Adversarial Generation of Time-Frequency Features with application in audio synthesis

Andrés Marafioti[1], Nicki Holighaus[1], Nathanaël Perraudin[2], and Piotr Majdak[1]

[1]Acoustics Research Institute, Austrian Academy of Sciences, Wohllebengasse 12–14, 1040 Vienna, Austria.
[2]Swiss Data Science Center, ETH Zürich, Universitätstrasse 25, 8006 Zürich

***Abstract*—Time-frequency (TF) representations provide powerful and intuitive features for the analysis of time series such as audio. But still, generative modeling of audio in the TF domain is a subtle matter. Consequently, neural audio synthesis widely relies on directly modeling the waveform and previous attempts at unconditionally synthesizing audio from neurally generated TF features still struggle to produce audio at satisfying quality. In this contribution, focusing on the short-time Fourier transform, we discuss the challenges that arise in audio synthesis based on generated TF features and how to overcome them. We demonstrate the potential of deliberate generative TF modeling by training a generative adversarial network (GAN) on short-time Fourier features. We show that our TF-based network was able to outperform the state-of-the-art GAN generating waveform, despite the similar architecture in the two networks.**

I. Introduction

The recent advent of machine learning and particularly in deep learning have allowed for impressive progress in the field of generative modeling. Recurrent neural networks, variational autoencoders [1] and in particular generative adversarial networks (GANs) [2] are able to generate realistic-looking images [3], [4]. Despite this success, unsupervised generation of natural sound remains a challenge, as the waveform representation has strong dependencies at very different scales. Although it was shown [5] that classic recurrent neural networks can be successfully applied, most state-of-the-art contributions rely on dilated convolutions, made popular by WaveNet [6]. Long-term dependencies can be achieved by conditional generation [7], [8], [9]. Nevertheless, sampling from these architectures is computationally expensive to the point that a second neural network has been used [10], [11] to accelerate the generation at the cost of an overall more complex system. In contrast, adversarial generation with GANs enables the direct generation of the whole signal at once, see, e.g., [12], [13], which have demonstrated the potential of applying GANs to audio. Generative Adversarial Networks rely on two competing neural networks that compete and are trained simultaneously: The generator, which produces new data from samples of a random variable and the discriminator, which attempts to distinguish real (training) and fake (generated) data. During training, it is the generator's objective to fool the discriminator, while the discriminator attempts to classify real and fake data accurately. Since their introduction, GANs have been adopted quickly and a number of variations improve on the original concept, e.g., [14] and [15], which we follow here.

Successful previous work in neural audio generation relies almost exclusively on the waveform representation for training and generation. There are numerous reasons to believe, however, that time-frequency (TF) coefficients provide a representation that can be more easily learned. For example, discriminative networks for audio have been observed to learn TF-like features and discriminative models trained on the waveform directly struggle to outperform networks trained on TF features [16], [17], even in the presence of more training data. Neural networks with TF input have show promising results in problems like source separation [18], [19], audio in-painting [20] and even conditioned speech synthesis [21]. In complex generative models, e.g., [7], [22], the generation of time-frequency features is successfully used as an intermediate step. Why then, one could ask, has generation of audio in terms of an invertible time-frequency representation only seen limited success? STFT as TF representation

We consider generative modeling of what is arguably the best understood time-frequency representation, namely the short-time Fourier transform (STFT). The short-time Fourier transform is an ubiquitous tool in audio analysis and processing and has been subject to extensive studies for over half a century, e.g., [23], [24], [25] to name but a few. Instead of the complex STFT coefficients, analysis often relies on the magnitude (spectrogram) only, which represents the distribution of the signal energy in time-frequency space. In fact, the logarithmic magnitude (log-magnitude) is often preferred, it shows a more even distribution of values, see Figure 5, and provides a better match to the, approximately logarithmic, perception of sound pressure in humans. The phase component is often discarded or superficially treated, since it is much harder to understand. In contrast to the phase itself, the partial derivatives of the phase provide an intuitive interpretation as local instantaneous frequency and time [26], [27]. In fact, the phase-magnitude relations for the STFT [28], [29] suggest that the phase derivatives can even be estimated from the magnitude. Phase estimation and reconstruction for the STFT, from the magnitude or the phase derivatives, is unreliable in areas of small magnitude [30], requires sufficient redundancy [31] and is, at best, locally stable [32], [33], [34]. The Griffin-Lim algorithm [35], often applied in this context, is known to produce artifacts and require careful initialization, especially at low redundancy. Additionally, a generated magnitude estimate rarely corresponds to the true STFT of any time-domain signal, it is *inconsistent*.

This is a consequence of the transform redundancy, which implies that the STFT maps time-domain signals into a lower-dimensional subspace of all possible magnitudes. Synthesis from inconsistent magnitude may significantly change the signal characteristics and lead to artifacts. Moreover, any phase reconstruction method is bound to perform poorly. On the other hand, phase reconstruction, or even stable synthesis from valid STFT coefficients is not possible without redundancy, such that consistency is an essential consideration when generating STFTs.

We propose steps to overcome those difficulties, taking inspiration from properties of the continuous STFT [28], [29], [23] and from the recent progress in phaseless reconstruction (PLR) [36]. We provide guidelines for the choice of short-time Fourier transform (STFT) parameters that ensure the reliability of PLR and further provide methods for monitoring whether a generative model in training succeeds to generate consistent magnitude estimates. Eventually, we demonstrate the value of our guidelines in the context of unsupervised audio synthesis with GANs. We introduce TiFGAN, the first GAN to generate audio successfully using a TF representation and improving on the current state-of-the-art for audio synthesis with GANs[1]. We performed perceptual and objective evaluation. Our software, complemented by instructive examples, is available to the public at https://github.com/tifgan/stftGAN.

II. Structure of Continuous Short-Time Fourier Transforms

To understand how to overcome the difficulties that arise in the generation of short-time Fourier features, it is necessary to first understand continuous short-time Fourier transforms. The short-time Fourier transform of $f \in \mathbf{L}^2(\mathbb{R})$ with respect to the window $\varphi \in \mathbf{L}^2(\mathbb{R})$, with $\|\varphi\| = 1$ is given by

$$V_\varphi f(x,\omega) = \int_{\mathbb{R}} f(t)\overline{\varphi(t-x)}\mathrm{e}^{-2\pi i\omega t}\,dt \tag{1}$$

and maps $\mathbf{L}^2(\mathbb{R})$ to a subspace of $\mathbf{L}^2(\mathbb{R}^2)$. It can be inverted using the inverse STFT

$$f(t) = \int_{\mathbb{R}^2} V_\varphi f(x,\omega)\varphi(t-x)\mathrm{e}^{2\pi i\omega t}\,d(x,\omega) = V_\varphi^*(V_g f)(t), \tag{2}$$

where V_φ^* is the adjoint operator of V_φ. For general functions $F \in \mathbf{L}^2(\mathbb{R}^2)$, $V_\varphi V_\varphi^*(F)$ is the orthogonal projection of F onto the range of V_φ and $V_\varphi^*(F) = V_\varphi^* V_\varphi V_\varphi^*(F)$. Hence, if F is not in the range of V_φ, then $f = V_\varphi^*(F) \in \mathbf{L}^2(\mathbb{R})$ is still a valid time-domain signal, but $V_\varphi f \neq F$. F is an *inconsistent* time-frequency representation of f and the time-frequency structures of F will be distorted in $V_\varphi f$. Although consistency is a non-issue in time-frequency signal analysis, since $V_\varphi(f)$ is always consistent, it is crucial to ensure consistency when generating or processing signals in the time-frequency domain.

