NORTHWESTERN UNIVERSITY

Interpretable Speech Representation and Editing

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Max Morrison

EVANSTON, ILLINOIS

June 2024

ABSTRACT

Interpretable Speech Representation and Editing

Max Morrison

Generative machine learning has begun to revolutionize how we develop media content, including speech content for podcasts, social media, film dialogue, and video game dialogue. In this dissertation, I describe my research contributions using recent machine learning techniques to advance speech editing technologies. Contributions include advances in the state-of-the-art in speech representation (e.g., pitch, periodicity, and pronunciation representations) and controllable speech synthesis (e.g., improved editing quality and accuracy), as well as novel editing capabilities such as fine-grained pronunciation control and editing the timbral correlates of volume.

Table of Contents

List of Tables

List of Figures

violet indicates agreement between PPGs and SPPGs. (**right**) Red indicates disagreement between SPPGs and PPGs. For both plots, only phonemes with a maximal probability above 0.1% are depicted.
`figure-2-2-0070-000751.wav` 52

2.3 **Average framewise phoneme accuracy |** Accuracy of PPGs computed from five input representations. The wav2vec 2.0 [**7**] input representation has the best PPG accuracy when averaged over all datasets (see legend). **N.B.** The base wav2vec 2.0 model of Charsiu [**127**] was trained on some of my Common Voice test partition as well as the TIMIT training partition, making Charsiu's results on those datasets unreliable upper bounds. 57

2.4 **Acoustic phoneme similarities |** (**top**) Row b column c is $\ln \mathcal{S}_{b,c} = \ln \mathbb{E}\left[\lambda_c G_{c,t}; \lambda_b G_{b,t} \geq \lambda_z G_{z,t}\, \forall z\right]$, the log of the average class-weighted probability assigned to phoneme c when phoneme b is the maximum model prediction. Averages are taken over all frames of my validation partition of Common Voice [**3**] using my PPG model trained with class-balancing on Mel spectrogram inputs. Red boxes show that the corresponding unvoiced fricative (/f/, /s/, /sh/) to each voiced fricative (/v/, /z/, /zh/) is assigned relatively high probability, and vice versa. Class-balanced training and class-weighting are used to remove column banding indicative of natural phoneme frequency. Remaining column banding (e.g., "ah" is brighter than "ao") is caused by differences in self-similarity (i.e., the yellow diagonal). (**bottom**)

Zooming in to the row of voiced fricative "v" demonstrates that corresponding unvoiced fricative "f" is usually the next most likely prediction when "v" is most likely. 60

2.5 Pitch posteriorgrams $D \in \mathbb{R}^{|F| \times T}$ produced by my reimplementation of the baseline FCNF0 pitch tracker [**2**] (**middle**) and my proposed FCNF0++ (**bottom**), where $D_{f,t} = p(y_t = f|x_t)$ is the time-varying categorical distribution produced by performing inference on adjacent input audio frames $x_1, \ldots, x_T$. The input audio (**top**) is the same as in Figure 1.1: the speech utterance "I am sitting in a room, different from the one you are in now". To produce these visualizations, I apply softmax to the $|F|$-dimensional network logits inferred for each time frame to produce normalized distributions and take the natural log. Greater brightness indicates higher probability. The y-axis frequency ranges are representative of the pitch bin ranges of the baseline and proposed models. My proposed methods produce a sharper peak during pitched frames and encourage uniform probability in unpitched regions, making it easy to identify these regions algorithmically (Section 2.3).

`figure-1-1-sitting.wav` 67

2.6 **Decoding methods for neural pitch estimation |** I compare Viterbi decoding with linear interpolation of unvoiced regions. (**top**) Pitch contours produced via each decoding method. (**bottom**) entropy-based periodicity contour (**black**) (Section 2.3) and three

noise floor.

`figure-3-4-0016-000321.wav` 105

3.5 **Speech reconstruction example |** I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) perform speech synthesis, (3) encode the synthesized speech in my representation, and (4) overlay the speech representation inferred from synthesized speech on the input representation to demonstrate the reconstruction accuracy of my speech representation. **Blue** SPPGs (**top**) are inputs, **red** SPPGs are inferred from synthesized speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

`figure-3-5-0108-000345-original.wav`

`figure-3-5-0108-000345-reconstructed.wav` 107

4.1 **Subjective evaluation interface for pitch-shifting and time-stretching |** Participants recruited on Amazon Mechanical Turk listen to 15 sets of three audio files, where each set consists of the same edit (a pitch-shift or time-stretch) applied to an original audio recording using three methods: (1) my proposed method for

pitch-shifting (Section 4.1) or time-stretching (Section 4.2) as well as DSP baselines (2) TD-PSOLA [**80**] and (3) WORLD [**69**]. 111

4.2 **Pitch-shifting example |** I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) modify the pitch by −600 (**left**) and +600 (**right**) cents, (3) perform speech synthesis using the modified pitch contours to produce pitch-shifted speech, (4) encode the pitch-shifted speech in my representation, and (5) overlay the speech representation inferred from pitch-shifted speech on the input representation (*after the* ±600 *cent shift has been applied*) to demonstrate pitch disentanglement. **N.B., the pitch range (i.e., the y-axis) varies between the left and right figure.** **Blue** SPPGs (**top**) are inputs, **red** SPPGs are inferred from pitch-shifted speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from pitch-shifted speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

`figure-4-2-0108-000430-original.wav`

`figure-4-2-0108-000430-(-600¢).wav`

`figure-4-2-0108-000430-(+600¢).wav` 113

4.3 **Comparing voiced/unvoiced decisions from periodicity and PPGs |** (**top**) SPPG (Section 2.1) with voiced phonemes in **green**

and unvoiced phonemes in **red**. (**middle**) Entropy-based periodicity (Section 2.3) with voiced frames in **green**, unvoiced frames in **red**, and voicing threshold α in **orange**. (**bottom**) Voicing decisions derived from the SPPG (**blue**) and periodicity (**orange**). The SPPG framewise voiced/unvoiced decisions exhibit clearly improved alignment with voiced and unvoiced phonemes.

`figure-4-3-0083-000365.wav` 116

4.4 **Time-stretching example |** I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) interpolate the voiced and silent frames of my proposed representation to increase (**left**) or decrease (**right**) the total duration by a factor of $\sqrt{2}$, (3) perform speech synthesis using the interpolated representation to produce time-stretched speech, (4) encode the time-stretched speech in my representation, and (5) overlay the speech representation inferred from time-stretched speech on the input representation (*after interpolation*) to demonstrate the preservation of speech features during time-stretching. **N.B., the duration (i.e., the x-axis) varies between the left and right figure.** **Blue** SPPGs (**top**) are inputs, **red** SPPGs are inferred from time-stretched speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that

threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

`figure-4-4-0082-000568-original.wav`

`figure-4-4-0082-000568-(0.71x).wav`

`figure-4-4-0082-000568-(1.41x).wav` 118

4.5 **Frequency spectra of speech reconstruction and volume-matched editing of timbral correlates of volume |** I modify the single-band A-weighted loudness (Section 2.4) of an unedited speech recording by 0 (i.e., reconstruction; **blue**), −10 (**orange**), and +10 dBA (**green**) using my proposed method for jointly editing volume and its timbral correlates. I then perform frame-resolution A-weighted loudness matching using DSP-based gain scaling (Section 3.2.1) with the original audio. This produces reconstructed and perceptual-loudness-edited speech recordings with equal volume. I plot the frequency spectra of speech reconstruction as well as volume-matched editing of the timbral correlates of volume to show that my proposed method captures the expected behavior of perceptually louder speech having relatively more high-frequency content. In voice quality literature, this increase in the relative amount of high-frequency content has been found to explain over 20% of intraspeaker acoustic variations [**54**]. Likewise, most musical instruments produce relatively more high frequency content when played with more input energy (e.g., hitting a drum harder or bowing a violin string faster or with

more pressure). My proposed system learns the timbral correlates of volume within a speech dataset and enables disentangled control.

