Real-Time, Streamable Differentiable DSP Vocoder for Articulatory Synthesis

Drake Lin

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-202
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-202.html

December 1, 2024

Acknowledgement

I would like to thank Professor Gopala Anumanchipalli for his mentorship and support in my research at UC Berkeley. This work would not be possible without the Berkeley Speech Group and the significant contributions from Louis Liu and valuable assistance provided by Bohan Yu. I would also like to thank Professor Kannan Ramchandran for serving as the second reader of this work. Finally, I extend my deepest thanks to my parents, Min and Meng, for their unwavering support and love.

Real-Time, Streamable Differentiable DSP Vocoder for Articulatory Synthesis

by Drake Lin

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Gopala Anumanchipalli
Research Advisor

05/06/2024

(Date)

****

Kannan RAMCHANDRAN

Professor Kannan Ramchandran
Second Reader

05/06/2024

(Date)

Real-Time, Streamable Differentiable DSP Vocoder for Articulatory Synthesis

by

Drake Lin

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master's of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Spring 2024

Abstract

Real-Time, Streamable Differentiable DSP Vocoder for Articulatory Synthesis

by

Drake Lin

Master's of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Gopala Anumanchipalli, Chair

Articulatory synthesis, the process of generating speech from physical movements of human articulators, offers unique advantages due to its physically grounded and compact input features. However, recent advancements in the field have prioritized audio quality without a focus on streaming latency. In this paper, we propose a real-time streaming differentiable digital signal processing (DDSP) articulatory vocoder that can synthesize speech from electromagnetic articulography (EMA), fundamental frequency (F0), and loudness data. Our best model achieves a transcription word error rate (WER) of 8.9%, which is 4.0% lower than a state-of-the-art baseline. The same model can also generate 5 milliseconds of speech in less than 2 milliseconds on CPU in a streaming fashion, opening the door for downstream real-time low-latency audio applications.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my gratitude to Professor Gopala Anumanchipalli for his mentorship and support in my research throughout my time at UC Berkeley, and the Berkeley Speech Group for the wonderful memories and enriching experiences throughout the years. Specifically, this work would not be possible without the significant contributions from Louis Liu and valuable assistance provided by Bohan Yu.
I would also like to thank Professor Kannan Ramchandran for serving as the second reader of this work. Finally, I extend my deepest thanks to my parents, Min and Meng, for their unwavering support and love throughout my life – past, present, and future.

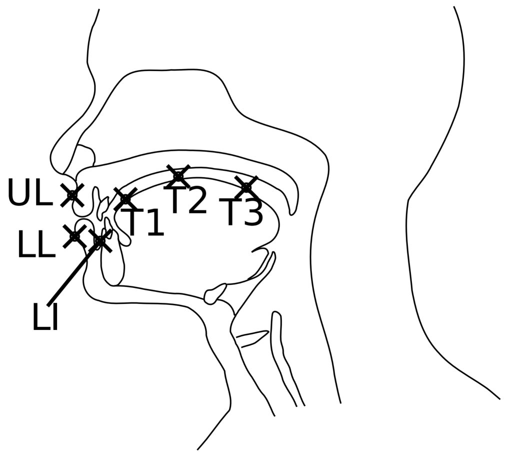
Chapter 1

Introduction

1.1 Background

Articulatory synthesis is a speech synthesis method that generates speech from articulatory features, the physical movements of human articulators such as the lips, tongue and jaw. This method traces its origins to the early 20th century when researchers sought to synthesize speech using mechanical models of the vocal tract. As technology improved, researchers began developing more complex and precise computational models, culminating with the integration of neural networks in the past couple decades that have brought improvements to the naturalness and intelligibility of synthesized speech. Unlike speech synthesis methods that rely on other intermediate speech representations such as Mel spectrograms, articulatory synthesis offers several unique advantages. Since the input features to an articulatory synthesizer are physically grounded [6], it is easily interpretable and controllable, allowing for nuanced and dynamic adjustments in speech sound production. By closely mimicking human speech mechanics, the method has the potential to produce more natural-sounding and expressive speech, especially in languages with complex phonetic and phonological rules. The parameter space in articulatory synthesis is also more compact compared to other methods, leading to data processing efficiency and reducing the computational load [2], critical for real-time speech applications. For instance the dataset [29] contains only 6 x-y points per sample, in contrast to the 128 Mel bins typically used in Mel spectrogram-based vocoders.

Articulatory synthesis is applicable to a variety of downstream tasks. It has shown promise in assisting patients with vocal cord disorders communicate better [10, 20] and help decode brain signals to speech waveforms [4], revolutionizing communication for locked-in patients. Additionally, articulatory synthesis has applications in silent speech systems, directly converting articulatory movements into speech without vocalization [10]. Furthermore, articulatory synthesis can be utilized in voice conversion tasks by manipulating a speaker's unique articulatory parameters, potentially enhancing user security and privacy in digital communications [7].



label	location	label	location
UL	Upper lip	T1	Tongue tip
LL	Lower lip	T2	Tongue body
LI	Lower incisor	T3	Tongue dorsum

Figure 1.1: Electromagnetic articulography (EMA) sensor locations. Diagram from [29].

1.2 Motivations

Many of the aforementioned tasks, such as real-time communication aids and assistive technologies, depend on the ability to operate in a real-time and streaming fashion. However, to our knowledge, no current model exists that currently meets these needs while delivering high-quality performance. Most models, including the state-of-the-art HiFi-CAR model [2], are designed for high-quality output without an emphasis on the low-latency or streamable aspects necessary for real-time interaction.

To solve this problem, we utilize Differentiable Digital Signal Processing (DDSP) [11]. DDSP models combines traditional digital signal processing (DSP) techniques with modern neural network architectures, creating a powerful hybrid model that benefits from the strength of both domains. The neural network encoder converts input features such as pitch, loudness, and spectral features, into control signals that dynamically guide the DSP modules to generate audio. The DSP modules are differentiable such that the model can be trained end-to-end, hence the name "Differentiable Digital Signal Processing". Importantly, these DSP components are designed based on well-understood signal processing techniques that inherently model the speech production process, introducing a strong natural inductive bias that simplifies the model and enhances its efficiency. Thus, DDSP vocoders are ideal for real-time applications, since they are both lightweight and high-quality.

