IOP Publishing Mach. Learn.: Sci. Technol. 4 (2023) 035023 https://doi.org/10.1088/2632-2153/acee44

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
21 April 2023

REVISED
3 August 2023

ACCEPTED FOR PUBLICATION
8 August 2023

PUBLISHED
21 August 2023

PAPER

Data-driven modeling of noise time series with convolutional generative adversarial networks*

Adam Wunderlich** and **Jack Sklar**

Communications Technology Laboratory, National Institute of Standards and Technology, Boulder, CO 80305, United States of America
** Author to whom any correspondence should be addressed.

E-mail: adam.wunderlich@nist.gov

Keywords: time series, generative adversarial network (GAN), power law noise, shot noise, impulsive noise, fractional gaussian noise, fractional Brownian motion

Abstract

Random noise arising from physical processes is an inherent characteristic of measurements and a limiting factor for most signal processing and data analysis tasks. Given the recent interest in generative adversarial networks (GANs) for data-driven modeling, it is important to determine to what extent GANs can faithfully reproduce noise in target data sets. In this paper, we present an empirical investigation that aims to shed light on this issue for time series. Namely, we assess two general-purpose GANs for time series that are based on the popular deep convolutional GAN architecture, a direct time-series model and an image-based model that uses a short-time Fourier transform data representation. The GAN models are trained and quantitatively evaluated using distributions of simulated noise time series with known ground-truth parameters. Target time series distributions include a broad range of noise types commonly encountered in physical measurements, electronics, and communication systems: band-limited thermal noise, power law noise, shot noise, and impulsive noise. We find that GANs are capable of learning many noise types, although they predictably struggle when the GAN architecture is not well suited to some aspects of the noise, e.g. impulsive time-series with extreme outliers. Our findings provide insights into the capabilities and potential limitations of current approaches to time-series GANs and highlight areas for further research. In addition, our battery of tests provides a useful benchmark to aid the development of deep generative models for time series.

1. Introduction

Noise, commonly defined as an unwanted, irregular disturbance, is a fundamental aspect of all real-world signals that arises from a multitude of natural and man-made sources. Furthermore, the unpredictable, stochastic nature of noise makes it a significant impediment to measurement, data analysis, and signal processing. Efforts to understand, mitigate, and harness the effects of noise over the last century have led to the extensive development of many physical and mathematical models, e.g. see [1–4] for overviews.

With recent advances in computational hardware and measurement equipment, it is now possible to collect, store, process, and analyze much larger quantities of data than previously possible. Consequently, flexible, data-driven methods for signal modeling and processing are increasingly becoming feasible in many areas of science and engineering [5]. One form of data-driven modeling that has rapidly progressed in recent years is deep generative modeling, a type of unsupervised machine learning that uses deep neural networks to implicitly represent complicated, high-dimensional data distributions defined by a target training set [6–9]. Deep generative models, and most notably, generative adversarial networks (GANs), have been used to successfully synthesize highly-realistic images, audio, video, and text [10–13]. Moreover, because deep

* U.S. government work not protected by U.S. copyright.

generative models can learn unknown, unstructured high-dimensional target distributions, they represent a potentially powerful class of methods for many data analysis and signal processing problems [6–9].

Since noise is a key component of realistic signals, and given the flourishing interest in GANs in particular, it is important to ask: To what extent are GANs capable of learning various noise types? Answering this question provides insight into potential applications and limitations of GANs and related generative models.

Prior related works includes the application of GANs to image denoising [14–18], image noise adaption [19, 20], image texture synthesis [21–23], and underwater acoustic noise modeling [24]. Although the aforementioned investigations provide some evidence that GANs can learn noise and related distributions, they are limited to particular classes of noise and domain-specific applications. Further, because most prior related studies focus on images, they give little insight into time series, which are of primary interest in many domains.

The literature on GANs for time series has predominantly focused on audio applications. Many time-series GAN models leverage prior work on GANs for images by training the generator to produce an image-domain, time-frequency representation, such as a spectrogram, which is then mapped into a time series, e.g. [25–30]. Additionally, there has been some work on GANs that directly model time series, e.g. using recurrent neural networks [31, 32], or convolutional neural networks [25, 33].

In the present work, we empirically investigate the ability of general-purpose GANs for time series to learn noise modeled as a real-valued, discrete-time random process. Namely, as outlined in section 2, we examine four wide-ranging classes of noise commonly encountered in physical measurement, electronics, and communication: band-limited thermal noise, power law noise, shot noise, and impulsive noise. Within each noise class, we consider multiple random process models over a broad range of parameter values. The mathematical noise models that we consider include stationary, nonstationary, Gaussian, non-Gaussian, and long-memory random processes.

Our evaluations focus on two complementary GAN models for time series based on the popular deep convolutional GAN (DCGAN) [34] architecture: a direct time-series model, WaveGAN [25], and an image-domain model that uses a complex-valued, short-time Fourier transform (STFT) representation of the time-series [28, 30]. Details are provided in section 3. The GAN architectures were selected for their general-purpose nature, relative simplicity, and straightforward implementation. A prior investigation assessed the effectiveness of these GAN models for synthetic baseband communication signals in the presence of additive white noise and signal distortions arising from stochastic communication channels [35].

Given the extraordinary number and breadth of noise models [1–4] and GAN architectures [10–13], it is not feasible to examine all possibilities, and our investigations are necessarily limited in scope. In particular, we do not aim to comprehensively evaluate all published GAN models for time series or to propose a single GAN architecture that works optimally for all noise types. Instead, our goal is to assess the effectiveness of simple, general-purpose, convolutional GAN models for time series. Nonetheless, to our knowledge, this investigation is the most extensive appraisal of GAN performance across a wide range of noise models thus far.

Our empirical studies yield new insights into the capabilities and potential limitations of current approaches to time-series GANs and highlight areas for further research. In addition, our battery of tests provides a useful benchmark to aid future developments. Python software implementing our experiments and evaluations as well as training datasets and results are publicly available [36, 37].

2. Noise models and simulation methods

In this section, we review the classical noise models and the simulation methods used to generate target distributions for our experiments. Specific parameter choices for our synthetic noise data sets are given in section 6.

The mathematical models presented here were selected because they cover disparate, well-known noise types and because there are accurate, computationally efficient methods for simulation and parameter estimation. This set of noise models is not comprehensive, and descriptions of additional types of noise can be found in [1–4]. For simplicity, we focus on real-valued random processes.

Below, using standard notation, we denote the set of real numbers as $\mathbb{R}$ and the set of integers as $\mathbb{Z}$. All simulated time series were 4096 samples long, which provided a good balance between realism and computational complexity. Throughout, unitless quantities, e.g. time, are used.

2.1. Band-limited thermal noise

Thermal noise, also called Johnson–Nyquist noise, arises from the thermal motion of charge carriers inside an electrical conductor [1–3]. Thermal noise is commonly modeled as a zero-mean white process, i.e. a

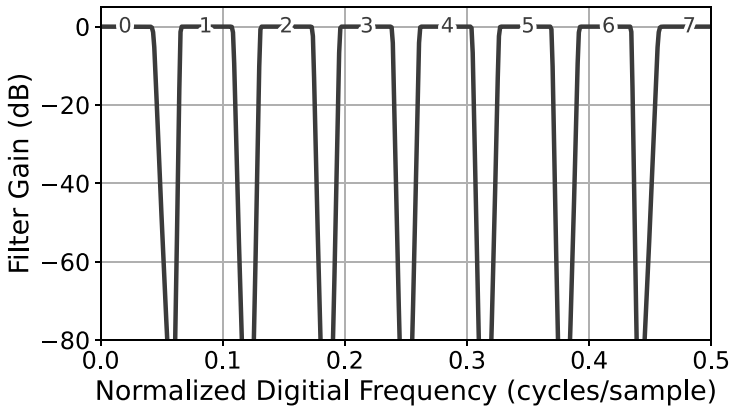

Figure 1. Frequency response of each digital filter used to simulate band-limited thermal noise, indexed 0 through 7.

sequence of independent, identically distributed (i.i.d.) random variables with zero mean and finite variance [4, 38]. In the case of radio frequency electronics, thermal noise is band-limited by system components. For this reason, band-limited (or filtered) thermal noise is of interest in many contexts [4, 39].

To simulate band-limited thermal noise, we first generated a white standard normal sequence and then filtered it with a digital bandpass filter. Specifically, we applied a 40th order digital Butterworth filter, implemented using cascaded second-order sections and zero-phase filtering [40]. Frequency responses for the eight bandpass filters used to generate our target distributions are shown in figure 1.

2.2. Power law noise

Power law noise, also called colored, fractional, or fractal noise, arises in electronics as well as a diverse array of other physical phenomena [1–3, 41]. Power law noise is characterized by a power spectral density (PSD), $S(f)$, that is proportional to a power of frequency, f, at low frequencies, i.e. $S(f) \propto |f|^{\eta}$, where η is a real number. Specific integer powers are often associated with 'colors of noise', e.g. random processes with $\eta = -1$, 0, and 1 are called pink, white, and blue noise, respectively [42, chapter 3]. When η is near -1, the process is also known as $1/f$ noise, flicker noise, or excess noise [2, 41].

We consider two well-known mathematical models for power law noise: fractional Gaussian noise (FGN) and fractional Brownian motion (FBM) [43]. FGN can be interpreted as a generalization of discrete-time white Gaussian noise, and FBM can be interpreted as a generalization of the continuous-time Brownian motion (or Wiener) process [43]. The above models arise in the study of self-similar (or fractal) processes, as well as so-called long memory (or persistent) processes [38, 44, 45].

FGN is a zero-mean, stationary, discrete-time Gaussian process, Y_t, $t \in \mathbb{Z}$, with autocovariance sequence

$$\mathrm{Cov}(Y_t, Y_{t+k}) = \frac{\sigma_Y^2}{2}\left(|k-1|^{2H} + |k+1|^{2H} - 2|k|^{2H}\right), \tag{1}$$

where $k \in \mathbb{Z}$, $\sigma_Y^2 = \mathrm{Var}[Y_t]$, and $H \in (0,1)$ is called the Hurst index [38, 43, 44]. Successive time steps of FGN are negatively correlated when $H \in (0, 0.5)$ and positively correlated when $H \in (0.5, 1)$. When $H = 0.5$, successive time steps are uncorrelated, and FGN reduces to classical white Gaussian noise. Near $f = 0$, the PSD for FGN is proportional to $|f|^{1-2H}$ [38, 43, 45].

FGN arises as the increment process of the continuous-time FBM process, $B_H(t), t \in \mathbb{R}$ [38, 43, 44]. Namely,

$$Y_t = B_H(t+1) - B_H(t), \quad t \in \mathbb{Z}. \tag{2}$$

For the rigorous definition of FBM, see, e.g. [43, 44]. FBM is a nonstationary, zero-mean Gaussian process with $B_H(0) = 0$, and in the special case $H = 0.5$, FBM reduces to a classical Brownian motion process. The PSD for the nonstationary FBM process can be given a rigorous interpretation [46], where the PSD is proportional to $|f|^{-(2H+1)}$ [38, 43, 46].

In summary, as $f \to 0$, the PSDs for FGN and FBM are proportional to $|f|^{\eta}$, where $\eta \in (-1, 1)$ for FGN and $\eta \in (-3, -1)$ for FBM, respectively, with $H \in (0, 1)$.

To synthesize discrete-time FGN and FBM, we implemented the exact approach of Perrin *et al* [47], which utilizes the fast circulant embedding method [48] to generate FGN and applies cumulative summation to obtain discrete-time FBM. All FGN simulations set $\sigma_Y^2 = 1$. Examples of synthetic FBM time series are

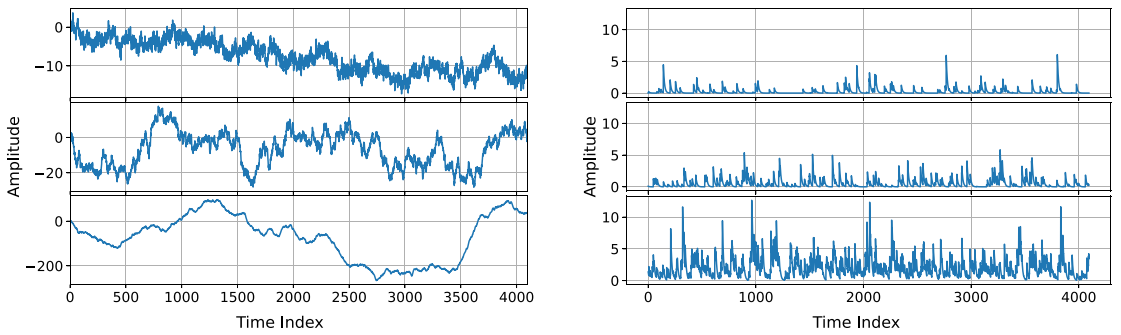

Figure 2. Left: Example fractional Brownian motion time series with $H = 0.2$, 0.5, and 0.8 from top to bottom. Right: Example shot noise time series with one-sided exponential pulse type and event rates $\nu = 0.25$, 0.5, and 2 from top to bottom.

shown in figure 2 (Left) for $H = 0.2$, 0.5, and 0.8. It can be seen that as the Hurst index increases, FBM tends to deviate further from the origin.

2.3. Shot noise

Shot noise, also called Poisson noise or photon noise, arises from the random arrival of discrete charge carriers in electronics and photons in optics [1–4]. Shot noise can be modeled using a filtered Poisson process of the form

$$X(t) = \sum_{n=1}^{N(t)} A_n p(t - \tau_n), \quad t > 0, \tag{3}$$

where $N(t)$, the number of events in the interval $(0, t]$, is a homogeneous Poisson point process with event rate ν and event times, $\{\tau_n\}$ [3, 4, 49–51]. If $N(t) = 0$, then the sum is taken to be zero. Above, $p(t)$ is a deterministic pulse function, and the pulse amplitudes, $\{A_n\}$, are independent, identically distributed, and independent of $N(t)$. For a finite time interval of length T, the number of events, N is Poisson distributed with mean νT, and the event times, $\{\tau_1, \tau_2, \ldots, \tau_N\}$, are uniformly distributed on the interval [49, p 140].

Following Theodorsen *et al* [51], we assumed that the pulse amplitudes follow an exponential distribution with mean β. We considered two pulse functions, one-sided exponential and Gaussian, taken from Howard [4, p 506]. Table 1 summarizes the pulse functions, where $u(t)$ denotes the unit step function equal to one for $t \geqslant 0$ and zero otherwise, and σ_d is a pulse duration parameter. Table 1 also lists the integrals $I_1 = \int_{-\infty}^{\infty} p(t)\,\mathrm{d}t$ and $I_2 = \int_{-\infty}^{\infty} p(t)^2\,\mathrm{d}t$ of each pulse function, which are used for the event rate estimator introduced in section 5.3.