We propose to generate only the log-magnitude of the STFT, so the first question is how to generate a consistent STFT from a given log-magnitude. For this purpose, we can apply the so-called phase-magnitude relations of the STFT [28], [29]. They are derived from the fact that the STFT $V_{\varphi_1} f$ with a standard Gaussian window, $\varphi_1(t) = e^{-\pi t^2}$, can be related to the Bargman transform [39] of f, yielding that

$$(x,\omega) \mapsto \mathrm{e}^{-\pi i x\omega + \pi\frac{|z|^2}{2}}(\mathcal{V}_{\varphi_1} f)(x,-\omega) \tag{3}$$

is an analytic function. Applying the Cauchy-Riemann equations [40] and setting $\varphi_\lambda(t) = e^{-\pi t^2/\lambda}$ to be a dilated Gaussian, $M_\lambda = |V_{\varphi_\lambda} f|$ and $\phi_\lambda = \arg(V_{\varphi_\lambda} f)$, results in the relations

$$\frac{\partial}{\partial x}\phi_\lambda = \lambda^{-1}\frac{\partial}{\partial \omega}\log M_\lambda, \tag{4}$$

$$\frac{\partial}{\partial \omega}\phi_\lambda = -\lambda\frac{\partial}{\partial x}\log M_\lambda - 2\pi x, \tag{5}$$

where λ is the time-frequency ratio of φ_λ, i.e. the ratio of the standard deviations of $\varphi_\lambda/\|\varphi_\lambda\|_1$ and $\widehat{\varphi_\lambda}/\|\widehat{\varphi_\lambda}\|_1$. The relations (4) and (5) hold for every point (x,ω), except when $V_{\varphi_\lambda} f(x,\omega) = 0$. Hence, given the log-magnitude $\log M_\lambda$, and setting $\tilde{\phi}_\lambda(0,0)$ to some arbitrary real number, we can obtain a phase by integration using (4) and (5), so long as we avoid integrating through points (x,ω) with $M_\lambda = 0$. If $\log M_\lambda$ is consistent, then $S_\lambda = \exp(\log M_\lambda + i\tilde{\phi}_\lambda)$ will be a consistent STFT, i.e. there is an $f \in \mathbf{L}^2(\mathbb{R})$, such that $S_\lambda = V_{\varphi_\lambda} f$. For other windows g, similar equations exist [41], with additional terms resulting from STFT not being analytic, but in practice, the approximations provided by (4), (5) are often good enough.

Given a generated function $\tilde{M}_{\log}$ with $\exp(\tilde{M}_{\log}) \in \mathbf{L}^2(\mathbb{R}^2)$, it still remains to determine whether it is a consistent STFT log-magnitude, however. One way to achieve this would be to perform the phase reconstruction procedure outlined above and evaluate the projection error, i.e., compute $\|S_\lambda - V_{\varphi_\lambda} V_{\varphi_\lambda}^*(S_\lambda)\|$, which equals 0 if and only if $\tilde{M}_{\log}$ was consistent. While this is a valid approach, a more direct criterion is desirable in practice. Once more, analyticity of $V_{\varphi_\lambda} f$ can be employed to arrive at the conclusion that $\tilde{M}_{\log}$ is consistent, if and only if

$$\left(\lambda\frac{\partial^2}{\partial x^2} + \lambda^{-1}\frac{\partial^2}{\partial \omega^2}\right)\tilde{M}_{\log} = -2\pi. \tag{6}$$

Although the relation (6) is known [29], the bijectivity of this relationship has, to our knowledge, never been stated nor exploited to test consistency.

III. Structure of the Discrete Short-Time Fourier Transform

We now formally introduce the discrete STFT and discretizations of the properties discussed in the previous section. For any positive $j \in \mathbb{N}$, we set $\underline{j} = [0,\dots,j-1]$. The short-time Fourier transform [24] of a finite, real signal $s \in \mathbb{R}^L$, with

[1]During the preparation of this manuscript, the work [37] became publicly available, to our knowledge the first time that phase information was successfully included in the generation process. Usage of the time-direction phase derivative enabled their model, GANSynth, to produce significantly better results than previous methods, providing further evidence for the importance of phase information. The authors kindly provided us with details of their implementation, enabling a preliminary discussion of similarities and differences to our work.

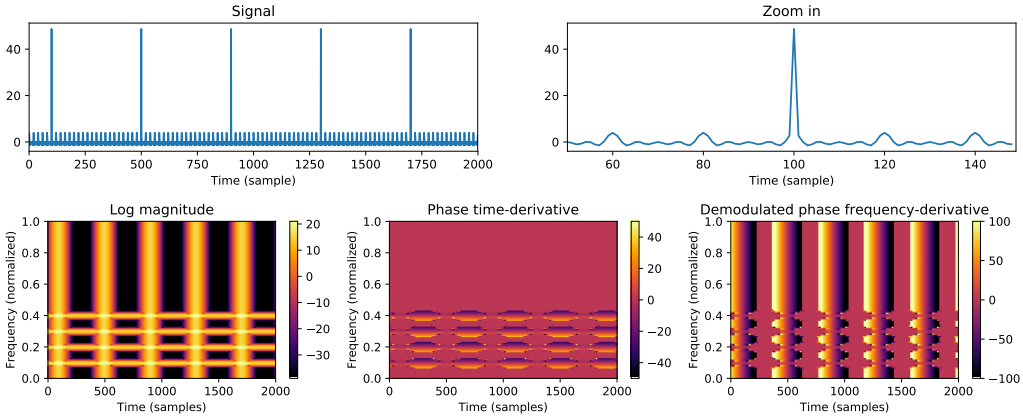

Fig. 1. Signal representations. Top row: waveform of a test signal (pure tone and pulses). Bottom row: STFT features: log magnitudes (left), time-direction phase derivatives (center) and frequency-direction phase derivatives (right). For small log magnitude, phase derivatives were set to zero. The frequency-direction derivative was computed after demodulation [38].

the *window* $g \in \mathbb{R}^L$, time step $a \in \mathbb{N}$ and $M \in \mathbb{N}$ frequency channels is given by

$$\mathrm{STFT}_g(s)[m,n] = \sum_{l \in \underline{L}} s[l] g[l-na] e^{-2\pi i m l/M}, \tag{7}$$

for $n \in \underline{N}, m \in \underline{M}$, with indices to be understood modulo L. The vectors $S_g[\cdot, n] \in \mathbb{C}^N$ and $S_g[m, \cdot] \in \mathbb{C}^M$ are called the n-th (time) frame and m-th channel of the STFT, respectively. The ratio M/a is a measure of the transform redundancy and the STFT is overcomplete (or redundant) if $M/a > 1$. Since s and g are real, all time frames are conjugate symmetric and it is sufficient to store the first $M_{\mathbb{R}} = \lfloor M/2 \rfloor$ channels, such that the STFT matrix has size $M_{\mathbb{R}} \times N$. If $f, \varphi(\cdot - L/2) \in \mathbf{L}^2(\mathbb{R})$ are approximately time- and bandlimited in $[0, L)$ and $(-1/2, 1/2]$, respectively, then with $s[l] = f(l)$ for $l \in \underline{L}$ and $g[l] = \varphi(l)$ for $l \in [-\lfloor L/2 \rfloor, \lceil L/2 \rceil - 1]$, the approximation

$$\mathrm{STFT}_g s[m,n] \approx V_\varphi f(na, m/M), \tag{8}$$

is valid for $n \in \underline{N}$ and $m \in \underline{M_{\mathbb{R}}}$.

