RFWAVE: MULTI-BAND RECTIFIED FLOW FOR AUDIO WAVEFORM RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advancements in generative modeling have significantly enhanced the reconstruction of audio waveforms from various representations. While diffusion models are adept at this task, they are hindered by latency issues due to their operation at the individual sample point level and the need for numerous sampling steps. In this study, we introduce RFWave, a cutting-edge multi-band Rectified Flow approach designed to reconstruct high-fidelity audio waveforms from Mel-spectrograms or discrete acoustic tokens. RFWave uniquely generates complex spectrograms and operates at the frame level, processing all subbands simultaneously to boost efficiency. Leveraging Rectified Flow, which targets a straight transport trajectory, RFWave achieves reconstruction with just 10 sampling steps. Our empirical evaluations show that RFWave not only provides outstanding reconstruction quality but also offers vastly superior computational efficiency, enabling audio generation at speeds up to 160 times faster than real-time on a GPU. An online demonstration is available at: `https://rfwave-demo.github.io/rfwave/`.

1 INTRODUCTION

Audio waveform reconstruction significantly enhances the digital interactions by enabling realistic voice and sound generation for diverse applications. This technology transforms low-dimensional features, derived from raw audio data, into perceptible sounds, improving the audio experience on various platforms such as virtual assistants and entertainment systems. Autoregressive models and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have been applied to this task, greatly advancing audio quality beyond traditional signal processing methods (Kawahara et al., 1999; Morise et al., 2016). Autoregressive methods, while effective, are hindered by slow generation speeds due to their sequential prediction of sample points (Oord et al., 2016; Kalchbrenner et al., 2018; Valin & Skoglund, 2019). In contrast, GANs predict sample points in parallel, resulting in faster generation speeds and maintaining high-quality output (Kumar et al., 2019; Yamamoto et al., 2020; Kong et al., 2020a; Siuzdak, 2023; Du et al., 2023). Consequently, GAN-based methods deliver impressive performance and are extensively utilized in real-world audio generation applications.

Despite the advancements, GAN-based waveform reconstruction models face challenges such as the necessity for complex discriminator designs and issues like instability or mode collapse (Thanh-Tung et al., 2018). In response, diffusion models for waveform reconstruction have been explored, offering stability during training and the ability to reconstruct high-quality waveforms (Chen et al., 2020; Kong et al., 2020b; gil Lee et al., 2022; Nguyen et al., 2024; Huang et al., 2022; Koizumi et al., 2022). However, these models are at least an order of magnitude slower compared to GANs. The slow generation speed in these diffusion-based waveform reconstruction models is primarily due to two factors: (1) the requirement of numerous sampling steps to achieve high-quality samples, and (2) the operation at the waveform sample point level. The latter often involves multiple upsampling operations to transition from frame rate resolution to sample rate resolution, increasing the sequence length and consequently leading to higher GPU memory usage and computational demands.

In this paper, we propose RFWave, a diffusion-type waveform reconstruction method designed to match the speed of GAN-based methods while maintaining the training stability and high sample quality of diffusion models. To overcome the challenge of slow sampling, we employ Rectified Flow (Liu et al., 2023; Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023), which connects data and noise along a straight line, thereby enhancing sampling efficiency. To address the GPU memory and computational demands of sample point-level modeling, our model operates at the level of Short-Time Fourier Transform (STFT) frames, enabling more efficient processing and reducing GPU memory usage. RFWave with only 10 sampling steps can generate high-quality audio, achieving an inference speed of up to 160 times real-time on an NVIDIA GeForce RTX 4090 GPU. Additionally, the incorporation of three enhanced loss functions and an optimized sampling strategy further elevates the overall quality of the reconstructed waveforms. To our knowledge, RFWave stands as the fastest diffusion-based audio waveform reconstruction model, and it delivers superior audio quality. Furthermore, it can reconstruct waveforms from Mel-spectrograms and discrete acoustic tokens, enhancing its versatility and applicability in various audio generation tasks. Our main contributions are as follows:

1. By integrating the Rectified Flow and 3 enhanced loss functions – energy-balanced loss, overlap loss, and STFT loss – our model can reconstruct high-quality waveforms with a drastically reduced number of sampling steps.
2. We utilize a multi-band strategy, coupled with the high-efficiency ConvNeXtV2 (Woo et al., 2023) backbone, to generate different subbands concurrently. This not only assures audio quality by circumventing cumulative errors, but also boosts the synthesis speed.
3. Our model operates at the level of STFT frames, not individual waveform sample points. This approach significantly accelerates processing and reduces GPU memory usage.
4. We propose a technique for selecting sampling time points based on the straightness of the Rectified Flow transport trajectories, which enhances sample quality for free.

2 BACKGROUND

In this section, we describe the basic formulation of the Rectified Flow and present releated works, while also detailing our correlation and distinction.

Rectified Flow Rectified Flow (Liu et al., 2023) presents an innovative Ordinary Differential Equation (ODE)-based framework for generative modeling and domain transfer. It introduces a method to learn a mapping that connects two distributions, π_0 and π_1 on $\mathbb{R}^d$, based on empirical observations:

$$\frac{\mathrm{d}Z_t}{\mathrm{d}t} = v(Z_t, t), \quad \text{initialized from } Z_0 \sim \pi_0 \text{, such that } Z_1 \sim \pi_1, \tag{1}$$

where $v\colon \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ represents a velocity field. The learning of this field involves minimizing a mean square objective function,

$$\min_v \mathbb{E}_{(X_0,X_1)\sim\gamma} \left[\int_0^1 || \frac{\mathrm{d}}{\mathrm{d}t} X_t - v(X_t, t) ||^2 \, \mathrm{d}t \right], \quad \text{with} \quad X_t = \phi(X_0, X_1, t), \tag{2}$$

where $X_t = \phi(X_0, X_1, t)$ represents a time-differentiable interpolation between X_0 and X_1, with $\frac{\mathrm{d}}{\mathrm{d}t} X_t = \partial_t \phi(X_0, X_1, t)$. The γ represents any coupling of (π_0, π_1). An illustrative instance of γ is the independent coupling $\gamma = \pi_0 \times \pi_1$, which allows for empirical sampling based on separately observed data from π_0 and π_1. Liu et al. (2023) recommended a simple choice of

$$X_t = (1-t)X_0 + tX_1 \implies \frac{\mathrm{d}}{\mathrm{d}t} X_t = X_1 - X_0. \tag{3}$$

This simplification results in linear trajectories, which are critical for accelerating the inference process. Typically, the velocity field v is represented using a deep neural network. The solution to (2) is approximated through stochastic gradient methods. To approximate the ODE presented in (1), numerical solvers are commonly employed. A prevalent technique is the forward Euler method. This approach computes values using the formula

$$Z_{t+\frac{1}{n}} = Z_t + \frac{1}{n} v(Z_t, t), \qquad \forall t \in \{0, \ldots, n-1\}/n, \tag{4}$$

where the simulation is executed with a step interval of $\epsilon = 1/n$ over n steps.

The velocity field has the capacity to incorporate conditional information, which is particularly essential in applications such as text-to-image generation and waveform reconstruction from compressed acoustic representation. Consequently, in such contexts, $v(Z_t, t)$ in (2) is modified to $v(Z_t, t \mid \mathcal{C})$, where $\mathcal{C}$ represents the conditional information pertinent to the corresponding X_1.

Diffusion Models for Audio Waveform Reconstruction Diffusion models (Song et al., 2020; Ho et al., 2020) have become the de-facto choice for high-quality generation in the realm of generative models. Diffwave (Kong et al., 2020b) and WaveGrad (Chen et al., 2020) were pioneering efforts to reconstruct waveforms using diffusion models, achieving performance comparable to autoregressive models and GANs. PriorGrad (gil Lee et al., 2022) enhances both speech quality and inference speed by employing a data-dependent prior distribution. Meanwhile, FreGrad (Nguyen et al., 2024) simplifies the model and reduces denoising time through the use of Discrete Wavelet Transform (DWT). Multi-Band Diffusion (MBD) (Roman et al., 2023) leverages a diffusion model to reconstruct waveforms from discrete EnCodec (Défossez et al., 2022) tokens. Despite these advancements, the current fastest generation speed of these methods is only about 10 to 20 times faster than real-time, which limits their application for real-time use, especially when combined with large-scale Transformer-based acoustic models (Wang et al., 2023; Du et al., 2024). In contrast, our method achieves speeds up to 160 times faster than real-time, significantly enhancing real-time applicability.

Estimating Complex Spectrograms Waveform reconstruction from complex spectrograms can be effectively achieved using the ISTFT. Notably, Vocos (Siuzdak, 2023) and APNet2 (Du et al., 2023), utilizing GANs as their model framework, estimate magnitude and phase spectrograms from the input Mel-spectrograms, which can be transformed to complex spectrograms effortlessly. Both models operate at the frame level, enabling them to achieve significantly faster inference speeds compared to HiFi-GAN (Kong et al., 2020a), which uses multiple upsampling layers and operates at the level of waveform sample points. Morever, these models preserve the quality of the synthesized waveform, demonstrating their superiority in both speed and fidelity without a trade-off. In this paper, we directly estimate complex spectrograms using Rectified Flow and focus on frame-level operations, aiming to enhance both the efficiency and quality of our waveform synthesis process. Notably, the distributions of real and imaginary parts of the complex spectrograms appear more homogeneous in comparison to the distributions magnitude and phase.

Multi-band Audio Waveform Reconstruction Both Multi-band MelGAN (Yang et al., 2021) and Multi-band Diffusion (Roman et al., 2023) employ multi-band strategies, albeit for different purposes within their respective frameworks. Multi-band MelGAN, specifically, uses Pseudo-Quadrature Mirror Filters (PQMF) (Johnston, 1980) to divide frequency bands. This division results in each subband's waveform being a fraction of the original waveform's length, based on the number of subbands. By reshaping these subbands into feature dimensions and utilizing a unified backbone for modeling, Multi-band MelGAN is able to operate on considerably shorter signals. This strategy significantly enhances the efficiency of the model, leading to accelerated training and inference processes. Multi-band Diffusion utilizes an array of band-pass filters to separate the frequency bands and models each subband with a distinct model. This approach ensures that errors in one band do not negatively impact the others. In our research, we simplify the process of frequency band division by directly choosing the appropriate dimensions from the complex spectrograms. Furthermore, we enhance efficiency by modeling all subbands together in parallel with a single, unified model. This strategy improves the processing speed and also helps in reducing error accumulation across different subbands.

3 METHOD

Our model utilizes a multi-band Rectified Flow to directly predict the complex spectrogram. It operates at the STFT frame level and incorporates a highly efficient ConvNeXtV2 (Woo et al., 2023) backbone. With only 10 steps of sampling, the model is capable of producing high-quality waveforms. In this section, we present the structure of the multi-band Rectified Flow model, which can operate with noisy sample in either the time or frequency domain. Additionally, we describe the corresponding normalization techniques, three enhanced loss functions, and the strategy for selecting sampling time points.

3.1 MULTI-BAND RECTIFIED FLOW

Model Structure The model structure is depicted in Figure 1. All frequency bands, each distinguished by a unique subband index, share the same model. The subbands of a given sample are grouped together into a single batch for processing, which facilitates simultaneous training or

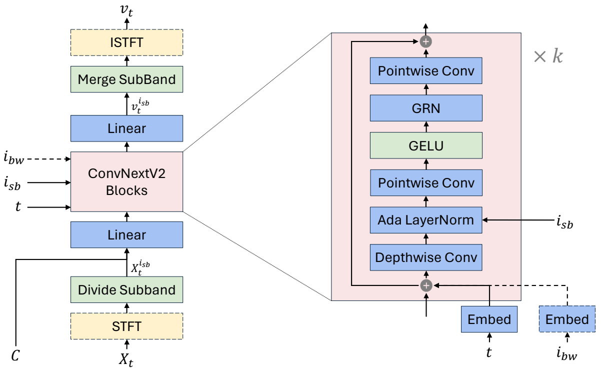


Figure 1: The overall structure for RFWave. i_{sb} is the subband index, C is the conditional input, which can be an Encodec token or Mel-spectrogram, and i_{bw} is the EnCodec bandwidth index. Modules enclosed in a dashed box, as well as dashed arrows, are considered optional.

inference. This significantly reduces inference latency. Moreover, independently modeling the subbands reduces error accumulation. As discussed in (Roman et al., 2023), conditioning higher bands on lower ones can lead to an error accumulation, which means inaccuracies in the lower bands can adversely affect the higher bands during inference.