`figure-4-5-0073-000047-original.wav`

`figure-4-5-0073-000047-reconstruction.wav`

`figure-4-5-0073-000047-(-10dBA).wav`

`figure-4-5-0073-000047-(+10dBA).wav` 122

4.6 **Example of jointly editing volume and the timbral correlates of volume |** I (1) encode a speech utterance in my proposed representation (Chapter 2), (2) modify the single-band A-weighted loudness by -10 (**left**) and $+10$ (**right**) dBA, (3) perform speech synthesis using modified A-weighted loudness to produce speech in which volume and its timbral correlates are jointly edited, (4) encode the synthesized speech in my representation, and (5) overlay the speech representation inferred from synthesized speech on the input representation (*after* ± 10 *dBA edits have been applied to the A-weighted loudness*) to demonstrate accurate reconstruction of input features during loudness and volume editing. **Blue** SPPGs (**top**) are inputs, **red** SPPGs are inferred from synthesized speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity,

and 6 dBA for volume. **N.B., the loudness range (i.e., the y-axis) varies between the left and right figure.**

`figure-4-6-0037-000659-original.wav`

`figure-4-6-0037-000659-(-10dBA).wav`

`figure-4-6-0037-000659-(+10dBA).wav` 123

4.7 **Spectral balance editing example** | I (1) encode a speech utterance in my proposed representation (Chapter 2), (2) perform speech synthesis using the encoded representation with resampling augmentation factors (Section 3.2.1) of $r_f = \sqrt{2}/2$ (**left**) and $r_f = \sqrt{2}$ (**right**), (3) encode the synthesized speech in my representation, and (4) overlay my speech representation inferred from synthesized speech on the input representation to demonstrate accurate reconstruction while editing the spectral balance. My editing method produces corresponding changes in vowels in the inferred SPPG, such as predicting "ao" instead of "aa" (**right**) and "ay" instead of "eh" (**left**). To further demonstrate this behavior, Figure 4.8 zooms in on just the phonemes that were changed in the SPPG. **Blue** SPPGs (**top**) are inferred from a recording of the source speaker, **red** SPPGs are inferred from speech synthesized with modified spectral balance, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that

threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

🔊 `figure-4-7-0047-000757-original.wav`

🔊 `figure-4-7-0047-000757-(r_f=0.71).wav`

🔊 `figure-4-7-0047-000757-(r_f=1.41).wav` 127

4.8 **Spectral balance editing produces corresponding changes in phoneme probabilities |** I visualize all frames in which the absolute difference between SPPGs inferred from ground truth audio and audio with my proposed spectral balance editing exceeds 35%, using the same audio as in Figure 4.7. **Blue** indicates where SPPGs inferred from original audio are assigned at least 35% more probability than SPPGs inferred from edited audio. **Red** indicates where SPPGs inferred from audio with spectral balance editing are assigned at least 35% more probability than SPPGs inferred from ground truth audio. On the **left** ($r_f = 0.71$), we see that "er" is replaced by "ow" (1.4 seconds), "aa" is replaced with "ao" (1.7 seconds), "eh" is replaced with "w" (2.8 seconds), and "t" is replaced with "d" (3.4 seconds). These replacements all correspond with less energy in the high-frequencies. On the **right** ($r_f = 1.41$), we see that "dh" is replaced with "th" (1.3 seconds), "z" is replaced with "s" (2.1 seconds), "l" is replaced with "ah" (2.7 seconds). These replacements all correspond with more energy in high frequencies. Interestingly, "s" (an unvoiced fricative) replaces "z" (the corresponding voiced

fricative), but periodicity remains the same (Figure 4.7). This can be explained by the fact that English pronunciations of, e.g., pluralizations, are commonly "z", but are labeled as "s" in the lexically-derived PPG training data (Section 2.1.3.1). 128

4.9 **Voice conversion example |** I (1) encode speech utterances from a male (left) and a female (right) source speaker in my representation (Chapter 2), (2) shift the pitch by the difference of the mean base-two log-F0 in voiced regions between the source speaker and a target speaker of the opposite gender, (3) perform speech synthesis using the target speaker index and the source speaker representation (with mean-corrected pitch) to synthesize the source speech content using the speaker embedding associated with the target speaker, (4) encode the synthesized speech in my representation, and (5) overlay my representation inferred from synthesized speech on the input representation to demonstrate accurate reconstruction during voice conversion. **Blue** SPPGs (**top**) are inferred from a recording of the source speaker, **red** SPPGs are inferred from speech synthesized in the voice of the target speaker, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

of the word "tomato" (**blue**) and inferred from the synthesized speech (**red**). For readability, phoneme rows in the PPGs with maximum probability $< 10\%$ are omitted. The accurate reconstruction of PPGs (**magenta**) indicates preservation of (potentially edited) phonetic content in the generated speech. In the center, the input (**blue**) PPG is interpolated only within the (**gray**) edit region to be halfway (i.e., 50%) between the /eh/ in the left PPG and the /aa/ in the right PPG using SLERP [**98**]. Note that the reconstruction of interpolating "ey" (**left**) and "aa" (**right**) is "ae" or "eh" (**center**). This is consistent with interpolating vowels in formant space (F1, F2 - F1) [**47**] and indicates that one pronunciation can be represented more than one way in a PPG. **(bottom)** Pronunciation distances between synthesized speech and the original audio. My proposed distance (Section 2.1.4) is more robust to resynthesis artifacts and accurately captures pronunciation interpolation without a transcript.

`figure-4-11-source.wav`

`figure-4-11-conversion.wav`

`figure-4-11-interpolation.wav`

`figure-4-11-manual.wav` 135

4.12 **Accent conversion example |** I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) perform my proposed regex-based accent conversion to manually convert from the original, South African accent to an American Midwestern accent,

(3) perform speech synthesis to produce accent-edited speech, (4) encode the accent-edited speech in my representation, and (6) overlay the speech representation inferred from synthesized speech on the input representation to demonstrate reconstruction of both prosody and target pronunciation during accent conversion. My sequential rule set for this example is as follows: (1) `reallocate`([“dh”, “ah”], [“th”, “ah”]), (2) `reallocate`([“n”, “aa”, “t”], [“n”, “ah”, “t”]), (3) `reallocate`(“er”, “r”), (4) `reallocate`(“ae”, “eh”), where `reallocate`(b, c) reallocates all probability in regex matches for phoneme sequence b to corresponding phonemes in phoneme sequence c. **Blue** SPPGs (**top**) are inputs, **red** SPPGs are inferred from accent-edited speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

`figure-4-12-0082-000322-original.wav`

`figure-4-12-0082-000322-midwestern.wav` 138

4.13 **Automatic onomatopoeia example** | I (1) encode an example recording of cat vocalizations in my proposed representation (Chapter 2), (2) perform speech synthesis using the speaker index of a target speaker to produce a vocal imitation, (3) encode the

synthesized vocal imitation in my representation, and (4) overlay the speech representation inferred from synthesized speech on the input representation to demonstrate reconstruction of my speech representation during vocal imitation. **Blue** SPPGs (**top**) are inputs, **red** SPPGs are inferred from synthesized speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

`figure-4-13-source.wav`

`figure-4-13-0082-000322-target.wav`

`figure-4-13-0082-000322-imitation.wav`

CHAPTER 1

Introduction

Speech is an efficient and foundational medium of human communication, conveying information via words (i.e., the *lexical* content), prosody (i.e., the pitch, duration and loudness of speech), and pronunciation. Speech communication technologies—such as the representation, transmission, playback, and editing of speech—are central to modern communication mediums (e.g., film, podcasts, radio, television, social media, and video games). While the lexical content of speech is amenable to representation, transmission, and editing via written language, prosody and pronunciation are *acoustic* properties that describe the spoken realization of the lexical content, and are influenced by factors such as prominence [**16, 77**], emotion [**21, 13**], and social and cultural contexts (e.g., regional dialects and accents). These factors are not well-suited for representation via lexical features. This is because prosody and pronunciation are continuous, time-varying speech attributes that vary at rates that can be entirely independent of lexical features. Further, lexical features impose a coarse discretization on both speech pronunciations (as categorical phones or phonemes) and phoneme durations (as an integer number of time frames). Therefore, prosody and pronunciation require specialized representations to be accurately and interpretably represented or modified.