1.3 Research Objective

We aim to design a real-time, streamable DDSP articulatory vocoder that leverages the detailed vocal tract information provided by articulatory data, augmented with F0 and loudness data to provide source information. This integration would potentially enhance the naturalness and expressiveness of the synthesized speech while staying lightweight and streamable. The proposed vocoder must be capable of generating high-quality speech in a streaming fashion while maintaining a latency of less than 5 milliseconds per sample. This undercuts the 200Hz sample rate of most articulatory data, crucial for the model to be considered real-time capable. Additionally, our audio quality must match or exceed the current state-of-the-art systems.

Chapter 2

Related Work

2.1 Articulatory Synthesis

In the deep learning era, there are generally three different frameworks for articulatory synthesis. The first approach involves predicting the acoustic parameters from articulatory features, followed by speech synthesis with traditional signal-processing-based vocoders. This method leverages existing, well-established vocoders like WORLD [25], which are known for their efficiency and robustness. These tried and tested vocoders provide a stable baseline for quality and intelligibility. Several studies have demonstrating its effectiveness in generating intelligible speech [8, 30]. However, since these models are not trained end-to-end, their reliance on intermediate acoustic parameter prediction can introduce errors which are propagated in the synthesized speech.

The second approach focuses on predicting intermediate spectrograms from articulatory features [10, 18], which are then transformed into speech using Generative Adversarial Network (GAN)-based vocoders [32, 19]. These GAN-based vocoders can generate high-quality speech from less detailed spectrograms, which is advantageous when dealing with articulatory data that may be inherently noisy or imprecise. However, this approach faces latency challenges, since there are two distinct models used: one for spectrogram generation and another for converting these spectrograms into speech. The computational overhead of maintaining two complex models also impacts resource efficiency.

The third and most direct approach is the synthesis of speech directly from articulatory features using models such as HiFi-CAR [19, 26]. This method eliminates the need for intermediate representations like acoutsic parameters or spectrograms, reducing the synthesis complexity and error accumulation. Among them, [2] is the state-of-the-art (SOTA) model in terms of synthesis intelligibility and inference speed. However, direct modeling techniques such as HiFi-CAR can be computationally expensive and require extensive data as the model must learn to model the intricate correlations between articulatory movements and resulting sound.

2.2 Differentiable Digital Signal Processing

Differentiable Digital Signal Processing (DDSP) is a framework that integrates classical digital signal processing (DSP) techniques with modern neural network architectures to create highly efficient and expressive models for audio synthesis [11]. At its core, a DDSP model consists of two components: a neural network encoder and traditional digital signal processing (DSP) modules, as shown in Figure 2.1 from [11]. The neural network encoder act as a feature extractor, transforming input features like fundamental frequency (F0), loudness, and spectral features into control signals. These signals include filter coefficients and amplitudes for harmonic synthesis, which dictate the input of the DSP modules that generate the output audio. The DSP modules, such as filters, oscillators, and noise generators, are designed to be differentiable to support back-propagation. Thus, the model can be trained end-to-end.

By utilizing DSP techniques that inherently model the physical and perceptual properties of audio generation, DDSP models are more lightweight and require less data to train compared to models that attempt to learn synthesis from scratch. Instead of learning to model raw waveforms, DDSP models need only to learn to model the control signals, leaving the synthesis task to the controllable DSP modules. This domain-appropriate inductive bias results in more natural periodicity and frequency changes, as well as better controllability as the control signals can directly be manipulated [14, 11].

There are two main architectures of DDSP vocoders: the source-filter model [31, 1, 21] and the harmonic-plus-noise (H+N) model [27, 12, 28]. The source-filter model is a classical approach to speech synthesis based on the human vocal system, where the ”source” generates raw voiced/unvoiced sounds which is then shaped by the ”filter”, which represents the vocal

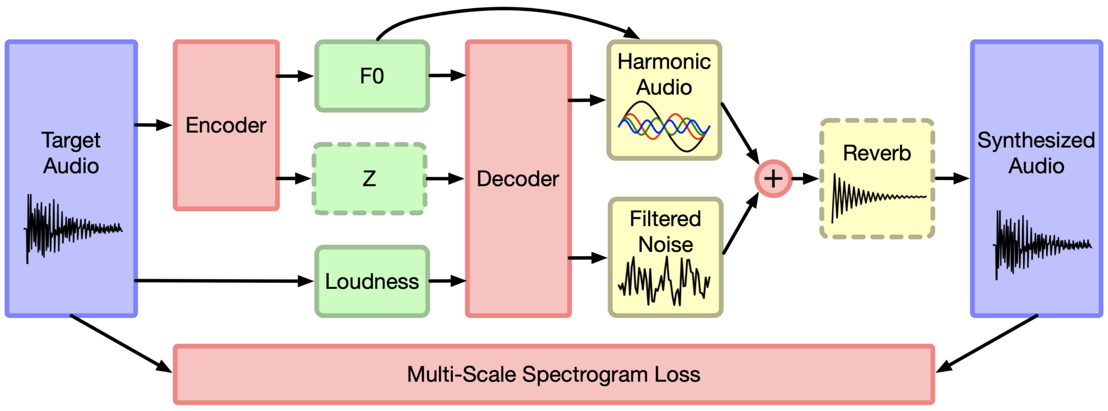


Figure 2.1: DDSP autoencoder architecture from [11]. Red components are trainable neural networks, green components are the latent representation, and yellow components are deterministic synthesizers and effects.

tract. The Harmonic-plus-Noise model divides speech into two components: harmonic tones, which represent the periodic part of speech produced by vocal cord vibrations, and noise, which models the aperiodic parts of speech produced by airflow in the vocal tract. The H+N model is extremely adaptable, which is particularly beneficial when neural networks are learning the control signals.

DDSP has seen wide-spread applications in music generation due to the music's natural periodicity and structure [11, 34]. The technique has also been applied to timbre transfer [24, 9, 13], extracting the quality of an instrument or voice and imposing it on another audio source. DDSP has also been used for singing voice synthesis [3, 33] and speech synthesis [31, 1, 21, 27, 12, 28], where the controllability and fluidity of DDSP is highly valued.

Chapter 3

Methods

Following [11], our proposed model has two parts: a neural network encoder and a DSP vocoder. We augment the EMA data, which contains vocal tract filter information, with F0 and loudness, which contains source information, to serve as the model inputs. The model has two parallel implementations: one for offline training, which processes entire samples at once, and another for real-time streaming, where samples are inputted one batch at a time. The overall architecture and performance remains the same, but slight implementation differences will be noted when relevant. In the real-time streaming mode, the initial forward pass does not produce any audio output. Subsequent forward passes will output the audio between $[t_{n-1}, t_n]$. For instance, with the inputs at $f_{model} = 200$ Hz and output audio sampling frequency set as $f_s = 16$kHz, each forward pass outputs 80 audio samples.