For a finite time interval, $(0, T]$, the mean and autocovariance of the shot noise process are time-dependent, approaching steady-state values as $t, T \to \infty$ [4]. Therefore, to approximate a weak-sense stationary discrete-time shot noise process, we generated a length $2L$ process of duration $T = (2L - 1)\Delta_t \gg \sigma_d$ and then discarded the first L samples. Namely, defining a discrete-time grid $t_m = m\Delta t$, for $m = 0, 1, \ldots, 2L - 1$, we drew N from a Poisson distribution with mean νT, where $T = (2L - 1)\Delta t$. Next, we randomly generated N integers $\{m_1, m_2, \ldots, m_N\}$ from a discrete uniform distribution on $[0, 2L - 1]$ and drew $\{A_1, A_2, \ldots, A_N\}$ independently from an exponential distribution with mean β. Then, we formed the impulse sequence

$$f[m] = \sum_{n=1}^{N} A_n \delta_{m,m_n}, \tag{4}$$

where δ_{m,m_n} is a Kronecker delta function, and performed the discrete convolution of $f[m]$ with the sampled pulse function, $p(t_m)$, retaining the $2L$ samples in the middle of the convolution result. Last, we discarded the first L samples to remove any transients and approximate a steady-state realization of a length L discrete-time shot noise process. The validity of the steady-state simulated shot noise time series was verified by checking that there was close agreement between the empirical autocovariance function and the theoretical asymptotic autocovariance function [4]. For all shot noise simulations, we set $\sigma_d = 1$, $\beta = 1$, and $\Delta t = 0.1$. Example synthetic shot noise time series with a one-sided exponential pulse function and event rates $\nu = 0.25$, 0.5, and 2 are shown in figure 2 (Right).

Table 1. Pulse functions used to simulate synthetic shot noise.

Pulse type	$p(t)$	I_1	I_2
One-sided exponential	$\frac{1}{\sigma_d}\exp[-t/\sigma_d]u(t)$	1	$\frac{1}{2\sigma_d}$
Gaussian	$\frac{1}{\sigma_d\sqrt{2\pi}}\exp[-t^2/(2\sigma_d^2)]$	1	$\frac{1}{2\sigma_d\sqrt{\pi}}$

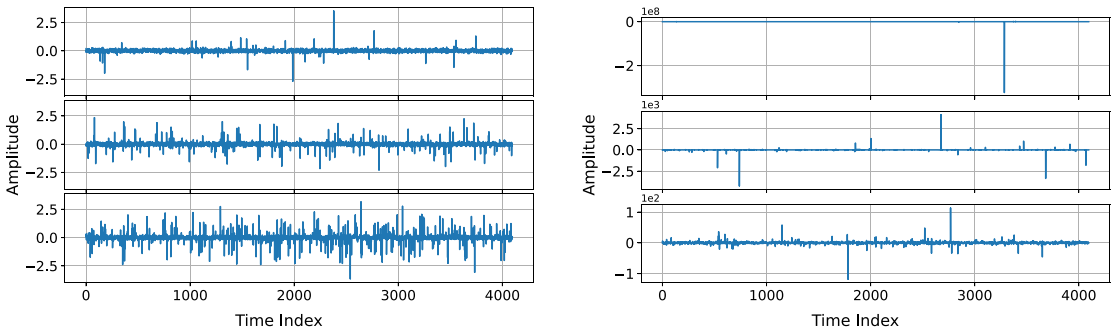

Figure 3. Example time series for impulsive noise. Left: Bernoulli–Gaussian with $\sigma_w = 0.1$ and $\sigma_i = 1$ for $p = 0.01$ (Top), $p = 0.05$ (Middle), and $p = 0.1$ (Bottom). Right: standard symmetric α-stable with $\alpha = 0.5$ (Top), $\alpha = 1$ (Middle), and $\alpha = 1.5$ (Bottom).

2.4. Impulsive noise

Impulsive noise, consisting of random, large bursts of short duration arising from either naturally occurring or man-made sources, is a limiting factor for many communication scenarios [52–55], including wireless [56, 57], digital subscriber line [58, 59], power line [60, 61], and undersea acoustic environments [62, 63]. Many models for impulsive noise have been developed; see Shongwe *et al* [64] for an overview. We focused on two well-studied impulsive noise models that were straightforward to implement and evaluate: the Bernoulli–Gaussian (BG) and symmetric alpha-stable models, described below. These models both define non-Gaussian, memoryless, white processes with a power spectrum that is constant across all frequencies. Impulse noise models with memory have also been proposed, e.g. see [61, 64], but such models are outside the scope of the present study.

A simple impulsive noise model that has been applied in many contexts is the BG model [52, 54, 55, 61], independently defined at each discrete time step as

$$X_{\mathrm{BG}} = N_w + BN_i, \tag{5}$$

where N_w and N_i are independent, zero-mean, normal random variables with variances σ_w^2 and σ_i^2, respectively, and B is a Bernoulli random variable with mean p, i.e. the probability that $B = 1$ is p, where p is called the impulse probability. Above, N_w corresponds to a thermal noise background and N_i is intermittent impulsive noise. The probability density function (PDF) for X_{BG} is the Gaussian mixture

$$f(x) = (1-p)\mathcal{N}(x;0,\sigma_w^2) + p\mathcal{N}(x;0,\sigma_w^2+\sigma_i^2) \tag{6}$$

where $\mathcal{N}(x;\mu,\sigma^2)$ denotes the PDF for a normal distribution with mean μ and variance σ^2.

We simulated independent BG noise at each time step using equation (5) with $\sigma_w = 0.1$ and $\sigma_i = 1$. Example time series are shown in figure 3 (Left) for $p = 0.01$, 0.05, and 0.1. Corresponding PDFs are plotted on a logarithmic scale on the left side of figure 4.

Another popular model for impulsive noise is the symmetric α-stable ($S\alpha S$) family of distributions, a subclass of the stable (a.k.a. Levy α-stable) family of distributions, which are used to model heavy-tailed, non-Gaussian phenomena [53, 65–68]. The PDF of a $S\alpha S$ distribution can be succinctly expressed in terms of its characteristic function as

$$f(x;\alpha,\gamma,\delta) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\left(i\delta u - \gamma|u|^{\alpha}\right) \mathrm{e}^{-iux}\,\mathrm{d}u, \tag{7}$$

where $i^2 = -1$, $\alpha \in (0,2]$ is the characteristic exponent, $\gamma > 0$ is the scale parameter, and $\delta \in \mathbb{R}$ is the location parameter[2]. A $S\alpha S$ distribution is said to be 'standard' if $\delta = 0$ and $\gamma = 1$. The special cases $\alpha = 1$

[2] Many different parameterizations for stable distributions are in use; see Nolan [65, sections 1.3, 3.5] for an overview. Here, we used the parameterization of Nikias and Shao [66].

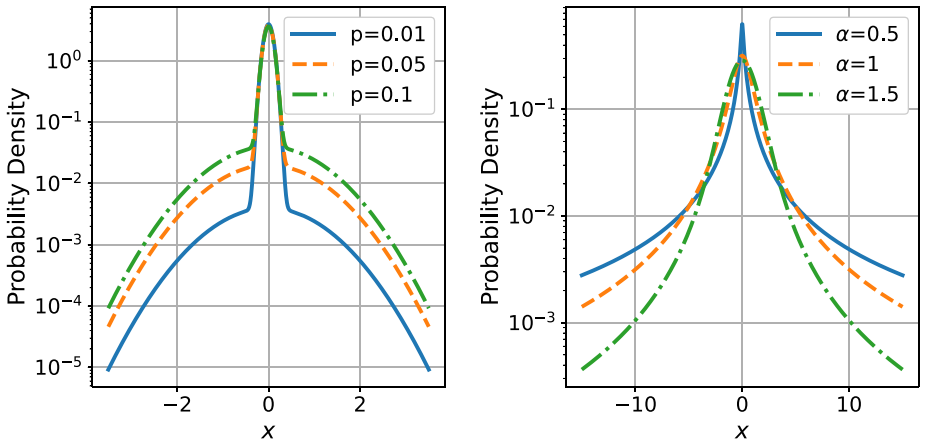

Figure 4. Example probability density functions for impulsive noise. Left: Bernoulli–Gaussian with $\sigma_w = 0.1$ and $\sigma_i = 1$. Right: standard symmetric α-stable.

and $\alpha = 2$ correspond to Cauchy and normal distributions, respectively. As α decreases, the PDF has a sharper peak and the tails become heavier [65, 66].

We considered discrete time $S\alpha S$ processes where the value at each time step is an independent standard $S\alpha S$ random variable with parameter α. To simulate standard $S\alpha S$ variates, we used the 'pylevy' Python module [69], which implements a method of Chambers *et al* [70] for generating stable random variables; see also [65]. Example time series are shown in figure 3 (Right) for three values of α. Corresponding PDFs are plotted on a logarithmic scale on the right side of figure 4.

Comparing the example time series plots in figure 3, we see that the range of BG noise is fairly consistent across different impulse probabilities, p. On the other hand, the range of $S\alpha S$ noise varies by several orders of magnitude for different values of the characteristic exponent, α. These observations are consistent with the corresponding PDFs shown in figure 4. Namely, because BG noise is a mixture of two Gaussian distributions, BG noise has rapidly decaying 'light' tails, whereas $S\alpha S$ noise has slowly decaying 'heavy' tails with a higher probability of extreme values [71].

3. GAN models

We implemented two CNN-based GAN models for our experiments that are based on the widely used DCGAN model [34]: a 1-D convolutional model trained directly on time series, called WaveGAN [25] and a 2D convolutional model trained on the complex-valued STFT. Both models were designed to generate time series of length 4096. Details on these models are given below. We start with a brief introduction to GANs.

3.1. Basic GAN theory

Since the introduction of GANs 2014, research on GANs and related deep generative modeling frameworks has developed quickly and spawned a large literature. For reviews, see [6–13].

Given a training set drawn from a high-dimensional target distribution, p_d, e.g. consisting of images or time series, the basic idea of a GAN is to train two deep neural networks, a generator network, G, and a discriminator (or critic) network, D, together dynamically. The generator generates samples from a generator distribution, p_g, where the aim is to match the target distribution. The discriminator seeks to assess the realism of generated samples, i.e. determine if samples are 'real' or 'fake.' The generator is fed a random vector, z, drawn from a specified latent distribution, e.g. multivariate uniform, which it maps to a generated sample, $G(z)$. The discriminator maps sample data, x, to $D(x)$, a real number, with larger values indicating greater confidence that the sample belongs to the target distribution.

The generator and discriminator networks are typically trained using a backpropagation implementation of stochastic gradient descent with a specified loss, or objective function [72]. Many different approaches to training GANs have been investigated to avoid failure modes such as inadequate convergence and mode collapse, where the generator output in insufficiently diverse. The GAN models that we investigated were trained with the widely-used Wasserstein GAN loss with gradient penalty [73], which seeks to minimize the Wasserstein distance between the generated distribution and the target distribution [74]. Training details for our models are provided in section 4.

Table 2. WaveGAN generator architecture.

Operation	Filter shape	Output shape
$z \sim \text{Uniform}(-1, 1)$		$(n, 100)$
Dense	(100, 4096)	$(n, 4096)$
Reshape		$(n, 1024, 4)$
ReLU		$(n, 1024, 4)$
Transpose Conv1-D (stride = 4)	(25, 1024, 512)	$(n, 512, 16)$
ReLU		$(n, 512, 16)$
Transpose Conv1-D (stride = 4)	(25, 512, 256)	$(n, 256, 64)$
ReLU		$(n, 256, 64)$
Transpose Conv1-D (stride = 4)	(25, 256, 128)	$(n, 128, 256)$
ReLU		$(n, 128, 256)$
Transpose Conv1-D (stride = 4)	(25, 128, 64)	$(n, 64, 1024)$
ReLU		$(n, 64, 1024)$
Transpose Conv1-D (stride = 4)	(25, 64, 2)	$(n, 1, 4096)$
Tanh		$(n, 1, 4096)$

Table 3. WaveGAN discriminator architecture.

Operation	Filter shape	Output shape
$x \sim G(z)$		$(n, 1, 4096)$
Conv1-D (stride = 4)	(25, 2, 64)	$(n, 64, 1024)$
LReLU($\alpha = 0.2$)		$(n, 64, 1024)$
Conv1-D (stride = 4)	(25, 64, 128)	$(n, 128, 256)$
LReLU($\alpha = 0.2$)		$(n, 128, 256)$
Conv1-D (stride = 4)	(25, 128, 256)	$(n, 256, 64)$
LReLU($\alpha = 0.2$)		$(n, 256, 64)$
Conv1-D (stride = 4)	(25, 256, 512)	$(n, 512, 16)$
LReLU($\alpha = 0.2$)		$(n, 512, 16)$
Conv1-D (stride = 4)	(25, 512, 1024)	$(n, 1024, 4)$
LReLU($\alpha = 0.2$)		$(n, 1024, 4)$
Reshape		$(n, 4096)$
Dense	(1024, 1)	$(n, 1)$

3.2. WaveGAN

WaveGAN [25] is a direct time series GAN designed for audio generation based on a 1-D flattened version of the 2D DCGAN model [34]. Tables 2 and 3 outline our implementation of the WaveGAN generator and discriminator, respectively. In these tables, Dense, Conv 1-D and Transpose Conv 1-D, denote dense fully connected layers, one-dimensional convolutional layers, and transposed convolutional layers, respectively. Also, Tanh, ReLU, and LReLU indicate hyperbolic-tangent (Tanh), rectified linear unit (ReLU), and leaky rectified linear unit (LReLU) activation functions. The filter dimensions for convolutional layers correspond to kernel length, number of input channels, and number of output channels, respectively. Similarly, the filter dimensions for the dense layers correspond to input length and output length, respectively. The first output shape dimension, n, denotes the batch size. Compared to the original WaveGAN model, which was designed to produce time series of length 16 384 our only modification was to change the dense layer to support the 4096 length of our synthetic noise waveforms.

In the discriminator, WaveGAN includes an additional operation, called 'phase shuffle,' consisting of a random circular shift on the activation output of each convolutional layer. Our implementation applied a random circular shift between −2 and 2 time steps, as recommended by Donahue *et al* [25].

3.3. Short-time fourier transform GAN (STFT-GAN)

GANs based on STFT representations have been proposed for audio generation, e.g. see [28, 30]. Here, we used a similar model, denoted STFT-GAN, based on the DCGAN architecture [34].

The discrete STFT for a real-valued time series is calculated by dividing the time series into shorter segments of equal length, multiplying by a window function, and then calculating the one-sided discrete Fourier transform on each segment [75, 76]. Unless stated otherwise, we used a Hann window of length 128 with 50% segment overlap, which for a 4096 length time series produces a (one-sided) STFT with

Table 4. STFT-GAN generator architecture.