The model maps a noisy sample (X_t) to its velocity (v_t). For each subband, the subband's noisy sample ($X_t^{i_{sb}}$) is fed into the ConvNeXtV2 backbone to predict its velocity ($v_t^{i_{sb}}$) conditioned on time (t), the subband index (i_{sb}), the conditional input (C, the Mel-spectrogram or the EnCodec (Défossez et al., 2022) tokens), and an optional EnCodec bandwidth index (i_{bw}). The detailed structure of the ConvNeXtV2 backbone is shown in Figure 1. We employ Fourier features as described in (Kingma et al., 2021). The $X_t^{i_{sb}}$, C, and Fourier features are concatenated along the channel dimension and then passed through a linear layer, forming the input that is fed into a series of ConvNeXtV2 blocks. The sinusoidal t embedding, along with the optional i_{bw} embedding, are element-wise added to the input of each ConvNeXtV2 block. The i_{bw} is utilized during the decoding of EnCodec tokens, enabling a single model to support EnCodec tokens with various bandwidths. Furthermore, the i_{sb} is incorporated via an adaptive layer normalization module, which utilizes learnable embeddings as described in (Siuzdak, 2023; Xu et al., 2019). The other components are identical to those within the ConvNeXtV2 architecture, details can be found in (Woo et al., 2023).

Our methodology offers two modeling options. The first involves mapping Gaussian noise to the waveform directly in the time domain, wherein X_0, X_1, X_t and v_t all reside in the time domain. The second option maps Gaussian noise to the complex spectrogram, placing X_0, X_1, X_t and v_t in the frequency domain. Notably, $X_t^{i_{sb}}$ and $v_t^{i_{sb}}$ are consistently represented in the frequency domain, as detailed in the following paragraphs, ensuring that the neural network runs at the frame level. By processing frame-level features, our model achieves greater memory efficiency compared to diffusion vocoders like PriorGrad, which operate at the level of waveform sample points. While PriorGrad (gil Lee et al., 2022) can train on 6-second[1] audio clips at 44.1 kHz within 30 GB of GPU memory, our model is capable of handling 177-second clips with the same memory resources.

Operating with X_t in the Time Domain and Waveform Equalization Since our model is designed to function at the frame level, when X_t and v_t are in the time domain (specially, X_1 is the waveform and X_0 is noise of the identical shape, with X_t and v_t derived from (3)), the use of STFT and ISTFT, as illustrated in Figure 1, becomes necessary. The dimension of X_t and v_t adhere to $[1, T]$[2], where T is the waveform length in sample points. After the STFT operation, we extract the subbands by equally dividing the full-band complex spectrogram, getting $X_t^{i_{sb}}$. Each subband $X_t^{i_{sb}}$ is then processed independently by the backbone as an individual sample to predict the corresponding $v_t^{i_{sb}}$. These predictions are subsequently merged back together before the ISTFT operation. The $X_t^{i_{sb}}$ processed by the backbone thus has dimensions of $[2d_s, F]$, where d_s represents the number of

---

[1] The duration of an audio clip is calculated as the batch size multiplied by the number of segment frames and hop length, divided by the sampling rate.

[2] For simplicity, the batch dimension is not included in the discussion.

frequency bins in a subband's complex spectrum and F the number of frames. The real and imaginary parts of each subband are interleaved to form a $2d_s$-dimensional feature.

White Gaussian noise has uniform energy across frequency bands, but waveform energy profiles vary significantly. For example, speech energy decays exponentially with frequency, while music maintains a consistent distribution (Schnupp et al., 2011). These differences challenge diffusion model training, making energy equalization across bands beneficial (Roman et al., 2023). In the time-domain model, a bank of Pseudo-Quadrature Mirror Filters (PQMF) is employed to decompose the input waveform into subbands. Subsequently, these subbands are equalized and then recombined to form the equalized waveform. The performance of the PQMF bank exhibits a modest enhancement compared to the array of band-pass filters employed in (Roman et al., 2023). It's important to note that the PQMF is solely utilized for waveform equalization and holds no association with the division of the complex spectrogram into subbands. For waveform equalization, mean-variance normalization is employed, utilizing the exponential moving average of mean and variance of each waveform subband computed during training. This approach ensures that the transformation can be effectively inverted using the same statistics.

Alternative Approach: Operating with X_t in the Frequency Domain and STFT Normalization When X_t and v_t reside in the frequency domain (i.e., X_1 is the waveform's complex spectrogram and X_0 is noise of the identical shape), STFT and ISTFT, as shown in Figure 1, are unnecessary. The dimensions of X_t and v_t are $[2d, F]$, where d is the number of frequency bins in the complex spectrogram. By equally partitioning the full-band complex spectrogram, $X_t^{i_{sb}}$ is extracted, resulting in a shape of $[2d_s, F]$, which is then processed by the ConvNeXtV2 backbone. In the frequency-domain model, the waveform is transformed to a complex spectrogram without equalization. The preprocessing involves the dimension-wise mean-variance normalization of the complex spectrogram.

During inference, operating with X_t and v_t in the time domain requires both STFT and ISTFT at each sampling step. In contrast, when X_t and v_t are in the frequency domain, only a single ISTFT is needed after the entire sampling process. Despite this computational overhead, our experiments show that the time-domain configuration achieves slightly better performance, particularly in preserving high-frequency details. Sampling algorithms for the two distinct approaches, one in the time domain and the other in the frequency domain, are provided in Appendix Section A.9.1.

3.2 LOSS FUNCTIONS

Energy-balanced Loss In preliminary experiments, we noticed low-volume noise in expected silent regions. We attribute this to the property of Mean Square Error (MSE) used in (2). The MSE measures the absolute distortion between the predicted values and the ground truth. In silent regions, small absolute errors contribute minimally to the overall MSE loss, so the model does not prioritize eliminating them during training. This results in the model's inability to effectively suppress minor deviations in silent areas, potentially leading to perceptible noise. In contrast, larger errors in high-amplitude regions have a significant impact on the MSE loss, causing the model to focus more on reducing errors in these areas during training.

We propose energy-balanced loss to mitigate this problem. Our energy-balanced loss is designed to weight errors differently depending on the region's volume (or energy) accross the time-axis. Specifically, for each frequency subband, we compute the standard deviation along the feature dimension of the ground truth velocity to construct a weighting coefficient of size $[1, F]$. This vector is reflective of the frame-level energy of the respective subband, as depicted in Figure A.1. Subsequently, both the ground truth and predicted velocity are divided by this vector before proceeding to the subsequent steps. For the frequency domain model, the training objective defined in (2) is adjusted as follows:

$$\min_{v} \mathbb{E}_{X_0\sim\pi_0,(X_1,\mathcal{C})\sim D}\left[\int_0^1 ||\ (X_1 - X_0)/\sigma - v(X_t, t \mid \mathcal{C})/\sigma\ ||^2 \,\mathrm{d}t\right], \quad \text{with } \sigma = \sqrt{\mathrm{Var}_1(X_1 - X_0)} \quad\text{and}\quad X_t = tX_1 + (1-t)X_0, \tag{5}$$

where D represents the dataset with paired X_1 and $\mathcal{C}$, and Var_1 calculates the variance along the feature dimension. For the time domain model, this energy balancing operation precedes the ISTFT process. This approach helps to minimize the relative error in low-volume regions. Our experimental results demonstrate that this method enhances overall performance, benefiting not just the silent parts.

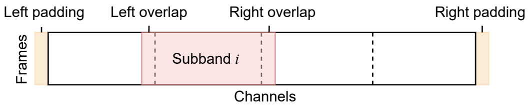


Figure 2: An illustration of dividing complex spectrograms into subbands. The area highlighted in pink represents a subband, while the section enclosed by the two dashed vertical lines indicates the main section.

Overlap Loss Within the multi-band structure, each subband is predicted independently, potentially resulting in inconsistencies among them. To mitigate these inconsistencies, we introduce an overlap loss. This involves maintaining overlaps between the subbands when dividing the full-band complex spectrograms. A detailed illustration of this scheme is provided in Figure 2 and described in detail in Appendix Section A.3. In this paper, we employ an 8-dimensional overlap. During the training phase, the MSEs of the overlapped predictions is minimized. In the inference phase, the overlaps are removed, and all subbands are merged to recreate the full-band complex spectrograms.

As each subband is predicted, the model internally maintains consistency between its overlap section and the main section. The overlap serves as an anchor to maintain consistency among the subbands. Modeling higher bands based on lower bands rather than predicting all subbands in parallel can increase consistency between subbands. However, at the inference stage, if the lower bands are incorrectly predicted, the higher bands conditioned on them would also be inaccurate (Roman et al., 2023). RFWave addresses this by using overlap loss to maintain consistency between subbands during training. At the inference stage, subbands are predicted independently, so any error occurring in one subband does not negatively affect the others.

STFT Loss The magnitude spectrogram derived loss (Arik et al., 2018) is extensively utilized in GAN-based vocoders, such as HiFi-GAN (Kong et al., 2020a) and Vocos (Siuzdak, 2023), both of which leverage a Mel-spectrogram loss. Nevertheless, its use in diffusion-based vocoders is relatively rare. This might stem from its lack of direct compatibility with the formalization of a noise prediction diffusion model. Here we adopt STFT loss for RFWave. According to (5), the model's output serves as an approximation for velocity:

$$v(X_t, t \mid \mathcal{C}) \approx \frac{\mathrm{d}}{\mathrm{d}t} X_t = X_1 - X_0. \tag{6}$$

Hence, at time t, an approximation for X_1 is:

$$\widetilde{X}_1 \approx X_0 + v(X_t, t \mid \mathcal{C}). \tag{7}$$

The STFT loss can be applied on the approximation $\widetilde{X}_1$, incorporating the spectral convergence loss and log-scale STFT-magnitude loss as proposed in (Arik et al., 2018), which are detailed in Appendix Section A.4. According to our experimental findings, the STFT loss effectively reduces artifacts in the presence of background noise.

3.3 SELECTING TIME POINTS FOR EULER METHOD

Liu et al. (2023) employ equal step intervals for the Euler method, as illustrated in (4). Here, we propose selecting time points for sampling based on the straightness of transport trajectories. The time points are chosen such that the increase in straightness is equal across each step. The straightness of a learned velocity field v is defined as $S(v) = \int_0^1 \mathbb{E} \mid\mid (X_1 - X_0) - v(X_t, t \mid \mathcal{C}) \mid\mid^2 \mathrm{d}t$, which describes the deviation of the velocity along the trajectory. A smaller $S(v)$ means straighter trajectories. This approach ensures that the difficulty of each Euler step remains consistent, requiring the model to take more steps in more challenging regions. Using the same number of sampling steps, this method outperforms the equal interval approach. The implementation details are provided in Appendix Section A.5, while the algorithm is presented in Appendix Section A.9.2. We refer to this approach as **equal straightness**.

4 EXPERIMENTS AND ANALYSIS

4.1 EXPERIMENTAL SETUP

Overview We evaluate RFWave using both Mel-spectrograms and discrete EnCodec tokens as inputs. For Mel-spectrogram inputs, we first benchmark RFWave against existing diffusion vocoders to demonstrate its superiority. We then compare it with widely-used GAN models to highlight its practical applicability and advantages. For discrete EnCodec tokens, we evaluate RFWave's efficiency in reconstructing high-quality audio from compressed representations across diverse domains. Finally, we conduct ablation studies and further analysis to examine the effects of the individual components of RFWave.

Data For Mel-spectrogram inputs, we conduct two evaluations. When benchmarking against diffusion vocoders, we train separate models on LibriTTS (Zen et al., 2019) (speech), MTG-Jamendo (Bogdanov et al., 2019) (music), and Opencpop (Wang et al., 2022) (vocal) datasets and test each model on its respective dataset to ensure comprehensive comparison across various audio categories. When comparing RFWave to widely used GAN-based models, we train a model on LibriTTS and evaluate its in-domain performance on the LibriTTS test set. Additionally, we assess the out-of-domain generalization ability of this LibriTTS-trained model by testing it on the MUSDB18 (Rafii et al., 2017) test subset.