The inverse STFT with respect to the *synthesis window* $\tilde{g} \in \mathbb{R}^L$ can be written as $\tilde{s}[l] = \sum_{n \in \underline{N}} \sum_{m \in \underline{M}} \mathrm{STFT}_g s[m,n] \tilde{g}[l - na] e^{2\pi i m l/M}$, $l \in \underline{L}$. We say that $\tilde{g}$ is a *dual window* for g, if $\tilde{s} = s$ for all $s \in \mathbb{R}^L$, [42], [43], [25].

Despite common practice, it is often advantageous to choose the number of channels M to be smaller than L_g. Rather, a, M should be chosen to respect the time- and frequency-concentration of the window g, leading to the rule of thumb

$$aM \approx L \frac{\sigma_g}{\sigma_{\hat{g}}}, \tag{9}$$

with the standard deviations $\sigma_g = \sigma(g/\|g\|_1)$ and $\sigma_{\hat{g}} = \sigma(\hat{g}/\|\hat{g}\|_1)$. For Gaussian windows, it is known that at integer redundancy, [44], and conjectured for other redundancies, [45], that choosing a, M according to (9) is strictly optimal in terms of overall transform stability, expressed by the frame bound ratio [46]. Finally, we strongly suggest using STFTs with sufficient redundancy $M/a \geq 4$, in particular when the generation of magnitude-only coefficients with subsequent phase reconstruction is considered.

phase Inconsistencies due to algo implementation !?

! Implementations of STFT in SciPy and Tensorflow delay the window by $\lfloor L_g/2 \rfloor$ samples, introducing a phase skew dependent on the (stored) window length L_g and with severe effects on any phase analysis and processing if not accounted for. Conversion between (7) and other conventions is considered in the appendix D and [38], [47].

A. Phase recovery and the phase-magnitude relationship

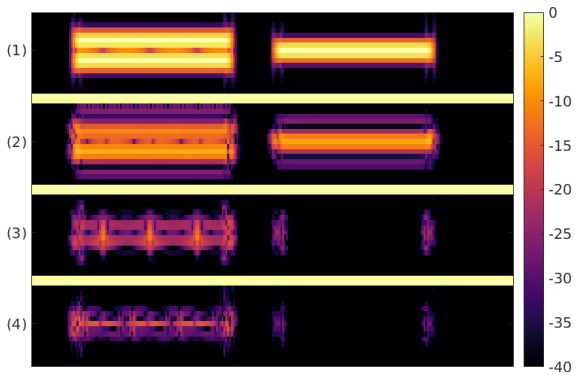

Fig. 2. Overview of spectral changes resulting from different phase reconstruction methods. (1) Original log-magnitude, (2-4) log-magnitude differences between original and signals restored with (2) cumulative sum along channels (initialized with zeros), (3) PGHI from phase derivatives (4) PGHI from magnitude only and phase estimated from (10).

The direct discrete approximation of the phase-magnitude relationship (4), (5) results in

$$\begin{aligned} \partial_n \phi_g[m,n] &\approx \frac{aM}{\lambda} \partial_m M_g[m,n], \\ \partial_m \phi_g[m,n] &\approx -\frac{\lambda}{aM} \partial_n M_g[m,n] - 2\pi na/M, \end{aligned} \tag{10}$$

with $\lambda = \sigma_g/\sigma_{\hat{g}}$. The accuracy of (10) depends on the proximity of the window g to a Gaussian and the STFT parameters

a, M. Optimal results are obtained for Gaussian windows at redundancy $M/a = L$. In general, STFT phase derivative estimation requires sufficient redundancy ($M/a \geq 4$) to be reliable. While STFTs with $M/a = 4$ perform decently and are considered in our network architecture, we have reason to believe that a further, moderate increase in redundancy has the potential to further elevate synthesis quality.

Once the phase derivatives are estimated, it may seem straightforward to restore the phase from its time-direction derivative by summation along frequency channels as proposed in [37]. Even on real, unmodified STFTs, the resulting phase misalignment introduces cancellation between frequency bands resulting in energy loss, see Figure 2 (2) for a simple example. In practice, such cancellations often leads to clearly perceptible changes of timbre[2]. Moreover, in areas of small STFT magnitude, the phase is known to be unreliable [30] and highly sensitive to distortions [32], such that it cannot be reliably modelled and synthesis from generated phase derivatives is likely to introduce more distortion. Phase-gradient heap integration (PGHI) [36] relies on the phase-magnitude relations (10) and bypasses phase instabilities by avoiding integration through areas of small magnitude, leading to significantly better and more robust phase estimates $\tilde{\phi}$, see Figure 2 (4). PGHI often outperforms more expensive, iterative schemes relying on alternate projection, e.g., Griffin-Lim [35], [48], [49], at the phaseless reconstruction (PLR) task.

PLR generally relies heavily on consistent STFT magnitude, cf. Section III-B below, for good results. Note that the integration step in PGHI can also be applied if phase derivative estimates from some other source are available, e.g., when training a network to learn time- and frequency-direction phase derivatives, see Figure 2(3) for an example.

B. Consistency of the STFT

Inversion of the STFT in general and PLR in particular require an overcomplete representation, i.e. $M/a > 1$. In many regards, higher redundancy implies improved robustness and readability of an STFT, but in the context of generative modeling, it also introduces an additional layer of complication: The space of valid STFTs with a given window is a lower dimensional subspace of all complex-valued matrices of size $M_{\mathbb{R}} \times N$. In other words, out of all matrices of the right size, only few are valid STFTs or STFT log-magnitudes. A given coefficient matrix may be very far from the STFT of any time-domain signal, even if it *looks* correct. To prevent degradation, it is important to ensure that the generated data S^{gen} is *consistent*, i.e., it satisfies $S^{\text{gen}} \approx \text{STFT}_g(s)$, for some signal $s \in \mathbb{R}^L$. The synthesis $\text{iSTFT}_{\tilde{g}}(S^{\text{gen}})$ implicitly projects onto the subspace of valid STFTs [46]. Therefore, consistency can be evaluated by computing the *projection error* $\text{e}^{\text{proj}} = \|S^{\text{gen}} - S^{\text{proj}}\|$, where

$$S^{\text{proj}} = \text{STFT}_g(\text{iSTFT}_{\tilde{g}}(S)), \tag{11}$$

$\|\cdot\|$ denotes the Euclidean norm and $\tilde{g}$ is a dual window, see Section III. If e^{proj} is large, its effects on the synthesized signal are unpredictable and, in particular if PLR is required, degraded synthesis quality must be expected.

Here, we instead propose to exploit the consistency relation (6), the direct discrete approximation of which suggests that the second order partial derivatives of M_g satisfy

$$\lambda \partial_n^2 M_g[m,n]/a^2 + \lambda^{-1} \partial_m^2 M_g[m,n]/b^2 \approx -2\pi/L. \tag{12}$$

In practice, and in particular at moderate redundancy, (12) is prone to numerical errors, however. We were more successful when tracking a measure inspired by the sample Pearson correlation coefficient. Let

$$\text{DM}_n = |\partial_n^2 \tilde{M} + \pi a^2 \lambda^{-1}/L|, \ \text{DM}_m = |\partial_m^2 \tilde{M} + \pi b^2 \lambda/L|, \tag{13}$$

where the terms $\pi a^2 \lambda^{-1}/L$ and $\pi b^2 \lambda/L$ are obtained by distributing the shift $2\pi/L$ in (12) equally to both terms on the left hand side. We define the consistency $\varrho(\tilde{M})$ of $\tilde{M}$ as

$$\varrho(\tilde{M}) := r_{\text{DM}_n, \text{DM}_m}, \tag{14}$$

where $r_{X,Y}$ is the sample Pearson correlation coefficient [50] of the paired sets of samples (X, Y). If the equality is satisfied in (12), then $\varrho(\tilde{M}) = 1$. Conversely if $\varrho(\tilde{M}) \approx 0$, then (12) surely violated and the representation is inconsistent. Although the usage of ϱ as consistency measure should still be considered experimental, its application is discussed in Section III-C below.