3.1 Encoder

The encoder architecture is shown in Figure 3.1. Its inputs are the speech's fundamental frequency F0, loudness, and EMA sampled at $f_{model} = 200$ Hz. The encoder converts the

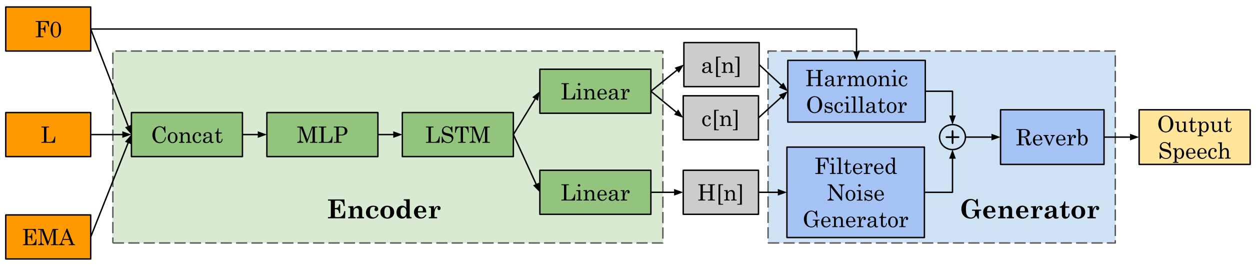


Figure 3.1: Model architecture. Only the green encoder modules are trainable. The gray blocks are control signals. F0 is pitch, L is loudness, a[n] is the global amplitude, $\mathbf{c}$[n] is the harmonic distribution, $\mathbf{H}$[n] is the filter frequency response.

features into control signals of the same sampling frequency f_{model}. Any seq-to-seq model is viable in place of the LSTM (Long Short-Term Memory) unit; we experimented with GRUs (Gated Recurrent Unit), BiLSTMs, BiGRUs, convolutional layers, and transformers, but ended up selecting LSTM for its superior combination of streamability, synthesis quality, and lightweight nature.

The input features are first concatenated along the channel dimension, then processed by an MLP (multi-layer perceptron) layer to extract features and project the concatenated input into a higher-dimensional space. The output of the MLP is then passed into the LSTM to model the temporal nature of speech. Finally, the output of the LSTM is passed into three linear heads to produce the control units. The first two heads have $K + 1$ units, where K denotes the total number of harmonics used. The first unit's output is the global amplitude $a_{sin}[n]$ and $a_{cos}[n]$ respectively, while the rest of the output corresponds to the time-varying harmonic distribution $\mathbf{c}_{sin}[n]$ and $\mathbf{c}_{cos}[n]$, which is a vector of length K at each time point n. Before being fed into the harmonic oscillator, the $\mathbf{c}_{sin}[n]$ and $\mathbf{c}_{cos}[n]$ whose corresponding harmonic frequencies are greater than Nyquist frequency are set to -1e-10 to avoid aliasing. The processed $\mathbf{c}_{sin}[n]$ and $\mathbf{c}_{cos}[n]$ are then normalized using softmax. The output of the other linear head is the frequency response $\mathbf{H}[n]$ of the time-varying filters, which is a vector of size M at each time point n.

For the real-time streaming mode, the LSTM forward pass is modified to include the previous hidden state alongside each new sample. Since the other components of the encoder operate solely on the current sample timestamp, the encoder's performance remains consistent whether processing is conducted offline or in real-time.

3.2 Digital Signal Processing (DSP) Generator

For the DSP modules, we iterated on the DSP generators of [11]. The outputs of the encoder control two DSP modules: a harmonic oscillator and a filtered-noise generator. The harmonic oscillator generates the voiced components of speech while the filtered-noise generator synthesizes the unvoiced components. The outputs of these two modules are added and passed into a reverb module to produce the final synthesized speech.

Harmonic Oscillator

The harmonic oscillator generates a sum of sinusoids whose frequencies are multiples of F0. The k-th sinusoid x_k is controlled by global amplitudes $a_{sin}[n]$ and $a_{cos}[n]$, harmonic weights $c_{sin,k}[n]$ and $c_{cos,k}[n]$, and a frequency contour $f_k[n]$, as shown below in equation 3.1. We differ from previous DDSP methods by using both cosine and sine waves in the harmonic oscillator. This is crucial for the GAN training objective, since it makes it more challenging for the discriminator to distinguish between real and generated speech.

$$x_k[n] = a_{sin}[n]c_{sin,k}[n]\sin(\phi_k[n]) + a_{cos}[n]c_{cos,k}[n]\cos(\phi_k[n]) \tag{3.1}$$

$\phi_k[n] = 2\pi \sum_{m=0}^{n} f_k[m]$ is the instantaneous phase and $f_k[n] = kF_0[n]$ is the integer multiple of F0. The harmonic distributions $\mathbf{c}[n]$ output from the encoder has K values $(c_1[n], c_2[n], ..., c_K[n])^T$ for each time point n and satisfy

$$\sum_{k=0}^{K} c_k[n] = 1 \quad \text{and} \quad c_k[n] \geq 0 \tag{3.2}$$

Thus, the output of the harmonic oscillator can be calculated as

$$x[n] = \sum_{k=1}^{K} x_k[n] = \sum_{k=1}^{K} a_{sin}[n] c_{sin,k}[n] \sin(\phi_k[n]) + a_{cos}[n] c_{cos,k}[n] \cos(\phi_k[n]) \tag{3.3}$$

Since $F_0[n]$, $a_{sin}[n]$, $a_{cos}n$, $\mathbf{c}_{sin}[n]$ and $\mathbf{c}_{cos}[n]$ are all sampled at $f_{model} = 200\text{Hz}$, we need to first upsample them back to the sampling frequency $f_s = 16\text{kHz}$ of the speech signals before calculating the above equations, i.e. upsample by a factor of $u = 80$. Here u is also the frame size. We upsample using the traditional signal-processing method by first inserting $u - 1$ zeros between every two samples and then convolving with a Hann window of size $2u + 1$.

In the real-time streaming mode, we simply save the previous values of $F_0[n]$, $a_{sin}[n]$, $a_{cos}n$, $\mathbf{c}_{sin}[n]$ and $\mathbf{c}_{cos}[n]$ and use those values with the new input for upsampling. Additional care is also taken to ensure phase information is saved between forward passes.