For discrete EnCodec token inputs, we follow convention by training a universal model on a large-scale dataset. This dataset combines Common Voice 7.0 (Ardila et al., 2019) and clean data from DNS Challenge 4 (Dubey et al., 2022) for speech, MTG-Jamendo (Bogdanov et al., 2019) for music, and FSD50K (Fonseca et al., 2021) and AudioSet (Gemmeke et al., 2017) for environmental sounds. Recognizing the lack of a comprehensive test set for universal audio codec models, we constructed a unified evaluation dataset comprising 900 test audio samples from 15 external datasets, covering speech, vocals, and sound effects. Detailed information about this test set is provided in Table A.4.

Baseline and Evaluation Metrics We conducted a comprehensive benchmark comparison of RFWave against a series of state-of-the-art models. For Mel-spectrogram inputs, we utilized PriorGrad (gil Lee et al., 2022) and FreGrad (Nguyen et al., 2024) as baselines for diffusion-based models. Vocos (Siuzdak, 2023) and BigVGAN (gil Lee et al., 2023) served as baselines for GAN-based methods, reflecting their widespread use in real-world applications. For discrete EnCodec token inputs, we compared with EnCodec (Défossez et al., 2022) and MBD (Roman et al., 2023).

We trained PriorGrad and FreGrad on LibriTTS, Opencpop, and MTG-Jamendo using their open-source details. For Vocos, BigVGAN, EnCodec, and MBD, we used public pre-trained models. We followed the authors' recommended sampling steps: 6 for PriorGrad, 50 for FreGrad, and 20 for MBD. Model sources are in Table A.2.

For objective evaluation, we use ViSQOL (Chinen et al., 2020) to assess perceptual quality, employing speech-mode for 22.05/24 kHz waveforms and audio-mode for 44.1 kHz waveforms. Additional metrics include the Perceptual Evaluation of Speech Quality (PESQ) (Rix et al., 2001), the F1 score for voiced/unvoiced classification (V/UV F1), and the periodicity error (Periodicity) (Morrison et al., 2022). For subjective evaluation, we conduct crowd-sourced assessments, employing a 5-point Mean Opinion Score (MOS) to determine the audio naturalness, ranging from 1 ('poor/unnatural') to 5 ('excellent/natural'), more details provided in the Appendix Section A.10.

Implementation We use the time-domain model with three enhanced loss functions as the default. The RFWave backbone contains 8 ConvNeXtV2 blocks, and the complex spectrogram is divided into 8 equally spanned subbands. For evaluation, we use 10 sampling steps. Table A.3 lists the parameters used for extracting Mel-spectrograms and complex spectrograms from datasets with different sample rates. Further implementation details and computational resource requirements are provided in the Appendix Sections A.1 and A.2, respectively.

4.2 EXPERIMENTAL RESULTS ON MEL-SPECTROGRAMS INPUT

Comparison with Diffusion-based Method In this portion, we compare RFWave[3] with PriorGrad and FreGrad by training separate models for each method on the LibriTTS, MTG-Jamendo, and Opencpop datasets and evaluating each model on its respective test set. Table 1 presents overall results, while Table A.5 provides detailed metrics for each audio category. RFWave consistently

---

[3] This evaluation used an earlier RFWave version without the optimized sampling method (equal straightness) from Section 3.3. Despite this, the results convincingly demonstrate RFWave's superiority over other diffusion vocoders.

outperforms PriorGrad and FreGrad in both objective and subjective metrics. The superior MOS score achieved by RFWave is likely due to its ability to produce clearer and more consistent harmonics, particularly in high-frequency ranges, which contributes to better overall audio quality. A collection of spectrograms and their corresponding analyses can be found in Appendix Section A.6.

Comparison with GAN-based Method When comparing RFWave to widely used state-of-the-art GAN-based models (BigVGAN, Vocos), we used models trained on LibriTTS. Besides evaluating on the LibriTTS testset, to assess the models' robustness and extrapolation capabilities, following the methodology in (gil Lee et al., 2023), we also evaluated these models' performance on the MUSDB18 dataset. It can be observed in Table 2 that on the in-domain test set (LibriTTS), RFWave is generally on par with state-of-the-art GAN-based models. However, on the out-of-domain test set (MUSDB18), according to the results in Table 3, RFWave shows significant advantages over BigVGAN and Vocos, even though BigVGAN employed Snake activation (Ziyin et al., 2020) to enhance its out-of-domain data generation capabilities. This demonstrates that RFWave, as a diffusion-type model, offers clear advantages in generalization and robustness compared to GAN-based models. For detail, we observe that GAN-based methods tend to generate horizontal lines in the high-frequency regions of the spectrograms, as exemplified in Figure A.7. In contrast, RFWave consistently produces clear high-frequency harmonics, even when applied to out-of-domain data.

Table 1: Average Mean Opinion Score (MOS) and objective evaluation metrics for RFWave, PriorGrad and FreGrad across various test sets. MOS is provided with 95% confidence interval.

Model	MOS ↑	PESQ ↑	ViSQOL ↑	V/UV F1 ↑	Periodicity ↓
RFWave	**3.95**±0.09	**4.202**	**4.456**	**0.979**	**0.070**
PriorGrad	3.75±0.09	3.612	4.347	0.974	0.082
FreGrad	2.99±0.14	3.640	4.179	0.973	0.087
Ground truth	4.00±0.09	-	-	-	-

Table 2: MOS and objective evaluation metrics for RFWave, BigVGAN and Vocos on LibriTTS.

Model	MOS ↑	PESQ ↑	ViSQOL ↑	V/UV F1 ↑	Periodicity ↓
RFWave	**3.82**±0.12	**4.304**	4.579	0.967	0.091
BigVGAN	3.78±0.11	4.240	**4.712**	**0.978**	**0.067**
Vocos	3.74±0.10	3.660	4.696	0.958	0.104
Ground truth	3.91±0.10	-	-	-	-

Table 3: MOS for RFWave, BigVGAN and Vocos on MUSDB18.

Model	Vocals	Drums	Bass	Others	Mixture	Average
RFWave	**3.46**±0.14	3.65±0.10	**3.62**±0.10	**3.54**±0.11	**4.04**±0.11	**3.67**±0.05
BigVGAN	3.42±0.13	**3.68**±0.09	3.50±0.12	3.33±0.13	3.58±0.10	3.51±0.05
Vocos	3.04±0.15	3.54±0.11	2.96±0.12	2.88±0.15	3.08±0.14	3.10±0.06
Ground truth	3.62±0.14	3.71±0.14	3.75±0.12	3.79±0.12	4.17±0.10	3.80±0.05

4.3 EXPERIMENTAL RESULTS ON DISCRETE ENCODEC TOKENS INPUT

We evaluated RFWave against EnCodec and MBD across various bandwidths for discrete EnCodec tokens inputs. For RFWave, we used Classifier-Free Guidance (CFG) (Ho & Salimans, 2021) with a 2.0 coefficient. CFG showed little improvement for Mel-spectrogram inputs, but effective for EnCodec tokens, likely due to CFG tending to be more impactful when the input is more compressed.

Table 4 displays the average MOS and objective metrics for EnCodec tokens with varying bandwidths across different auditory contexts. Meanwhile, Table A.6 offers the detailed results for each category. RFWave excels in all metrics, achieving optimal scores, except for ViSQOL. While using a larger bandwidth generally improves performance, for RFWave, an increase in bandwidth from 6.0 kbps to 12.0 kbps results in only a slight enhancement in MOS. EnCodec's GAN-based decoder attains the highest ViSQOL. Different model families may produce distinct waveform footprints, with ViSQOL showing a subtle bias for GAN-based models.

Table 4: Average MOS and objective metrics for RFWave, EnCodec and MBD across various test sets.

Bandwidth	Model	MOS ↑	PESQ ↑	ViSQOL ↑	V/UV F1 ↑	Periodicity↓
1.5 kbps	RFWave$_{\text{(CFG2)}}$	**3.17**±0.22	**1.797**	3.108	**0.914**	**0.193**
	EnCodec	2.23±0.23	1.708	**3.518**	0.906	0.199
	MBD	3.01±0.19	1.699	2.982	0.901	0.212
3.0 kbps	RFWave$_{\text{(CFG2)}}$	**3.52**±0.25	**2.444**	3.570	**0.939**	**0.145**
	EnCodec	2.79±0.25	1.934	**3.793**	0.930	0.166
	MBD	3.06±0.23	2.310	3.402	0.922	0.171
6.0 kbps	RFWave$_{\text{(CFG2)}}$	**3.69**±0.16	**2.936**	3.892	**0.954**	**0.117**
	EnCodec	3.10±0.15	2.432	**4.091**	0.951	0.126
	MBD	3.43±0.15	2.488	3.582	0.929	0.168
12.0 kbps	RFWave$_{\text{(CFG2)}}$	**3.73**±0.16	**3.270**	4.124	**0.965**	**0.099**
	EnCodec	3.55±0.15	2.892	**4.291**	0.963	0.105
	MBD	-	-	-	-	-
Ground truth		4.07±0.14	-	-	-	-

4.4 ANALYSIS

Ablations We evaluate the model's performance in both frequency and time domains, then incrementally add the three loss functions and equal straightness to the better-performing model for further analysis. We conduct ablation studies on LJSpeech (Ito & Johnson, 2017), using 250 test sentences for Mel-spectrogram-based waveform reconstruction, summarized in Table 5. Despite the metrics not consistently correlating with human evaluations, they accurately captured the quality improvements brought about by design modifications (Roman et al., 2023). The model operating in the time domain outperforms its frequency domain counterpart, as the ISTFT operation in the former (Figure 1) introduces periodic signals. The inverse Discrete Fourier Transform (IDFT) matrix comprises sinusoidal functions of frequencies $[0, 1/N, ..., (N-1)/N]$, where N is the number of FFT points (Oppenheim & Schafer, 1975). This mechanism is similar to the Snake activation (Ziyin et al., 2020) in BigVGAN (gil Lee et al., 2023), which also introduces periodic signals into the generator. Energy-balanced loss and equal straightness improves the overall performance. The STFT loss enhances PESQ scores but adversely affects ViSQOL and periodicity error. This occurs because the model tends to prioritize magnitude over phase when applying STFT loss. This trade-off is essential for effectively eliminating artifacts, especially in the presence of background noise. Although overlap loss slightly decreases PESQ, omitting it results in a noticeable transition between subbands. Example spectrograms illustrating the effects of omitting STFT loss and overlap loss are provided in Figures A.8 and A.9, respectively.

Table 5: Objective metrics assess the model's performance in the frequency domain and the time domain, where three loss functions and equal straightness are applied incrementally.

Setting	PESQ ↑	ViSQOL ↑	V/UV F1 ↑	Periodicity ↓
frequency	3.872	4.430	0.948	0.144
time	4.127	4.551	0.957	0.124
+ energy-balanced loss	4.181	4.598	0.965	0.110
+ overlap loss	4.158	4.599	0.959	0.122
+ STFT loss	4.211	4.578	0.961	0.119
+ equal straightness	4.275	4.654	0.966	0.100

Futher Analysis Using the model with an 8-layer ConvNeXtV2 backbone, time domain, and three loss functions as the baseline, we first experimented with the DDPM (Ho et al., 2020) approach. Utilizing the noise schedule from DiffWave and PriorGrad, the performance with 50 sampling steps fell short of the baseline, as demonstrated in Table 6. This highlights that **Rectified Flow is critical for our model to efficiently generate high-quality audio samples**. Next, we experimented with a

ResNet (He et al., 2016) backbone with a similar number of parameters. This configuration slightly reduced efficiency and performance in objective metrics (Table 6), indicating that the **ConvNeXtV2 backbone enhances both efficiency and audio quality**. Finally, we experimented with models of different sizes. Specifically, we tested ConvNextV2 backbones of half and double sizes, with detailed configurations provided in A.1. **Increasing the backbone size improves performance but decreases efficiency** (Table 6).

Table 6: Objective metrics for further analysis. xRT stands for the speed at which the model can generate speech in comparison to real-time.