C. Performance of the consistency measure

The purpose of the consistency measure ϱ is to determine whether a generated log-magnitude is likely to be close to the log-magnitude STFT of a signal, i.e. it is consistent. As discussed above, consistency is crucial to prevent degraded synthesis quality. Hence, is important to evaluate its the dependence of its properties on changes in the redundancy, the window function and its sensitivity to distortion.

In a first test, we compute the mean and standard deviation of ϱ on a speech and a music dataset, see Section IV for details, at various redundancies, using Gaussian and Hann windows with time-frequency ratio $\lambda \approx 4$ and STFT parameters satisfying $aM/L = 4$, see Figure 3. We note that a Gaussian random matrix takes surprisingly high values for ϱ and, thus, ϱ is not reliable below redundancy 4. For Gaussian windows, mean consistency increases with redundancy, while the standard deviation decreases, indicating that ϱ becomes increasingly reliable and insensitive to signal content. This analysis suggests that in a redundancy of 8 or 16 could lead to notable improvements. At redundancy 4, spectrograms for both types of data score reliably better than the random case, with speech scoring higher than music. The Hann window scores worse than the Gaussian on average in all conditions, with a drop in performance above $M/a = 16$. This indicates that ϱ is only suitable to evaluate consistency of Hann window log-magnitudes for redundancies in the range 6 to 16.

In a second test, we fix a Gaussian STFT with redundancies 4 and 8 and evaluate the behaviour of ϱ under deviations from true STFT magnitudes. To this end, we add various levels of uniform Gaussian noise to the STFT before computing the

[2]See https://tifgan.github.io for examples.

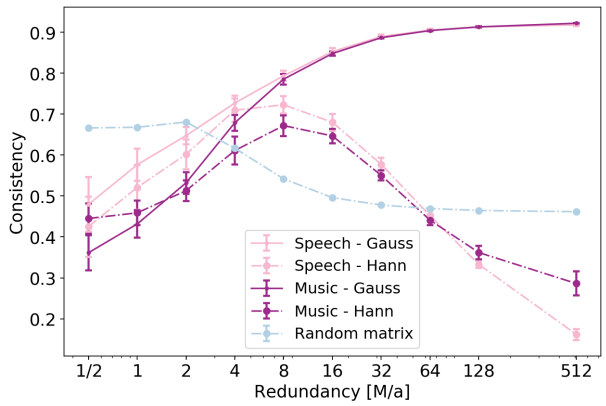

Fig. 3. Consistency as function of the redundancy for various time-domain windows. Random matrix from Gaussian distribution.

log–magnitude, see Figure 4. At redundancy 8 we observe a monotonous decrease of consistency with increasing noise level. In fact, the consistency converges to the level of random noise at high noise levels. Especially for music, ϱ is sensitive to even small levels of noise. At redundancy 4, the changes are not quite as pronounced, but the general trend is similar. While this is not a full analysis of the measure ϱ, it is reasonable to expect that models that match the value of ϱ closely generate approximately consistent log-magnitudes

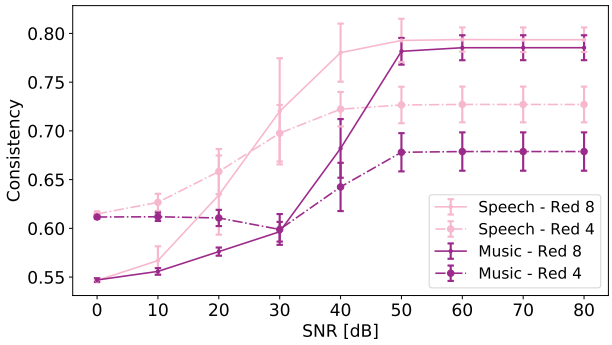

Fig. 4. Consistency as a function of SNR obtained by adding complex Gaussian noise to the STFT coefficients.

Furthermore, the results suggest that ϱ has a low standard deviation across data of the same distribution. In the context of GANs, where no obvious convergence criterion applies, ϱ can thus assist the determination of convergence and divergence by tracking

$$\gamma = \left| \mathbb{E}_{\mathbf{M}\sim\mathbb{P}_{\mathbf{M}_{\text{real}}}} \left[\varrho(\mathbf{M})\right] - \mathbb{E}_{\mathbf{M}\sim\mathbb{P}_{\mathbf{M}_{\text{fake}}}} \left[\varrho(\mathbf{M})\right] \right|. \tag{15}$$

IV. Time Frequency Generative Adversarial Network (TiFGAN)

Similar to WaveGAN and SpecGAN [12], TiFGAN is an adaptation of DCGAN [51], originally proposed for image generation. Our changes to DCGAN/SpecGAN are detailed below. The TiFGAN architecture, depicted in Figure 6, additionally relies on the guidelines and principles for generating short-time Fourier data that we presented in Sections II and III. For the purpose of this contribution, we restrict to generating 1 second of audio data, sampled at 16 kHz. More precisely, we generate $L = 16384$ samples of audio. For the short-time Fourier transform, we fix the minimal redundancy that we consider reliable, i.e., $M/a = 4$ and select $a = 128, M = 512$, such that $M_{\mathbb{R}} = 257$, $N = L/a = 128$ and the STFT matrix S_g is of size $\mathbb{C}^{M_{\mathbb{R}}\times N}$. This implies that the frequency step $b = L/M = 32$, such that we chose for the analysis window g a (sampled) Gaussian with time-frequency ratio $\lambda = 4 = aM/L$. Since the Nyquist frequency is not expected to hold significant information for the considered signals, we drop it to arrive at a representation size of 256×128, which is well suited to processing using strided convolutions.

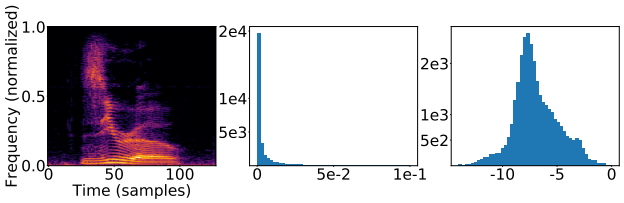

Fig. 5. From left to right: log magnitude, distribution of the magnitude, distribution of the log magnitude.

Since the distribution of values in the magnitude of the STFT is not well suited for a GAN (see Figure 5) we use log-magnitude coefficients. We first normalize the STFT magnitude to have maximum value 1, such that the log-magnitude is confined in $(-\infty, 0]$. Then, the dynamic range of the log-magnitude is limited by clipping at $-r$ (in our experiments $r = 10$), before scaling and shifting to the range of the generator output $[-1, 1]$, i.e. dividing by $r/2$ before adding constant 1. The network trained to generate log-magnitudes will be referred to as TiFGAN-M. Generation of, and synthesis from, the log-magnitude STFT is the main focus of this contribution. Nonetheless, we also trained a variant architecture TiFGAN-MTF for which we additionally provided the time- and frequency-direction derivatives of the (unwrapped, demodulated) phase[3] [38], [26].

