Supplementary Material

A. Detail of the the GAN architecture

Table 2 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 2. 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.

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

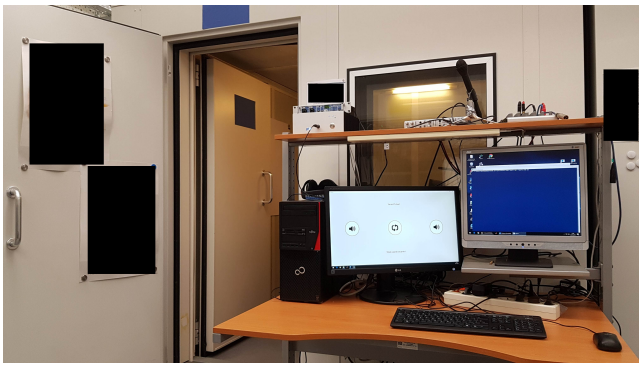

Figure 9. Sound booth from the outside with equipment for external monitoring of ongoing tests.

B. Listening test location

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

C. Comparison to GANSynth:

A direct comparison between the results of the very recent GANSynth architecture (Engel et al., 2019), 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 (Karras et al., 2018) 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 2.3, 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 supplementary material.

D. Short-time Fourier phase conventions

In Section 2.2, 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 point-wise multiplication of the STFT with a predetermined matrix of phase factors. Common *phase conventions* and the conversion between them are discussed in (Dolson, 1986; Arfib et al., 2011). The 3 most wide-spread conventions, the last of which is rarely described, but frequently implemented in software frameworks, are presented here:

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 low-pass 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.

Time-invariant STFT: Given by

$$\begin{aligned}&\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n]\\ &\qquad=\sum_{l=-\lfloor L_g/2\rfloor}^{\lceil L_g/2\rceil-1} s[l+na]g[l]e^{-2\pi iml/M},\end{aligned} \tag{12}$$

the time-invariant STFT can be interpreted as filtering the signal s with the band-pass 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 2.3, 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:

$$\begin{aligned}\mathrm{STFT}_g(s)[m,n] &= e^{-2\pi imna/M}\,\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n]\\ &= W[m,n]\,\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n].\end{aligned} \tag{13}$$

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 $\phi_g[m,n] = \phi_g^{\mathrm{ti}}[m,n] - 2\pi imna/M$.

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 vector 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

$$\mathrm{STFT}_g^{\mathrm{sti}}(s)[m,n] = \sum_{l=0}^{L_g-1} s[l+na]g[l]e^{-2\pi iml/M}. \tag{14}$$

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 $s^{(n)} = [s[na]g[0],\ldots,s[na+M-1]g[M-1]]^T \in \mathbb{R}^M$. Comparing (12) with (14), 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

$$\begin{aligned}&\mathrm{STFT}_g^{\mathrm{sti}}(s)[m,n]\\ &= e^{-2\pi im\lfloor L_g/2\rfloor/M}\,\mathrm{STFT}_g^{\mathrm{ti}}(s[\cdot+\lfloor L_g/2\rfloor])[m,n].\end{aligned} \tag{15}$$

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

$$\begin{aligned}&\mathrm{STFT}_g^{\mathrm{ti}}(s)[m,n+\lfloor L_g/2\rfloor/a]\\ &= e^{2\pi im\lfloor L_g/2\rfloor/M}\,\mathrm{STFT}_g^{\mathrm{sti}}(s)[m,n],\end{aligned} \tag{16}$$

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., $n \in [0,\ldots,\lfloor(L-L_g)/a\rfloor]$. If an STFT according to the convention (14), 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 $s[L-1]$.