Filtered-Noise Generator

This module generates noise signals and filters them with the learned linear time-varying finite impulse response (LTV-FIR) filters. The encoder output $\mathbf{H}[n]$ is the frequency response of the LTV-FIR filters. To avoid dealing with complex numbers, we view $\mathbf{H}[n]$ as half of the transfer function of a zero-phase filter, since the frequency response of a zero-phase filter is always real and symmetric. We take the inverse FFT to get the zero-phase filter coefficients in the time domain, and shift it to be a causal, linear-phase filter. Finally, we apply a Hann window to the linear-phase filter in the time domain to balance the time-frequency resolution, and denote the processed filters to be $\mathbf{h}[n]$. To get the output of the filtered-noise generator, each $\mathbf{h}[n]$ is convolved with a noise signal of length u, i.e., a frame of noise, and then overlap-added with a hop size of u. Noise is generated from a uniform distribution between $[-1, 1]$, and all convolutions are calculated through FFT.

For real-time streaming, we implement the overlap-add method to save and add previous noise outputs with the current outputs. This approach ensures continuity and coherence in the resulting audio.

Reverb Module

The outputs of the harmonic oscillator and filtered-noise generator are combined to form synthesized speech. The combined output is then processed by a reverb module, which

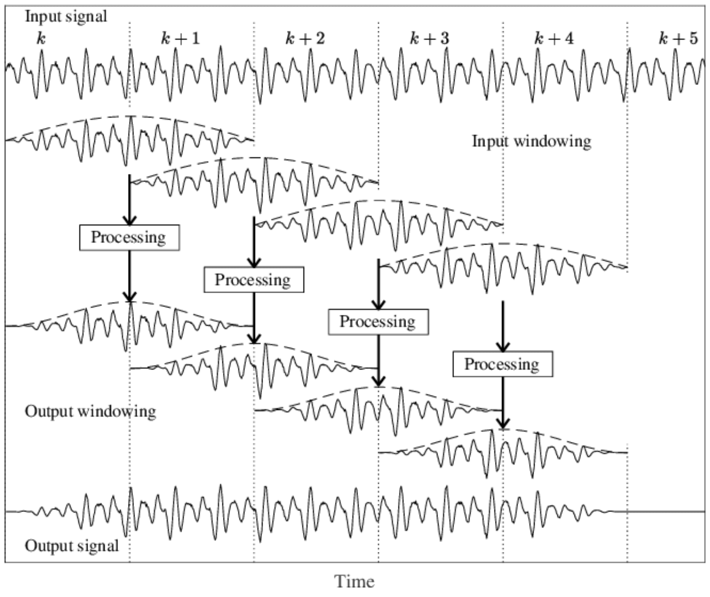


Figure 3.2: Diagram illustrating the overlap-add method from [5]. In this process, the output from each step is saved, then overlapped and added to the output of the subsequent step. This technique achieves the same result as processing the entire sample in one go.

consists of a causal single layer convolution layer that models room reverberation, an essential characteristic of realistic audio. While many neural synthesis algorithms incorporate reverberation implicitly, we have chosen to separate the reverberation into a distinct post-synthesis convolution step to enhance interpretability. In addition to room reverberation, the 1 layer convolution also helps with de-noising and modeling mouth radiation.

For real-time streaming, the weights of the learned reverb filter are directly convolved to each output frame using FFT. These processed outputs are then stored to overlap-add with the next input, ensuring seamless audio continuity between frames.

3.3 Loss Functions

Multi-Scale Spectral Loss

We used the multi-scale spectral loss as defined in [11]:

$$\mathcal{L}_{MSS} = \sum_{i \in W} ||S_i - \hat{S}_i||_1 + \alpha || \log S_i - \log \hat{S}_i ||_1 \tag{3.4}$$

where S and $\hat{S}$ are the magnitude spectrograms of the ground truth audio and the generated audio respectively. α is chosen to be 1 in this paper. $W = [2048, 1024, 512, 256, 128, 64]$ is the set of FFT sizes, and the frame overlap is set to be 75%.

Multi-Resolution Adversarial Loss

As mentioned in [1], training only on multi-scale spectral loss for audio often results in over-smoothed spectral predictions. L1 / L2 losses aim to reduce large discrepancies and capture the low-frequency components of the input, averaging out rapid changes in spectral details which results in muffled-sounding audio.

In order to capture the high-frequency components of the spectrograms, following the work of [15], we utilize multi-resolution spectrogram discriminators. The architecture of one sub-discriminator is shown in Figure 3.3.

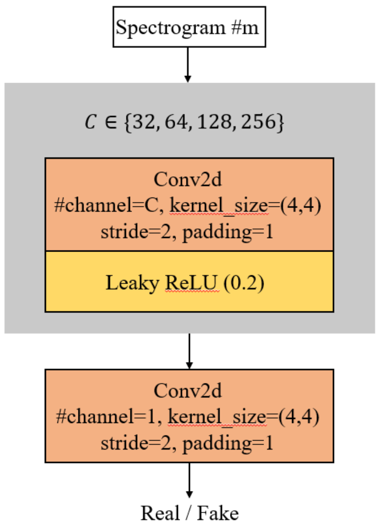


Figure 3.3: The architecture of one sub-discriminator. There are 6 sub-discriminators in total, each with the same architecture but different input spectrogram resolutions.

Note that the input spectrograms are calculated from the synthesized speech / ground truth speech with different parameters, such as window size, hop size, and number of points for FFT. For each sub-discriminator, the adversarial loss is calculated as Least Squares GAN (LSGAN) described in [23]:

$$\min_{D_i} \mathcal{L}_{LSGAN}(D_i; G) = \frac{1}{2}\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D_i(S(x)) - 1)^2] + \frac{1}{2}\mathbb{E}_{z \sim p_z(z)}[(D_i(S(G(z))))^2] \tag{3.5}$$

$$\min_{G} \mathcal{L}_{LSGAN}(G; D_i) = \mathbb{E}_{z \sim p_z(z)}[(D_i(S(G(z))) - 1)^2] \tag{3.6}$$

where S is the STFT, D_i is the i-th sub-discriminator, G is the DDSP vocoder, and z is the input features, which are the spectrograms of the real and generated audio.

The overall loss functions for the generator and discriminator are:

$$\mathcal{L}(G) = \mathcal{L}_{MSS} + \frac{\lambda}{R}\sum_{i=1}^{R}\mathcal{L}_{LSGAN}(G; D_i) \tag{3.7}$$

$$\mathcal{L}(D) = \frac{1}{R}\sum_{i=1}^{R}\mathcal{L}_{LSGAN}(D_i; G) \tag{3.8}$$

where λ controls the weight of the LSGAN loss.