Operation	Filter shape	Output shape
$z \sim \text{Uniform}(-1, 1)$		$(n, 100)$
Dense	(100, 4096)	$(n, 4096)$
Reshape		$(n, 1024, 2, 2)$
ReLU		$(n, 1024, 2, 2)$
Transpose Conv2-D (stride = 2)	(5, 5, 1024, 512)	$(n, 512, 4, 4)$
ReLU		$(n, 512, 4, 4)$
Transpose Conv2-D (stride = 2)	(5, 5, 512, 256)	$(n, 256, 8, 8)$
ReLU		$(n, 256, 8, 8)$
Transpose Conv2-D (stride = 2)	(5, 5, 256, 128)	$(n, 128, 16, 16)$
ReLU		$(n, 128, 16, 16)$
Transpose Conv2-D (stride = 2)	(5, 5, 128, 64)	$(n, 64, 32, 32)$
ReLU		$(n, 128, 32, 32)$
Transpose Conv2-D (stride = 2)	(5, 5, 64, 2)	$(n, 2, 65, 65)$
Tanh		$(n, 2, 65, 65)$

Table 5. STFT-GAN discriminator architecture.

Operation	Filter size	Output shape
$x \sim G(z)$		$(n, 2, 65, 65)$
Conv2-D (stride = 2)	(5, 5, 2, 64)	$(n, 64, 32, 32)$
LReLU($\alpha = 0.2$)		$(n, 64, 32, 32)$
Conv2-D (stride = 2)	(5, 5, 64, 128)	$(n, 128, 16, 16)$
LReLU($\alpha = 0.2$)		$(n, 128, 16, 16)$
Conv2-D (stride = 2)	(5, 5, 128, 256)	$(n, 256, 8, 8)$
LReLU($\alpha = 0.2$)		$(n, 256, 8, 8)$
Conv2-D (stride = 2)	(5, 5, 256, 512)	$(n, 512, 4, 4)$
LReLU($\alpha = 0.2$)		$(n, 512, 4, 4)$
Conv2-D (stride = 2)	(5, 5, 512, 1024)	$(n, 1024, 2, 2)$
LReLU($\alpha = 0.2$)		$(n, 1024, 2, 2)$
Reshape		$(n, 4096)$
Dense	(4096, 1)	$(n, 1)$

dimensions of 65×65. In this case, the constant-overlap-add constraint is satisfied, and the STFT can be inverted to obtain a time series of the original length [75].

Tables 4 and 5 outline the architectures for the STFT-GAN generator and discriminator, respectively, which are composed of five 2-D convolutional layers with 5×5 kernels. The notation in the tables is similar to that used previously, with Conv 2D and Transpose Conv 2D indicating two-dimensional convolutional and transposed convolutional layers, and n denoting the batch size. Because the discrete STFT is complex-valued, each STFT entry requires two channels, corresponding to the real and imaginary parts, respectively.

4. Training and implementation

4.1. Baseline implementation

Following the original WaveGAN training implementation [25], both models were trained using Wasserstein GAN loss with gradient penalty [73] and the Adam optimizer [77]. According to the implementation recommendations of Gulrajani *et al* [73], the gradient penalty weight was set equal to 10. WaveGAN used hyperparameter settings of $\alpha = 10^{-4}$, $\beta_1 = 0.5$, and $\beta_2 = 0.9$ for the learning rate and moment decay rates, respectively [25]. STFT-GAN differed by setting $\beta_1 = 0$, as recommended by Gulrajani *et al* [73].

Consistent with the original Wasserstein GAN implementation [74], WaveGAN was trained with an imbalanced discriminator-generator update rule, where the discriminator weights were updated five times for each generator update. In contrast, STFT-GAN was trained with a balanced discriminator-generator update rule, where the discriminator weights were updated once for each generator update. The balanced update rule for STFT-GAN was selected based on limited tests carried out for a prior study [35], where we found that balanced updates yielded improved convergence for STFT-GAN.

Each model was trained with a target data set of size $2^{14} = 16\,384$, for 500 epochs with a batch size of 128. These parameter values were found to be sufficient for convergent training across all experiments. The data

accompanying this paper [37] include GAN training history files as well as plots of GAN loss and discriminator output during training.

Prior to training, target distribution training sets were scaled using feature min-max scaling, which scales each feature, i.e. time sample or pixel value[3], to the interval $[-1, 1]$, the range of the hyperbolic tangent output activation of the generator. Specifically, minimum and maximum values of each feature were estimated over the training set of size 16 384. Because the generator's output activation is a hyperbolic tangent function, the raw generated data was in the range $[-1, 1]$. Raw generated data was rescaled using the inverse feature min-max transformation with the minimum and maximum values estimated from the training set. Therefore, the range of generated data was restricted to the range of the training dataset.

4.2. Quantile data transformation for impulsive noise

As we will see later, the impulsive noise types were particularly challenging for our baseline GAN models. Consequently, for the impulsive noise types, we also investigated replacing the feature min-max scaling of the target data with a quantile transformation [78, section 7.4.1] applied independently to each channel to make the data approximately follow a standard normal distribution. The motivations for this transformation were twofold: (1) it ensured that the distribution of each channel was unimodal with 'light' tails [71], and (2) it effectively limited the impact of outliers.

We implemented the quantile transformation using the 'quantile_transform' method in the scikit-learn Python library [79]. This method is based on the formula $Y = F_Y^{-1}(F_X(X))$, where X is an input random variable with continuous cumulative distribution function (CDF) $F_X(x)$, and Y is an output random variable with desired continuous CDF $F_Y(y)$. In our case, $F_Y(y)$ is the CDF for a standard normal distribution. The transformation formula follows from the fact that the random variable $F_X(X)$ has a uniform distribution on the interval $[0, 1]$ [78, section 7.4.1]. In practice, to apply this method to a sample of X, F_X is replaced by the empirical CDF.

The quantile transformation for a given training set was estimated using 1024 uniformly-spaced quantiles for each target distribution channel. Any data values exceeding the most extreme quantiles were clipped to those values. For WaveGAN, the transformation was fit directly to the time series values, whereas for STFT-GAN, the transformation was fit on the real and imaginary channels of the target STFT distribution separately. For both models, a scaled-tanh activation was used at the end of the generator to limit the absolute-maximum value of generated data to the absolute maximum of the target quantile-transformed distribution. Finally, the inverse quantile transformation was applied to each channel of the generated data to return it to the original range.

While the quantile transformation method is included in the commonly used scikit-learn Python library, to our knowledge, it has not been previously examined as a preprocessing step for GAN training.

5. Evaluation methods

Performance evaluation of generative models, and GANs in particular, is a difficult problem and an active research area. Recent developments are summarized in two review papers by Borji [80, 81]. Two important aspects of generative model quality are fidelity, i.e. the degree of realism in generated samples, and diversity, i.e. how well generated samples capture the full range of variation of the target distribution [80, 82].

We assessed fidelity and diversity using general-purpose metrics introduced by Naeem *et al* [82], described below. In addition, we further evaluated generative fidelity in terms of median PSD and characteristic parameters for each noise type. Evaluations for each noise type were conducted using test sets of size 4096 from the target and generated time series distributions. In particular, the target distribution test sets were synthesized independently from the training sets.

5.1. Density and coverage metrics

In an effort to address shortcomings of other evaluation measures, Naeem *et al* [82] proposed general-purpose metrics named *density* and *coverage* to assess generative model fidelity and diversity, respectively.

Suppose that a suitable distance measure for the data is identified, and denote test samples of real (target) data as $X_1, X_2, \ldots, X_N$ and fake (generated) data as $Y_1, Y_2, \ldots, Y_M$. For a given real data sample, X_i, let $\mathrm{NND}_k(X_i)$ be the distance from X_i to the kth nearest neighbor among the real data sample excluding itself,

[3] The real and imaginary channels of complex-valued STFT pixels were scaled separately.

and let $B(x, r)$ denote the ball centered at x with radius r. Also, let $\mathcal{I}[S]$ be the indicator function that equals one if the proposition S is true and zero otherwise.

For a given fake sample, Y_j, Naeem *et al* [82] define density as the expected number of real sample neighborhoods that contain Y_j divided by the expected number of such neighborhoods when the target and generated distributions are the same. Namely, for a given test sample, Naeem *et al* propose the estimator[4]

$$\widehat{\text{density}} = \frac{1}{kM} \sum_{j=1}^{M} \sum_{i=1}^{N} \mathcal{I}\left[Y_j \in B(X_i, \text{NND}_k(X_i))\right], \tag{8}$$

where division by kM ensures that $E[\widehat{\text{density}}] = 1$ when the real and fake distributions are the same [82, Lemma 1]. Note that while density is always greater than or equal to zero, it may be greater than one, depending on the density of real data around the fake data. Density values close to one indicate excellent generative model fidelity. On the other hand, values near zero indicate poor fidelity. Naeem *et al* [82] do not comment on how to interpret density values much larger than one, so additional assessments of generative fidelity are likely needed in that circumstance.

To evaluate generative diversity, Naeem *et al* [82] define coverage as the fraction of real samples whose neighborhoods contain at least one fake sample. For a given test sample, Naeem *et al* estimate coverage as

$$\widehat{\text{coverage}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}\left[\exists j \text{ s.t. } Y_j \in B(X_i, \text{NND}_k(X_i))\right]. \tag{9}$$

Because coverage is essentially the probability that a real sample is 'close' to a fake sample, it is bounded between zero and one. Coverage values close to one indicate good generative diversity, i.e. generated samples cover the full support of the target distribution. Conversely, coverage values near zero imply poor generative diversity, which may result from mode collapse (a.k.a. mode dropping), as demonstrated by Naeem *et al* [82].

Under the condition that the real and fake distributions are identical, Naeem *et al* show that [82, lemma 2]

$$E[\widehat{\text{coverage}}] = 1 - \frac{(N-1)\cdots(N-k)}{(M+N-1)\cdots(M+N-k)}. \tag{10}$$

Moreover, they propose that the hyperparameter, k, should be selected to ensure that the expected value of the coverage estimator is close to one when the real and fake distributions are the same. In our evaluations, we used test sets of size $M = N = 4096$ and implemented the above density and coverage estimators with $k = 10$, implying that $E[\widehat{\text{coverage}}] \approx 0.999$ when the target and generated distributions are identical.

Implementation of the above density and coverage metrics requires defining a suitable measure of distance between data points. We chose to use a normalized version of dynamic time warping (DTW) distance [83, 84], a widely-used, general-purpose distance measure for time series that has been shown to be highly effective for time series indexing, classification, and clustering [84–87]. Technically, DTW distance is not a metric [86, 88], but it does satisfy the requirements of a dissimilarity measure [89, 90] and is *nearly* a metric in many circumstances [86, 88, 91].[5]

DTW distances between time series were calculated using the 'fast' methods from the `dtaidistance` python package [92], setting the maximal warping window size to 32. The warping window size parameter was selected based on computational feasibility considerations and limited preliminary experiments. To obtain a robust distance measure that was insensitive to data scaling, each time series was first normalized by its maximum absolute value prior to DTW estimation. Estimated DTW distances were then normalized by the window size to ensure values between zero and one. Normalized DTW distances were computed between each target time series as well as between each target and generated time series in the test sets of size 4096. Subsequently, density and coverage were estimated using equations (8) and (9), respectively, with $k = 10$.

Approximate 95% confidence intervals for the density metric were estimated using the percentile bootstrap method [93], where bootstrap resampling with replacement was performed over the generated test sample 10 000 times. Preliminary experiments indicated that additionally bootstrapping over the target distribution sample resulted in bootstrap estimates that were uniformly lower than the original point

[4] We use 'hat' notation to distinguish point estimators from the quantity being estimated.

[5] It is unclear if the non-metricity of DTW distance impacts the effectiveness of the density and coverage metrics, and further study may be necessary to clarify this issue.

estimate, so bootstrap resampling was therefore restricted to the generated sample only. Approximate 95% confidence intervals for the coverage metric were estimated using the classical Wilson score method for a binomial proportion [94].

To our knowledge, the combination of the density and coverage metrics above with DTW distance for time series, as well as the procedures for confidence intervals, have not been previously proposed and are novel.

5.2. Power spectral density

The median PSD for each test set was estimated with the multitaper method, a versatile nonparametric approach [95, 96]. Specifically, we used the implementation in the Python 'Spectrum' package [97] with the time half-bandwidth parameter set to $NW = 4$, the first $k = 7$ Slepian sequences, the FFT length set to 4096, and the fast 'eigen' method for result weighting. These parameter choices are typical and were found to yield consistent results. After applying the multitaper method to estimate the PSD for each time series in the test set, we calculated the median value in each frequency bin. The uncertainties in the median PSD estimate across the test set were negligible in the context of our evaluations. This procedure was carried out on both target and generated distributions across all noise types.

Denote the one-sided median PSDs for the target and generated distributions as $P_t(f_d)$ and $P_g(f_d)$, respectively, where $f_d \in [0, 0.5]$ is normalized digital frequency with units of cycles per sample. To evaluate the faithfulness of P_g relative to P_t, we used a one-sided version of Georgiou's 'geodesic distance' for power spectra [98], defined as[6]

$$d_g(P_g, P_t) = \sqrt{\int_0^{0.5} \left(\log \frac{P_g(f_d)}{P_t(f_d)}\right)^2 \frac{df_d}{0.5} - \left(\int_0^{0.5} \log \frac{P_g(f_d)}{P_t(f_d)} \frac{df_d}{0.5}\right)^2}. \tag{11}$$

In our evaluations, we used a natural logarithm, but the choice of logarithm is arbitrary. The geodesic distance can be interpreted as the length of a geodesic connecting points on a manifold of PSDs [99]. Technically, d_g is a pseudo-metric, because it is insensitive to scaling, i.e. $d_g(P_g, P_t) = d_g(P_g, \kappa P_t)$ for any $\kappa > 0$ [98]. Because the first term depends on the difference of log-transformed power spectra, it reflects differences in areas of both low and high PSD. We estimated the geodesic PSD distance by approximating the above formula on a discrete frequency grid.

5.3. Noise model parameters

For each noise type, except for band-limited thermal noise, we assessed how well the generated time series distribution matched the target distribution in terms of characteristic noise parameters. Later, boxplots are used to compare distributions of estimated noise parameters for target and generated time series distributions. Boxplots of parameter estimates for target distributions with known ground truth characterize the inherent bias and variability of the estimators and hence provide a basis for assessing generated data.

For power law noise distributions, we evaluated the accuracy of the the Hurst index, H, using the well-studied 'discrete variations' method [100–102] implemented with a second-order difference filter.

Under the assumption that the shot noise pulse amplitudes follow an exponential distribution, which is true for our target distributions, we assessed the shot noise event rate, ν, using the (apparently novel) estimator

$$\hat{\nu} = \frac{2\hat{\mu}_X^2 I_2}{\hat{\sigma}_X^2 I_1^2}, \tag{12}$$

where $\hat{\mu}_X$ and $\hat{\sigma}_X^2$ are the estimated mean and variance of the shot noise time series, and where $I_1 = \int_{-\infty}^{\infty} p(t)\,dt$ and $I_2 = \int_{-\infty}^{\infty} p(t)^2\,dt$ are integrals of the known pulse function, $p(t)$; see table 1. A derivation is given in the appendix.