Prosody and pronunciation are important elements of communication that encode sentiment [**62**] and meaning [**16, 92**]. As such, contextually-appropriate prosody is crucial

both for human communication as well as text-to-speech systems that interact with human listeners. In some cases, contextually-inappropriate prosody and pronunciation can be unsettling: voice assistants that sound unintelligible, incorrect, or even threatening. Concretely, this could be the result of poor rendering of the prosody associated with punctuation (e.g., the difference between "Let's eat, grandma!" and "Let's eat grandma!") or due to a word spoken in one accent sounding like another word in another accent (e.g., "rise up lights" in a midwestern American accent sounds like "razor blades" in an Australian accent). In other cases, contextually-inappropriate prosody and pronunciation restricts the utility of speech synthesis systems for content creation of podcasts, movies, video games, and social media that require competitively high-fidelity speech (e.g., by emphasizing an incorrect syllable or mispronouncing heteronyms such as "*en*trance" and "en*trance*").

Manually editing speech attributes in post-production (e.g., changing the prosody or pronunciation) requires substantial effort by domain-expert audio engineers to achieve high perceptual quality. Performing overdubs (i.e., corrections made via re-recording) properly requires both significant audio engineering skills, vocal skills to perform natural prosody matching and maintain a consistent distance to the microphone, and expensive studio time or equipment. These workflow inefficiencies and prerequisite of a high level of technical competencies are due to a lack of speech editing software that enables precise, fine-grained edits (e.g., fixing a mispronunciation) using interpretable controls that directly correspond to the desired speech attribute. For example, it is far more intuitive to specify changing an "ey" sound to an "aa" sound (Section 4.7; Figure 4.11) than manually performing frequency response corrections by aligning the automation curves of

equalizers with the time of the "ey" sound, the frequencies corresponding to the formants (Figure 1.1), and the per-formant gain changes in dB associated with the timbral differences between "ey" and "aa". As well, it is simpler to specify changes in voice quality that correlate with volume in dB (Section 4.3) than to attempt to mimic the corresponding changes in frequency spectrum (Figure 4.5) dynamic range, and transients associated with quieter or louder speech using a multi-band compressor. **This inability for prior speech editing systems to service the needs of speech content creators stems from three primary challenges.**

- **Challenge 1** | The lack of a disentangled, interpretable representation of speech that facilitates control over the desired speech attributes of interest. This causes many disparate systems to be built, each demonstrating a small subset of the desired editing capabilities and workflows—and often with low perceptual audio quality.
- **Challenge 2** | The lack of a speech generation system compatible with the aforementioned interpretable representation, which can produce high-fidelity speech that is accurate relative to what is specified in the representation.
- **Challenge 3** | The lack of available editing capabilities. This causes content creators to reach for less intuitive or less capable tools, or tools for which they have had to develop substantial niche expertise (e.g., using compression, equalization, additive noise, or concatenative synthesis to change one phoneme to sound like another).

In this dissertation, I address these three challenges and develop the underlying technology that supports a new paradigm of speech editing tools that allow both novices

and experts access to efficient and intuitive methods for creating and editing high-quality speech recordings. I first develop a disentangled, interpretable representation of speech (Chapter 2; Figure 1.2). Then, I demonstrate that this representation can be paired with existing speech generation systems to perform high-fidelity speech synthesis (Chapter 3). Finally, I demonstrate that my interpretable speech synthesizer can be used for high-fidelity speech editing—including both improving existing editing capabilities and extending the available set of controls of speech editing systems (Chapter 4). Beyond the development of novel speech editing tools well-suited for professional speech content creation (e.g., for movies, video games, and podcasts), my contributions make possible new research directions in explainable text-to-speech systems; audiovisual feedback for language or accent learning; and real-time, fine-grained voice control for individuals with speech or language disorders.

1.1. Background

In order to understand my contributions, some knowledge of speech production, audio processing, machine learning, and speech editing is required. The remainder of this chapter provides an overview of this prerequisite knowledge, followed by a description of each of my research contributions within this dissertation.

1.1.1. Speech production

The human vocal mechanism consists of an air pressure system (the lungs), a vibratory system (the *larynx*), and a resonator system (the nasal and oral cavities). When the vocal cords of the larynx are engaged, air passing through the larynx produces vibrations in the

glottis (i.e., the space between the vocal cords) at a frequency controlled by the strength of the contraction of the muscles of the larynx. The resonator system then performs a spectral shaping of the signal based on the positions of *articulators*, which include the tongue, lips, teeth, gums, glottis, and more. The loudness of the signal is determined by the amount of air expelled by the lungs as well as the engagement of the vocal cords. Speech sounds produced with engaged vocal cords are *voiced* sounds, meaning that the speech signal contains repetition at a fundamental frequency (F0)—as well as energy at harmonics H1, H2, ... (Figure 1.1; bottom). This is opposed to *unvoiced* speech, in which the speech does not contain any significant amount of repetition within the human auditory range of 20 Hz - 20 kHz. The *formants* F1, F2, F3, ... are spectral peaks of the time-varying filter produced by the resonant system that shape the relative energy of harmonics. Note that in practice—and excluding outliers—male speakers typically produce F0 within the range 50-200 Hz, and female speakers produce F0 within 100-550 Hz [**123**].

My proposed speech representation (Chapter 2; Figure 1.2) can loosely be viewed as a computational parameterization of speech production. Specifically, I use an automatic pitch estimator to model both F0 (Section 2.2) and voicing or vocal cord engagement (Section 2.3), perceptually-weighted loudness to model air pressure (Section 2.4), and a phonetic posteriorgram (PPG) (Section 2.1) to disambiguate phonemes, such as is done by the spectral filtering produced by the placement of the articulators in the resonator.

While some prior work has attempted to produce a more precise computational model of speech production using, e.g., distances between articulatory features [**118**], these works are limited both by the lack of availability of high-quality annotations of articulatory

positions, as well as the ability to be used as an intuitive control interface by downstream users. For example, a novice can specify they want a "p" sound and not a "d" sound, but is unlikely to quickly infer that the distinction between "p" and "d" corresponds to alveolar (tongue against roof of mouth) as opposed to bilabial (closed lips) articulator positions. Niche demographics of experienced users (e.g., clinical speech pathologists) are capable of making such distinctions in articulatory positions, and future work may demonstrate the utility of such articulatory models for at least these specific users—especially if larger datasets of articulatory positions or other methods to acquire more data become available.

1.1.2. Audio processing

Computers represent speech waveforms as a sequence of quantized values sampled at a constant interval in time (Figure 1.1; top). The audio sampling rate is application dependent: common audio sampling rates include 44.1 kHz (for high-fidelity speech applications such as podcasts and film) and 8 kHz (for high-bandwidth speech applications such as telephone communications). While audio utilizes a high sampling rate, its repetitive and relatively low-dimensional structure make audio signals well-suited for domain-specific feature representations that summarize key elements of the audio signal while compressing the temporal dimension for faster transmission and processing.

The most widely-used audio representation that exemplifies this temporal compression is the magnitude spectrogram (Figure 1.1; middle). Let $x = x_1, \ldots, x_T$ be a sequence of T frames of speech, where one frame of speech consists of window size W speech samples and is offset from the previous frame by hop size H samples. The spectrogram $S \in \mathbb{R}^{|\Omega| \times T}$ is

the time-varying energy of x at a set of fast-Fourier transform (FFT) frequencies $\omega \in \Omega$.[1]

$$S_{\omega,t} = \left|\mathrm{FFT}(x_t, \omega)\right| \tag{1.1}$$

The magnitude spectrogram allocates its representational capacity linearly along the frequency axis. However, humans perceive frequency logarithmically, with each doubling in frequency producing a perceptually equal frequency shift that corresponds to a musical octave. As well, humans do not perceive equal energy at different frequencies as equally loud. The logarithmic nature of human speech perception is addressed by change-of-basis techniques such as the Mel spectrogram (Figure 1.1; bottom), which filters each spectrogram frame using a set of triangular filters, where the center frequencies of the filters are determined using human perceptual data. Similarly, A-weighted loudness [**65**] (Section 2.4) addresses the uneven perception of loudness at various frequencies by weighting each frequency channel of the magnitude spectrogram using weights determined via psychoacoustic experiments of human perception of loudness.

While time-frequency representations such as the Mel spectrograms are commonly used in loss functions [**40**], as output representations [**96**], and input representations [**60, 81**], they are not well-suited to be directly modified by users for the purposes of speech editing, as the speech attributes of interest (e.g., the prosody and pronunciation) are *entangled*, meaning that one cannot independently edit the attributes of interest. Further,

[1]N.B., a list of symbols is available in Appendix A.
[2]This speaker icon 🔊 followed by an `audio file name` references a corresponding audio recording in the accompanying directory of supplementary audio.