Chapter 4

Results

We experimented with the MNGU0 EMA dataset [29], which contains 75 minutes of 16 kHz single-speaker speech paired with 200 Hz of EMA. F0 is extracted from the paired speech using CREPE [17] with a hop size of 80. Loudness is calculated as the absolute amplitude envelope, where the maximum absolute value of each non-overlapping speech frame is extracted, with a frame size of 80 [16, 22]. Thus, EMA, F0, and loudness are all sampled at 200 Hz. During each training epoch, we randomly crop 1 second of the corresponding waveform, EMA, F0 and loudness for every utterance as the input. We use the original train-test split with 1129 training utterances and 60 test samples, which are 71.3 minutes and 3.7 minutes respectively. From the training utterances, we choose 60 for validation.

4.1 Experimental Setup

For our DDSP model, we set the number of harmonics as $K = 50$ and the number of frequency bands $M = 65$. We trained 5 models, varying the hidden dimension of LSTM to be of size [64, 128, 256, 512, 1024] to examine the tradeoff between model size and performance. We designated these models as DDSP-64, DDSP-128, DDSP-256, DDSP-512, and DDSP-1024,

Table 4.1: Model sizes.

Model	Params.
HiFi-CAR	13.5M
DDSP-64	**56K**
DDSP-128	191K
DDSP-256	708K
DDSP-512	2.7M
DDSP-1024	10.7M

respectively. The number of parameters per model are shown in Table 4.1. For the generator, we use the Adam optimizer with initial learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, a batch size of 32, and 6400 training epochs. The learning rate is multiplied by 0.3 at epoch milestones [2400, 4800].

For the multi-resolution discriminators, we use 6 different resolutions: window size = [64, 128, 256, 512, 1024, 2048]. The number of points for FFT is the same as each of the window size, and the overlap is 75%. We use the Adam optimizer with initial learning rate 5e-6, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate is multiplied by 0.3 at epoch milestones [2400, 4800].

4.2 Baseline

We train a modified HiFi-CAR from [2] which has EMA, F0 and loudness as its input (denoted as "HiFi-CAR") to serve as our baseline.

4.3 Synthesis Results

We used Whisper-Large, a pre-trained automatic speech recognition model, to evaluate the synthesized test set speech of models and compare the mean character error rate (CER) and word error rate (WER). The transcription results are shown in Table 4.2. Compared to the modified HiFi-CAR baseline, the DDSP model performance depends on the encoder size. The models with LSTM hidden dimension greater than 256 outperform the baseline, with the biggest model DDSP-1024 achieving a WER of 8.9%, 4.0% lower than baseline. Notably, this is impressive as the DDSP-256 model is 20x smaller than the baseline, as shown in Table 4.1.

To objectively measure synthesis quality, we use the Mel Cepstral Distortion (MCD) metric. This metric quantifies the differences between two mel frequency cepstral coefficient vectors derived from generated and reference speech. Specifically, we use MCD-DTW, an improved MCD metric that incorporates the Dynamic Time Warping algorithm to find the

Table 4.2: Transcription results using Whisper-LARGE

Model	CER (%)	WER (%)
HiFi-CAR	7.7	12.9
DDSP-64	9.1	17.4
DDSP-128	8.2	14.7
DDSP-256	6.8	11.8
DDSP-512	6.7	11.3
DDSP-1024	**5.8**	**8.9**

Table 4.3: Mel Cepstral Distortion

Model	MCD
HiFi-CAR	**3.62 ± 0.74**
DDSP-64	3.92 ± 0.78
DDSP-128	3.74 ± 0.76
DDSP-256	3.73 ± 0.75
DDSP-512	3.71 ± 0.64
DDSP-1024	3.64 ± 0.65

minimum MCD between two speeches. The results are tabulated in 4.3. None of the DDSP models are better than the baseline, but the largest model DDSP 1024 achieves comparable results. This outcome is somewhat expected given that the DDSP model operates under a causal framework, whereas the baseline HiFi-CAR model is non-causal. The HiFi-CAR's ability to leverage future contextual information possibly results in "smoother" speech, generating lower MCD values.

We also ran a Mean Opinion Score (MOS) test on the DDSP-1024 model, HiFi-CAR model, and ground truth to assess subjective perceptions of naturalness, clarity and speech quality. We randomly selected the same 7 test samples from each model and performed the conventional 5-scale MOS test with 23 participants, who were asked to rate the quality of audio samples on a scale from 1 (poor) to 5 (excellent). The results of the MOS test in table 4.4 indicates that the DDSP model performed very well in comparison to HiFi-CAR. The MOS difference between the DDSP model and the ground truth is 0.34 versus HiFi-CAR's difference of 0.99. Thus, we conclude the DDSP model results in more "natural" speech than HiFi-CAR.

In our subjective evaluation, the DDSP vocoder's output is very human-like and clear. However, it does display some limitations upon closer listening. Specifically, as seen in the lower frequency areas of Figure 4.1, the synthesized speech can occasionally sound muddy around complex passages where overlapping harmonics can blur the speech. Furthermore, audio artifacts occasionally appear, particularly with sounds that require precise articulatory control such as fricatives. These artifacts result in unnatural fluctuations in audio, which detracts from speech quality. However, these artifacts are brief and minor, and the resulting speech is still highly intelligible.

Table 4.4: Mean Opinion Score

Model	MOS
HiFi-CAR	3.24±0.93
DDSP-1024	3.89±0.75
Ground Truth	**4.23±0.75**

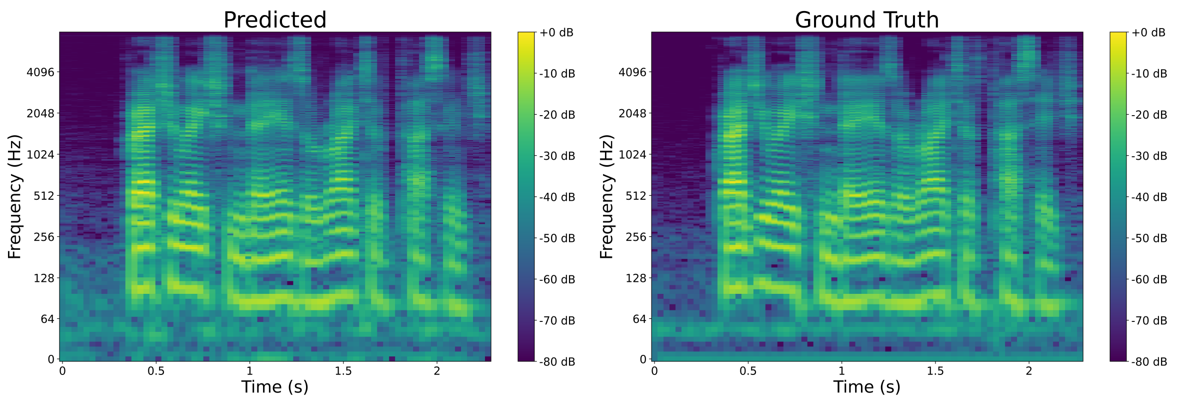


Figure 4.1: Spectrogram of model prediction and ground truth for phrase “Graduation is a rite of passage.”