For each of the impulsive noise models, we evaluated two characteristic parameters. Namely, for BG noise, we assessed the impulse probability, p, and the scale parameter ratio, $\theta = \sqrt{\sigma_w^2 + \sigma_i^2}/\sigma_w$, which measures the relative dispersion of the mixture components; see equation (6). The BG parameters were estimated by fitting a two-component Gaussian mixture model using the iterative expectation maximization method implemented in the scikit-learn Python library [79]. To assess $S\alpha S$ noise, we estimated the characteristic exponent, α, and the scale parameter, γ, using the 'fast' methods of Tsihrintzis and Nikias [103].

[6] Consistent with Georgiou's definition, the integrals were normalized by the length of the integration interval.

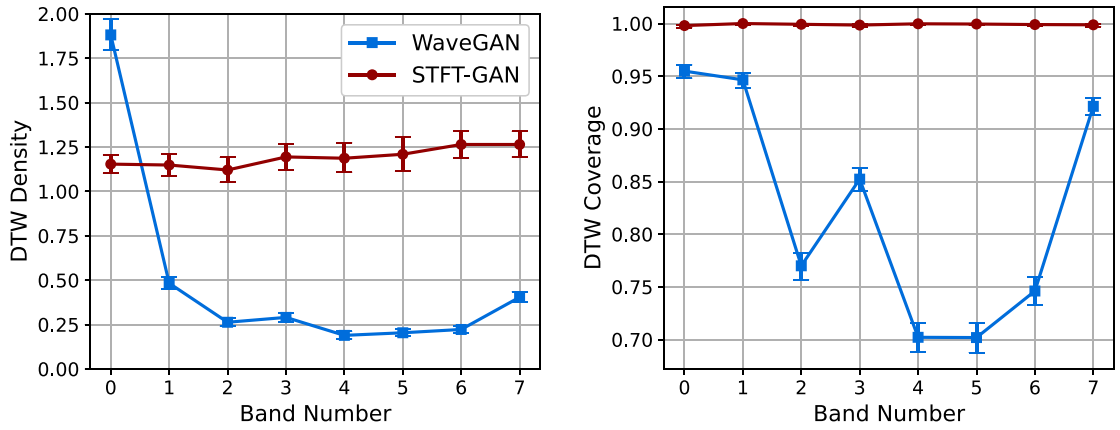

Figure 5. DTW density and coverage results for band-limited thermal noise.

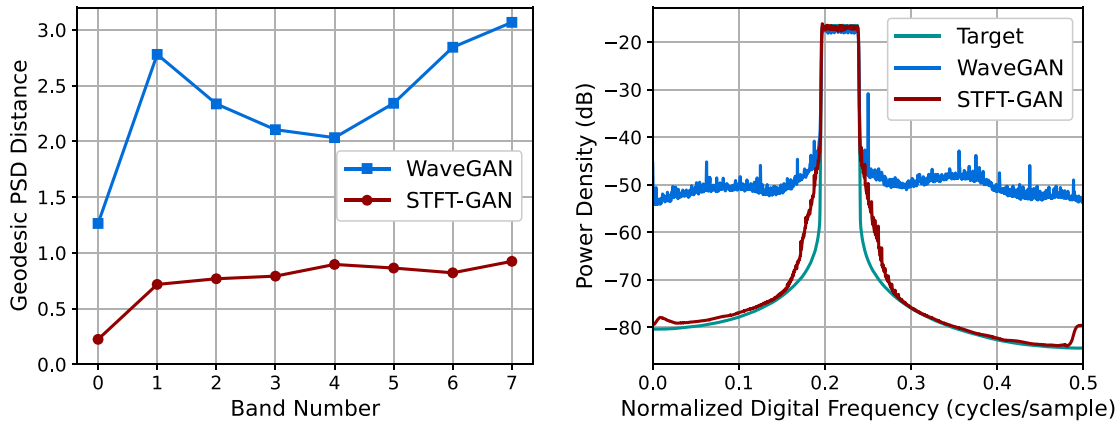

Figure 6. PSD results for band-limited thermal noise. Left: geodesic PSD distance. Right: median PSD comparison for band #3.

6. Experimental results

6.1. Band-limited thermal noise

The eight digital bandpass filters shown in figure 1 were used to simulate eight target data sets of band-limited thermal noise, where each data set contained noise limited to a single band. DTW density and coverage results are plotted in figure 5 and estimated geodesic PSD distance is plotted in figure 6 (Left), where the bands are ordered in terms of increasing center frequency. It is evident that STFT-GAN yielded uniformly better density, coverage, and PSD fidelity than WaveGAN. Median estimated PSDs for band number 3 are shown in figure 6 (Right); other bands are similar. We see that STFT-GAN more closely tracked the target PSD out-of-band, whereas WaveGAN suffered from a limited dynamic range.

6.2. Power law noise

The power law noise models from section 2.2 were evaluated for target Hurst indices of $H = 0.05$, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 0.95. DTW density and coverage results are shown in figure 7. PSD distance results and boxplots of estimated Hurst indices are given in figure 8. For FGN, STFT-GAN performed well, achieving near-ideal density and coverage as well as excellent median PSD fidelity and Hurst indices. On the other hand, WaveGAN generally performed poorly on the density and coverage metrics, except for $H = 0.95$, and also exhibited inferior median PSD fidelity.

For FBM, in addition to the baseline 65×65 STFT size, we also tested an STFT dimension of 129×65, resulting from a window segment length of 256 with 75% overlap. We denote the baseline and modified models as STFT-GAN (65×65) and STFT-GAN (129×65), respectively. Examining DTW density and coverage results, performance was generally excellent except for the largest Hurst indices of 0.9 and 0.95,

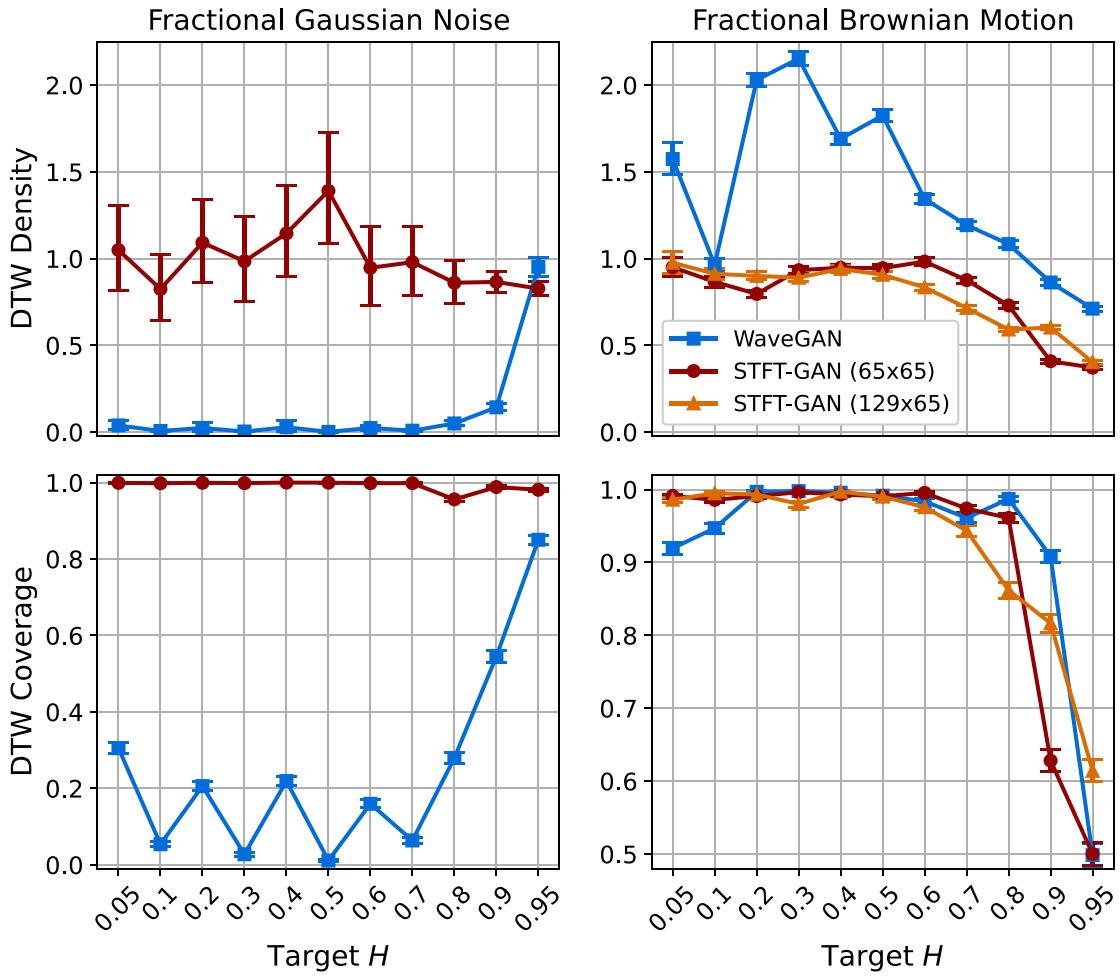

Figure 7. DTW density and coverage results for power law noise,.

where all models exhibited a drop-off in DTW coverage, indicating inadequate sample diversity. In terms of median PSD distance and estimated Hurst indices, STFT-GAN (129×65), which had higher frequency resolution than the baseline model, achieved superior PSD and Hurst index fidelity over the full parameter range. Figure 9 compares the median PSDs for the $H = 0.9$ case, illustrating better PSD accuracy for STFT-GAN (129×65) at low frequencies.

Example target and generated time series for FBM with $H = 0.5$, which corresponds to the classical Brownian motion process, are plotted in figure 10. Qualitatively, the example generated time series are consistent with a Brownian motion process.

6.3. Shot noise

Target distributions defined by the shot noise model described in section 2.3 were assessed with the two pulse types in table 1 for event rates $\nu = 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75$, and 3.0. DTW density and coverage results are shown in figure 11. PSD distance results and estimated event rate boxplots are given in figure 12.

For shot noise with the one-sided exponential pulse type, WaveGAN exhibited very good DTW density and coverage, while STFT-GAN did poorly on those metrics. Both models had excellent median PSD fidelity and similar event rate performance.

On the other hand, for shot noise with the smoother Gaussian pulse type, STFT-GAN performed better overall than WaveGAN, with STFT-GAN exhibiting excellent DTW density and coverage values near one, and superior PSD distance results. Although WaveGAN also had excellent DTW coverage, slightly better than STFT-GAN, it achieved worse fidelity as measured by PSD distance.

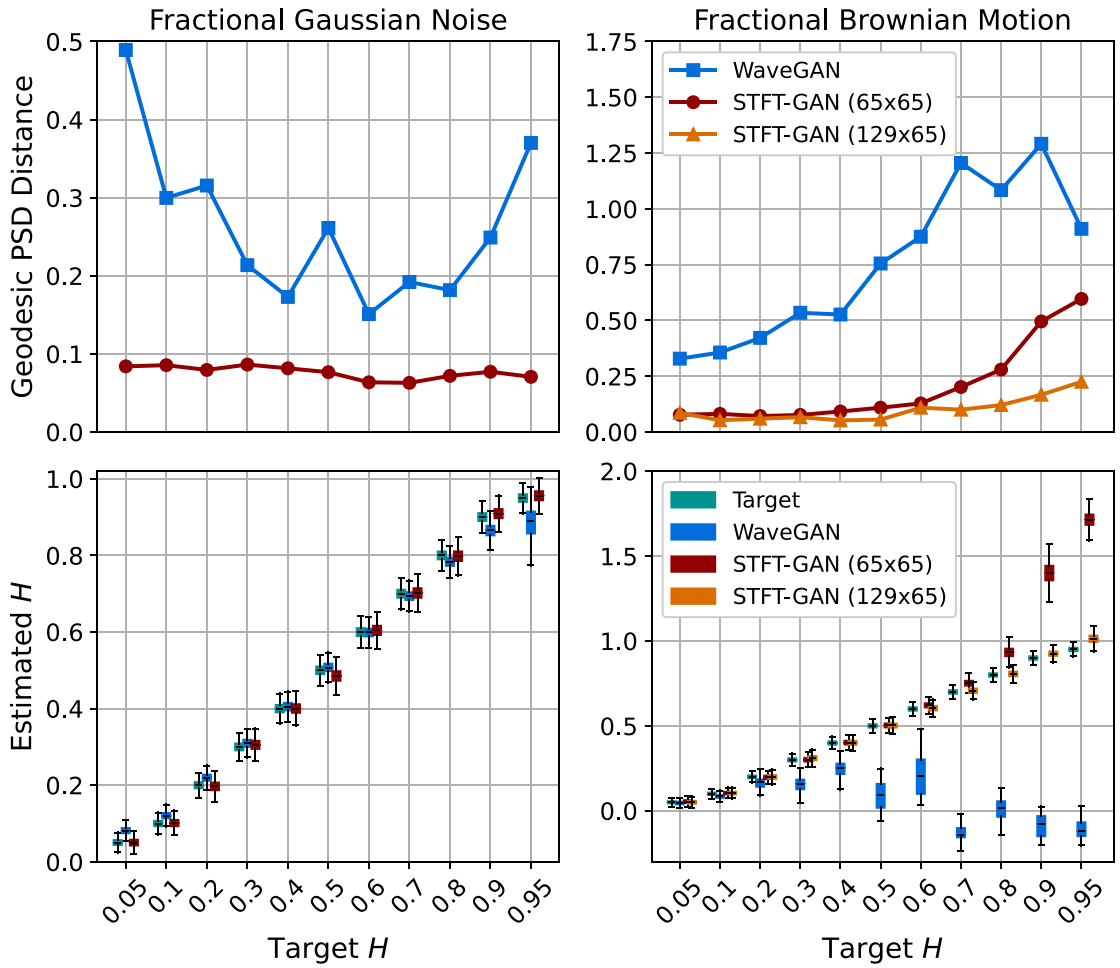

Figure 8. Additional fidelity evaluations for power law noise. Top: geodesic PSD distance plots. Bottom: boxplots of estimated Hurst index.

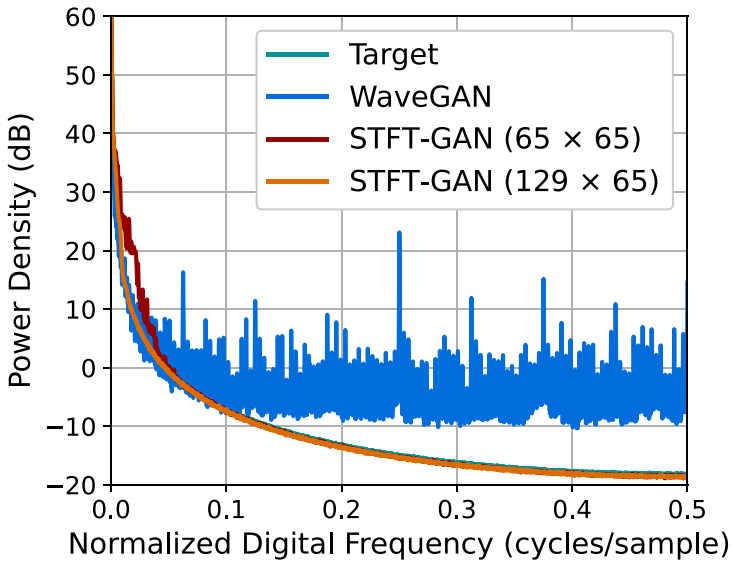

Figure 9. Median PSD comparisons for FBM with $H = 0.9$.