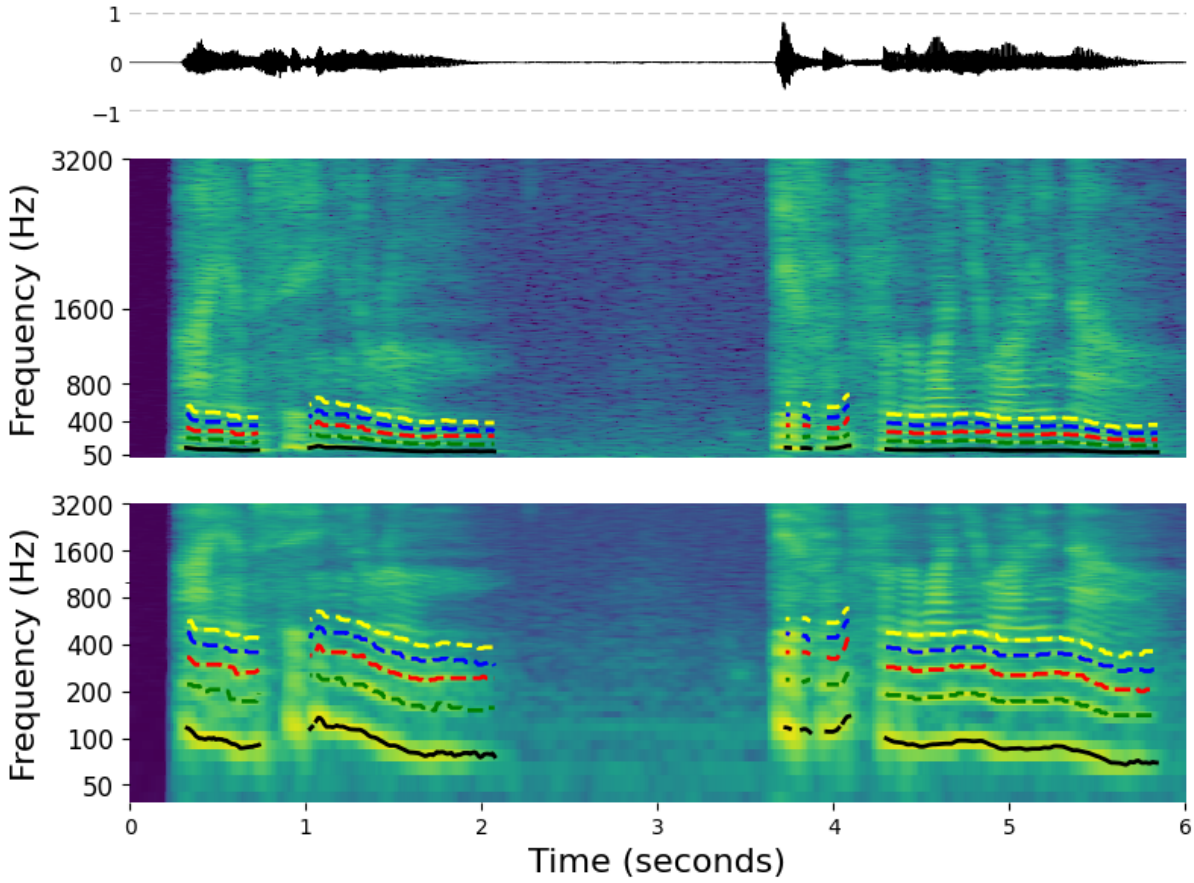

Figure 1.1. (**top**) Audio waveform of the speech utterance "I am sitting in a room, different from the one you are in now." (**middle**) The corresponding magnitude spectrogram depicting the distribution of energy across the frequency spectrum. (**bottom**) The corresponding Mel spectrogram, which reallocates more representational capacity to better capture the relevant frequency range for speech processing. The black line overlaying the lowest yellow band indicates the fundamental frequency (F0), while the parallel, higher-frequency bands indicate harmonics H1 (**green**), H2 (**red**), H3 (**blue**), and H4 (yellow). Harmonic estimation is performed using my proposed neural pitch estimator (Section 2.2) and Viterbi-based estimation method (Section 4.4).
`figure-1-1-sitting.wav`[2]

the process of editing a time-frequency representation by specifying individual energy levels of each spectrogram bin is not an intuitive or efficient editing process for novices or

experts. My proposed representation (Chapter 2) addresses these difficulties and demonstrates speech reconstruction quality on-par with the commonly used Mel spectrogram, while also enabling a range of state-of-the-art speech editing controls that are difficult or impossible with Mel spectrograms.

1.1.3. Machine learning

Once the speech to be edited is encoded in a suitable representation—such as a Mel spectrogram or my proposed representation (Chapter 2)—and the desired edits have been performed, the final step is to render the edited representation into a new speech recording that contains the original speech, but with the desired edits (Chapter 4). In audio terminology, this process of converting from acoustic audio features to a speech waveform is called *vocoding*, and has traditionally been performed using digital signal processing (DSP) techniques that leverage closed-form mathematical models and domain-specific assumptions about audio and speech [**69, 80**]. DSP-based methods are often fast, but their vocoding performance has been far surpassed by machine learning models that learn to model the conditional distribution of the speech waveform given input acoustic features using a large dataset of speech recordings [**40, 52**]. This general practice of training a model on a dataset to learn a distribution (i.e., *machine learning*) is utilized throughout my dissertation to produce state-of-the-art pitch estimation (Section 2.2) and pronunciation representation (Section 2.1), as well as to generate speech waveforms from Mel spectrograms, my proposed representation (Chapter 2), and other audio representations used as baselines for comparison (Chapter 3). The particular type of machine learning utilized throughout my work is *deep neural networks (DNNs)* [**51**], which perform gradient

descent in small batches to optimize the parameters of a customizable, non-linear function according to a customizable loss function. DNNs have demonstrated the capability to learn to model high-dimensional, non-linear conditional distributions from sufficiently large datasets. While the field of research on neural networks has grown large, the relevant DNN concepts employed in my dissertation research are relatively standard and few: convolutional neural networks [**50**], Transformer neural networks [**108**], and generative adversarial networks (GANs) [**26**]. Given the current ubiquity of these techniques and availability of high-quality educational materials (e.g., the Annotated Transformer [**93**]), I do not further describe these general techniques in this dissertation. Architectures, loss functions, and any non-standard techniques of DNNs used in this work are described in their corresponding sections.

1.1.4. Speech editing

Accurate computational representation and editing of speech advances the creative workflows of multiple multi-billion-dollar speech content creation industries as well as diagnostic practices for clinical speech pathologists servicing patients with speech and language disorders [**68**]. For example, the global market value of the film & video game industry in 2023 was $285.62 billion USD in 2023, while the global market for social media was $219.06 billion in 2023 and for podcasts was $27.73 billion [**17**]. All of these content creation applications necessitate fast, high-quality representation or editing of speech recordings.

A key challenge with developing systems for representing and editing speech is determining what representations and edits are both feasible to develop and useful for the

target users. Given the variety of creative and professional tasks that speech technology is used for, users may desire different levels of control and different editing capabilities. For example, animators utilize automated translation services (e.g., Papercup) to produce "dubbed" versions of films suitable for distribution in different companies, audio engineers use automatic pitch correction (e.g., Melodyne) to quantize F0 to musical notes, and voice actors and sound effect artists use a wide variety of tools and techniques to sculpt custom character voices for films and video games. As such, improving the creative workflow can mean improving the quality of existing state-of-the-art editing capabilities, or proposing new and demonstrably useful editing capabilities. Thus, my contributions both improve the state-of-the-art of existing editing capabilities of known utility (e.g., pitch-shifting) and propose new speech editing controls (e.g., fine-grained pronunciation control). These contributions are made possible by my proposed disentangled speech representation for fine-grained speech editing (Chapter 2), which allows the capabilities I develop to be easily integrated into any existing neural speech editing systems by replacing the existing speech representation with my proposed representation.