Setting	GPU xRT ↑	PESQ ↑	ViSQOL ↑	V/UV F1 ↑	Periodicity ↓
baseline	162.59	4.211	4.578	0.961	0.119
DDPM, 50 steps	34.40	2.790	4.499	0.939	0.174
ResNet	150.95	3.865	4.509	0.955	0.128
half size	228.70	4.128	4.553	0.959	0.120
double size	111.62	4.285	4.636	0.968	0.100

4.5 INFERENCE SPEED

We perform inference speed benchmark tests using an NVIDIA GeForce RTX 4090 GPU. The implementation was done in PyTorch (Paszke et al., 2019), and no specific hardware optimizations were applied. The inference was carried out with a batch size of 1 sample, utilizing the LJSpeech test set, resampled to model's sampling rate. Table 7 displays the model size, synthesis speed and GPU memory consumption of the models. RFWave is more than twice as fast as BigVGAN and consumes less GPU memory, thereby eliminating latency as a barrier for practical applications. This speed advantage becomes even more pronounced when synthesizing high-resolution audio (44.1/48 kHz). Frame-level models like RFWave significantly outperform sample-point-based models in high-resolution audio synthesis in terms of speed. RFWave can easily adjust the complex spectrogram's window length and hop length for 44.1 kHz sampling, maintaining low computational complexity, whereas BigVGAN requires additional upsampling blocks. Vocos serves as a strong baseline, given its efficiency in requiring only a single forward pass and operating at the frame level.

Table 7: Model footprint and synthesis speed. xRT stands for the speed at which the model can generate speech in comparison to real-time.

Model	Parameters (M)	GPU xRT ↑	GPU Memory (MB)	Sampling steps
RFWave	18.1	162.59	780	10
BigVGAN	107.7	72.68	1436	1
Vocos$_{\text{(ISTFT)}}$	13.5	**2078.20**	**590**	1
PriorGrad	2.6	16.67	4976	6
FreGrad	**1.7**	7.50	2720	50
MBD	411.0	4.82	5480	20
RFWave$_{\text{(44.1KHz)}}$	20.5	152.58	902	10
BigVGAN$_{\text{(44.1KHz)}}$	116.5	39.03	1740	1

5 CONCLUSION

In this study, we propose RFWave, a multi-band Rectified Flow approach for audio waveform reconstruction. The model has been carefully designed to overcome the latency issues associated with diffusion models. RFWave stands out for its ability to generate complex spectrograms by operating at the frame level, processing all subbands concurrently. This concurrent processing significantly enhances the efficiency of the waveform reconstruction process. The empirical evaluations conducted in this research have demonstrated that RFWave achieves exceptional reconstruction quality. Moreover, it has shown superior computational efficiency by generating audio at a speed that is 160 times faster than real-time, comparable to GAN-based methods, making it practical for real-world applications.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

Sercan Arik, Heewoo Jun, and Gregory Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, PP:1–1, 11 2018. doi: 10.1109/LSP.2018.2880284.

Evelina Bakhturina, Vitaly Lavrukhin, Boris Ginsburg, and Yang Zhang. Hi-fi multi-speaker english tts dataset. *arXiv preprint arXiv:2104.01497*, 2021.

Dawn AA Black, Ma Li, and Mi Tian. Automatic identification of emotional cues in chinese opera singing. *ICMPC, Seoul, South Korea*, 2014.

Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The mtg-jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, United States, 2019. URL http://hdl.handle.net/10230/42015.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Michael Chinen, Felicia S. C. Lim, Jan Skoglund, Nikita Gureev, Feargus O'Gorman, and Andrew Hines. Visqol v3: An open source production ready objective speech and audio metric. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, 2020. doi: 10.1109/QoMEX48832.2020.9123150.

Soonbeom Choi, Wonil Kim, Saebyul Park, Sangeon Yong, and Juhan Nam. Children's song dataset for singing voice research. In *International Society for Music Information Retrieval Conference (ISMIR)*, volume 4, 2020.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *CoRR*, abs/2210.13438, 2022. doi: 10.48550/ARXIV.2210.13438. URL https://doi.org/10.48550/arXiv.2210.13438.

Hui-Peng Du, Ye-Xin Lu, Yang Ai, and Zhen-Hua Ling. Apnet2: High-quality and high-efficiency neural vocoder with direct prediction of amplitude and phase spectra. *arXiv preprint arXiv:2311.11545*, 2023.

Zhihao Du, Qian Chen, Shiliang Zhang, Kai Hu, Heng Lu, Yexin Yang, Hangrui Hu, Siqi Zheng, Yue Gu, Ziyang Ma, Zhifu Gao, and Zhijie Yan. Cosyvoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens. *CoRR*, abs/2407.05407, 2024. doi: 10.48550/ARXIV.2407.05407. URL https://doi.org/10.48550/arXiv.2407.05407.

Harishchandra Dubey, Vishak Gopal, Ross Cutler, Ashkan Aazami, Sergiy Matusevych, Sebastian Braun, Sefik Emre Eskimez, Manthan Thakker, Takuya Yoshioka, Hannes Gamper, and Robert Aichner. Icassp 2022 deep noise suppression challenge, 2022.

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2021.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 776–780. IEEE, 2017.

Sang gil Lee, Heeseung Kim, Chaehun Shin, Xu Tan, Chang Liu, Qi Meng, Tao Qin, Wei Chen, Sungroh Yoon, and Tie-Yan Liu. Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=_BNiN4IjC5.

Sang gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. BigVGAN: A universal neural vocoder with large-scale training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=iTtGCMDEzS_.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Rongjie Huang, Max W. Y. Lam, J. Wang, Dan Su, Dong Yu, Yi Ren, and Zhou Zhao. Fastdiff: A fast conditional diffusion model for high-quality speech synthesis. In *International Joint Conference on Artificial Intelligence*, 2022. URL https://api.semanticscholar.org/CorpusID:248300058.

Eric J. Humphrey, Simon Durand, and Brian McFee. Openmic-2018, September 2018. URL https://doi.org/10.5281/zenodo.1432913.

Keith Ito and Linda Johnson. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

J. Johnston. A filter family designed for use in quadrature mirror filter banks. In *ICASSP '80. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pp. 291–294, 1980. doi: 10.1109/ICASSP.1980.1171025.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pp. 2410–2419. PMLR, 2018.

Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain De Cheveigne. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech communication*, 27(3-4): 187–207, 1999.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 21696–21707. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b578f2a52a0229873fefc2a4b06377fa-Paper.pdf.

Yuma Koizumi, Heiga Zen, Kohei Yatabe, Nanxin Chen, and Michiel Bacchiani. Specgrad: Diffusion probabilistic model based neural vocoder with adaptive noise spectral shaping. In Hanseok Ko and John H. L. Hansen (eds.), *23rd Annual Conference of the International Speech Communication Association, Interspeech 2022, Incheon, Korea, September 18-22, 2022*, pp. 803–807. ISCA, 2022. doi: 10.21437/INTERSPEECH.2022-301. URL https://doi.org/10.21437/Interspeech.2022-301.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020a.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020b.

Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=XVjTT1nw5z.

Zhaorui Liu and Zijin Li. Music Data Sharing Platform for Computational Musicology Research (CCMUSIC DATASET), November 2021. URL https://doi.org/10.5281/zenodo.5676893.

Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1877–1884, 2016.

Max Morrison, Rithesh Kumar, Kundan Kumar, Prem Seetharaman, Aaron Courville, and Yoshua Bengio. Chunked autoregressive GAN for conditional waveform synthesis. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=v3aeIsY_vVX.

Tan Dat Nguyen, Ji-Hoon Kim, Youngjoon Jang, Jaehun Kim, and Joon Son Chung. Fregrad: Lightweight and fast frequency-aware diffusion vocoder. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2024.

Tu Anh Nguyen, Wei-Ning Hsu, Antony d'Avirro, Bowen Shi, Itai Gat, Maryam Fazel-Zarani, Tal Remez, Jade Copet, Gabriel Synnaeve, Michael Hassid, et al. Expresso: A benchmark and analysis of discrete expressive speech resynthesis. *arXiv preprint arXiv:2308.05725*, 2023.

Frederico S Oliveira, Edresson Casanova, Arnaldo Candido Junior, Anderson S Soares, and Arlindo R Galvão Filho. Cml-tts: A multilingual dataset for speech synthesis in low-resource languages. In *International Conference on Text, Speech, and Dialogue*, pp. 188–199. Springer, 2023.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing*. Prentice Hall international editions. Prentice-Hall, 1975. ISBN 9789380618388. URL https://books.google.com/books?id=vSzuVLBbp6cC.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina

Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.

Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. Musdb18-a corpus for music separation. 2017.

A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 2, pp. 749–752 vol.2, 2001. doi: 10.1109/ICASSP.2001.941023.

Robin San Roman, Yossi Adi, Antoine Deleforge, Romain Serizel, Gabriel Synnaeve, and Alexandre Défossez. From discrete tokens to high-fidelity audio using multi-band diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=dOanKg3jKS.

Jan Schnupp, Israel Nelken, and Andrew King. *Auditory neuroscience: Making sense of sound*. MIT press, 2011.

Yao Shi, Hui Bu, Xin Xu, Shaoji Zhang, and Ming Li. Aishell-3: A multi-speaker mandarin tts corpus and the baselines. *arXiv preprint arXiv:2010.11567*, 2020.

Hubert Siuzdak. Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis. *CoRR*, abs/2306.00814, 2023. doi: 10.48550/ARXIV.2306.00814. URL https://doi.org/10.48550/arXiv.2306.00814.

Yang Song, Jascha Narain Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ArXiv*, abs/2011.13456, 2020. URL https://api.semanticscholar.org/CorpusID:227209335.

Shinnosuke Takamichi, Kentaro Mitsui, Yuki Saito, Tomoki Koriyama, Naoko Tanji, and Hiroshi Saruwatari. Jvs corpus: free japanese multi-speaker voice corpus. *arXiv preprint arXiv:1908.06248*, 2019.

Hiroki Tamaru, Shinnosuke Takamichi, Naoko Tanji, and Hiroshi Saruwatari. Jvs-music: Japanese multispeaker singing-voice corpus. *arXiv preprint arXiv:2001.07044*, 2020.

Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. On catastrophic forgetting and mode collapse in generative adversarial networks. *CoRR*, abs/1807.04015, 2018. URL http://arxiv.org/abs/1807.04015.

Jean-Marc Valin and Jan Skoglund. Lpcnet: Improving neural speech synthesis through linear prediction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5891–5895. IEEE, 2019.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.

Yu Wang, Xinsheng Wang, Pengcheng Zhu, Jie Wu, Hanzhao Li, Heyang Xue, Yongmao Zhang, Lei Xie, and Mengxiao Bi. Opencpop: A high-quality open source chinese popular song corpus for singing voice synthesis. In Hanseok Ko and John H. L. Hansen (eds.), *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, pp. 4242–4246. ISCA, 2022. doi: 10.21437/INTERSPEECH.2022-48. URL https://doi.org/10.21437/Interspeech.2022-48.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext V2: co-designing and scaling convnets with masked autoencoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 16133–16142. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01548. URL https://doi.org/10.1109/CVPR52729.2023.01548.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/2f4fe03d77724a7217006e5d16728874-Paper.pdf.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020.

Geng Yang, Shan Yang, Kai Liu, Peng Fang, Wei Chen, and Lei Xie. Multi-band melgan: Faster waveform generation for high-quality text-to-speech. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 492–498, 2021. doi: 10.1109/SLT48900.2021.9383551.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

Liu Ziyin, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1583–1594. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1160453108d3e537255e9f7b931f4e90-Paper.pdf.

A APPENDIX

A.1 IMPLEMENTATION DETAILS

The RFWave backbone contains 8 ConvNeXtV2 blocks. Within each ConvNeXtV2 block, the depth-wise convolutional layer featuring a large kernel utilizes a kernel size of 7 and has a channel dimension of 512. The first and last 1x1 point-wise convolutional layers in the sequence possess channel dimensions of 512 and 1536, respectively. In Subsection 4.4, the half-size variant contains 8 ConvNeXtV2 blocks with point-wise convolutional layers of 384 and 1152, while the double-size variant features 16 ConvNeXtV2 blocks with point-wise convolutional layers of 512 and 1536.

During the extraction of complex coefficients for the model, we use the orthonormal Fast Fourier Transform (FFT) and its inverse (IFFT), with the normalization convention of dividing by $1/\sqrt{N}$ for both operations, here N is the FFT size. This approach ensures the spectrogram extracted is within a more reasonable range for modeling.

In terms of loss functions, we assign a weight of 1 to either the Rectified Flow loss or its energy-balanced variant. If employed, the overlap loss and STFT loss are assigned a weight of 0.01. These weights are determined by aligning the L2-norm of the gradients from the overlap loss and STFT loss with approximately 1/10 of that from the Rectified Flow loss.