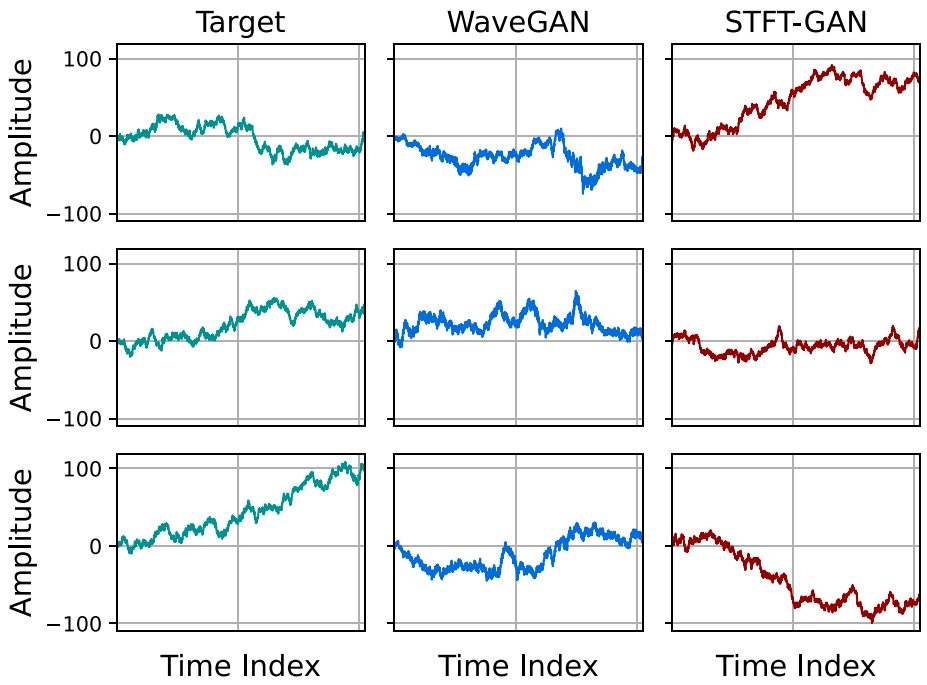

Figure 10. Examples of target and generated time series for FBM with $H = 0.5$. The STFT-GAN results are with an STFT dimension of 129×65.

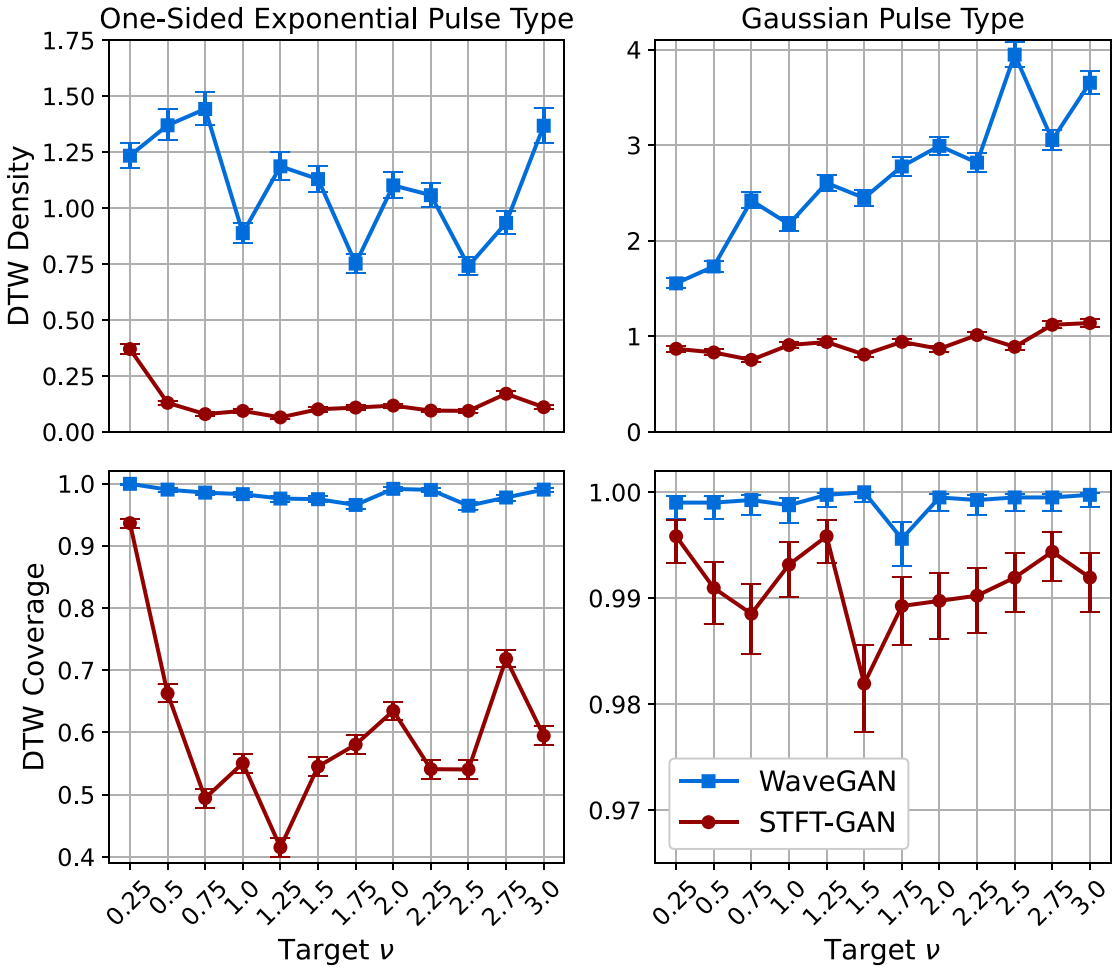

Figure 11. DTW density and coverage results for shot noise.

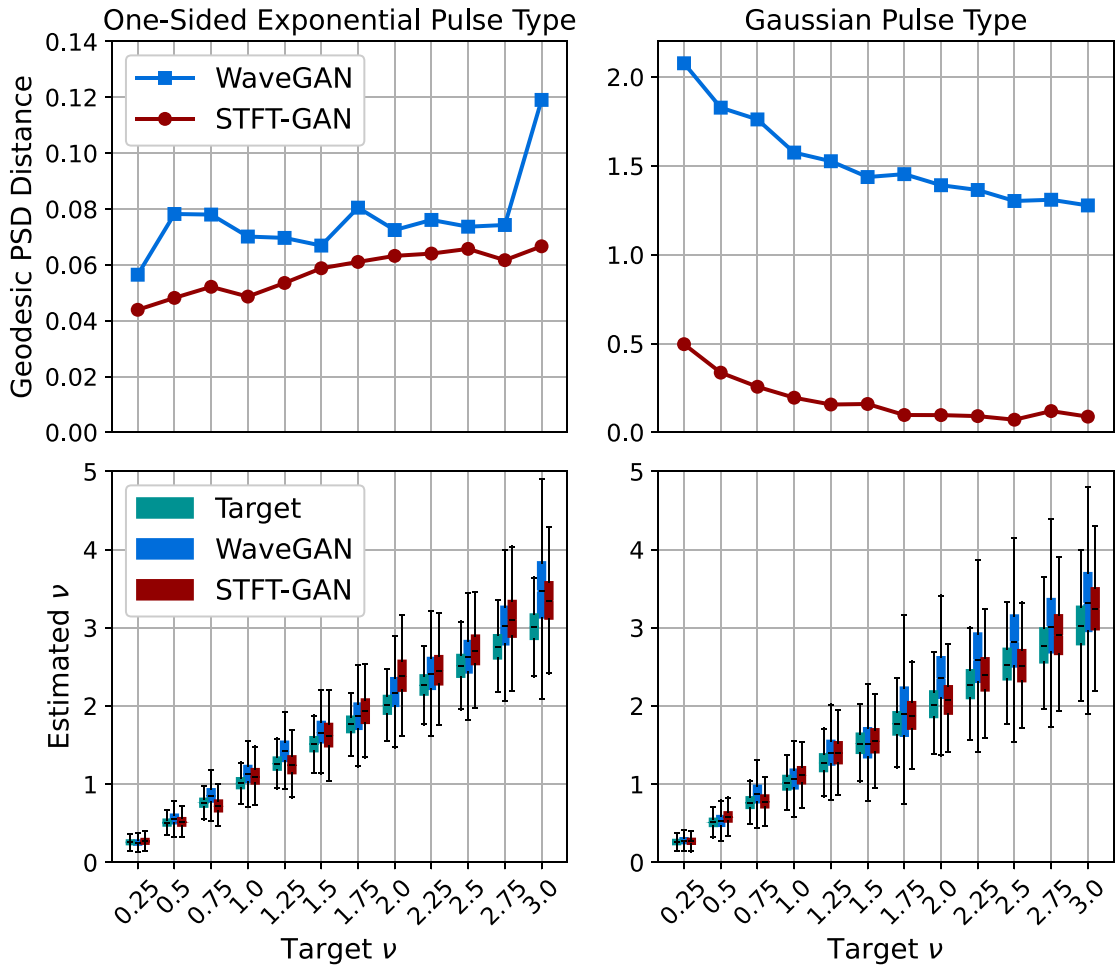

Figure 12. Additional fidelity evaluations for shot noise. Top: geodesic PSD distance plots. Bottom: estimated event rate boxplots.

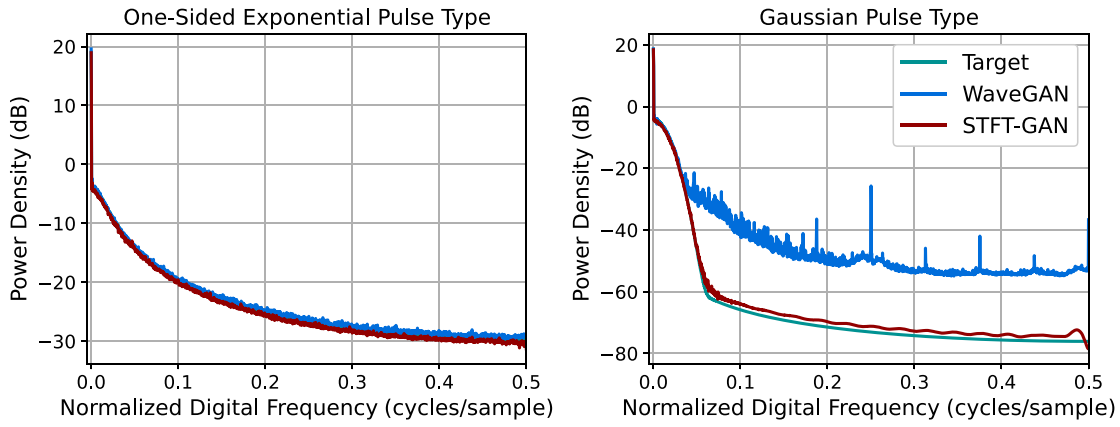

Figure 13. Median PSDs for Shot Noise with event rate $\nu = 1$.

Figure 13 compares median PSDs for the two pulse types when the target event rate is $\nu = 1$. These plots illustrate the inability of WaveGAN to recover the larger PSD dynamic range for the Gaussian pulse type.

Representative example target and generated time series for one-sided exponential shot noise for a target event rate of $\nu = 0.25$ are plotted in figure 14. From this figure, it can be seen that WaveGAN correctly learned to generate non-negative shot noise time series, while STFT-GAN generated time series with occasional small negative values.

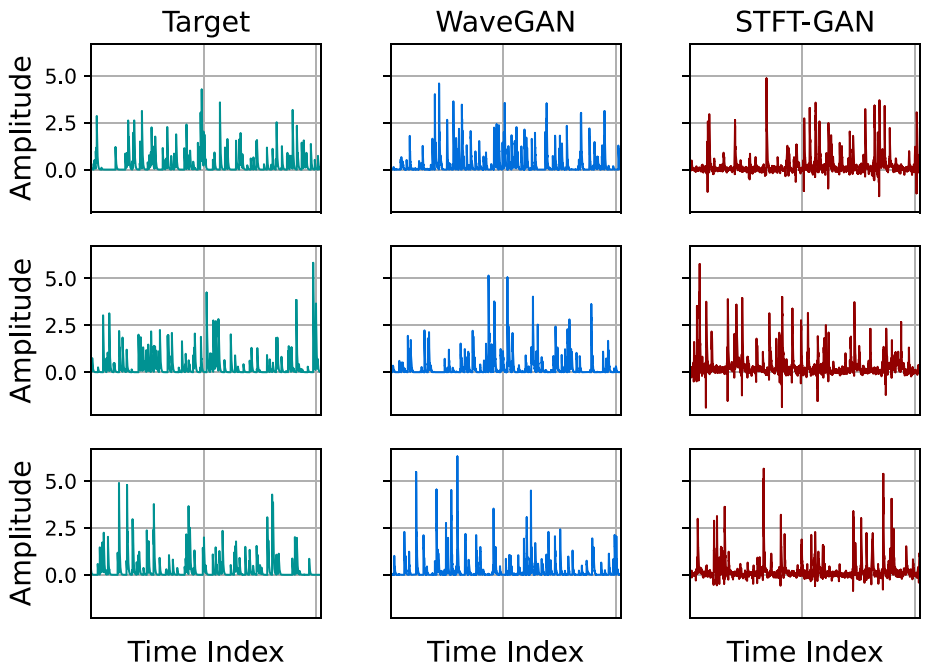

Figure 14. Examples of time series for one-sided exponential shot noise with target event rate $\nu = 0.25$.