For TiFGAN-M, the phase derivatives are estimated from the generated log-magnitude following (10). For both TiFGAN-M and TiFGAN-MTF, the phase is reconstructed from the phase derivative estimates using phase-gradient heap integration (PGHI) [36], which requires no iteration, such that reconstruction time is comparable to simply integrating the phase derivatives. For synthesis from the STFT, we use the *canonical dual window* [42], [46], precomputed using the Large Time-Frequency Analysis Toolbox (LTFAT) [52], [53].

a) GAN architecture: Our architecture is based on DCGAN [51] and similar to WaveGAN and SpecGAN [12], we add both a fully-connected layer at the beginning of the generator (and end of the discriminator), to improve the distribution of information over the whole representation, as well as one convolutional layer each to generator and discriminator to

[3]Phase derivatives were obtained using the `gabphasegrad` function in the Large Time-Frequency Analysis Toolbox (LTFAT) [52], [53].

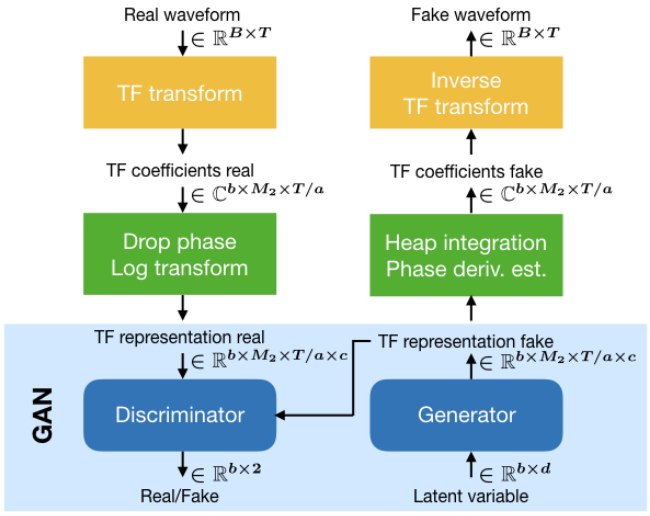

Fig. 6. The general architecture with parameters $T = 16384$, $a = 128$, $M_{\mathbb{R}} = 257$ $c = 1, 3$, $d = 100$. Here $B = 64$ is the batch size. The orange and green steps are done at a pre/post-processing stage.

enable the generation of larger matrices. Moreover, we generate data of size $(256, 128)$, a rectangular array of twice the width and four times the height of DCGANs output, such that we also adapted the filter shapes to better reflect and capture the rectangular shape of the training data[4]. Precisely, we use filters of size $(12, 3)$ instead of $(5, 5)$ and further reduce the number of filter channels of the fully-connected layer and the first convolutional layer of the generator by a factor of 2 compared to SpecGAN. Since these two layers comprise the majority of parameters, our architecture only has 10% more parameters than WaveGAN in total. However, when only the parameters of the other convolutional layers are considered, TiFGAN is 44% larger than WaveGAN. Details can be found in Appendix A.

b) Training: During training of TiFGAN, we monitored the relative consistency γ of the generator (15) in addition to the adversarial loss, negative critic and gradient penalty. In the optimization phase, networks that failed to train well, could often be detected to diverge in consistency and discarded after less than $50k$ steps of training (1 day), while promising candidates quickly started to converge towards the consistency of the training data, i.e., $\gamma \to 0$, see Figure 7. Networks with smaller γ synthesized better audio, but when trained for many steps, they were sometimes less reliable in terms of semantic audio content, e.g., for speech they were more likely to produce gibberish words than with shorter training. Our networks were trained for 200k steps as this seemed to provide reasonably good results in both semantic and audio quality. We optimized the Wasserstein loss [15] with the gradient penalty hyperparameter set to 10 using the ADAM optimizer [54] with $\alpha = 10^{-4}$, $\beta_1 = 0.5$, $\beta_2 = 0.9$ and performed 5 updates of the discriminator for every update of the generator. For the reference condition, we used the pre-trained WaveGAN network provided by [12][5].

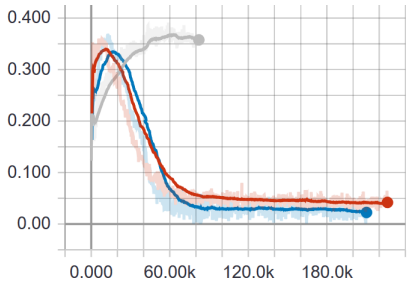

Fig. 7. Eq. (15) for three different networks. Gray shows a failed network. Red is obtained for the TiFGAN-M and Blue for the TiFGAN-MTF described in section IV

c) Comparison to SpecGAN [12]: Our network is purposefully designed to be similar to SpecGAN to emphasize that the main cause for improved results is the handling of time-frequency data according to the guidelines in Section III. Nonetheless, the larger target size required some changes to the network architecture, see above[6]. SpecGAN relies on an STFT of redundancy $M/a = 2$ with Hann window of length $L_g = 256$, time step $a = 128$ and $M = 256$ channels. According to Section III, this setup is not very well suited to generative modeling. PLR in particular is expected to be unreliable, which is evidenced by the results reported in [12], which employ the classical Griffin-Lim algorithm [35] for PLR. Finally, SpecGAN performs a normalization per frequency channel over the entire dataset, preventing the network to learn the natural relations between channels in the STFT log-magnitude, which are crucial for consistency, as shown in Section III.

A. Evaluation

To evaluate the performance of TiFGAN, we trained the variant TiFGAN-M and TiFGAN-MTF using the procedure outlined above on two datasets from [12]: (a) Speech, a subset of spoken digits "zero" through "nine" (sc09) from the "Speech Commands Dataset" [55]. The dataset is not curated, some samples are noisy or poorly labeled, the considered subset consists of approximately 23,000 samples. (b) Music, a dataset of 25 minutes of piano recordings of Bach compositions, segmented into approximately 19,000 overlapping samples of 1 s duration.

a) Evaluation metrics: For speech and music, we provide audio examples online[7]. For speech, we performed listening tests and evaluated the inception score (IS) [56] and Fréchet inception distance (FID) [57], using the pre-trained classifier provided with [12]. For the real data and both variants of TiFGAN, we moreover computed the consistency ϱ (14) and

[4]When training on piano data, we also observed that, when using square filters, the frequency content of note onsets was unnaturally dispersed over time. This effect was notably reduced after switching to tall filters

[5]https://github.com/chrisdonahue/wavegan

[6]Note that SpecGAN is of equal size as WaveGAN and uses $(5, 5)$ filters.

[7]https://tifgan.github.io

	vs TiFGAN-M	vs TiFGAN-MTF	vs WaveGAN	Cons	RSPE (dB)	IS	FID
Real	86%	90%	94%	0.70	-22.0	7.98	0.5
TiFGAN-M	–	67%	75%	0.67	-13.8	5.97	26.7
TiFGAN-MTF	33%	–	55%	0.68	-7.5	4.48	32.6
WaveGAN	25%	45%	–	–	–	4.62	41.6

TABLE I

RESULTS OF THE EVALUATION. FIRST THREE LEFT COLUMNS: PREFERENCE (IN %) OF THE CONDITION SHOWN IN A ROW OVER THE CONDITIONS SHOW IN A COLUMN. CONS: AVERAGED CONSISTENCY MEASURE ρ. RSPE: AS IN EQ. (16). IS: INCEPTION SCORE. FID: FRÉCHET INCEPTION DISTANCE.

the relative spectral projection error (RSPE) in dB, after phase reconstruction from the log-magnitude, i.e.,

$$10\log_{10}\left(\frac{\||S|-|S^{\text{proj}}|\|}{\|S\|}\right), \tag{16}$$

where $|S| = |\mathrm{STFT}_g(s)|$ in the case of real data and $|S| = exp(\tilde{M})$, with the generated log-magnitude $\tilde{M}$, for the generated data. Finally, $S^{\text{proj}} = \mathrm{STFT}_g(\mathrm{iSTFT}_{\tilde{g}}(S))$, as in Section III-B, where PGHI was applied to obtain S from $|S|$ (and generated phase derivatives in that case of TiFGAN-MTF).