Training details are also worth mentioning. Audio samples are randomly cropped to lengths of 32512 and 65024 for 22.05/24 kHz and 44.1 kHz waveforms, respectively. This is equivalent to a crop window of 128 frames for both sampling rates. We use a batch size of 64. The model optimization is performed using the AdamW optimizer with a starting learning rate of 2e-4 and beta parameters of (0.9, 0.999). A cosine annealing schedule is applied to reduce the learning rate to a minimum of 2e-6.

A.2 COMPUTATIONAL RESOURCE REQUIRED FOR THE EXPERIMENTS

The bulk of the experiments were carried out on personal computers and GPU servers, which were sourced from cloud service providers. Primarily, we utilized Nvidia-4090-24G and Nvidia-A100-80G GPUs for these tasks.

The specific hardware used and the corresponding time taken for training RFWave on various datasets are as follows:

Table A.1: GPU configurations and training duration for various datasets

Dataset	GPU configuration	Training duration
LJspeech	1×4090	2 days
LibriTTS	2×A100	5 days
Opencpop	1×4090	1 day
MTG-Jamendo	2×A100	7 days
EnCodec mixed dataset	4×A100	10 days

A.3 DEVIDING INTO SUBBANDS

The complex spectrogram is circularly padded in the feature dimension with a size of $(d_{ol}, d_{ol} - 1)$, where d_{ol} signifies the overlap size. The dimension of each subband's main section, denoted as d_m, is calculated by dividing $d - 1$ by the total number of subbands. Here, d represents the dimension of the complex spectrograms. The last subband is an exception, having an extra feature dimension but one less padding dimension. Subbands are extracted by applying a sliding window along the feature dimension, where the window has a size of $d_m + 2d_{ol}$ and shifts by d_m[4].

---

[4] The feature dimension of each subband, represented as d_s, equals $(d_m + 2d_{ol})$, accounting for the interleave of real and imaginary parts.

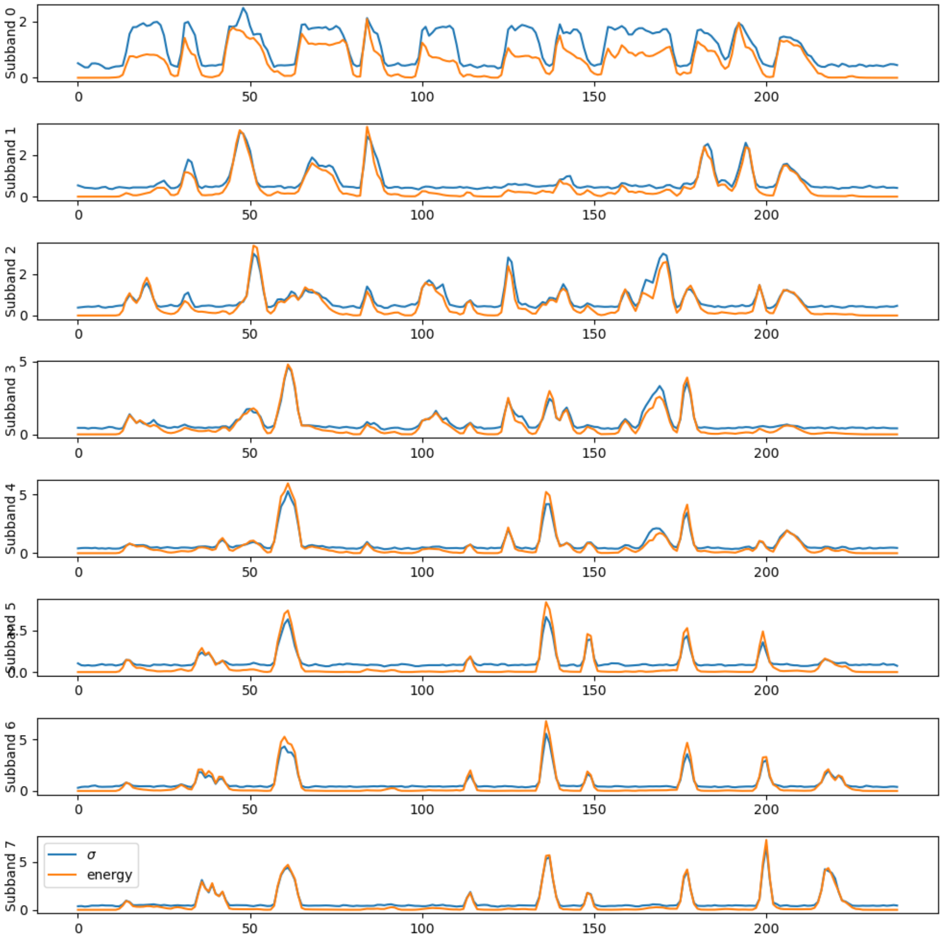


Figure A.1: The energy and weighting coefficients, represented by σ, display a consistent variation throughout the frames.

A.4 STFT LOSS

Spectral convergence (SC) loss

$$\| |X_1| - |\widetilde{X}_1| \|_F / \| |X_1| \|_F, \tag{8}$$

where X_1 is the ground truth and $\widetilde{X}_1$ is the estimated complex spectrogram using (7). The Frobenius norm, $\|\cdot\|_F$, applied over time and frequency, emphasizes large spectral components in SC loss.

Log-scale STFT-magnitude loss

$$\| \log(|X_1| + \epsilon) - \log(|\widetilde{X}_1| + \epsilon) \|_1, \tag{9}$$

where the L^1 norm, $\|\cdot\|_1$, along with a small constant ϵ, is used to accurately capture small-amplitude components in the log-scale STFT-magnitude loss.

A.5 SELECTING TIME POINTS FOR EULER METHOD

Straightness is calculated from a single batch with a size of 96. The Euler method uses 100 equal interval steps to estimate the integral in the straightness definition. Time points are selected such that the increase in straightness remains consistent across each interval. An example with 10 Euler steps is provided in Figure A.2. The time points are calculated only once per model, resulting in negligible computational load.

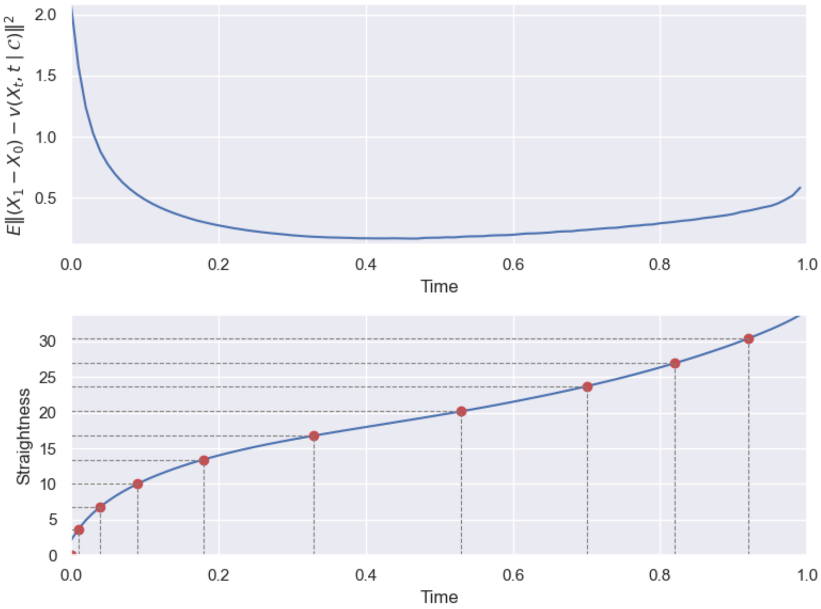


Figure A.2: Deviation (top) and straightness (bottom) over time: red dots mark Euler method time points, with constant increase in straightness across intervals.

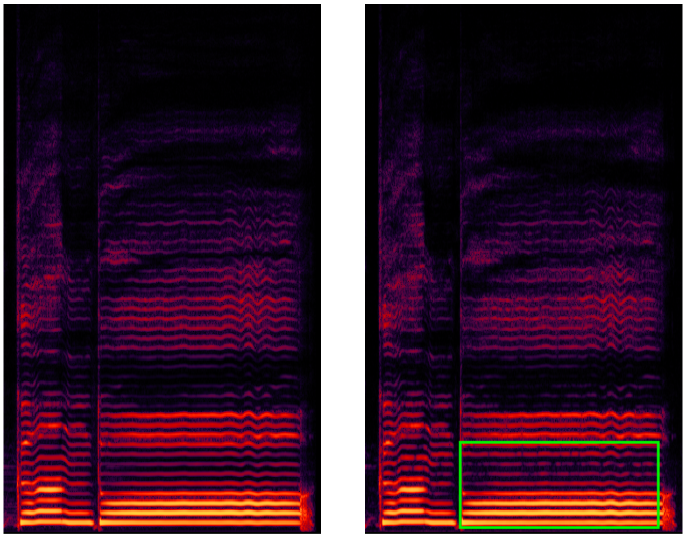
A.6 SPECTROGRAM EXAMPLES FOR COMPARISON WITH DIFFUSION-BASED METHOD

For all the spectrogram figures, we used Adobe Audition for spectrogram visualization, with the dynamic range being [-132, 0] dB.

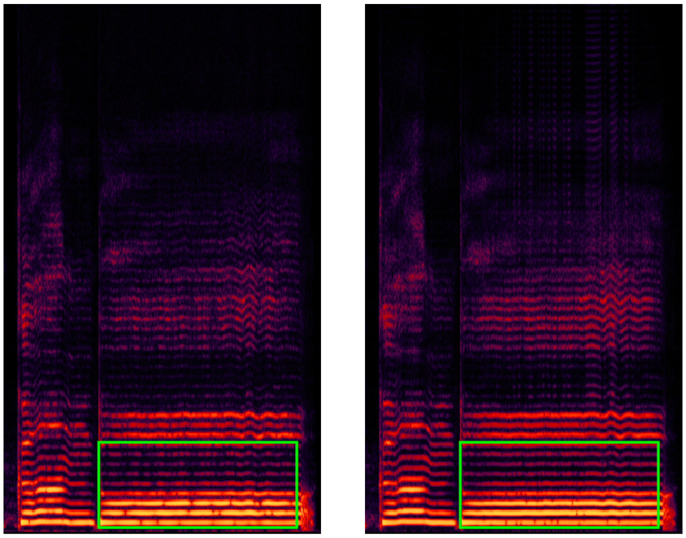
Figure A.3 showcases spectrogram examples generated by various models using the Opencpop dataset. When compared to the ground truth spectrogram, RFWave is seen to produce clean and stable harmonics, whereas the harmonics generated by other models exhibit minor discontinuities.

Figure A.4 exhibits spectrogram examples generated by different models utilizing the LibriTTS dataset. RFWave generate clear high-frequency harmonics, while PriorGrad and FreGrad result in blurred high-frequency harmonics.

Figure A.5 displays spectrogram examples generated by diverse models from the Jamendo dataset. Both RFWave and PriorGrad generate commendable spectrograms, with RFWave edging out slightly in the high-frequency range.

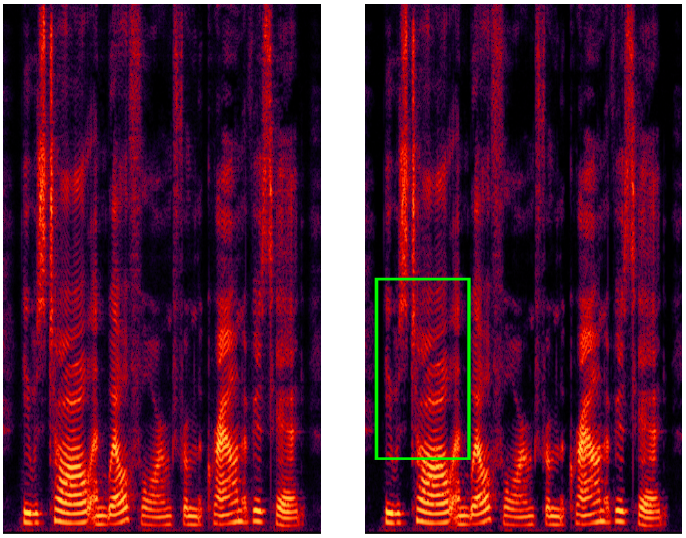


(a) Ground truth (b) RFWave

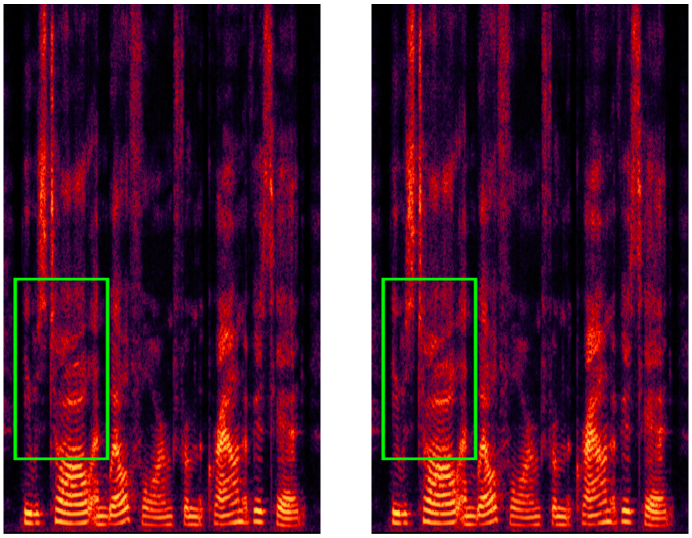


(c) PriorGrad (d) FreGrad

Figure A.3: Examples of spectrograms from Opencpop. Within the region marked out by the green box, the harmonics generated through PriorGrad and FreGrad show minor discontinuities.