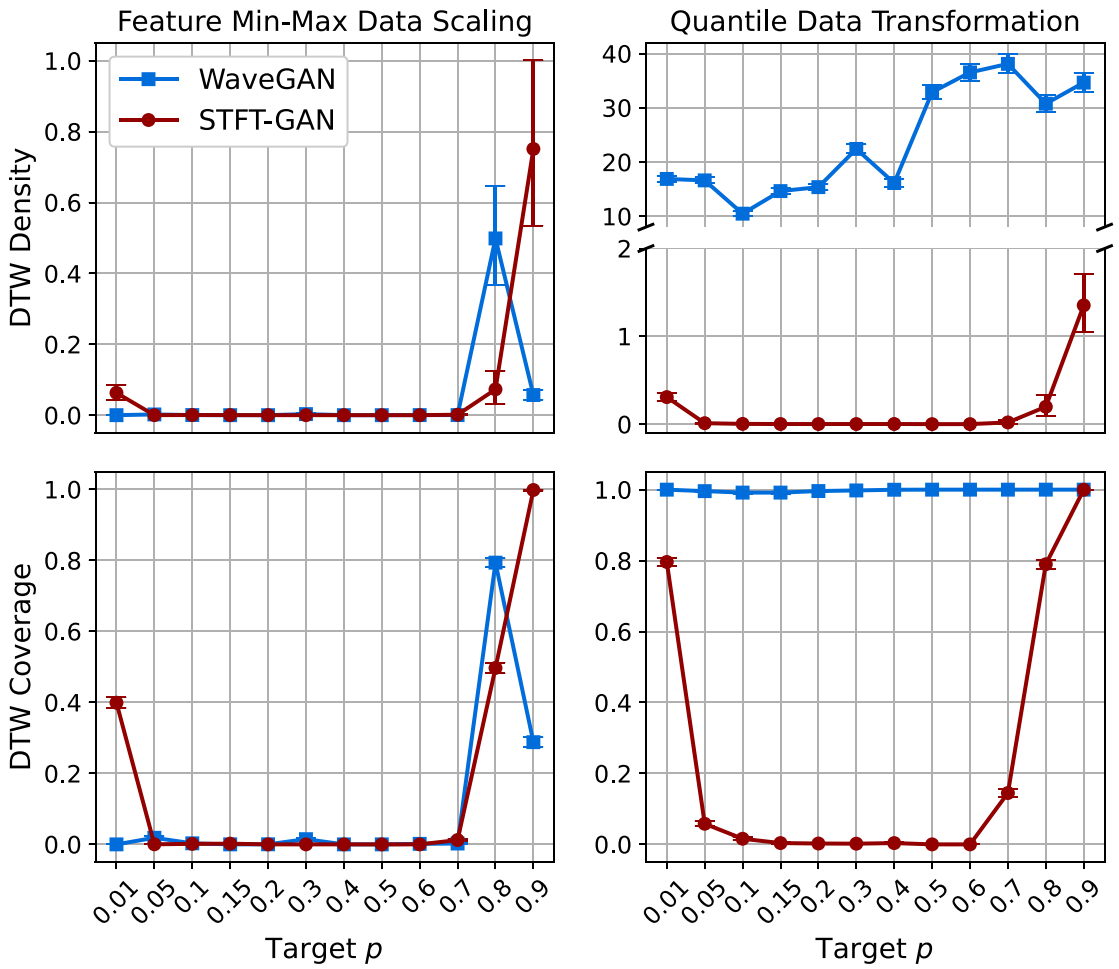

Figure 15. DTW density and coverage results for Bernoulli-Gaussian noise. Note that a broken y-axis is used in the upper right plot to display results for both models.

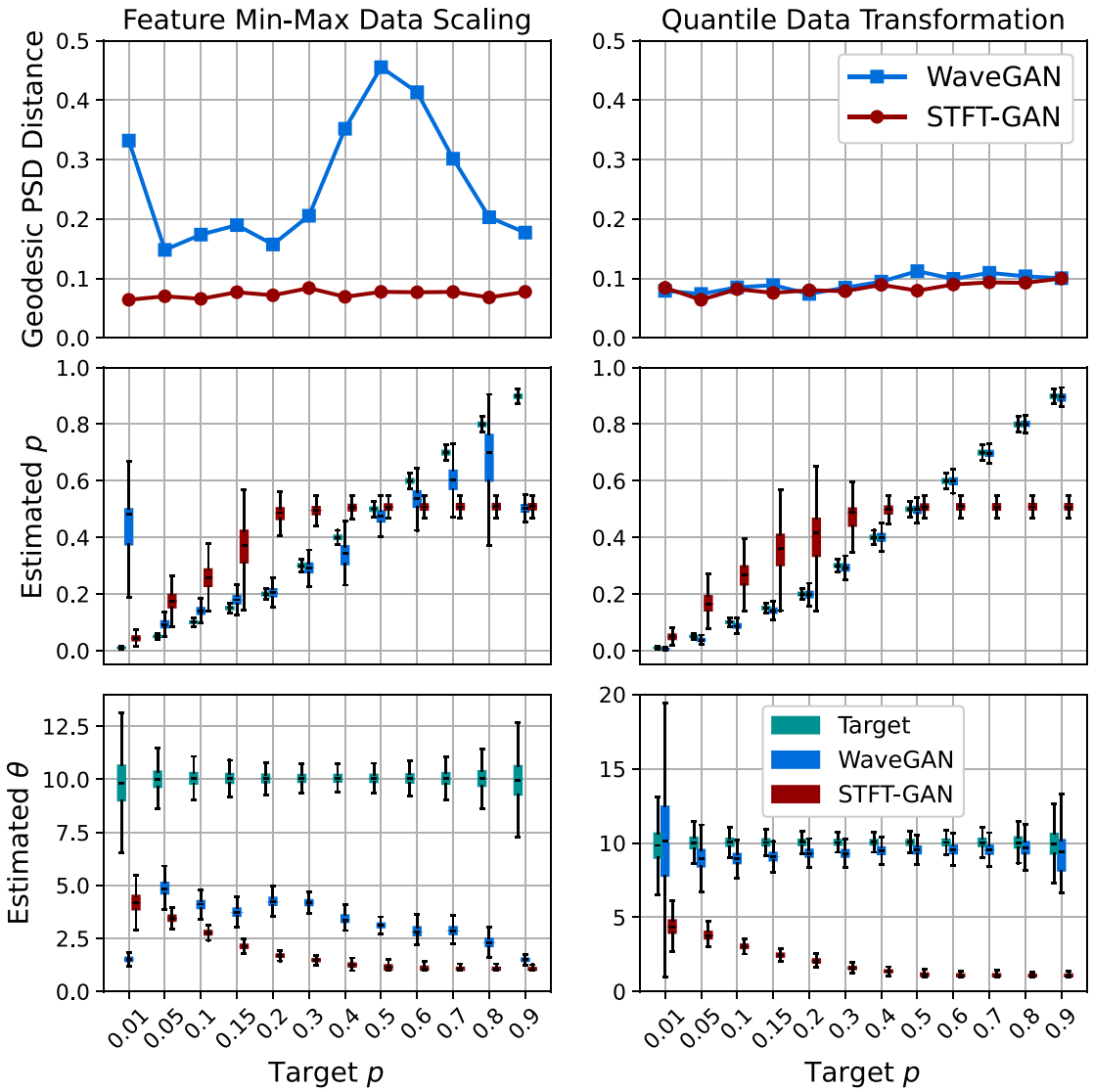

Figure 16. Additional fidelity evaluations for Bernoulli-Gaussian noise. Top: geodesic PSD distance plots. Middle: boxplots of estimated impulse probability. Bottom: boxplots of estimated scale parameter ratio.

6.4. Impulsive noise

Last, the ability of the two GAN models to learn impulsive noise defined by the BG and $S\alpha S$ models described in section 2.4 was evaluated. Specifically, BG noise with $\sigma_w = 0.1$ and $\sigma_i = 1$, i.e. a scale parameter ratio of $\theta = \sqrt{\sigma_w^2 + \sigma_i^2}/\sigma_w \approx 10.05$, was assessed for impulse probabilities of $p = 0.01$, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Standard $S\alpha S$ noise with location and scale parameters equal to zero and one, respectively, was evaluated for characteristic exponents $\alpha = 0.5$, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, and 1.5. As described in section 4, the WaveGAN and STFT-GAN models were trained with the two different preprocessing schemes described in section 4: (1) a baseline implementation using feature min-max scaling and (2) an implementation applying a quantile data transformation, which transforms each channel to an approximate standard normal distribution.

Figures 15 and 16 present the aggregate results for BG noise. Both GAN models performed poorly with feature min–max scaling, as seen from the DTW density and coverage plots as well as the estimated impulse probability and scale parameter boxplots. In particular, DTW coverage indicates that both models experienced partial or full mode collapse for all but the largest target impulse probabilities, p.

WaveGAN clearly improved with the quantile data transformation, exhibiting excellent DTW coverage, although the DTW density metric was abnormally large[7]. Also, WaveGAN accurately recovered the target impulse probability and scale ratio across most scenarios, except for the extreme $p = 0.01$ case, where the

[7] Additional investigations are required to understand the implications of density values much larger than one. Naeem *et al* [82] do not comment on how to interpret this outcome.

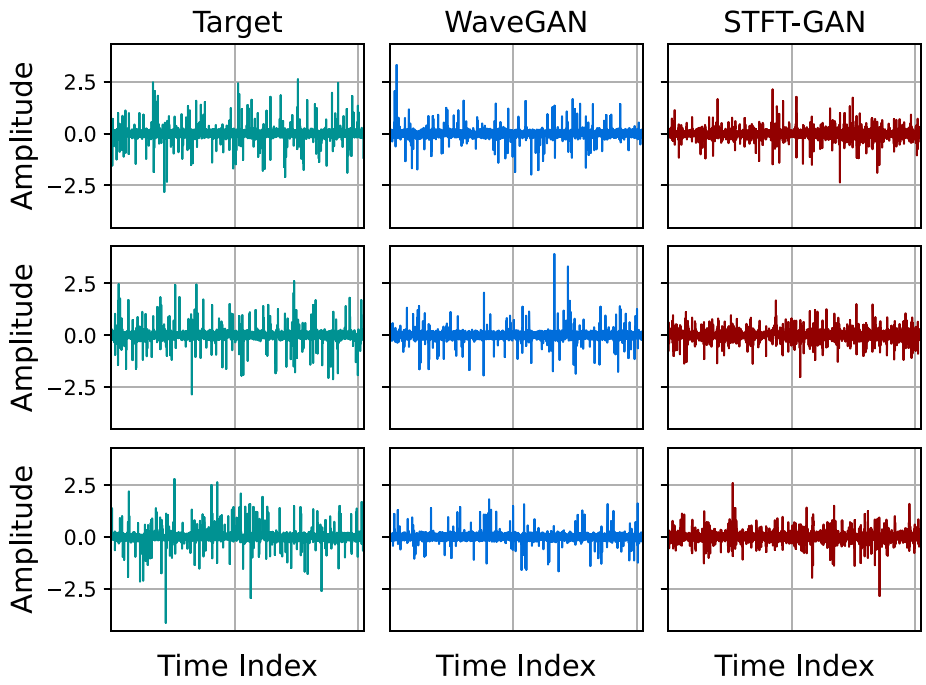

Figure 17. Example time series for GANs trained on Bernoulli-Gaussian noise with quantile data transformation and target impulse probability, $p = 0.05$.

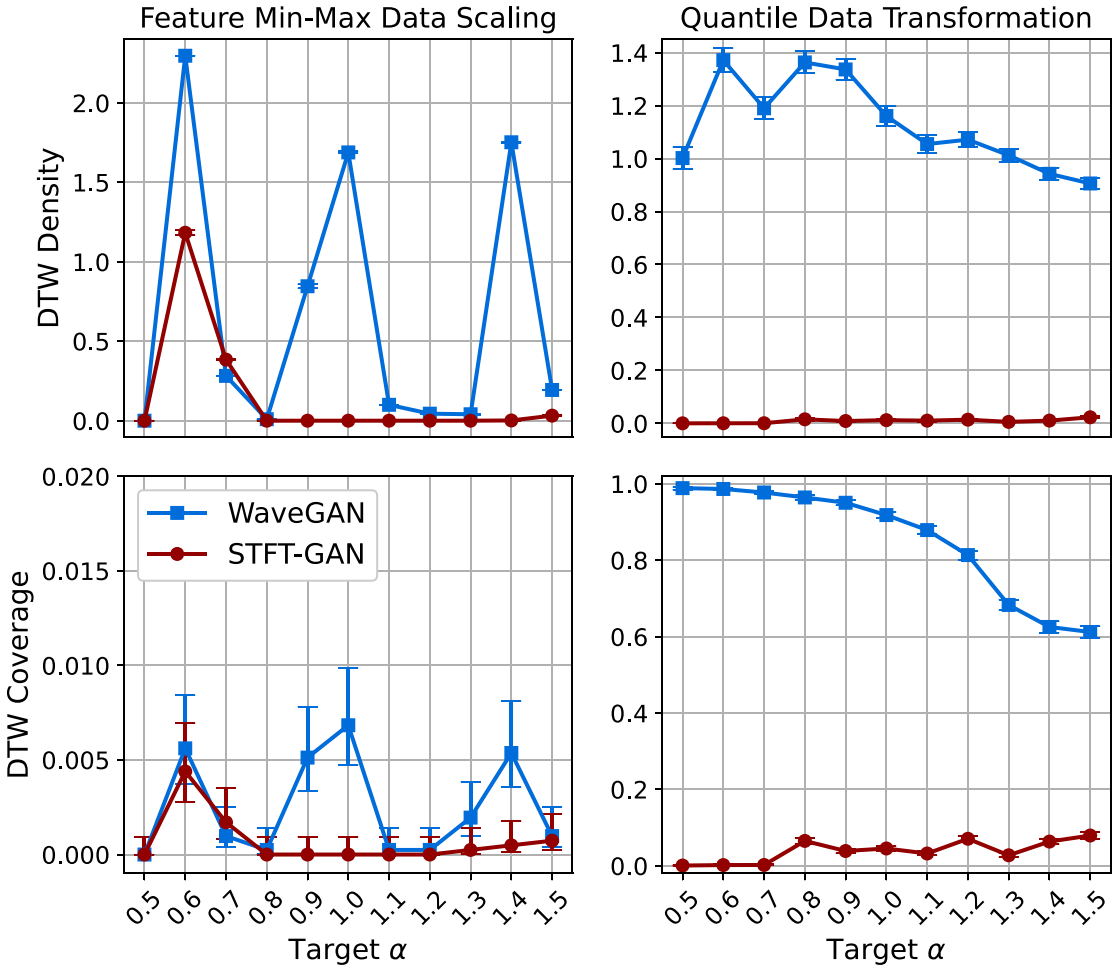

Figure 18. DTW density and coverage results for symmetric α-stable noise.