Listening tests where performed in a sound booth and sounds were presented via headphones, see Appendix B. The task involved pairwise comparison of preference between four conditions: real data extracted from the dataset, TiFGAN-M generated examples, TiFGAN-MTF generated examples, and WaveGAN generated examples. In each trial, listeners were provided with two sounds from two different conditions. Signals were selected at random from 600 pre-generated examples per condition. Each of the six possible combinations was repeated 80 times in random order, yielding 480 trials per listener. The test lasted approximately 45 minutes including breaks which subjects were allowed to take at any time. Seven subjects were tested and none of them were the authors. A post-screening showed that one subject was not able to distinguish between any of the conditions and thus was removed from the test, yielding in 2880 valid preferences in total from six subjects.

b) Results: The results are summarized in Table IV-0c. On average, the subjects preferred the real samples over WaveGAN's in 94%. For TiFGAN-MTF, the preference decreased to 90% and for TiFGAN-M further to 86%. The large gap between generated and real data can be explained by the experimental setup that enables a very critical evaluation. Nonetheless, it is apparent that TiFGAN-M performed best in the direct comparison to real data by a significant margin. Comparison of the other pairings leads to a similar conclusion: Subjects preferred TiFGAN-MTF over WaveGAN in 55%, TiFGAN-M over WaveGAN in 75% and TiFGAN-M over TiFGAN-MTF in 67%. While TiFGAN-M clearly outperformed the other networks, TiFGAN-MTF was only slightly more often preferred over WaveGAN.

The analysis of IS and FID leads to similar conclusions: TiFGAN-M showed a large improvement on both measures over the other conditions, with still a large gap to the real-data performance. On the other hand, comparing WaveGAN to TiFGAN-MTF, the results for both measures are mixed.

For both TiFGAN-M and TiFGAN-MTF, the consistency was similar, close to those obtained for the real dataset. Nevertheless, for TiFGAN-MTF, the relative spectral projection error (RSPE) of -7.5dB was substantially larger than that for TiFGAN-M (-13.8 dB), suggesting that TiFGAN-MTF was not able to learn the phase derivatives as accurate as those reconstructed by TiFGAN-M. To look more deeply into this, for TiFGAN-MTF, we discarded the phase derivatives and estimated new phase derivatives based on the log magnitudes (as done for TiFGAN-M). By doing so, the average RSPE decreased to -12.5dB, which confirms our finding that phase reconstruction provides better results than phase generation by our network.

In summary, TiFGAN-M provided a substantial improvement over the previous state-of-the-art in unsupervised adversarial audio generation. Although the results for TiFGAN-MTF are not as clear, we believe that direct generation of phase has the potential to provide results on par or better than the magnitude alone and should be systematically investigated. Further research will focus on avoiding discrepancies between the phase derivatives and the log-magnitude.

V. CONCLUSIONS

In this contribution, we considered the difficulties of generating TF features and how to overcome them in order to avoid degraded quality, artifacts, and distortion at the synthesis stage. We introduced a new measure assessing the quality of a TF representation, i.e., the consistency measure. It is computationally cheap and can be used to a priori estimate the potential success of phase recovery. In the context of adversarial neural networks, it can ease the assessment of convergence at training time.

We applied the discussed considerations to GANs which results in the introduction of TiFGAN, a GAN directly generating STFT representations. Two variants of TiFGANs were trained on speech and music. In terms of psychoacoustic evaluation and numeric measures, our TiFGAN outperformed the current state of the art GAN-based networks demonstrating the potential of TF representations in generative modeling.

In the future, further extensions of the proposed approach are planned towards logarithmic and perceptual frequency scales [58], [59], [60], [61], [62].

ACKNOWLEDGMENTS

This work has been supported by Austrian Science Fund (FWF) project MERLIN (Modern methods for the restoration of lost information in digital signals;I 3067-N30). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. We would like to thank the authors of [37] for providing us with details of their implementations prior to presenting it at ICLR 2019, allowing us to have their approach as a comparison.

REFERENCES

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

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

[3] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[4] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[5] 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," *CoRR*, vol. abs/1612.07837, 2016. [Online]. Available: http://arxiv.org/abs/1612.07837

[6] 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," *CoRR abs/1609.03499*, 2016.

[7] J. Shen, R. Pang, R. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," *CoRR*, vol. abs/1712.05884, 2017. [Online]. Available: http://arxiv.org/abs/1712.05884

[8] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," 2017.

[9] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," 2017.

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

[11] T. L. Paine, P. Khorrami, S. Chang, Y. Zhang, P. Ramachandran, M. A. Hasegawa-Johnson, and T. S. Huang, "Fast wavenet generation algorithm," *arXiv preprint arXiv:1611.09482*, 2016.

[12] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[13] S. Pascual, A. Bonafonte, and J. Serra, "Segan: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.

[14] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[16] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6964–6968.

[17] Z. Zhu, J. H. Engel, and A. Hannun, "Learning multiscale features directly from waveforms," *arXiv preprint arXiv:1603.09509*, 2016.

[18] Z.-C. Fan, Y.-L. Lai, and J.-S. R. Jang, "Svsgan: Singing voice separation via generative adversarial network," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 726–730.

[19] J. Muth, S. Uhlich, N. Perraudin, T. Kemp, F. Cardinaux, and Y. Mitsufuji, "Improving dnn-based music source separation using phase features," *arXiv preprint arXiv:1807.02710*, 2018.

[20] A. Marafioti, N. Perraudin, N. Holighaus, and P. Majdak, "A context encoder for audio inpainting," *Preprint, submitted to IEEE TASLP.*, 2018. [Online]. Available: https://arxiv.org/pdf/1810.12138.pdf

[21] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. Saurous, "Tacotron: A fully end-to-end text-to-speech synthesis model," *CoRR*, vol. abs/1703.10135, 2017.

[22] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer," *arXiv preprint arXiv:1811.09620*, 2018.

[23] K. Gröchenig, *Foundations of Time-Frequency Analysis*, ser. Appl. Numer. Harmon. Anal. Birkhäuser, 2001.

[24] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 3, pp. 235–238, 1977.

[25] J. Wexler and S. Raz, "Discrete gabor expansions," *Signal Processing*, vol. 21, no. 3, pp. 207 – 220, 1990.

[26] M. Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, 1986.

[27] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *IEEE Trans. Signal Proc.*, vol. 43, no. 5, pp. 1068 –1089,, may 1995.

[28] M. Portnoff, "Implementation of the digital phase vocoder using the fast fourier transform," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 24, no. 3, pp. 243–248, 1976.

[29] F. Auger, É. Chassande-Mottin, and P. Flandrin, "On phase-magnitude relationships in the short-time fourier transform." *IEEE Signal Process. Lett.*, vol. 19, no. 5, pp. 267–270, 2012.

[30] P. Balazs, D. Bayer, F. Jaillet, and P. Søndergaard, "The pole behavior of the phase derivative of the short-time fourier transform," *Applied and Computational Harmonic Analysis*, vol. 40, no. 3, pp. 610–621, 2016.

[31] R. Balan, P. Casazza, and D. Edidin, "On signal reconstruction without phase," *Applied and Computational Harmonic Analysis*, vol. 20, no. 3, pp. 345–356, 2006.

[32] R. Alaifari and M. Wellershoff, "Stability estimates for phase retrieval from discrete gabor measurements," *arXiv preprint arXiv:1901.05296*, 2019.

[33] R. Alaifari and P. Grohs, "Phase retrieval in the general setting of continuous frames for banach spaces," *SIAM journal on mathematical analysis*, vol. 49, no. 3, pp. 1895–1911, 2017.