My speech representation and editing system (Figure 1.2) enable a unified approach to fast and high-fidelity editing of pitch, duration, volume, timbral correlates of volume, spectral balance, pronunciation, and speaker identity. Of these, timbral correlates of volume, spectral balance, and pronunciation control represent novel editing capacities developed in my research (Chapter 4). Physiologically, controlling the timbral correlates of volume corresponds to muscular contractions in the larynx and diaphragm and variations in air pressure produced by the lungs. This voice quality control is useful for increasing or decreasing the perceived intensity of the spoken content independent of volume. My

proposed disentangled spectral balance editing method changes the relative amount of energy at high or low frequencies. This produces a similar effect as, e.g., Alvin from *Alvin and the Chipmunks*, but without requiring voice actors to sing/speak unnaturally slowly. Lastly, fine-grained pronunciation control enables downstream applications such interpretable accent conversion, automatic onomatopoeia (i.e., producing speech recordings of speakers mimicking a designated non-speech sound), and fixing nuanced mistakes in pronunciation and accent during post-processing without having to rerecord.

While many neural speech editing systems have emerged over the past decade [**14, 49, 115, 120**], none of them adequately address the root cause of failure of speech editing systems: an inadequate speech representation. Entangled, non-informative, or non-intuitive speech representations prohibit accurate expression or synthesis of the user's intent. My dissertation uncovers and addresses key issues in disentangled and interpretable speech representation, and demonstrates the efficacies of resolving those issues by setting the state-of-the-art on a number of speech editing (and speech representation) tasks. Specifically, I find accurate representation of F0 (Section 2.2) and pronunciation (Section 2.1) to be key issues inhibiting editing capacities of prior works. I also address minor representational issues such as an improved representation of vocal cord engagement (i.e., the voiced/unvoiced decision) (Section 2.3) and a multi-band A-weighted loudness (Section 2.4) to trade-off reconstruction fidelity and disentanglement.

1.2. Contributions

My contributions in this dissertation can be organized as either improving the representation of speech, improving the speech synthesis system that produces speech from the

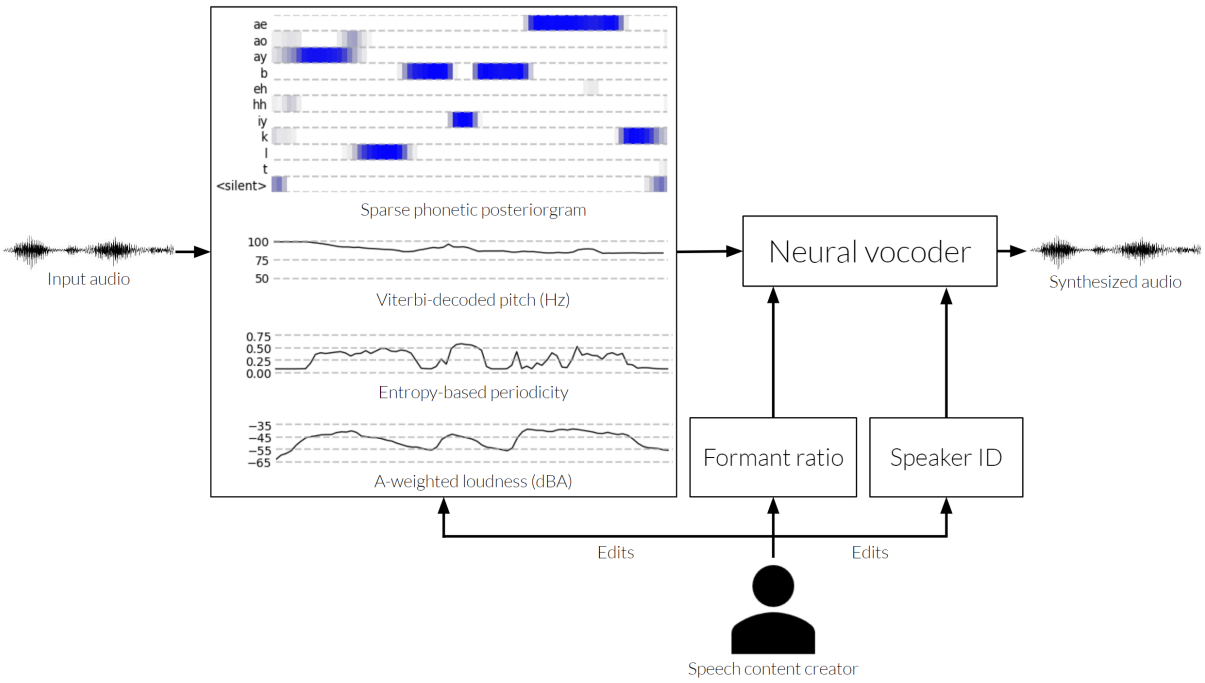

Figure 1.2. **My proposed speech representation (Chapter 2) and speech editing system (Chapter 3)** | Input audio (Arnold Schwarzenegger saying "I'll be back" from the movie The Terminator) is first encoded in my proposed disentangled and interpretable representation (Chapter 2) consisting of a sparse phonetic posteriorgram (SPPG) (Section 2.1), Viterbi-decoded neural pitch estimation (Section 2.2), entropy-based periodicity (Section 2.3), and multi-band A-weighted loudness. In the visualization of SPPGs (**top**; **blue**), Unused phonemes with maximal probability less than 10% are omitted for clarity. For interpretability and more intuitive control, I visualize the single-band A-weighted loudness instead of my proposed multi-band A-weighted loudness (Section 2.4; Figure 2.8). Speech content creators and producers edit the interpretable representation as well as the speaker identity (Section 3.2.2) and augmentation ratios (Section 3.2.1) to perform a variety of high-fidelity speech editing operations (Chapter 4) such as fine-grained (i.e., frame resolution) pitch-shifting, time-stretching, and loudness editing, as well as global (i.e., applied to the entire speech recording) editing of speaker identity and spectral balance.
`figure-1-2-schwarzenegger.wav`

proposed representation (including any potential edits), or proposing novel speech editing capabilities. My contributions are as follows.

- **Contributions to speech representation (Chapter 2)**

  - I develop an efficient neural pitch estimator with state-of-the-art pitch and periodicity accuracy on speech and music data (Sections 2.2-2.3).

  - I enable disentangled representation of pronunciation by advancing the state-of-the-art of a pronunciation representation called a phonetic posteriorgram (PPG) (Section 2.1).

- **Contributions to interpretable speech synthesis (Chapter 3)**

  - To improve the quality and efficiency of speech editing, I develop a fast, high-quality neural speech synthesizer that produces speech waveforms from my proposed speech representation. My interpretable speech synthesis system perform speech reconstruction with comparable perceptual accuracy to Mel spectrograms—a standard high-anchor for speech synthesis tasks.

- **Contributions to high-fidelity speech editing (Chapter 4)**

  - I demonstrate that my proposed interpretable speech synthesizer enables accurate, high-fidelity control over pitch, duration, volume, timbral correlates of volume, pronunciation, spectral balance, voice conversion, and speaker adaptation. Three of these controls (spectral balance, timbral correlates of volume, and pronunciation) represent novel speech editing capabilities not afforded by prior works.

Together, my contributions set the foundations for speech editing software that expedited and advances the workflows of speech content creators responsible for the development of many of the primary means of communication (e.g., film, podcasts, and social

media) used today. In Chapter 5, I describe future directions for research and development of speech technology made possible by my proposed contributions, such as real-time accent and pronunciation feedback for language learning and patients with speech and language disorders, querying audio databases via vocal imitation, and low-bandwidth interpretable speech coding.

CHAPTER 2

A disentangled, interpretable representation of speech

Different speech analysis and content creation tasks necessitate different representations of speech. For example, clinical speech pathologists utilize the international phonetic alphabet (IPA) phone set (as opposed to, e.g., the CMU pronunciation dictionary used in this thesis) due to having some physiological correspondence with the human speech mechanism[1], text-to-speech systems use representations that best serve as a bridge between the lexical and audio distributions (as measured by perceptual speech naturalness and fidelity) [**33, 49, 111**], and singers transcribe and read singing from lexically-annotated musical notation that coarsely represents F0 and loudness information. **What is the representation of speech most appropriate for editing tasks commonly used in speech content creation? Such a representation should possess four properties.**

(1) **Property 1 |** The ability to be computed from a speech recording without a transcript, which minimizes the need for manual or automatic lexical annotation—or even adherence to one or more languages as opposed to the more general set of speech sounds.

