonging to the same instrument, but with a pitch separated by 5 MIDI scales. We compute their embeddings Z_1 and Z_2, their mean embedding $(Z_1 + Z_2)/2$, and reconstruct the corresponding signals with the generator: $G(Z_1)$, $G(Z_2)$ and $G((Z_1 + Z_2)/2)$.

The results are displayed in Figure 6. The interpolation

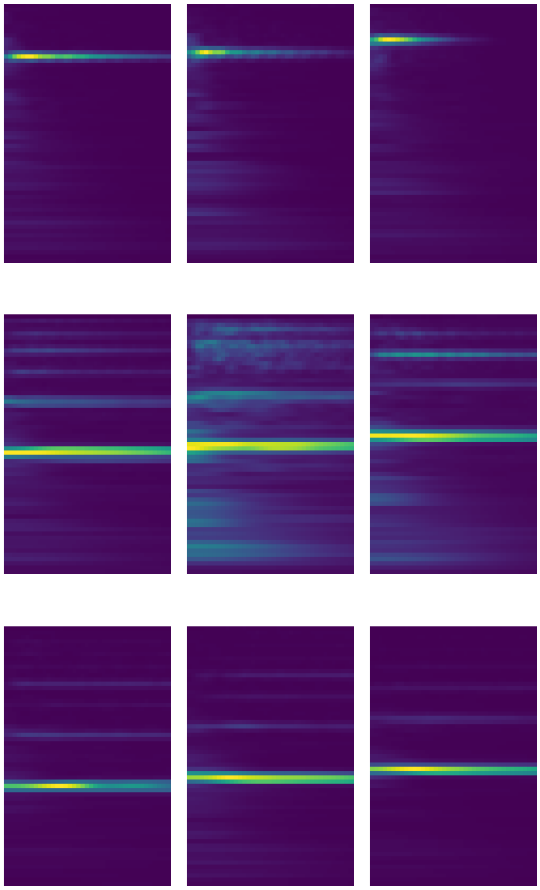

Figure 6. Pitch interpolation. Left column: $G(Z_1)$. Middle column: $G((Z_1 + Z_2)/2)$. Right column: $G(Z_2)$. Z_1 and Z_2 are the embeddings of samples from the test set. The generator interpolates the fundamental frequency with a simple arithmetic. The frequential displacement from left to right corresponds to 5 MIDI scales.

in the latent space does not result in a linear interpolation which would double the number of harmonics. It yields one fundamental frequency in each case. Furthermore, this fundamental frequency is indeed interpolated by this simple arithmetic. Observe that this is also the case of the partials, as can be seen in particular in the bottom example. However, this interpolation suffers from some artifacts. For instance, in the middle example, the partials at highest frequencies are cluttered and the resulting signal misses harmonicity. Yet, these results showcase the ability to transform signals via linear interpolations in the latent space with a simple unsupervised learning procedure and a predefined embedding.

The algorithm owes the ability to perform such pitch interpolations to the time-frequency scattering transform S_J used as an encoder, which regularizes small time-frequency deformations. As such, the pitch interval on which the interpolations can be performed is bounded by the size P of the Hann window used to filter the scalogram along the frequency axis.

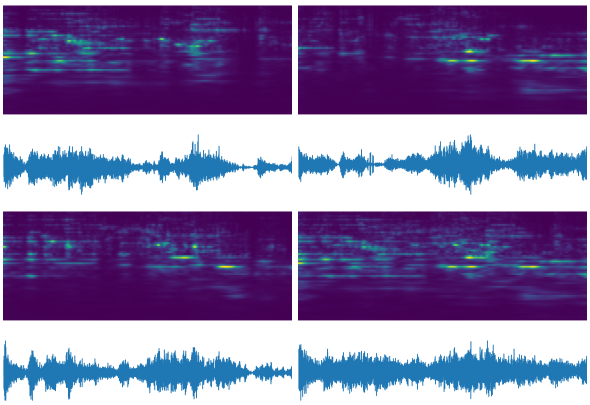

Figure 7. Interpolations in the latent and signal space. Top two signals: $G(Z_1)$ and $G(Z_2)$, where Z_1, Z_2 are Gaussian white noise realizations and G is trained on the Beethoven dataset. Bottom left: Latent interpolation $G((Z_1+Z_2)/2)$. Bottom right: $(G(Z_1) + G(Z_2))/2$. The latent interpolation is able to merge both signals while preserving the musical structure.

5.4 Waveform interpolation

Let us show results when interpolating waveforms from the Beethoven dataset, which have a high density of musical events. We take two random white noise realizations Z_1 and Z_2, and compare the effect on the waveforms of an interpolation in the latent space and in the signal space, with a network G trained on the Beethoven dataset.

The results are represented in Figure 7. The top two signals are the original signals $G(Z_1)$ and $G(Z_2)$, while the bottom left is the latent interpolation $G((Z_1 + Z_2)/2)$ and the bottom right the linear interpolation $(G(Z_1) + G(Z_2))/2$. The latent interpolation incorporates patterns from both signals but it respects the musical structure. It has successive musical notes with their harmonics, and it recovers a sound with silences. On the opposite, the linear interpolation merges both signals, which eliminates the silence regions while producing a cluttered log-spectrogram.

6. CONCLUSION

This paper introduces a causal musical synthesis network optimized through an inverse problem and which thus involves no learned encoder or discriminator. The encoder is defined from time-frequency signal priors in order to Gaussianize the input signal. The generator network maps back the resulting codes to raw waveforms. This network inverts the encoder and generates new signals whose scattering moments match those of the original signals. The resulting system synthesizes new realistic musical signals and performs the transformation of low-level attributes, such as pitch, by simple linear combinations in the latent space.

Synthesized signals do not reach the quality of state-of-the-art generating architectures but these first results show that this approach is a new promising avenue to synthesize audio signals directly from Gaussian white noise, without learning encoders or discriminators.

7. ACKNOWLEDGEMENTS

This work is supported by the ERC InvariantClass grant 320959 and an AMX grant from the MNESR.

8. REFERENCES

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

[2] Tomàs Angles and Stéphane Mallat. Generative networks as inverse problems with scattering transforms. In *International Conference on Learning Representations*, 2018.

[3] M. Blaauw and J. Bonada. Modeling and transforming speech using variational autoencoders. In *Interspeech*, pages 1770–1774, 2016.

[4] J. Chorowski, R.J. Weiss, R. A. Saurous, and S. Bengio. On using backpropagation for speech texture generation and voice conversion. In *International Conference on Audio and Speech Processing (ICASSP)*, 2018.

[5] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1068–1077, 2017.

[6] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[7] 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*, pages 2672–2680, 2014.

[8] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[9] E. Grinstein, N. Duong, A. Ozerov, and P. Pérez. Audio style transfer. *HAL preprint hal-01626389*, 2017.

[10] W.-N. Hsu, Y. Zhang, and J. Glass. Learning latent representations for speech generation and transformation. In *Interspeech*, pages 1273–1277, 2017.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[14] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations*, 2017.

[15] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[16] A. Van Den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, et al. Parallel WaveNet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*, 2017.

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

[18] A. Radford, L. Metz, and R. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[19] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

[20] A. Venkitaraman, A. Adiga, and C. S. Seelamantula. Auditory-motivated gammatone wavelet transform. *Signal Processing*, 94:608–619, 2014.