[34] S. Mallat and I. Waldspurger, "Phase retrieval for the cauchy wavelet transform," *Journal of Fourier Analysis and Applications*, vol. 21, no. 6, pp. 1251–1309, 2015.

[35] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[36] Z. Průša, P. Balazs, and P. Søndergaard, "A noniterative method for reconstruction of phase from stft magnitude," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 25, no. 5, pp. 1154–1164, 2017.

[37] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[38] D. Arfib, F. Keiler, U. Zölzer, V. Verfaille, and J. Bonada, "Time-frequency processing," *DAFX: Digital Audio Effects*, pp. 219–278, 2011.

[39] V. Bargmann, "On a hilbert space of analytic functions and an associated integral transform part i," *Communications on Pure and Applied Mathematics*, vol. 14, no. 3, pp. 187–214, 1961. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160140303

[40] J. B. Conway, *Functions of one complex variable [by] John B. Conway*. Springer-Verlag New York [New York], 1973.

[41] E. Chassande-Mottin, I. Daubechies, F. Auger, and P. Flandrin, "Differential reassignment," *IEEE signal processing letters*, vol. 4, no. 10, pp. 293–294, 1997.

[42] T. Strohmer, "Numerical algorithms for discrete Gabor expansions," in *Gabor Analysis and Algorithms: Theory and Applications*, ser. Appl. Numer. Harmon. Anal., H. G. Feichtinger and T. Strohmer, Eds. Birkhäuser Boston, 1998, pp. 267–294.

[43] A. Janssen, "From continuous to discrete weyl-heisenberg frames through sampling," *Journal of Fourier Analysis and Applications*, vol. 3, no. 5, pp. 583–596, 1997.

[44] M. Faulhuber and S. Steinerberger, "Optimal gabor frame bounds for separable lattices and estimates for jacobi theta functions," *Journal of Mathematical Analysis and Applications*, vol. 445, no. 1, pp. 407–422, 2017.

[45] T. Strohmer and S. Beaver, "Optimal OFDM system design for time-frequency dispersive channels," *IEEE Trans. Comm.*, vol. 51, no. 7, pp. 1111–1122, July 2003.

[46] O. Christensen, *An Introduction to Frames and Riesz Bases*, Second ed., ser. Applied and Numerical Harmonic Analysis. Birkhäuser Basel, 2016.

[47] Z. Pruša, "Stft and dgt phase conventions and phase derivatives interpretation," Acoustics Research Institute, Austrian Academy of Sciences, Tech. Rep., 2015.

[48] J. Le Roux, H. Kameoka, N. Ono, and S. Sagayama, "Fast signal reconstruction from magnitude stft spectrogram based on spectrogram consistency," in *Proc. Int. Conf. Digital Audio Effects*, vol. 10, 2010.

[49] N. Perraudin, P. Balazs, and P. L. Søndergaard, "A fast griffin-lim algorithm," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*. IEEE, 2013, pp. 1–4.

[50] L. Lyons, *A Practical Guide to Data Analysis for Physical Science Students*. Cambridge University Press, 1991.
[51] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.
[52] Z. Průša, P. L. Søndergaard, N. Holighaus, C. Wiesmeyr, and P. Balazs, "The Large Time-Frequency Analysis Toolbox 2.0," in *Sound, Music, and Motion*, ser. LNCS. Springer International Publishing, 2014, pp. 419–442.
[53] "Ltfat: The large time-frequency analysis toolbox." [Online]. Available: ltfat.github.io
[54] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.
[55] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.
[56] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.
[57] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.
[58] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.
[59] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant q transform," *The Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2698–2701, 1992.
[60] N. Holighaus, M. Dörfler, G. A. Velasco, and T. Grill, "A framework for invertible, real-time constant-q transforms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, pp. 775–785, 2013.
[61] N. Holighaus, G. Koliander, Z. Průša, and L. D. Abreu, "Characterization of analytic wavelet transforms and a new phaseless reconstruction algorithm," *Preprint, submitted to IEEE Trans. Sig. Proc.*, 2019. [Online]. Available: http://ltfat.github.io/notes/ltfatnote053.pdf
[62] T. Necciari, N. Holighaus, P. Balazs, Z. Průša, P. Majdak, and O. Derrien, "Audlet filter banks: A versatile analysis/synthesis framework using auditory frequency scales," *Applied Sciences*, vol. 8, no. 1:96, 2018.

APPENDIX A
DETAIL OF THE THE GAN ARCHITECTURE

Table II presents the details of the convolutional architecture used by TFGAN-M and TFGAN-MTF. Here $B = 64$ is the batch size.

Operation	Kernel Size	Output Shape
Generator		
Input z $\mathcal{N}(0, 1)$		$(B, 100)$
Dense	$(100, 256s)$	$(B, 256s)$
Reshape		$(B, 8, 4, 8s)$
DeConv 2D (Stride 2)	$(12, 3, 8s, 8s)$	$(B, 16, 8, 8s)$
LReLu ($\alpha = 0.2$)		$(B, 16, 8, 8s)$
DeConv 2D (Stride 2)	$(12, 3, 8s, 4s)$	$(B, 32s, 16, 4s)$
LReLu ($\alpha = 0.2$)		$(B, 32s, 16, 4s)$
DeConv 2D (Stride 2)	$(12, 3, 4s, 2s)$	$(B, 64, 32, 2s)$
LReLu ($\alpha = 0.2$)		$(B, 64, 32, 2s)$
DeConv 2D (Stride 2)	$(12, 3, 2s, s)$	$(B, 128, 64, s)$
LReLu ($\alpha = 0.2$)		$(B, 32, 32, 2d)$
DeConv 2D (Stride 2)	$(12, 3, s, c)$	$(B, 256, 128, c)$
Discriminator		
Input		$(B, 256, 128, c)$
Conv 2D (Stride 2)	$(12, 3, c, s)$	$(B, 128, 64, s)$
LReLu ($\alpha = 0.2$)		$(B, 128, 64, s)$
Conv 2D (Stride 2)	$(12, 3, s, 2s)$	$(B, 64, 32, 2s)$
LReLu ($\alpha = 0.2$)		$(B, 64, 32, 2s)$
Conv 2D (Stride 2)	$(12, 3, 2s, 4s)$	$(B, 32, 16, 4s)$
LReLu ($\alpha = 0.2$)		$(B, 32, 16, 4s)$
Conv 2D (Stride 2)	$(12, 3, 4s, 8s)$	$(B, 16, 8, 8s)$
LReLu ($\alpha = 0.2$)		$(B, 16, 8, 8s)$
Conv 2D (Stride 2)	$(12, 3, 8s, 16s)$	$(B, 8, 4, 16s)$
LReLu ($\alpha = 0.2$)		$(B, 8, 4, 16s)$
Reshape		$(B, 512s)$
Dense	$(512s, 1)$	$(B, 1)$

TABLE II
DETAILED ARCHITECTURE OF THE GENERATIVE ADVERSARIAL NETWORK. SCALE $s = 64$. CHANNELS $c = 1$ FOR THE MAGNITUDE NETWORK AND $c = 3$ FOR THE NETWORK THAT ALSO OUTPUTS THE DERIVATIVES.

APPENDIX B
LISTENING TEST LOCATION

Figures 8 and 9 show the physical setup of the listening test, including the sound booth and additional equipment.

Fig. 8. Inside of the sound booth used to perform the listening test.