[1]A *phone* is an atomic unit of speech sound, whereas a *phoneme* is a set of phones. Phones of the same phoneme set are called *allophones*. The one-to-many mapping of phonemes to phones is known as *allophonic variation*.

(2) **Property 2 |** *Interpretable* control parameters that accurately represent the underlying speech attributes to be edited (e.g., prosody and pronunciation).

(3) **Property 3 |** The ability to be used as an input for high-quality speech synthesis. Combined, properties 1 and 3 indicate that an appropriate representation for speech editing should be *invertible*.

(4) **Property 4 |** *Disentanglement*, wherein representations of various parameters can be independently edited without impacting each other (e.g., changing the pitch without changing pronunciation). Speech editing of multiple parameters requires each parameter to be disentangled relative to all other parameters.

My experiments presented in this thesis indicate that my representation (Figure 1.2) satisfies all four of these criteria for a representation for fine-grained speech editing and consists of four disentangled[2], interpretable, time-aligned features: sparse phonetic posteriorgrams (SPPGs) (Section 2.1), Viterbi-decoded neural pitch estimation (Section 2.2), entropy-based periodicity (Section 2.3), and multi-band A-weighted loudness (Section 2.4). In this chapter, I describe each of those speech representations and my contributions to their development. I also

[2]Attentive readers may note two remaining sources of entanglement: (1) entanglement of periodicity information between the periodicity and SPPGs (e.g., the voiced/unvoiced confidence can also be derived as the total probability assigned to voiced phonemes) and (2) entanglement between the multi-band loudness and the silence token in the SPPGs. Entanglement (1) can be solved by automatically or manually editing the periodicity when the total voiced/unvoiced confidence of the SPPG changes. Entanglement (2) does not impact any common type of speech edit, but can similarly be solved by automatically or manually editing the SPPG silence probability with edits to the A-weighted loudness.

describe existing speech representations that precede the development of my representation and compare them to my proposed speech representation, demonstrating that all prior representations fail to capture at least one of these properties (Section 2.5). In the following two chapters, I demonstrate that neural speech synthesis using my proposed representation produces accurate and high-fidelity speech reconstruction (Chapter 3) and speech editing (Chapter 4).

2.1. Sparse phonetic posteriorgrams (SPPGs)

To the best of my knowledge, my proposed sparse phonetic posteriorgram (SPPG) is the first representation of speech pronunciation that has the combined properties of being interpretable, suitable for fine-grained pronunciation control (Section 4.7), and amenable to high-fidelity speech synthesis (Chapter 3). A phonetic posteriorgram (PPG) [**28**] (Figure 1.2; top) is a non-stationary (i.e., time-varying) categorical distribution over acoustic units of speech (e.g., phones or phonemes). Representations similar to PPGs have been useful in speech generation due to their ability to disentangle pronunciation features from speaker identity, allowing accurate reconstruction of pronunciation (e.g., voice conversion) and coarse-grained pronunciation editing (e.g., foreign accent conversion). As such, all five top entries to the 2020 Voice Conversion Challenge [**125**] utilize representations similar to my proposed interpretable PPGs, but without the affordances of interpretability or control. This lack of interpretability and control is due to the construction of these representations as intermediary activations of automatic speech recognition (ASR) systems [**59**] or distributions over learned latent variables [**124, 126**] as opposed to a

distribution over interpretable phoneme categories. Beyond voice conversion, text-to-speech (TTS) systems that predict PPG representations from text as an intermediate step show improved pronunciation over comparable systems that predict speech directly from text [**103**] and have enabled accent conversion [**124**]. However, this accent conversion utilizes non-interpretable representations, such that any errors in the synthesized accent cannot be fixed without changing the entire speech recording. No prior work has demonstrated a system capable of fine-grained pronunciation editing suitable for fixing localized pronunciation errors such as those that occur due to mispronunciations or incorrect dialects or accents (by humans or TTS systems).

Text-to-speech (TTS) systems utilizing neural networks [**60, 49**]) typically use as an input representation the sequence of phonemes indices, extracted from the transcript via a grapheme-to-phoneme process. This representation's precision for accurately reconstructing pronunciation is limited by the granularity of the phoneme set, which coarsely discretized the continuous space of speech pronunciation induced by time-varying articulatory positions (Section 1.1.1). As well, this phoneme representation does not specify phoneme durations: phoneme durations are generated from lexical features, such as the phones or phonemes generated from text by a grapheme-to-phoneme (G2P) system. TTS systems that explicitly model the conditional distribution of phoneme durations given lexical features [**91**] offer a partial solution: phoneme durations can be inferred from ground truth speech and corresponding transcripts (e.g., via forced phoneme alignment [**127**]). However, this necessitates a highly accurate external forced phoneme aligner and quantizes phoneme durations to the frame resolution. In contrast, PPGs provide a more fine-grained representation of pronunciation, such as allowing for precise interpolation between

discretized lexical features (Figure 4.11). PPGs also preserve the precise alignment and permit training of speech synthesizers without access to the speech transcript—requiring a transcript only for the initial training of the generalizable PPG model.

Prior works have noted that pronunciation is preserved during voice and accent conversion when using representations like intermediate activations of ASR systems [**59**] or distributions over learned latent variables [**124**]—and have even used the term PPG to refer to some of these representations. While all of these are multi-dimensional, continuous-valued representations amenable to high-quality speech synthesis, none of these representations permit the interpretability, disentanglement, and control afforded by true PPGs built upon interpretable phonetic categories.

In this section, I improve the accuracy of state-of-the-art interpretable PPG representations on unseen datasets (Section 2.1.3) and demonstrate that interpretable PPGs exhibit comparable pitch modification accuracy to non-interpretable baseline speech representations [**15**]. I propose *sparse* PPGs (SPPGs) to prevent synthesis artifacts due to noise in low-probability PPG bins (Section 2.1.2). I further perform novel analyses of high-fidelity, interpretable PPG representations to show insights such as an interpretable acoustic pronunciation distance at the frame resolution (Section 2.1.4) as well as the first examples of interpretable, fine-grained pronunciation control, such as making a precise edit to a specified "ey" sound to sound more like an "aa" or interpolating between two phonemes (Section 4.7; Figure 4.11).

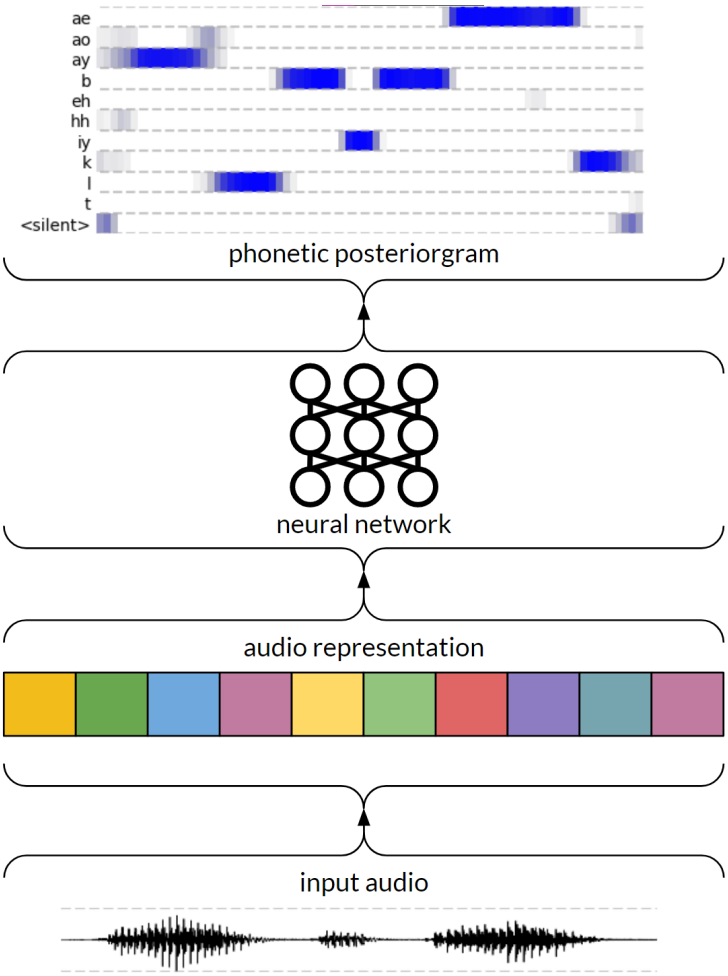

Figure 2.1. **Neural phonetic posteriorgrams (PPGs) |** My proposed method for producing PPGs using a neural network. I extract an audio representation (e.g., a Mel spectrogram; middle) from speech (bottom) and train a neural network to infer a time-varying categorical distribution over phoneme categories (i.e., a PPG; top).
`figure-1-2-schwarzenegger.wav`

2.1.1. Inferring PPGs using neural networks

Let $x_1, \ldots, x_T$ be a sequence of T adjacent frames of audio, where each frame consists of window size W samples and is offset from the previous frame by hop size H samples. I first encode $x_1, \ldots, x_T$ in an audio representation more suitable for a neural network

(e.g., a Mel spectrogram) and then train the neural network to infer PPG $G \in \mathbb{R}^{|P| \times T}$, a time-varying categorical distribution over phoneme set P (i.e., $G_{p,t}$ is inferred probability that the speech in frame t is phoneme p).