In summary, the DDSP vocoder achieves a remarkable degree of naturalness and intelligibility despite its shortcomings. The Mean Opinion Score illustrates its naturalness while the transcription metrics illustrates its intelligibility. This is particularly exciting given the model's casual real-time framework.

4.4 Inference Speed

To contextualize the model's inference speed, we inputted one second of samples (200 samples at 200 Hz) through the HiFi-CAR and DDSP models and measured the inference speed on a M2 Max CPU. The results are shown in table 4.5 .The DDSP models are much faster, with the largest DDSP-1024 model processing 2x faster than baseline while the smallest DDSP-64 model is nearly 5x faster. The DDSP-1024 model is 18 faster than real-time, while the DDSP-64 model is 42x faster than real-time. Interestingly, the inference times of DDSP-64, DDSP-128, and DDSP-256 show minimal variation, suggesting that the encoder is not the computational bottleneck below a certain parameter count. We explore this later.

To test the real-time streaming performance, we benchmarked the DDSP vocoder's inference time across different numbers of sample inputs. In practical scenarios, the vocoder must wait for an upstream model to generate the EMA, F0, and loudness inputs. If the upstream model cannot deliver per-sample inputs in real-time, the system may need to process multiple samples at a time. Thus, we test how the the DDSP vocoder handles batch streaming processing of samples by inputting 1 to 1024 input samples at time, where each sample corresponds to 5 ms of audio.

The results are shown in Table 4.6 and Graph 4.2. Notably, processing one sample, which represents 5 ms of audio, takes under 1 ms for the DDSP-64 model and under 2 ms for the DDSP-1024 model. This confirms that DDSP models can be effectively used for real-time streaming applications, since the DDSP models are faster than real-time even when processing the minimal number of sample inputs. Since all model sizes achieve this real-time limitation, applications can prioritize the trade-off between model size and model quality without compromising inference speed.

In addition to the general performance assessments, we also analyzed the inference times of specific individual model components, namely the encoder, harmonic, filtered noise, and reverb modules, to identify processing bottlenecks. The results, shown in Table 4.7 for the encoder and Table 4.8 for the DSP modules, indicate that the bottleneck is the harmonic module. The encoder, filtered noise, and reverb modules contribute little latency. Further investigation into the harmonic module concluded that the bottleneck operation is the harmonic upsampling process from 200 Hz to 16 kHz. This is currently implemented with a sparse Hann convolution, which effectively reduces spectral artifacts but is computationally intense. For future work, simplifying the upsampling process could potentially reduce inference time.

Table 4.6: Model inference times [ms] for different number of input samples

# Input	DDSP-64	DDSP-128	DDSP-256	DDSP-512	DDSP-1024
1	0.87 ± 0.03	0.92 ± 0.01	0.95 ± 0.02	1.04 ± 0.03	1.81 ± 0.06
2	1.04 ± 0.01	1.19 ± 0.05	1.31 ± 0.05	2.23 ± 0.07	3.92 ± 0.02
4	1.35 ± 0.01	1.62 ± 0.04	1.69 ± 0.04	2.81 ± 0.19	4.71 ± 0.22
8	2.01 ± 0.03	2.36 ± 0.02	2.38 ± 0.06	3.95 ± 0.14	5.93 ± 0.08
16	3.11 ± 0.03	3.60 ± 0.04	3.45 ± 0.02	5.17 ± 0.15	8.18 ± 0.17
32	4.70 ± 0.02	5.15 ± 0.05	5.39 ± 0.04	6.77 ± 0.11	12.10 ± 0.16
64	7.99 ± 0.04	8.78 ± 0.09	9.89 ± 0.08	11.30 ± 0.15	20.70 ± 0.15
128	15.07 ± 0.11	16.26 ± 0.14	17.13 ± 0.14	21.06 ± 0.13	39.18 ± 0.47
256	27.03 ± 0.21	28.75 ± 0.21	30.73 ± 0.24	37.46 ± 0.38	70.32 ± 0.23
512	50.97 ± 0.65	54.28 ± 1.25	57.96 ± 0.87	70.58 ± 0.28	135.49 ± 0.52
1024	92.93 ± 1.85	97.12 ± 1.84	107.09 ± 1.05	130.68 ± 1.68	260.13 ± 2.65

Table 4.5: CPU inference time for 1s of input.

Model	Inference Time [ms]
HiFi-CAR	107.4 ± 7.6
DDSP 64	**23.7 ± 0.6**
DDSP 128	24.3 ± 0.3
DDSP 256	24.7 ± 0.9
DDSP 512	29.8 ± 0.2
DDSP 1024	57.1 ± 1.0