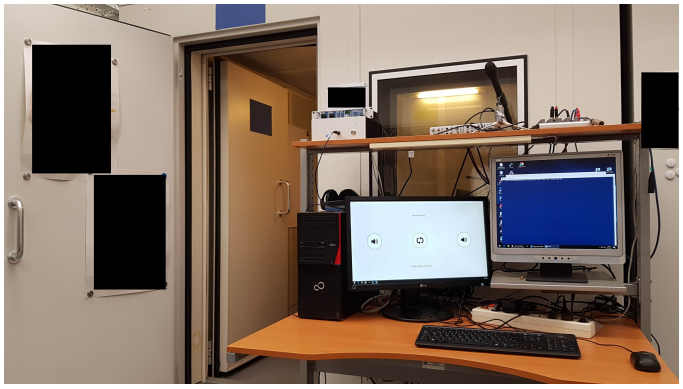

Fig. 9. Sound booth from the outside with equipment for external monitoring of ongoing tests.

APPENDIX C
COMPARISON TO GANSYNTH

A direct comparison between the results of the very recent GANSynth architecture [37], which obtained unprecedented audio quality for adversarial audio synthesis, and TFGAN is not straightforward, since GANSynth considers semi-supervised generation conditioned on pitch, while we considered unsupervised generation to facilitate the comparison with WaveGAN. Further, the network architecture [3] on which GANSynth is built is significantly larger and more sophisticated than our DCGAN-derived architecture. The adaptation of TFGAN to a comparable architecture and its application to semi-supervised generation generation is planned for future work. For now, we can only observe that a key ingredient of GANSynth is the usage of the time-direction phase derivative, which in fact corroborates our claim that careful modeling of the structure of the STFT is crucial for neural generation of time-frequency features. As discussed in Section III-A, the PLR method employed in GANSynth can be unreliable and synthesis quality can likely be further improved if a more robust method for PLR is considered. Audio examples for different PLR methods are provided in the webpage[8].

APPENDIX D
SHORT-TIME FOURIER PHASE CONVENTIONS

In Section III, we introduced the STFT in the so-called *frequency-invariant* convention. This is the convention preferred in the mathematical community. It arises from the formulation of the discrete STFT as a sliding window DFT. There are various other conventions, depending on how the STFT is derived and implemented. Usually, the chosen convention does not affect the magnitude, but only the phase of the STFT. When phase information is processed, it is crucial to be aware of the convention in which the STFT is computed, and adapt the processing scheme accordingly. Usually, the conversion between conventions amounts to the pointwise multiplication of the STFT with a predetermined matrix of phase factors. Common *phase conventions* and the conversion between them

[8]https://tifgan.github.io

are discussed in [26], [38]. The 3 most wide-spread conventions, the last of which is rarely described, but frequently implemented in software frameworks, are presented here:

a) Frequency-invariant STFT:: The m-th channel of the frequency-invariant STFT can be interpreted as demodulating the signal s with the pure frequency $e^{-2\pi iml/M}$, before applying a lowpass filter with impulse response $g[-\cdot]$. Therefore, the phase is expected to change slowly in the time-direction, it is already demodulated. The time-direction phase derivative indicates the distance of the current position to the local instantaneous frequency. On the other hand, the evolution of the phase in frequency-direction depends on the time position. Hence, the frequency-direction derivative indicates the (absolute) local group delay, sometimes called the instantaneous time.

b) Time-invariant STFT:: Given by

$$\begin{aligned}&\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n]\\ &\qquad=\sum_{l=-\lfloor L_g/2\rfloor}^{\lceil L_g/2\rceil-1} s[l+na]g[l]e^{-2\pi iml/M},\end{aligned}\tag{17}$$

the time-invariant STFT can be interpreted as filtering the signal s with the bandpass filters $g[-\cdot]e^{2\pi im(\cdot)/M}$. Hence, the phase is expected to revolve at roughly the channel center frequency in the time-direction and the time-direction phase derivative points to the (absolute) local instantaneous frequency. In the frequency direction, however, the phase in the frequency-direction changes slowly, i.e. it is demodulated in the frequency-direction. The frequency-direction phase derivative indicates the distance to the local instantaneous time. In each, the frequency- and time-invariant STFT, the phase is demodulated in one direction, but moves quickly in the other. In Section III-A, we propose to use the derivative of the demodulated phase in both directions, such that we must convert between the two conventions. This conversion is achieved simply by pointwise multiplication of the STFT matrix with a matrix of phase factors:

$$\begin{aligned}\mathrm{STFT}_g(s)[m,n]&=e^{-2\pi imna/M}\,\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n]\\&=W[m,n]\,\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n].\end{aligned}\tag{18}$$

Equivalently, if $\phi_g = \arg(\mathrm{STFT}_g(s))$ is the phase of the frequency-invariant STFT and $\phi_g^{\mathrm{ti}} = \arg(\mathrm{STFT}_g^{\mathrm{ti}}(s))$, then $\phi_g[m,n] = \phi_g^{\mathrm{ti}}[m,n] - 2\pi imna/M$.

c) Simplified time-invariant STFT:: In many common frameworks, including SciPy and Tensorflow, the STFT computation follows neither the frequency- nor time-invariant conventions. Instead, the window g is stored as a vetcor of length L_g with the peak not at $g[0]$, but at $g[\lfloor L_g/2\rfloor]$. The STFT is then computed as

$$\mathrm{STFT}_g^{\mathrm{sti}}(s)[m,n]=\sum_{l=0}^{L_g-1} s[l+na]g[l]e^{-2\pi iml/M}.\tag{19}$$

The above equation is slightly easier to implement, compared to the frequency- or time-invariant STFT, if $M \geq L_g$, since in that case, $g \in \mathbb{R}^{L_g}$ can simply be zero-extended to length M, after which the following holds: $\mathrm{STFT}_g^{\mathrm{sti}}(s)[\cdot,n] = \mathrm{DFT}_M(s^{(n)})[m]$, with $s^{(n)} = [s[na]g[0],\dots,s[na+M-1]g[M-1]]^T \in \mathbb{R}^M$. Comparing (17) with (19), we can see that the latter introduces a delay and a phase skew dependent on the (stored) window length L_g. In general, we obtain the equality

$$\begin{aligned}&\mathrm{STFT}_g^{\mathrm{sti}}(s)[m,n]\\&=e^{-2\pi im\lfloor L_g/2\rfloor/M}\,\mathrm{STFT}_g^{\mathrm{ti}}(s[\cdot+\lfloor L_g/2\rfloor])[m,n].\end{aligned}\tag{20}$$

If the hop size a is a divisor of $\lfloor L_g/2\rfloor$, then we can convert into a time-invariant STFT:

$$\begin{aligned}&\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n+\lfloor L_g/2\rfloor/a]\\&=e^{2\pi im\lfloor L_g/2\rfloor/M}\,\mathrm{STFT}_g^{\mathrm{sti}}(s)[m,n],\end{aligned}\tag{21}$$

or equivalently $\phi_g[m,n+\lfloor L_g/2\rfloor/a] = \phi_g^{\mathrm{ti}}[m,n+\lfloor L_g/2\rfloor/a] - 2\pi im(na+\lfloor L_g/2\rfloor)/M = \phi_g^{\mathrm{sti}}[m,n] - 2\pi imna/M$. Note that, additionally, SciPy and Tensorflow do not consider s circularly, but compute only those time frames, for which the window does not overlap the signal borders, i.e., $n \in [0,\dots,\lfloor (L-L_g)/a\rfloor]$. If an STFT according to the convention (19), with N time frames and aligned with the time-invariant STFT is desired, the signal s can be extended to length $L+L_g$ by adding $\lfloor L_g/2\rfloor$ zeros before $s[0]$ and $\lceil L_g/2\rceil$ zeros after $s[L-1]$.