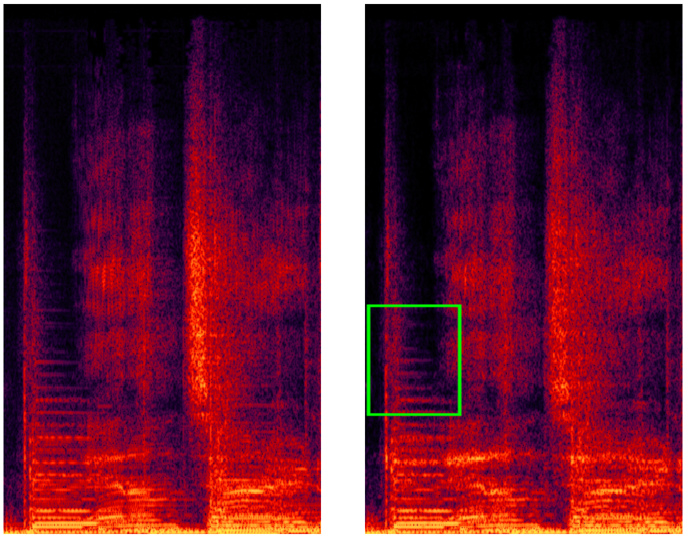


(a) Ground truth (b) RFWave

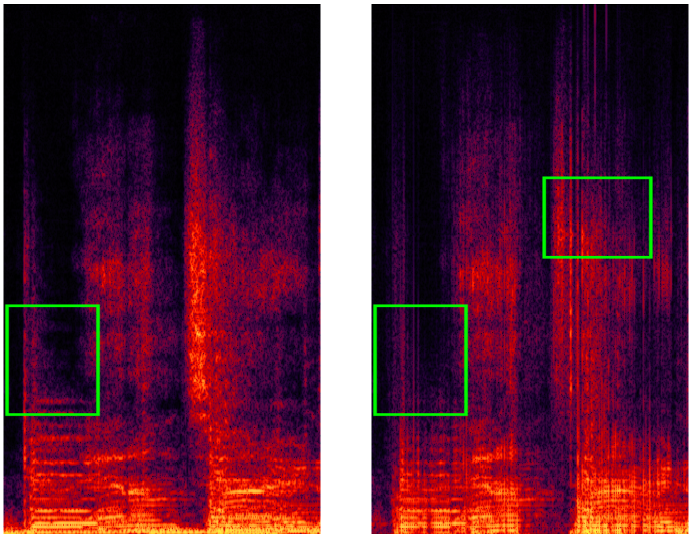


(c) PriorGrad (d) FreGrad

Figure A.4: Examples of spectrograms from LibriTTS. Within the green-boxed region, RFWave yields better harmonics than PriorGrad and FreGrad.



(a) Ground truth (b) RFWave



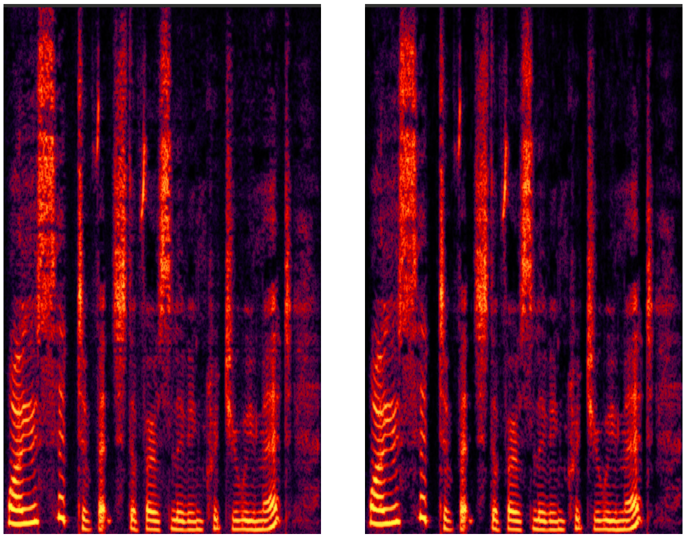
(c) PriorGrad (d) FreGrad

Figure A.5: Examples of spectrograms from MTG-Jamendo. Within the green-boxed region, RFWave generates more favorable high-frequency components compared to PriorGrad and FreGrad. Additionally, FreGrad gives rise to vertical line artifacts as highlighted by the top-right box.

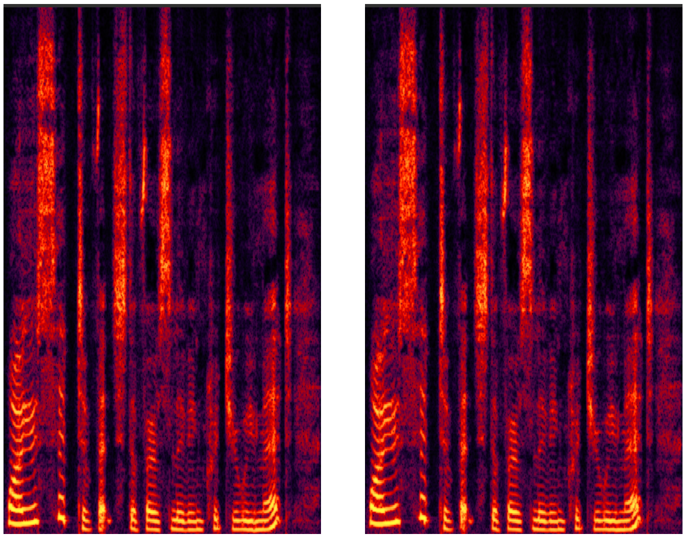
A.7 SPECTROGRAM EXAMPLES FOR COMPARISON WITH GAN-BASED METHOD

Figure A.6 illustrates that RFWave, BigVGAN and Vocos are capable of generating high-quality spectrograms when applied to the LibriTTS dataset. This demonstrates the effectiveness of each approach in handling in-distribution data. The spectrograms exhibit well-defined harmonics and minimal artifacts, highlighting the robustness of the models within their familiar domain.

Figure A.7 presents spectrogram examples generated on the MUSDB18 dataset, with the models having been trained on the LibriTTS dataset. A notable observation is that both BigVGAN and Vocos exhibit a tendency to produce horizontal lines in the high-frequency regions of the spectrograms. These artifacts are perceptually problematic, as they often result in a metallic sound quality that detracts from the naturalness of the audio. In stark contrast, RFWave consistently generates clear and well-defined high-frequency harmonics. This capability underscores RFWave's superior ability to maintain spectral fidelity, even when applied to out-of-distribution data, thereby enhancing the overall audio quality and realism.

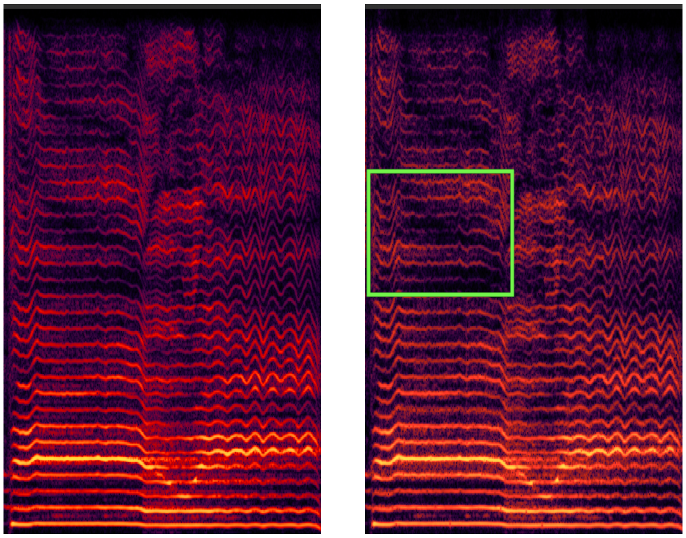


(a) Ground truth (b) RFWave

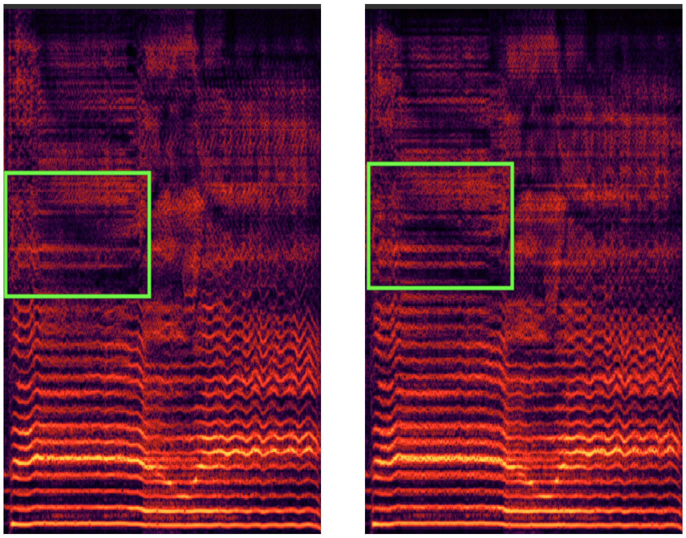


(c) BigVGAN (d) Vocos

Figure A.6: Examples of spectrograms from LibriTTS. All the methods produce high-quality spectrograms.