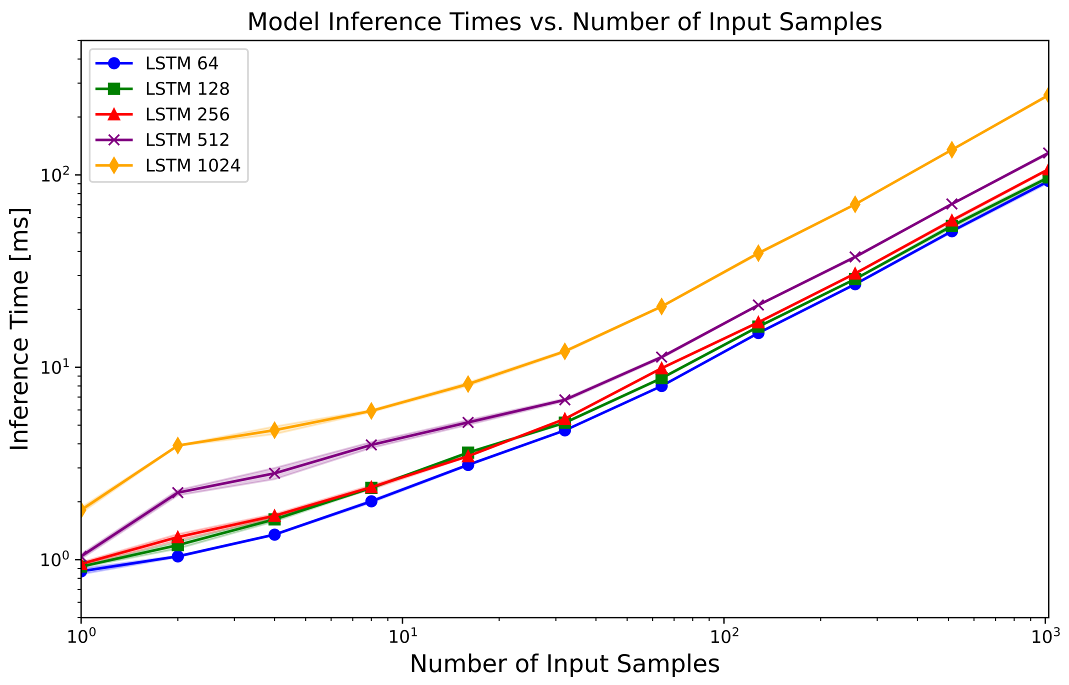


Figure 4.2: Model Inference Time vs Number of Input Samples

Table 4.7: Encoder Inference time versus Number of Sample Inputs

# Samples	DDSP-64	DDSP-128	DDSP-256	DDSP-512	DDSP-1024
1	0.15 ± 0.00	0.17 ± 0.00	0.19 ± 0.00	0.25 ± 0.00	0.89 ± 0.01
2	0.21 ± 0.00	0.26 ± 0.00	0.41 ± 0.01	1.18 ± 0.08	2.92 ± 0.09
4	0.27 ± 0.00	0.32 ± 0.00	0.42 ± 0.00	1.23 ± 0.07	3.21 ± 0.03
8	0.47 ± 0.00	0.55 ± 0.00	0.76 ± 0.00	1.61 ± 0.01	4.05 ± 0.11
16	0.55 ± 0.00	0.66 ± 0.00	0.96 ± 0.02	1.78 ± 0.01	5.27 ± 0.01
32	0.80 ± 0.01	1.00 ± 0.01	1.43 ± 0.01	2.46 ± 0.04	7.47 ± 0.04
64	1.11 ± 0.01	1.47 ± 0.01	2.50 ± 0.03	3.85 ± 0.06	13.00 ± 0.10
128	1.89 ± 0.06	2.49 ± 0.08	3.84 ± 0.11	6.96 ± 0.07	24.60 ± 0.10
256	3.45 ± 0.07	4.39 ± 0.03	6.78 ± 0.03	12.90 ± 0.10	45.80 ± 0.09
512	6.16 ± 0.13	8.16 ± 0.13	13.02 ± 0.05	24.30 ± 0.08	88.78 ± 0.24
1024	11.19 ± 0.07	15.58 ± 0.17	24.70 ± 0.12	47.94 ± 0.17	174.45 ± 0.45

Table 4.8: DSP Module Inference Time

# Samples	Harmonic Module [ms]	Noise Module [ms]	Reverb Module [ms]
1	0.50 ± 0.01	0.09 ± 0.00	0.07 ± 0.00
2	0.57 ± 0.00	0.10 ± 0.00	0.06 ± 0.00
4	0.93 ± 0.04	0.10 ± 0.00	0.07 ± 0.00
8	1.58 ± 0.02	0.10 ± 0.00	0.07 ± 0.00
16	2.46 ± 0.05	0.11 ± 0.00	0.08 ± 0.00
32	3.81 ± 0.06	0.14 ± 0.00	0.08 ± 0.00
64	6.55 ± 0.14	0.25 ± 0.00	0.12 ± 0.00
128	12.40 ± 0.07	0.42 ± 0.00	0.19 ± 0.02
256	22.73 ± 0.21	0.78 ± 0.00	0.35 ± 0.01
512	41.78 ± 0.08	1.39 ± 0.01	0.80 ± 0.00
1024	72.58 ± 0.33	2.87 ± 0.03	1.60 ± 0.02

4.5 Discussion

The DDSP model demonstrates increased synthesis intelligibility and quality with significantly lower inference time compared to the HiFi-CAR model. These advances underscores the effectiveness of DDSP's inductive bias towards audio production. The capability of the DDSP vocoder to operate in real-time streaming applications sets it apart from current models and enables downstream real-time applications. Additionally, our benchmarks show that the DDSP vocoder offers a flexible tradeoff between model size and latency, allowing for tailored application deployments that can optimize for either audio quality or latency. For instance, the DDSP-256 model is around 4x faster than the baseline while maintaining similar audio quality. In contrast, larger models like DDSP-1024 are 2x faster than the baseline but have much better audio quality. Since the DDSP-1024 is still 2.5x faster than real-time when processing one sample at a time, there still exists opportunity to improve audio quality at the expense of latency, either with increased parameters in the encoders and DSP modules or through a more complex model architecture.

Chapter 5

Conclusion

5.1 Conclusion

This paper presents a DDSP articulatory vocoder based on the harmonic-plus-noise model. With the strong inductive bias of DDSP, the model is fast, light-weight, and capable of synthesizing highly intelligible speech from EMA, F0 and loudness in a real-time and streaming fashion.

5.2 Real-time Streaming Takeaways

We summarize our takeaways for real-time streaming models as follows:

- More complex architectures often result in better performance, but for real-time streaming applications, simpler architectures tend to be more efficient. In this work, we found that MLP+LSTM performed sufficiently while maintaining faster processing times that other seq2seq methods such convolution networks and transformers.
- We observed that decreasing model parameter amount doesn't always lead to a significant reduction in model inference time. For instance, we observed a 50% decrease in inference time from DDSP-512 to DDSP-256, but didn't see the same drop from DDSP-256 to DDSP-128 and DDSP-64. Careful consideration must be put into balancing model complexity and inference quality.
- While libraries provide convenience, manually implementing standard functions may provide speedups given specific data atributes. Our implementation of sparse convolution was 10x faster than FFT convolve for our use case.
- In-place memory reallocation helps lower inference time in streaming applications by reducing memory overhead. In this work, we cut 5% of our inference time by reusing all vectors in each forward pass.