My neural network architecture consists of an input convolution layer, five Transformer encoder layers (self-attention and a feed-forward network) [**108**], and an output convolution layer that produces a categorical distribution via softmax activation over the $|P| = 40$ phonemes (including silence) from the CMU Pronunciation Dictionary phoneme set.[3] I use a kernel size of five for the input and output convolution layers. My Transformer layers use two attention heads and 512 channels. For each representation, I selected the number of layers and channels via hyperparameter search on a heldout validation partition from Common Voice [**3**] (Section 2.1.3.1). I fixed the number of channels at 128; trained using 3, 4, 5, and 6 layers; and then fixed the number of layers at best of these values and trained models with 128, 256, 512, and 1024 channels, selecting the best of these. In the event of divergence from overparameterization (i.e., *gradient confusion* [**95**]), I allowed one reload from checkpoint. I find 5 layers and either 256 (for EnCodec and Mel spectrograms) or 512 (for all others) channels to be optimal.

I use an Adam optimizer [**36**] with a learning rate of $2e^{-4}$ to optimize categorical cross entropy loss between predicted and ground truth phoneme categories at each ten millisecond frame. I train my PPG models for 200,000 steps using a variable batch size [**25**] of up to 150,000 frames per batch.

[3] `speech.cs.cmu.edu/cgi-bin/cmudict`

2.1.2. Inducing sparsity in PPGs

Noisy or uninformative regions in a speech editing representation (e.g., low-probability phoneme bins in the PPG) have relatively high conditional entropy with the speech waveform. I hypothesize that this high-entropy noise can be memorized by a neural speech synthesizer, inducing overfitting that harms generalization during reconstruction and editing. I address this by proposing *sparse phonetic posteriorgrams* (SPPGs). I explore three methods for producing SPPGs: (1) (**Top-k**) set all but the k most-probable phonemes at each frame to zero; (2) (**Threshold-k**) set all phonemes with probability less than k to zero; and (3) (**Percentile-k**) sort the phonemes of each frame by descending probability, add phoneme probabilities until the sum reaches k, and set all remaining phoneme probabilities to zero. I renormalize SPPG frames to sum to one. I find Percentile-k with $k = 0.85$ to be the best SPPG hyperparameter configuration (Section 2.1.3). An example of the difference between a PPG and corresponding SPPG is visualized in Figure 2.2.

2.1.3. Objective evaluation of PPGs

My evaluation in this section determines what audio input representation(s) (Section 2.1.3.2) is best for producing accurate PPGs—where accuracy is measured by the frame-wise phoneme accuracy relative to ground truth phoneme transcriptions—and evaluates the efficacy of my high-fidelity neural PPGs to produce an accurate, frame-resolution pronunciation distance.

In my original publication describing my proposed PPGs [**15**], I further demonstrate that a common speech synthesis system (VITS [**34**]) performs pitch-shifting using only my

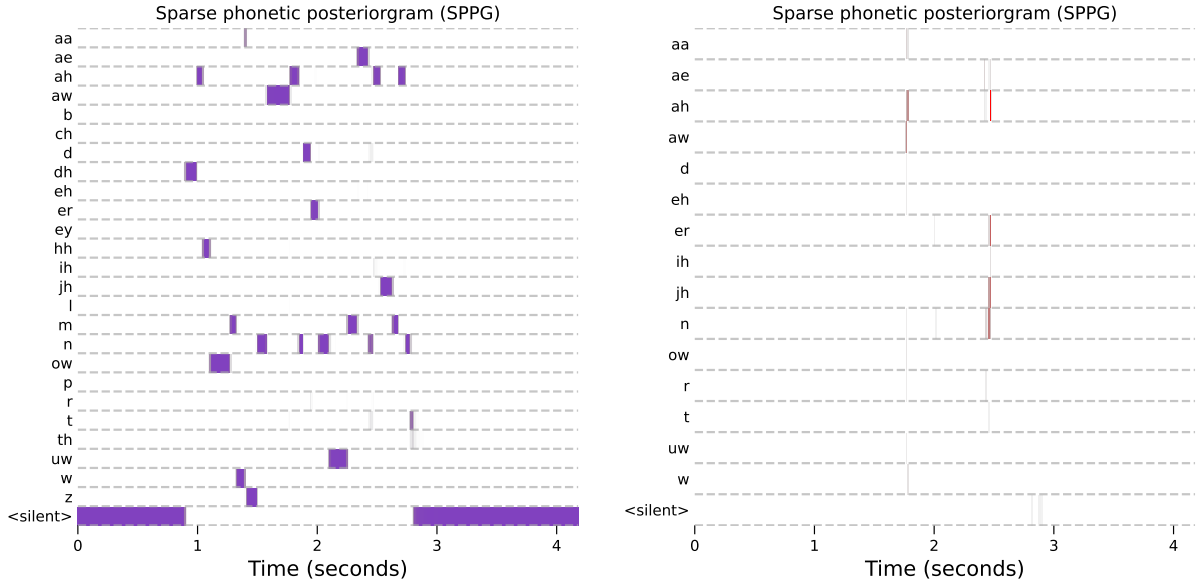

Figure 2.2. **Sparse phonetic posteriorgrams (SPPGs)** (**left**) An overlay of a PPG and its corresponding SPPG (using Percentile-k sparsification with $k = 0.85$). **Blue** indicates PPGs, **red** indicates SPPGs, and **violet** indicates agreement between PPGs and SPPGs. (**right**) **Red** indicates disagreement between SPPGs and PPGs. For both plots, only phonemes with a maximal probability above 0.1% are depicted.
`figure-2-2-0070-000751.wav`

proposed PPGs and neural pitch estimation with pitch and pronunciation accuracy approaching top non-interpretable speech representations such as wav2vec 2.0 [**7**] and Mels. I also corroborate the framewise phoneme accuracy results presented here with a subjective preference test of speech reconstruction, which verifies that improvements in framewise phoneme accuracy reduce audible mispronunciations or artifacts [**15**]. My follow-up work on performing speech reconstruction from my proposed representation [**71**] further improves pitch-shifting accuracy and enables speech reconstruction with comparable perceptual similarity as Mel spectrogram vocoding (Section 3.4)—a standard high-anchor

for quality and naturalness for text-to-speech systems—while enabling many high-fidelity speech editing capabilities not afforded by Mel spectrograms (Chapter 4).

PPGs allow a more nuanced representation of pronunciation than discrete phonemes. This is due to the PPG model expressing continuous-valued uncertainty over the most likely phonemes. This uncertainty can arise from errors in ground truth grapheme-to-phoneme (G2P) or Nonetheless, any network that outputs PPGs from audio input should broadly agree with high-quality, aligned phonetic transcriptions of the input audio. Therefore, I perform objective evaluation to determine the extent to which my PPG representations computed from each of the input representations described in Section 2.1.3.2 are accurate representations of the ground truth phoneme categories. I evaluate the framewise phoneme accuracy, or the proportion of frames where the ground-truth phoneme is assigned highest probability by the model. I perform evaluation on the test partitions of Common Voice and TIMIT, as well as all of Arctic.