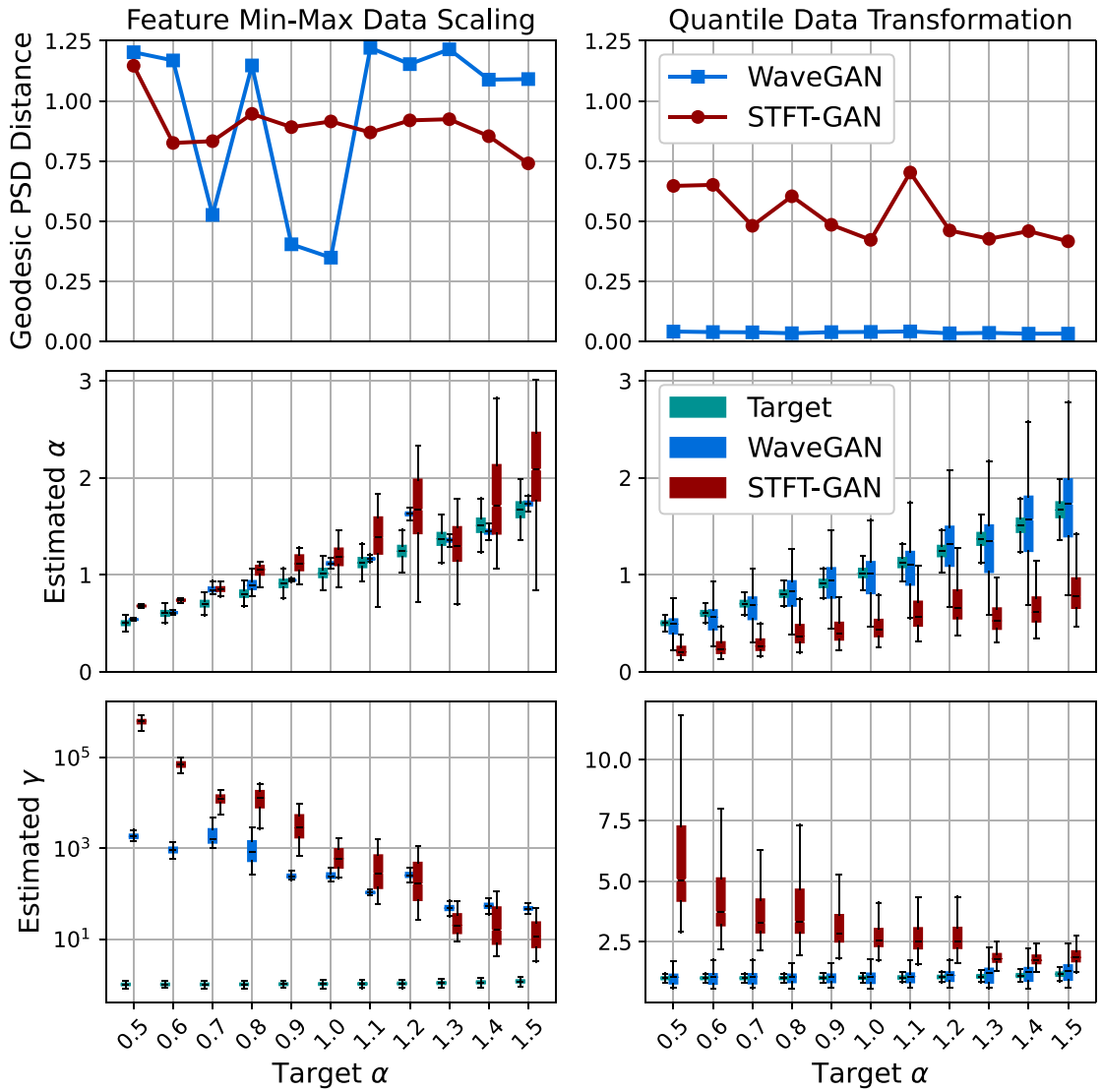

Figure 19. Additional fidelity evaluations for symmetric α-stable noise.

dispersion in scale ratio was very large. By contrast, the quantile data transformation did not appear to improve STFT-GAN performance.

Example target and generated time series for GANs with the quantile data transformation are shown in figure 17 for the case of $p = 0.05$ BG noise. From these plots, it is evident that STFT-GAN failed to recover the correct background noise level relative to the impulsive component, while WaveGAN better matched the target distribution.

Aggregate results for $S\alpha S$ noise are given in figures 18 and 19. The GAN models with feature min-max scaling suffered from mode-collapse during training across all tests, as evidenced by the near-zero DTW coverage results. By contrast, the quantile data transformation preprocessing step enabled WaveGAN to avoid mode-collapse during training, whereas STFT-GAN still suffered from poor diversity, as measured by DTW coverage. In terms of the fidelity metrics, WaveGAN clearly outperformed STFT-GAN, although the dispersion in the characteristic exponent was unacceptably large.

Example target and generated time series for GANs trained with a quantile data transformation on $S\alpha S$ noise with $\alpha = 1.0$ are shown in figure 20. From these plots, we see that WaveGAN produced short-duration impulses, whereas STFT-GAN produced impulses that were not as localized in time. These observations are consistent with the PSD distance results. Further, both models often produced time series with maximum impulse amplitudes that were too large, supporting the finding that they did not consistently recover the target characteristic exponent.

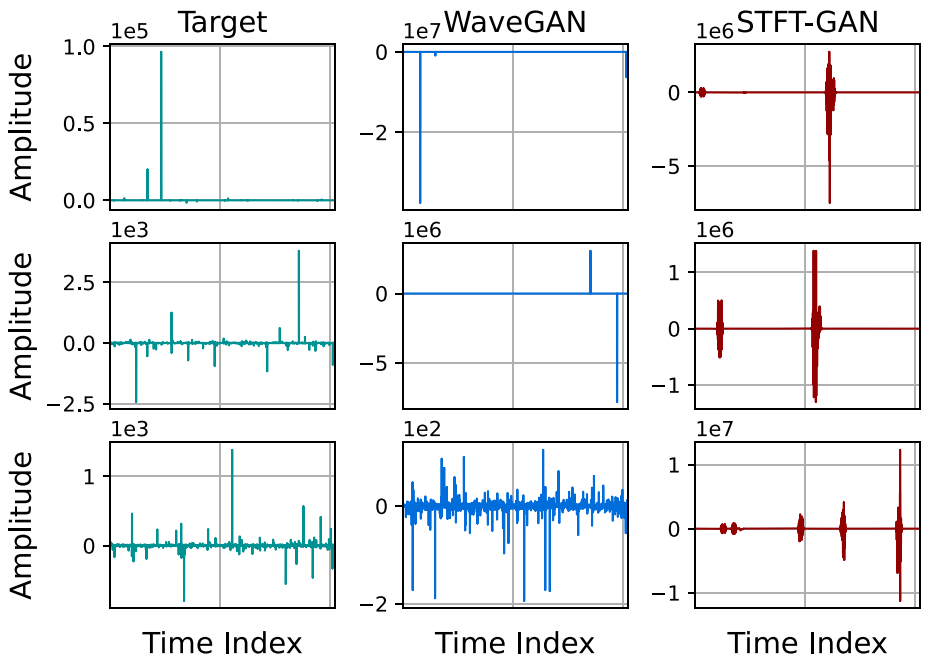

Figure 20. Example time series for GANs trained on symmetric α-stable noise with quantile data transformation and target characteristic exponent, $\alpha = 1.0$.

7. Discussion and conclusions

We examined the ability of two general-purpose GAN models for time series, WaveGAN and STFT-GAN, to faithfully learn several types of noise that frequently arise in physical measurements, signal processing, and communications. Specifically, we investigated four classes of noise with well-established models: band-limited thermal noise, power law noise, shot noise, and impulsive noise. In addition, within each noise class, we considered multiple types over a broad range of parameter values. Performance evaluations examined generative diversity, as measured by DTW coverage, and generative fidelity, as measured by DTW density, median PSD distance, and characteristic model parameters specific to each noise type.

For most noise types, either the time-domain WaveGAN model or the image-domain STFT-GAN model was more effective. Namely, STFT-GAN was better at learning a large (>60 dB) PSD dynamic range, as evidenced by the results for band-pass thermal noise, power law noise, and shot noise with the Gaussian pulse type. In addition, the flexibility afforded by the choice of STFT dimensions facilitated improvements in STFT performance, as shown by the FBM evaluations; see figure 8. These findings indicate that the time-frequency STFT data representation, which more directly encodes the frequency content of a signal, facilitates learning spectral characteristics.

On the other hand, there was evidence that WaveGAN was more effective at learning time-domain signal characteristics, which are especially important for discontinuous shot and impulsive noise. Namely, WaveGAN outperformed STFT-GAN for shot noise with one-sided exponential pulses and on both impulsive noise types when a quantile data transformation was applied in preprocessing. These findings suggest that a hybrid GAN model combining both time-domain and frequency-domain features may be beneficial in some settings. This is an interesting topic for further research.

GAN frameworks developed on standard image datasets with pixel values in the range [0, 255] typically rescale the data to the range $[-1, 1]$, which corresponds to the range of the hyperbolic tangent activation function that is often the generator output. Because most GAN research has focused on standard image datasets consisting of bounded data, finding effective approaches for unbounded target distributions is an important research question.

Indeed, for the challenging cases of non-Gaussian BG and $S\alpha S$ impulsive noise, we found that GANs were especially sensitive to the data scaling method applied prior to training. Specifically, STFT-GAN failed badly, regardless of the data scaling method. On the other hand, WaveGAN required a quantile data transformation, which altered the data distribution to approximate a standard normal distribution, to avoid mode collapse. These findings highlight the limitations of conventional min-max data scaling on training data and the need for further research into general-purpose GANs that can learn impulsive, non-Gaussian

time series. There has been some recent work specifically focused on GANs for heavy-tailed distributions, e.g. [24, 104, 105], but it is unclear how well these approaches generalize to other types of distributions.

The experimental evaluations presented here were necessarily limited in scope. Specifically, we did not attempt to evaluate all types of time series GANs or to propose a single GAN architecture that works optimally for all noise types. Moreover, we did not consider complex-valued noise, superpositions of multiple noise types, or time series with a deterministic component. Evaluating the ability of deep generative models to learn these categories of random processes is of high interest for future studies.

Our performance evaluations demonstrated the value in estimating multiple types of GAN performance measures, including both general-purpose and data-specific metrics that are easier to interpret. In particular, our evaluations showed the utility of the general-purpose density and coverage metrics based on DTW distance for time series. The development and study of general-purpose generative model evaluation measures, particularly for time series, remains an important research topic.

In conclusion, our findings demonstrate that general-purpose time series GANs based on commonly-used deep convolutional architectures are capable of accurately learning many types of classical noise models, including Gaussian and non-Gaussian distributions, as well as stationary and non-stationary random processes. These results give further evidence that GANs are a very promising class of generative models for blindly learning a wide variety of time series distributions. Moreover, our battery of tests with classical random processes provides a useful benchmark to aid further development of deep generative models for time series.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.18434/mds2-3034.

Appendix. Event rate estimation for shot noise

We derive the event rate estimator given in equation (12) for the shot noise model of section 2.3 under an exponential distribution for the pulse amplitudes. From classical results for filtered Poisson processes [4, 106], the mean and variance of the shot noise process defined by equation (3) are $\mu_X = \nu E[A_n] I_1$ and $\sigma_X^2 = \nu E[A_n]^2 I_2$, respectively, where $E[\cdot]$ denotes mathematical expectation, $I_1 = \int_{-\infty}^{\infty} p(t)\,\mathrm{d}t$, and $I_2 = \int_{-\infty}^{\infty} p(t)^2\,\mathrm{d}t$. Table 1 lists I_1 and I_2 for the pulse functions considered here. Now, assuming that A_n follows an exponential distribution with mean β, it follows that $E[A_n] = \beta$ and $E[A_n^2] = 2\beta^2$. Thus, $\mu_X = \nu\beta I_1$ and $\sigma_X^2 = \nu(2\beta^2) I_2$. Solving the μ_X equation for β, substituting the result into the σ_X^2 equation, solving for ν, and replacing μ_X and σ_X^2 by their sample estimates yields the estimator in equation (12). For the special case of a one-sided exponential pulse type, the event rate estimator is equivalent to equation (29) in [51] with $\epsilon = 0$.

ORCID iDs

Adam Wunderlich https://orcid.org/0000-0002-8463-9156
Jack Sklar https://orcid.org/0000-0002-5922-3247

References

[1] Vasilescu G 2006 *Electronic Noise and Interfering Signals: Principles and Applications* (Springer)
[2] Milotti E 2019 *The Physics of Noise* (Morgan & Claypool Publishers)
[3] Barrett H H and Myers K J 2004 *Foundations of Image Science* (Wiley)
[4] Howard R M 2016 *A Signal Theoretic Introduction to Random Processes* (Wiley)
[5] Montáns F J, Chinesta F, Gómez-Bombarelli R and Kutz J N 2019 Data-driven modeling and learning in science and engineering *C. R. Mécanique* **347** 845–55
[6] Foster D 2019 *Generative Deep Learning: Teaching Machines to Paint, Write, Compose and Play* (O'Reilly Media)
[7] Langr J and Bok V 2019 *GANs In Action: Deep Learning With Generative Adversarial Networks* (Manning Publications)
[8] Bond-Taylor S, Leach A, Long Y and Willcocks C G 2021 Deep generative modelling: a comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models *IEEE Trans. Pattern Anal. Mach. Intell.* **44** 7327–47
[9] Ruthotto L and Haber E 2021 An introduction to deep generative modeling *GAMM-Mitteilungen* **44** e202100008
[10] Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B and Bharath A A 2018 Generative adversarial networks: an overview *IEEE Signal Process. Mag.* **35** 53–65
[11] Hong Y, Hwang U, Yoo J and Yoon S 2019 How generative adversarial networks and their variants work: an overview *ACM Comput. Surv. (CSUR)* **52** 1–43
[12] Wang Z, She Q and Ward T E 2022 Generative adversarial networks in computer vision: A survey and taxonomy *ACM Comput. Surv. (CSUR)* **54** 1–38