5.3 Future Work

There are many opportunities to improve and extend this work. First, we'd like to explore the multi-speaker capabilities of our system. This can be achieved by incorporating a speaker embedding feature and training the model on more diverse datasets to handle multiple speakers effectively. Additionally, we plan to work on real-time streaming feature extraction to develop sequence-to-sequence model. While loudness extraction can be easily done in a real-time streaming manner, there currently lacks options for EMA and pitch extraction. Addressing this gap would enable the implementation of various sequence-to-sequence applications such as style transfer and voice anonymization. Another area of exploration is replacing the harmonic-plus-noise model with the source-filter model. While the harmonic-plus-noise model well represents audio, the source-filter is closer to the mechanisms of human speech production. This also enables more precise control over the speech output, which could in turn better model individual speaker characteristics.

Bibliography

[1] Prabhav Agrawal et al. “Ultra-lightweight Neural Differential DSP Vocoder For High Quality Speech Synthesis”. In: *arXiv preprint arXiv:2401.10460* (2024).

[2] P. Wu et. al. “Deep Speech Synthesis from Articulatory Representations”. In: *Proc. INTERSPEECH 2022 – 23rd Annual Conference of the International Speech Communication Association*. Incheon, Korea, 2022, pp. 779–783.

[3] Juan Alonso and Cumhur Erkut. “Latent space explorations of singing voice synthesis using DDSP”. In: *arXiv preprint arXiv:2103.07197* (2021).

[4] Gopala K Anumanchipalli, Josh Chartier, and Edward F Chang. “Speech synthesis from neural decoding of spoken sentences”. In: *Nature* 568.7753 (2019), pp. 493–498.

[5] Tom Bäckström. “Overlap-add Windows with Maximum Energy Concentration for Speech and Audio Processing”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 491–495. DOI: 10.1109/ICASSP.2019.8683577.

[6] Catherine P. Browman and Louis Goldstein. “Articulatory Phonology: An Overview”. In: *Phonetica* 49.3-4 (1992), pp. 155–180.

[7] Beiming Cao, Alan Wisler, and Jun Wang. “Speaker Adaptation on Articulation and Acoustics for Articulation-to-Speech Synthesis”. In: *Sensors* 22 (Aug. 2022), p. 6056. DOI: 10.3390/s22166056.

[8] Beiming Cao et al. “Articulation-to-Speech Synthesis Using Articulatory Flesh Point Sensors’ Orientation Information”. In: *Interspeech*. 2018.

[9] Michelle Carney et al. “Tone Transfer: In-Browser Interactive Neural Audio Synthesis.” In: *IUI Workshops*. 2021.

[10] Yu-Wen Chen et al. “EMA2S: An End-to-End Multimodal Articulatory-to-Speech System”. In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5.

[11] Jesse Engel et al. “DDSP: Differentiable Digital Signal Processing”. In: *International Conference on Learning Representations*. 2020.

[12] Giorgio Fabbro et al. *Speech Synthesis and Control Using Differentiable DSP*. 2020. arXiv: 2010.15084 [eess.AS].

[13] Ben Hayes, Charalampos Saitis, and György Fazekas. "Neural waveshaping synthesis". In: *arXiv preprint arXiv:2107.05050* (2021).

[14] Ben Hayes et al. "A review of differentiable digital signal processing for music & speech synthesis". In: *arXiv preprint arXiv:2308.15422* (2023).

[15] Won Jang et al. "UnivNet: A Neural Vocoder with Multi-Resolution Spectrogram Discriminators for High-Fidelity Waveform Generation". In: *Interspeech*. 2021.

[16] Cecilia Jarne. "A heuristic approach to obtain signal envelope with a simple software implementation". In: *Anales AFA* 29 (July 2018).

[17] Jong Wook Kim et al. "Crepe: A Convolutional Representation for Pitch Estimation". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 161–165.

[18] Miseul Kim et al. "Style Modeling for Multi-Speaker Articulation-to-Speech". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5.

[19] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17022–17033.

[20] Yuqin Lin et al. "End-to-End Articulatory Modeling for Dysarthric Articulatory Attribute Detection". In: *ICASSP 2020*. 2020, pp. 7349–7353.

[21] Zhijun Liu, Kuan Chen, and Kai Yu. "Neural Homomorphic Vocoder." In: *INTERSPEECH*. 2020, pp. 240–244.

[22] Alexis Deighton MacIntyre, Ceci Qing Cai, and Sophie K Scott. "Pushing the envelope: Evaluating speech rhythm with different envelope extraction techniques". In: *The Journal of the Acoustical Society of America* 151.3 (2022), pp. 2002–2026.

[23] Xudong Mao et al. "Least Squares Generative Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2016), pp. 2813–2821.

[24] Michael Michelashvili and Lior Wolf. "Hierarchical Timbre-Painting and Articulation Generation". In: *International Society for Music Information Retrieval Conference*. 2020.

[25] Masanori MORISE, Fumiya YOKOMORI, and Kenji OZAWA. "WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications". In: *IEICE Transactions on Information and Systems* E99.D.7 (2016), pp. 1877–1884.

[26] Max Morrison et al. "Chunked Autoregressive GAN for Conditional Waveform Synthesis". In: *ICLR 2022*. 2022.

[27] Shahan Nercessian. "Differentiable WORLD synthesizer-based neural vocoder with application to end-to-end audio style transfer". In: *arXiv preprint arXiv:2208.07282* (2022).

[28] Shahan Nercessian. “End-to-end zero-shot voice conversion using a DDSP vocoder”. In: *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2021, pp. 1–5.

[29] K. Richmond, P. Hoole, and S. King. “Announcing the electromagnetic articulography (day 1) subset of the mngu0 articulatory corpus”. In: *Interspeech*. 2011, pp. 1505–1508.

[30] Fumiaki Taguchi and Tokihiko Kaburagi. “Articulatory-to-speech Conversion Using Bi-directional Long Short-term Memory”. In: *Proc. Interspeech 2018*. 2018, pp. 2499–2503.

[31] Xin Wang, Shinji Takaki, and Junichi Yamagishi. “Neural source-filter waveform models for statistical parametric speech synthesis”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2019), pp. 402–415.

[32] 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)*. 2020, pp. 6199–6203.

[33] Takenori Yoshimura et al. “Embedding a Differentiable Mel-Cepstral Synthesis Filter to a Neural Speech Synthesis System”. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5.

[34] Yi Zhao et al. “Transferring Neural Speech Waveform Synthesizers to Musical Instrument Sounds Generation”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), pp. 6269–6273.