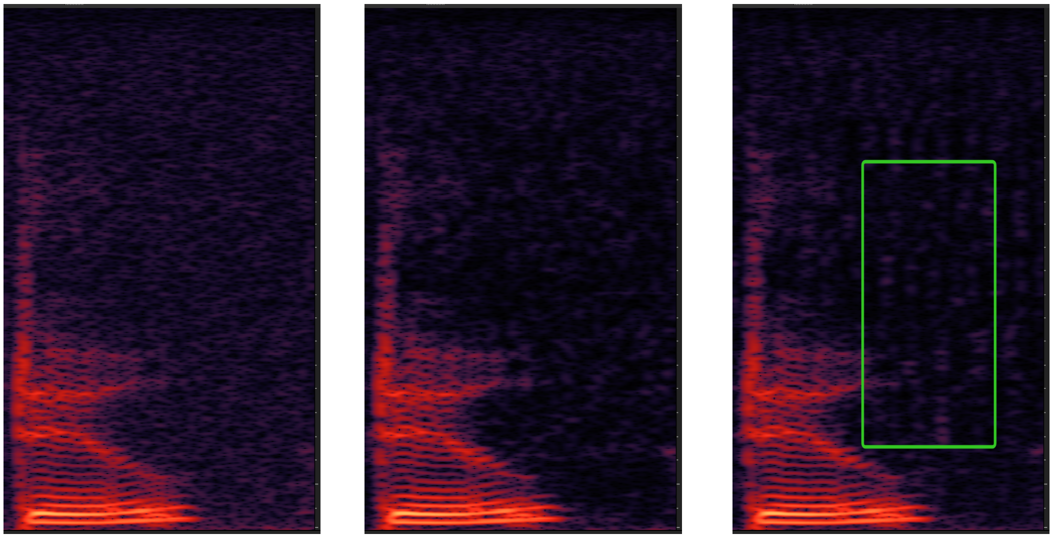


(a) Ground truth (b) RFWave



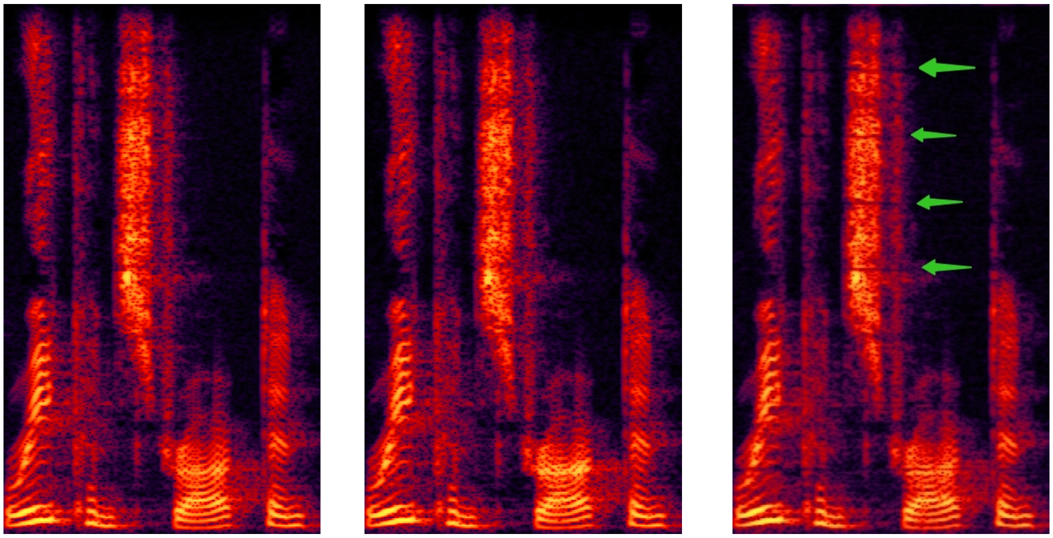
(c) BigVGAN (d) Vocos

Figure A.7: Examples of spectrograms from MUSDB18. Within the green-boxed region, both BigVGAN and Vocos have a propensity to generate horizontal lines within the high-frequency areas of the spectrograms. These artifacts present perceptual issues since they frequently lead to a metallic sound quality that diminishes the naturalness of the audio. In sharp contrast, RFWave produces well defined high-frequency harmonics, even when applied to out-of-distribution data.



(a) Ground truth (b) w/ STFT Loss (c) w/o STFT Loss

Figure A.8: The effect of STFT loss. There are some vertical patterns in the spectrogram of waveforms generated by a model without STFT loss, as highlighted by the green rectangular.



(a) Ground truth (b) w/ overlap Loss (c) w/o overlap Loss

Figure A.9: The effect of overlap loss. Omitting it results in a noticeable transition between subbands, as clearly illustrated by the green arrows.

A.8 TABLES

Table A.2: Code repositories for model training.

Model	repository
PriorGrad	https://github.com/microsoft/NeuralSpeech/tree/master/PriorGrad-vocoder
FreGrad	https://github.com/signofthefour/fregrad
Multi-Band Diffusion (MBD)	https://github.com/facebookresearch/audiocraft
EnCodec	https://github.com/facebookresearch/encodec
Vocos	https://github.com/gemelo-ai/vocos
BigVGAN	https://github.com/NVIDIA/BigVGAN

Table A.3: Parameters for extracting Mel-spectrograms and complex spectrograms.

Dataset	Sample rate (kHZ)	Window length	Hop length	FFT size	Mel bins
LJSpeech	22.05	1024	256	1024	100
LibriTTS	24	1024	256	1024	100
Opencpop	44.1	2048	512	2048	100
MTG-Jamendo	44.1	2048	512	2048	100
EnCodec Training	24	1280	320	1280	-

Table A.4: Construction of the general Encodec test dataset

Category	Subdirectory	Samples	Description
Speech	Expresso-test	50	From the Expresso dataset test set (Nguyen et al., 2023), randomly select 50 samples, segmenting any audio longer than 30 seconds into 10-second clips beforehand.
	HiFiTTS-test	50	Randomly select 50 samples from the test set of the Hi-Fi TTS dataset (Bakhturina et al., 2021).
	LibriTTS-test	50	Randomly select 50 samples from the test set of LibriTTS (Zen et al., 2019).
	Aishell3-test	50	Randomly select 50 samples from the test set of the Aishell3 dataset (Shi et al., 2020).
	JVS	50	Randomly select 50 samples from the entire JVS Corpus. (Takamichi et al., 2019).
	CML-TTS-test	50	Randomly select 50 samples from the test set of the CML-TTS dataset (Oliveira et al., 2023).
Vocal	Musdb-test-vocal	60	Segment the vocal track audio from the Musdb test set (Rafii et al., 2017) into 10-second clips, then randomly select 60 samples.
	CSD	20	Segment all audio in the CSD (Choi et al., 2020) into 10-second clips, then randomly select 20 samples.
	Opencpop	20	Randomly select 20 samples from the Opencpop dataset (Wang et al., 2022).
	ChineseOpera-monophonic	60	From the monophonic subset of the Chinese Opera Singing Dataset (Black et al., 2014), segment any audio longer than 30 seconds into 10-second clips, then randomly select 60 samples.
	JVS-Music	60	Randomly select 60 samples from the JVS-Music Corpus (Tamaru et al., 2020).
	RAVDESS	60	Randomly select 60 samples from the song subset of the RAVDESS corpus (Livingstone & Russo, 2018).
	Ccmusic-demo-vocal	20	From the demo audios of the Ccmusic Dataset (Liu & Li, 2021), select the vocal parts. Then, randomly select 20 samples from these for the test set, segmenting any audio longer than 30 seconds into 10-second clips beforehand.
Sound Effect	ESC-50	150	Randomly select 150 samples from the ESC-50 (Piczak, 2015).
	Musdb-test-accompaniment	40	Segment the accompaniment track audio from the test set of the Musdb dataset (Rafii et al., 2017) into 10-second clips, then randomly select 40 samples from these clips.
	Musdb-test-mixture	20	Segment the mixture track audio from the test set of the Musdb dataset (Rafii et al., 2017) into 10-second clips, then randomly select 20 samples from these clips.
	ChineseOpera-polyphonic	20	From the polyphonic subset of the Chinese Opera Singing Dataset (Black et al., 2014), segment any audio longer than 30 seconds into 10-second clips, then randomly select 20 samples.
	OpenMIC-test	40	Randomly select 40 samples from the test set of OpenMix 2018 (Humphrey et al., 2018).
	Ccmusic-demo-music	30	From the demo audios of the Ccmusic Dataset (Liu & Li, 2021), select the non-vocal parts. Then, randomly select 20 samples from these for the test set, segmenting any audio longer than 30 seconds into 10-second clips beforehand.

Table A.5: MOS and Objective evaluation metrics for RFWave, PriorGrad and FreGrad across various datasets.[5]

Dataset	Model	MOS ↑	PESQ ↑	ViSQOL ↑	V/UV F1 ↑	Periodicity ↓
LibriTTS	RFWave	3.83±0.14	4.228	4.595	0.968	0.090
	PriorGrad	3.70±0.16	3.820	4.134	0.960	0.100
	FreGrad	3.68±0.15	3.758	4.278	0.960	0.099
	Ground truth	3.88±0.15	-	-	-	-
Opencpop	RFWave	4.26±0.17	4.176	4.564	0.990	0.049
	PriorGrad	3.79±0.21	3.404	4.512	0.988	0.064
	FreGrad	3.73±0.20	3.522	4.507	0.986	0.074
	Ground truth	4.33±0.18	-	-	-	-
MTG-Jamendo	RFWave	3.90±0.15	-	4.317	-	-
	PriorGrad	3.79±0.14	-	4.412	-	-
	FreGrad	1.89±0.17	-	3.765	-	-
	Ground truth	3.93±0.15	-	-	-	-

---

[5] The MOS of the ground truth on LibriTTS differs from that presented in Table 2 because these two MOS tests were conducted separately.

Table A.6: MOS and objective evaluation metrics for RFWave, EnCodec and MBD across various test sets.

Test set	Bandwidth	Model	MOS	PESQ ↑	ViSQOL ↑	V/UV F1 ↑	Periodicity↓
Speech	1.5 kbps	RFWave$_{\text{(CFG2)}}$	3.29±0.47	1.774	3.102	0.913	0.178
		EnCodec	2.29±0.35	1.515	3.310	0.870	0.231
		MBD	2.94±0.29	1.659	2.793	0.901	0.195
	3.0 kbps	RFWave$_{\text{(CFG2)}}$	3.85±0.39	2.421	3.582	0.935	0.137
		EnCodec	2.85±0.47	1.967	3.746	0.919	0.166
		MBD	2.88±0.33	2.194	3.219	0.915	0.166
	6.0 kbps	RFWave$_{\text{(CFG2)}}$	4.05±0.26	2.974	3.913	0.952	0.109
		EnCodec	3.19±0.31	2.554	4.048	0.945	0.121
		MBD	3.51±0.30	2.372	3.410	0.924	0.161
	12.0 kbps	RFWave$_{\text{(CFG2)}}$	3.96±0.22	3.393	4.158	0.965	0.089
		EnCodec	3.52±0.22	3.104	4.250	0.961	0.095
		MBD	-	-	-	-	-
	Ground truth		4.13±0.23	-	-	-	-
Vocal	1.5 kbps	RFWave$_{\text{(CFG2)}}$	3.06±0.34	1.820	3.317	0.914	0.208
		EnCodec	1.94±0.38	1.900	3.931	0.941	0.166
		MBD	2.84±0.31	1.739	3.221	0.901	0.228
	3.0 kbps	RFWave$_{\text{(CFG2)}}$	3.47±0.34	2.467	3.793	0.942	0.152
		EnCodec	2.63±0.37	1.900	3.931	0.941	0.166
		MBD	3.26±0.45	2.426	3.655	0.929	0.175
	6.0 kbps	RFWave$_{\text{(CFG2)}}$	3.77±0.23	2.897	4.080	0.956	0.124
		EnCodec	2.92±0.23	2.310	4.206	0.956	0.131
		MBD	3.30±0.26	2.604	3.840	0.933	0.175
	12.0 kbps	RFWave$_{\text{(CFG2)}}$	3.83±0.31	3.147	4.273	0.964	0.109
		EnCodec	3.40±0.29	2.679	4.373	0.965	0.114
		MBD	-	-	-	-	-
	Ground truth		4.27±0.24	-	-	-	-
Soudn Effect	1.5 kbps	RFWave$_{\text{(CFG2)}}$	3.24±0.43	-	2.906	-	-
		EnCodec	2.62±0.42	-	3.314	-	-
		MBD	3.33±0.36	-	2.932	-	-
	3.0 kbps	RFWave$_{\text{(CFG2)}}$	3.11±0.59	-	3.334	-	-
		EnCodec	2.89±0.48	-	3.703	-	-
		MBD	3.11±0.54	-	3.331	-	-
	6.0 kbps	RFWave$_{\text{(CFG2)}}$	3.27±0.27	-	3.682	-	-
		EnCodec	3.17±0.24	-	4.020	-	-
		MBD	3.44±0.23	-	3.496	-	-
	12.0 kbps	RFWave$_{\text{(CFG2)}}$	3.33±0.28	-	3.942	-	-
		EnCodec	3.72±0.26	-	4.249	-	-
		MBD	-	-	-	-	-
	Ground truth		3.78±0.28	-	-	-	-

A.9 ALGORITHMS

We present a simplified version of the pseudocode along with a comprehensive Python implementation for the two sampling algorithms. The detailed Python code delves into the intricacies of subband processing and inverse equalization, offering deeper insights into the algorithmic steps involved. Additionally, we provide a Python implementation for selecting time points of equal straightness.

A.9.1 SAMPLING ALGORITHM

Following is the sampling algorithm (X_t in time domain or frequency domain) with its pseudocode and Python implementation.

For simplicity, in the pseudocode, we've omitted the subband-related operations. Here, $X_t^{\mathfrak{f}}$ represents the combination of all $X_t^{i_{sb}}$ shown in Figure 1, and $v_t^{\mathfrak{f}}$ represents the combination of all $v_t^{i_{sb}}$ shown in Figure 1. For a comprehensive understanding of the processing steps, please refer to the Python implementation.