[13] Pan Z, Yu W, Yi X, Khan A, Yuan F and Zheng Y 2019 Recent progress on generative adversarial networks (GANs): a survey *IEEE Access* **7** 36322–33
[14] Chen J, Chen J, Chao H and Yang M 2018 Image blind denoising with generative adversarial network based noise modeling *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 3155–64
[15] Wolterink J M, Leiner T, Viergever M A and Išgum I 2017 Generative adversarial networks for noise reduction in low-dose CT *IEEE Trans. Med. Imaging* **36** 2536–45
[16] Lin K, Li T H, Liu S and Li G 2019 Real photographs denoising with noise domain adaptation and attentive generative adversarial network *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*
[17] Ma Y, Wei B, Feng P, He P, Guo X and Wang G 2020 Low-dose CT image denoising using a generative adversarial network with a hybrid loss function for noise learning *IEEE Access* **8** 67519–29
[18] Cha S, Park T, Kim B, Baek J and Moon T 2021 GAN2GAN: generative noise learning for blind denoising with single noisy images *Int. Conf. on Learning Representations* (arXiv:1905.10488)
[19] Zhang T, Cheng J, Fu H, Gu Z, Xiao Y, Zhou K, Gao S, Zheng R and Liu J 2020 Noise adaptation generative adversarial network for medical image analysis *IEEE Trans. Med. Imaging* **39** 1149–59
[20] Miller K J and Du Bosq T 2021 A machine learning approach to improving quality of atmospheric turbulence simulation *Proc. SPIE, Infrared Imaging Systems: Design, Analysis, Modeling and Testing XXXII* vol 11740 p 117400N
[21] Zhou Y, Zhu Z, Bai X, Lischinski D, Cohen-Or D and Huang H 2018 Non-stationary texture synthesis by adversarial expansion *ACM Trans. Graph.* **37** 1–13
[22] Mauduit V, Abry P, Leonarduzzi R, Roux S G and Quemener E 2020 DCGAN for the synthesis of multivariate multifractal textures: How do we know it works? *2020 IEEE 30th Int. Workshop on Machine Learning for Signal Processing (MLSP)* pp 1–6
[23] Baradad Jurjo M, Wulff J, Wang T, Isola P and Torralba A 2021 Learning to see by looking at noise *Advances in Neural Information Processing Systems* vol 34 pp 2556–69 (available at: https://proceedings.neurips.cc/paper_files/paper/2021/file/14f2ebeab937ca128186e7ba876faef9-Paper.pdf)
[24] Zhou M, Wang J, Feng X, Sun H, Li J and Kuai X 2021 On generative-adversarial-network-based underwater acoustic noise modeling *IEEE Trans. Veh. Technol.* **70** 9555–9
[25] Donahue C, McAuley J and Puckette M 2019 Adversarial audio synthesis *Int. Conf. on Learning Representations (ICLR)* pp 1–16 (arXiv:1802.04208)
[26] Kumar K, Kumar R, de Boissiere T, Gestin L, Teoh W Z, Sotelo J, de Brébisson A, Bengio Y and Courville A C 2019 MelGAN: Generative adversarial networks for conditional waveform synthesis *Advances in Neural Information Processing Systems (NeurIPS 2019)* vol 32 (available at: https://papers.nips.cc/paper/2019/file/6804c9bca0a615bdb9374d00a9fcba59-paper.pdf) pp 1–12
[27] Smith K E and Smith A O 2020 Conditional GAN for timeseries generation (arXiv:2006.16477)
[28] Engel J, Agrawal K K, Chen S, Gulrajani I, Donahue C and Roberts A 2019 GANSynth: Adversarial neural audio synthesis *Int. Conf. on Learning Representations (ICLR)* pp 1–17 (arXiv:1902.08710)
[29] Marafioti A, Perraudin N, Holighaus N and Majdak P 2019 Adversarial generation of time-frequency features with application in audio synthesis *Proc. 36th Int. Conf. on Machine Learning (ICML)* (available at: http://proceedings.mlr.press/v97/marafioti19a/marafioti19a.pdf) pp 1–13
[30] Nistal J, Lattner S and Richard G 2021 Comparing representations for audio synthesis using generative adversarial networks *28th European Signal Processing Conf. (EUSIPCO)* pp 161–5
[31] Esteban C, Hyland S L and Rätsch G 2017 Real-valued (medical) time series generation with recurrent conditional GANs (arXiv:1706.02633)
[32] Yoon J, Jarrett D and van der Schaar M 2019 Time-series generative adversarial networks *Advances in Neural Information Processing Systems (NeurIPS 2019)* (available at: https://papers.nips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-paper.pdf) pp 1–11
[33] Wiese M, Knobloch R, Korn R and Kretschmer P 2020 Quant GANs: deep generation of financial time series *Quant. Financ.* **20** 1419–40
[34] Radford A, Metz L and Chintala S 2015 Unsupervised representation learning with deep convolutional generative adversarial networks (arXiv:1511.06434)
[35] Sklar J and Wunderlich A 2021 Feasibility of modeling orthogonal frequency-division multiplexing communication signals with unsupervised generative adversarial networks *J. Res. Natl Inst. Stand. Technol.* **126** 1–25
[36] Wunderlich A and Sklar J 2023 NoiseGAN: software for evaluating convolutional generative adversarial networks with classical random process noise models (available at: https://github.com/usnistgov/NoiseGAN)
[37] Wunderlich A and Sklar J Noise datasets for evaluating deep generative models 2023 (https://doi.org/10.18434/mds2-3034)
[38] Percival D B and Walden A T 2000 *Wavelet Methods for Time Series Analysis* (Cambridge University Press)
[39] Papoulis A 1965 *Probability, Random Variables and Stochastic Processes* (McGraw-Hill)
[40] Mitra S K 2001 *Digital Signal Processing: A Computer-Based Approach* 2nd edn (McGraw-Hill)
[41] Keshner M S 1982 1/f Noise *Proc. IEEE* **70** 212–8
[42] Rouphael T J 2014 *Wireless Receiver Architectures and Design: Antennas, RF, Synthesizers, Mixed Signal and Digital Signal Processing* (Academic)
[43] Mandelbrot B B and Van Ness J W 1968 Fractional Brownian motions, fractional noises and applications *SIAM Rev.* **10** 422–37
[44] Beran J, Feng Y, Ghosh S and Kulik R 2013 *Long-Memory Processes* (Springer)
[45] Veenstra J 2013 Persistence and anti-persistence: theory and software *PhD Dissertation* University of Western Ontario (available at: https://ir.lib.uwo.ca/etd/1119)
[46] Flandrin P 1989 On the spectrum of fractional Brownian motions *IEEE Trans. Inf. Theory* **35** 197–9
[47] Perrin E, Harba R, Jennane R and Iribarren I 2002 Fast and exact synthesis for 1-D fractional Brownian motion and fractional Gaussian noises *IEEE Signal Process. Lett.* **9** 382–4
[48] Dietrich C R and Newsam G N 1997 Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix *SIAM J. Sci. Comput.* **18** 1088–107
[49] Parzen E 1962 *Stochastic Processes* (Holden-Day)
[50] Snyder D L and Miller M I 1991 *Random Point Processes in Time and Space* 2nd edn (Springer)
[51] Theodorsen A, Garcia O E and Rypdal M 2017 Statistical properties of a filtered Poisson process with additive random noise: distributions, correlations and moment estimation *Phys. Scr.* **92** 054002
[52] Pighi R, Franceschini M, Ferrari G and Raheli R 2009 Fundamental performance limits of communications systems impaired by impulse noise *IEEE Trans. Commun.* **57** 171–82

[53] Tsihrintzis G A and Nikias C L 1995 Performance of optimum and suboptimum receivers in the presence of impulsive noise modeled as an alpha-stable process *IEEE Trans. Commun.* **43** 904–14
[54] Ghosh M 1996 Analysis of the effect of impulse noise on multicarrier and single carrier QAM systems *IEEE Trans. Commun.* **44** 145–7
[55] Herath S P, Tran N H and Le-Ngoc T 2012 On optimal input distribution and capacity limit of Bernoulli-Gaussian impulsive noise channels *2012 IEEE Int. Conf. on Communications (ICC)* pp 3429–33
[56] Blackard K L, Rappaport T S and Bostian C W 1993 Measurements and models of radio frequency impulsive noise for indoor wireless communications *IEEE J. Sel. Areas Commun.* **11** 991–1001
[57] Mirahmadi M, Al-Dweik A and Shami A 2013 BER reduction of OFDM based broadband communication systems over multipath channels with impulsive noise *IEEE Trans. Commun.* **61** 4602–15
[58] Kerpez K J and Gottlieb A M 1995 The error performance of digital subscriber lines in the presence of impulse noise *IEEE Trans. Commun.* **43** 1902–5
[59] Mann I, McLaughlin S, Henkel W, Kirkby R and Kessler T 2002 Impulse generation with appropriate amplitude, length, inter-arrival and spectral characteristics *IEEE J. Sel. Areas Commun.* **20** 901–12
[60] Meng H, Guan Y L and Chen S 2005 Modeling and analysis of noise effects on broadband power-line communications *IEEE Trans. Power Deliv.* **20** 630–7
[61] Fernandes V, Finamore W A, Ribeiro M V, Marina N and Karamachoski J 2017 Bernoulli–Gaussian distribution with memory as a model for power line communication noise *Proc. Brazilian Telecommunication Signal Processing Symp.* pp 328–32 (available at: https://biblioteca.sbrt.org.br/articlefile/620.pdf)
[62] Kuai X, Sun H, Zhou S and Cheng E 2016 Impulsive noise mitigation in underwater acoustic OFDM systems *IEEE Trans. Veh. Technol.* **65** 8190–202
[63] Wang J, Li J, Yan S, Shi W, Yang X, Guo Y and Gulliver T A 2020 A novel underwater acoustic signal denoising algorithm for Gaussian/non-Gaussian impulsive noise *IEEE Trans. Veh. Technol.* **70** 429–45
[64] Shongwe T, Vinck A H and Ferreira H C 2014 On impulse noise and its models *18th IEEE Int. Symp. on Power Line Communications and Its Applications* pp 12–17
[65] Nolan J P 2020 *Univariate Stable Distributions* (Springer)
[66] Nikias C L and Shao M 1995 *Signal Processing With Alpha-Stable Distributions and Applications* (Wiley)
[67] Shao M and Nikias C L 1993 Signal processing with fractional lower order moments: stable processes and their applications *Proc. IEEE* **81** 986–1010
[68] Georgiou P G, Tsakalides P and Kyriakakis C 1999 Alpha-stable modeling of noise and robust time-delay estimation in the presence of impulsive noise *IEEE Trans. on Multimed.* **1** 291–301
[69] Harrison P and Miotto J M 2020 PyLevy : Levy distributions for Python (available at: https://github.com/josemiotto/pylevy)
[70] Chambers J M, Mallows C L and Stuck B 1976 A method for simulating stable random variables *J. Am. Stat. Assoc.* **71** 340–4
[71] Nair J, Wierman A and Zwart B 2022 *The Fundamentals of Heavy-Tails: Properties, Emergence and Estimation* (Cambridge University Press)
[72] Goodfellow I, Bengio Y, Courville A and Bengio Y 2016 *Deep Learning* (MIT Press)
[73] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V and Courville A C 2017 Improved training of Wasserstein GANS *Advances in Neural Information Processing Systems (NeurIPS)* pp 1–11 (available at: https://papers.nips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-paper.pdf)
[74] Arjovsky M, Chintala S and Bottou L 2017 Wasserstein generative adversarial networks *Proc. Int. Conf. on Machine Learning (ICML)* pp 1–10 (available at: http://proceedings.mlr.press/v70/arjovsky17a/arjovsky17a.pdf)
[75] Smith J O 2011 *Spectral Audio Signal Processing* (W3K Publishing)
[76] Mallat S 2009 *A Wavelet Tour of Signal Processing* 3rd edn (Academic)
[77] Kingma D P and Ba J 2015 Adam: a method for stochastic optimization *Int. Conf. on Learning Representations (ICLR)* (arXiv:1412.6980)
[78] DasGupta A 2010 *Fundamentals of Probability: A First Course* (Springer)
[79] Pedregosa F *et al* 2011 Scikit-learn: machine learning in python *J. Mach. Learn. Res.* **12** 2825–30 (available at: www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf)
[80] Borji A 2019 Pros and cons of GAN evaluation measures *Comput. Vis. Image Underst.* **179** 41–65
[81] Borji A 2022 Pros and cons of GAN evaluation measures: new developments *Comput. Vis. Image Underst.* **215** 103329
[82] Naeem M F, Oh S J, Uh Y, Choi Y and Yoo J 2020 Reliable fidelity and diversity metrics for generative models *Int. Conf. on Machine Learning* (PMLR) pp 7176–85 (available at: http://proceedings.mlr.press/v119/naeem20a/naeem20a.pdf)
[83] Berndt D J and Clifford J 1994 Using dynamic time warping to find patterns in time series *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining* pp 359–70
[84] Keogh E and Ratanamahatana C A 2005 Exact indexing of dynamic time warping *Knowl. Inf. Syst.* **7** 358–86
[85] Ding H, Trajcevski G, Scheuermann P, Wang X and Keogh E 2008 Querying and mining of time series data: experimental comparison of representations and distance measures *Proc. VLDB Endowment* vol 1 pp 1542–52
[86] Mueen A and Keogh E 2016 Extracting optimal performance from dynamic time warping *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* pp 2129–30 (available at: www.cs.unm.edu/mueen/DTW.pdf)
[87] Bagnall A, Lines J, Bostrom A, Large J and Keogh E 2017 The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances *Data Min. Knowl. Discovery* **31** 606–60
[88] Lemire D 2009 Faster retrieval with a two-pass dynamic-time-warping lower bound *Pattern Recognit.* **42** 2169–80
[89] Han J, Kamber M and Pei J 2012 *Data Mining: Concepts and Techniques* 3rd edn (Morgan Kaufmann)
[90] Murphy K P 2012 *Machine Learning: A Probabilistic Perspective* (MIT Press)
[91] Casacuberta F, Vidal E and Rulot H 1987 On the metric properties of dynamic time warping *IEEE Trans. Acoust. Speech Signal Process.* **35** 1631–3
[92] Meert W, Hendrickx K, Van Craenendonck T, Robberechts P, Blockeel H and Davis J 2022 DTAIDistance (v2.3.10) (available at: https://github.com/wannesm/dtaidistance)
[93] Efron B and Tibshirani R J 1994 *An Introduction to the Bootstrap* (CRC Press)
[94] Agresti A and Coull B A 1998 Approximate is better than exact for interval estimation of binomial proportions *Am. Stat.* **52** 119–26
[95] Thomson D J 1982 Spectrum estimation and harmonic analysis *Proc. IEEE* **70** 1055–96
[96] Percival D B and Walden A T 2020 *Spectral Analysis for Univariate Time Series* (Cambridge University Press)

[97] Cokelaer T and Hasch J 2017 Spectrum: spectral analysis in python *J. Open Source Softw.* **2** 348
[98] Georgiou T T 2007 An intrinsic metric for power spectral density functions *IEEE Signal Process. Lett.* **14** 561–3
[99] Georgiou T T 2006 Distances between power spectral densities (arXiv:0607026)
[100] Istas J and Lang G 1997 Quadratic variations and estimation of the local Hölder index of a Gaussian process *Ann. Inst. Henri Poincare* B **33** 407–36
[101] Coeurjolly J-F 2001 Estimating the parameters of a fractional Brownian motion by discrete variations of its sample paths *Stat. Inference Stochastic Process.* **4** 199–227
[102] Coeurjolly J-F and Porcu E 2017 Properties and Hurst exponent estimation of the circularly-symmetric fractional Brownian motion *Stat. Probab. Lett.* **128** 21–27
[103] Tsihrintzis G A and Nikias C L 1996 Fast estimation of the parameters of alpha-stable impulsive interference *IEEE Trans. Signal Process.* **44** 1492–503
[104] Zhang X and Zhou J 2021 A heavy-tailed distribution data generation method based on generative adversarial network *2021 IEEE 10th Data Driven Control and Learning Systems Conf. (DDCLS)* pp 535–40
[105] Huster T, Cohen J, Lin Z, Chan K, Kamhoua C, Leslie N O, Chiang C-Y J and Sekar V 2021 Pareto GAN: extending the representational power of GANs to heavy-tailed distributions *Int. Conf. on Machine Learning* pp 4523–32 (available at: http://proceedings.mlr.press/v139/huster21a.html)
[106] Racicot R L 1971 Fitting a filtered Poisson process *IMA J. Appl. Math.* **7** 260–72