Algorithm 1 Simplified Sampling Algorithm (X_t in Time Domain)

```
Require:
$nn\_model$: pre-trained neural network model
$C$: conditional input
$\{t_i\}_{i=0}^{N}$: time steps where $0 = t_0 < t_1 < ... < t_N = 1$
Ensure:
Generated audio waveform $wave$

$X_0 \sim \mathcal{N}(0,1)^{[1,T]}$    // Initialize with Gaussian noise
$X_t^{\mathfrak{t}} \leftarrow X_0$

for $i \leftarrow 0$ to $N-1$ do
    $dt \leftarrow t_{i+1} - t_i$    // Calculate step interval
    $X_t^{\mathfrak{f}} \leftarrow$ STFT($X_t^{\mathfrak{t}}$)    // Convert $X_t$ to frequency domain
    $v_t^{\mathfrak{f}} \leftarrow nn\_model$.predict($X_t^{\mathfrak{f}}, t_i, C$)    // $v_t^{\mathfrak{f}}$ in frequency domain
    $v_t^{\mathfrak{t}} \leftarrow$ ISTFT($v_t^{\mathfrak{f}}$)    // Convert $v_t$ to time domain
    $X_t^{\mathfrak{t}} \leftarrow X_t^{\mathfrak{t}} + v_t^{\mathfrak{t}} \cdot dt$    // Euler step in time domain
end for

$X_1 \leftarrow X_t^{\mathfrak{t}}$
$wave \leftarrow X_1$

return $wave$
```

Detailed Algorithm 1: Sample Time Domain

```python
def sample_time_domain(model, mel, ts):
    '''
    :param model: A pre-trained time-domain RFWave model.
    :param mel: A batch of mel-spectrogram data, [batch_size, channels, num_frames].
    :param ts: The selected sampling time points.
    :return: The reconstructed audio waveform, [batch_size, num_frames * hop_length].
    '''

    batch_size, num_frames = mel.shape[0], mel.shape[2]
    noise_in_t = torch.randn([batch_size, model.hop_length * num_frames])
    # the STFT operation in Figure 1.
    noise_in_f = torch.stft(noise_in_t, **model.stft_kwargs)
    # Divide the noise spectrogram into subbands and reshape into the batch dimension
    noise_in_f = model.get_subband(noise_in_f)
    # repeat the mel-spectrogram to match the subbands accordingly.
    mel = torch.repeat_interleave(mel, model.num_bands, 0)

    z_in_t = noise_in_t
    z_in_f = noise_in_f
    vs = []  # used for selecting sampling time points
    for i in range(len(ts) - 1):
        t = ts[i]
        dt = ts[i + 1] - ts[i]
        # the model runs at frame-level feature.
        v_in_f = model.predict(z_in_f, t, mel)
        # place the subbands to construct a full band feature
        v_in_f = place_subband(model, v_in_f)
        # the ISTFT operation in Figure 1.
        v_in_t = torch.istft(v_in_f, **model.stft_kwargs)
        # the Euler step operates in the time-domain
        z_in_t = z_in_t + v_in_t * dt
        # the STFT operation in Figure 1.
        z_in_f = torch.stft(z_in_t, **model.stft_kwargs)
        z_in_f = get_subband(model, z_in_f)
        vs.append(v_in_t)
    # append the straight line velocity
    vs.append(z_in_t - noise_in_t)
    # inverse the waveform equalization.
    z_in_t = model.wave_equalizer.inverse(z_in_t)
    return z_in_t, vs
```

Algorithm 2 Simplified Sampling Algorithm (X_t in Frequency Domain)

```
Require:
  $nn\_model$: pre-trained neural network model
  $C$: conditional input
  $\{t_i\}_{i=0}^{N}$: time steps where $0 = t_0 < t_1 < ... < t_N = 1$
Ensure:
  Generated audio waveform $wave$

  $X_0 \sim \mathcal{N}(0,1)^{[d,F]}$    // Initialize with Gaussian noise
  $X_t^{\mathfrak{f}} \leftarrow X_0$

  for $i \leftarrow 0$ to $N - 1$ do
      $dt \leftarrow t_{i+1} - t_i$    // Calculate step interval
      $v_t^{\mathfrak{f}} \leftarrow nn\_model$.predict($X_t^{\mathfrak{f}}, t_i, C$)
      $X_t^{\mathfrak{f}} \leftarrow X_t^{\mathfrak{f}} + v_t^{\mathfrak{f}} \cdot dt$    // Euler step
  end for

  $X_1 \leftarrow X_t^{\mathfrak{f}}$
  $wave \leftarrow$ ISTFT($X_1$)

  return $wave$
```

Detailed Algorithm 2: Sample Frequency Domain

```python
def sample_freq_domain(model, mel, ts):
    '''
    :param model: A pre-trained frequency-domain RFWave model.
    :param mel: A batch of mel-spectrogram data, [batch_size, channels, num_frames].
    :param ts: The selected sampling time points.
    :return: The reconstructed audio waveform, [batch_size, num_frames * hop_length].
    '''
    batch_size, num_frames = mel.shape[0], mel.shape[2]
    noise_in_f = torch.randn([batch_size, model.n_fft + 2,num_frames])
    noise_in_f = model.get_subband(noise_in_f)
    mel = torch.repeat_interleave(mel, model.num_bands, 0)

    z_in_f = noise_in_f
    vs = []  # used for selecting sampling time points
    for i in range(len(ts) - 1):
        t = ts[i]
        dt = ts[i + 1] - ts[i]
        # the model runs at frame-level feature.
        v_in_f = model.predict(z_in_f, t, mel)
        # the Euler step operates in the frequency-domain
        z_in_f = z_in_f + v_in_f * dt
        vs.append(v_in_f)
    # append the straight line velocity
    vs.append(z_in_f - noise_in_f)
    # place the subbands to construct a full band spectrogram
    z_in_f = model.place_subband(z_in_f)
    # inverse the stft normalization
    z_in_f = model.stft_normalizer.inverse(z_in_f)
    # convert the complex spectrogram to waveform
    z_in_t = torch.istft(z_in_f, **model.stft_kwargs)
    return z_in_t, vs
```

A.9.2 SELECTING TIME POINTS OF EQUAL STRAIGHTNESS

Detailed Algorithm 3: Select Time Points

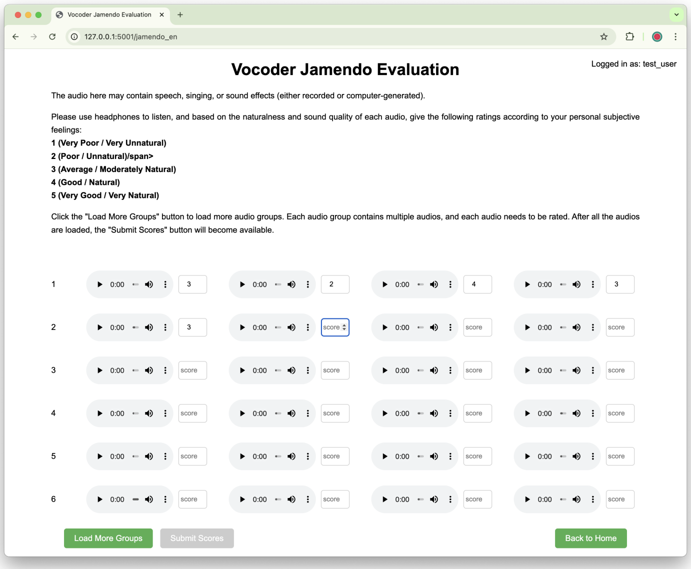
```python
def select_time_points(model, sample_fn, mel, N1=100, N2=10):
    '''
    This function is designed to select time points of equal straightness.
    :param model: A pre-trained RFWave model.
    :param sample_fn: sample_time_domain or sample_frequency_domain.
    :param mel: A batch of mel-spectrogram data, [batch_size, channels, num_frames].
    :param N1: The number of steps used for numerically estimating the straightness.
    :param N2: The number of steps for equal straightness
    :return:  A list of selected time points.
    '''

    # Step 1: Numerically estimate the trajectory
    eq_ts = torch.linspace(0, 1, N1 + 1)
    _, vs = sample_fn(model, mel, eq_ts)

    # Step 2: Calculate the deviation for each velocity step
    ds = []
    v_straight = vs[-1]
    for v in vs[:-1]:
        d = (v - v_straight).view(v.size(0), -1)
        d = torch.norm(d, p='fro', dim=1)
        ds.append(d.mean())

    # Step 3: calculate straightness
    # d_cum[-1] is the estimated straightness.
    d_cum = torch.cumsum(torch.stack(ds), dim=0)
    # s_inc is the increment of straightness for each step
    s_inc = d_cum[-1] / N2

    # Step 4: Select the time points of equal straightness
    ts = [0.]
    for i in range(1, N2):
        s_i = s_inc * i
        idx = torch.abs(d_cum - s_i).argmin()
        ts.append(idx / N1)
    return ts + [1.]
```

A.10 SUBJECTIVE EVALUATION

We conducted the subjective listening test on our self-developed website platform. The platform allows users to play audio samples and rate the audio naturalness using a 5-point Mean Opinion Score (MOS) scale, where 1 indicates "poor/unnatural" and 5 indicates "excellent/natural".

For each subjective evaluation, we invited 30 listeners to participate. For each comparison experiment to be evaluated, we randomly selected 20-40 groups of audio samples from the corresponding test set for one listener to rate. Each group of audio includes ground truth audio, audio synthesized by RFwave, and audio synthesized by other models. The order of audio within each set was randomly shuffled.

All participating listeners were required to use headphones and give ratings from 1 to 5 based on the audio naturalness. The rating results were directly submitted to our evaluation website. Figure A.10 shows the interface of the evaluation website.



Vocoder Jamendo Evaluation

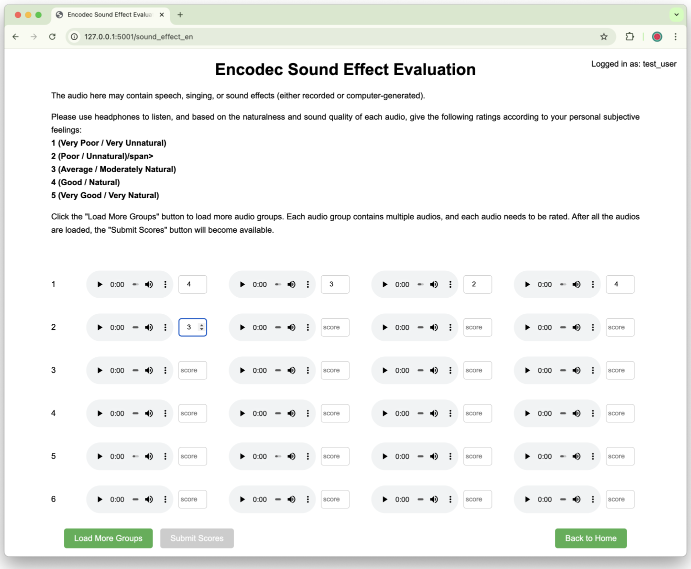
Logged in as: test_user

The audio here may contain speech, singing, or sound effects (either recorded or computer-generated).

Please use headphones to listen, and based on the naturalness and sound quality of each audio, give the following ratings according to your personal subjective feelings:
1 (Very Poor / Very Unnatural)
2 (Poor / Unnatural)/span>
3 (Average / Moderately Natural)
4 (Good / Natural)
5 (Very Good / Very Natural)

Click the "Load More Groups" button to load more audio groups. Each audio group contains multiple audios, and each audio needs to be rated. After all the audios are loaded, the "Submit Scores" button will become available.

1	3	2	4	3
2	3	score	score	score
3	score	score	score	score
4	score	score	score	score
5	score	score	score	score
6	score	score	score	score



Encodec Sound Effect Evaluation

Logged in as: test_user

The audio here may contain speech, singing, or sound effects (either recorded or computer-generated).

Please use headphones to listen, and based on the naturalness and sound quality of each audio, give the following ratings according to your personal subjective feelings:
1 (Very Poor / Very Unnatural)
2 (Poor / Unnatural)/span>
3 (Average / Moderately Natural)
4 (Good / Natural)
5 (Very Good / Very Natural)

Click the "Load More Groups" button to load more audio groups. Each audio group contains multiple audios, and each audio needs to be rated. After all the audios are loaded, the "Submit Scores" button will become available.

1	4	3	2	4
2	3	score	score	score
3	score	score	score	score
4	score	score	score	score
5	score	score	score	score
6	score	score	score	score

Figure A.10: Interface of our subjective evaluation platform.