Note that 100% framewise phoneme accuracy does not correspond to an ideal PPG. For 100% accuracy to be ideal, the phoneme transcriptions and alignments used for training and evaluation would have to be free of any noise. However, phoneme transcriptions used as training data are derived from an American English grapheme-to-phoneme (G2P) system. This means the phoneme transcriptions are derived solely from the transcript, without access to the speech recording or information about the speaker. Therefore, pronunciation variations due to speech rate, allophonic variation, prosodic context, or variations due to accent are not indicated in the forced phoneme alignment used as ground truth. This indicates that there is an unknown, dataset-specific ideal framewise phoneme accuracy that depends on the amount of noise in the data. In practice, I assume that

	Utterances	Speakers	Hours
Arctic [**39**]	7816	6	7.05
Common Voice [**3**]	864418	29493	1375.29
TIMIT [**22**]	6300	630	5.38
VCTK [**119**]	264468	109	82.37

Table 2.1. Statistics of each of the datasets used to train or evaluate PPGs.

higher accuracy is better, but utilize objective and subjective evaluation of downstream tasks (e.g., speech synthesis and editing) for verification. As well, my proposed, interpretable PPGs offer a potential interface for annotators aiming to perform manual verification of phoneme transcriptions and alignments (e.g., by using an inferred PPG as a starting point and making manual corrections as needed). I hypothesize that if this manual annotation was performed well such that a verifiably noise-free dataset existed, it could be used to determine the value of the "ideal" accuracy of a PPG on a different dataset.

The efficacy of sparsifying PPGs cannot be properly evaluated via framewise phoneme accuracy, because none of the proposed sparsification methods changes which phoneme is assigned highest probability. My proposed SPPG methods can instead be evaluated by training a speech synthesis system on each candidate PPG or SPPG representation and comparing these systems using my evaluation methodologies and objective metrics for speech reconstruction and editing (Section 3.3.1). Using these objective metrics for speech reconstruction and editing, my hyperparameter search over k for each method indicated Percentile-k with $k = 0.85$ produces the most accurate speech reconstruction and editing. I ablate speech editing between PPGs and Percentile-0.85 SPPGs in Section 4.8. Fine-grained control of pronunciation using SPPGs is demonstrated in Section 4.7.

2.1.3.1. Data for training and evaluating PPGs. I train my neural PPG model on a subset of the Common Voice dataset [**3**]. I perform objective evaluation of phoneme accuracy using a held-out partitions of Common Voice and TIMIT [**22**], as well as the full CMU Arctic [**39**] dataset. All datasets are English speech with a diverse range of speakers and accents. TIMIT and ARCTIC contain manually-aligned transcripts. For Common Voice, I use the alignments produced by the Charsiu forced-aligner [**127**]. The transcripts for Arctic and TIMIT are phonetically balanced (i.e., phonemes show up equally often in the transcript, but not necessarily for an equal number of frames of audio) and manually time-aligned. I partition Common Voice into train/valid/test partitions of proportions 80%/10%/10%. I report the number of utterances, speakers, and total hours of speech recordings of each dataset that I use to train or evaluate PPGs in Table 2.1.

2.1.3.2. Audio input representations. All representations are computed at a hopsize of ten milliseconds (ms) and a sample rate of 16,000 Hz, unless otherwise stated.

Mel spectrogram (Section 1.1.2; Figure 1.1) [80 channels] | Spectrograms are a common representation for speech research tasks. I use log-energy magnitude spectrograms computed from the raw audio with a window size of 1024 and bin the frequency channels into 80 Mel-spaced bands.

Wav2vec 2.0 [**7**] [768 channels] | Wav2vec 2.0 is a neural speech encoder with competitive ASR Phoneme Error Rate (PER) when fine-tuned on TIMIT. PER is similar to my objective metric of framewise phoneme accuracy (Section 2.1.3); however, PER does not require the alignment between the phonemes and the audio to be preserved. Wav2vec 2.0 uses a 20 ms hopsize. I apply nearest neighbors interpolation to double the number of timesteps to a 10 ms hopsize.

Charsiu [**127**] [768 channels] | Charsiu appends a convolutional layer to a pretrained wav2vec 2.0 base model that upsamples from the 20 ms hopsize to a 10 ms hopsize. The wav2vec 2.0 feature encoder is frozen and the rest of the model is fine-tuned to maximize a categorical cross entropy loss over ground truth derived via grapheme-to-phoneme and forced alignment [**64**]. I use the `W2V2-FC-10ms` model, which is among the state-of-the-art in forced phoneme alignment [**127**].

ASR bottleneck [**59**] [144 channels] | This is a pretrained ASR model with an encoder-decoder architecture. I use the bottleneck features output by the pretrained encoder, which is also used in voice conversion and TTS for its pronunciation-preserving qualities [**43, 103**].

EnCodec [**18**] [128 channels] | EnCodec converts audio into codebook indices of 32 codebooks—each containing 1024 codes and 128 channels—and then performs an element-wise sum over codebooks. EnCodec achieves competitive results on low-dimensional, invertible speech representation learning [**18**] and text-to-speech [**111**].

2.1.3.3. Objective evaluation results. Figure 2.3 provides results for my objective evaluation of PPGs. Wav2vec 2.0 achieves the highest overall accuracy across datasets, followed by Mel spectrograms. In Churchwell et al., [**15**] I perform a subjective evaluation of reconstruction using a Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) [**30**] style subjective evaluation. Each participant performs 10 MUSHRA trials. In each trial, one utterance in a test partition of the VCTK dataset [**119**] is reconstructed using only the PPGs (without sparsification) and a pitch contour. I compare PPGs computed from each of the five input representations. The reconstructions are

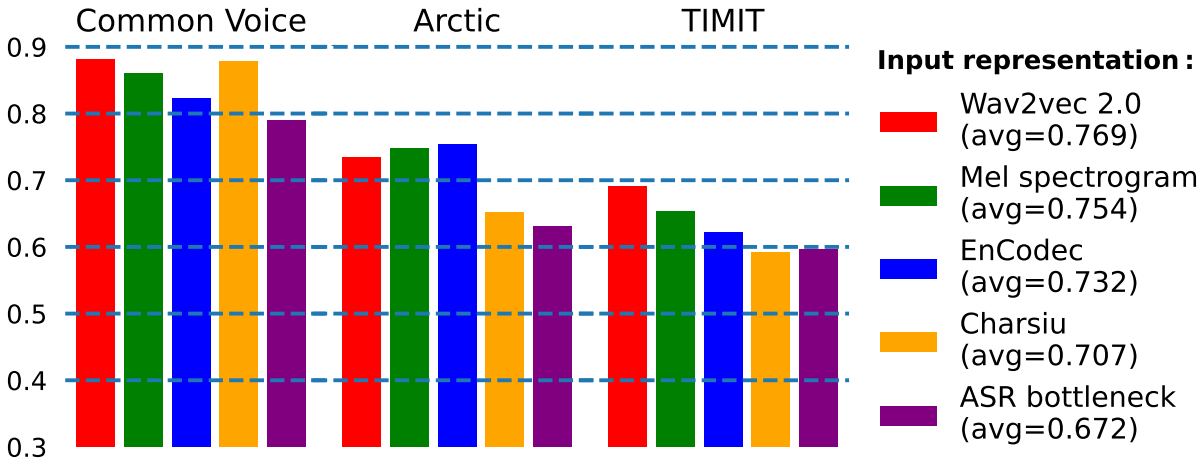

Figure 2.3. **Average framewise phoneme accuracy |** Accuracy of PPGs computed from five input representations. The wav2vec 2.0 [**7**] input representation has the best PPG accuracy when averaged over all datasets (see legend). **N.B.** The base wav2vec 2.0 model of Charsiu [**127**] was trained on some of my Common Voice test partition as well as the TIMIT training partition, making Charsiu's results on those datasets unreliable upper bounds.

rated in comparison to each other on a 0-100 quality scale using a set of sliders. Two references are also included in the comparison set: (1) the high-quality original speech audio (the high anchor) and (2) a low-quality, 4-bit quantization of the original audio (the low anchor). I recruited 50 participants. I filtered out 26 annotators that either failed the listening test or rated the low anchor (4-bit quantized audio) as higher quality than the high anchor (original audio). Results indicate that the perceptual quality of speech reconstruction with (non-sparsified) PPGs inferred from Mel spectrograms marginally outperforms wav2vec 2.0. Given the size, interpretability, generality, and relative simplicity of computing a Mel spectrogram, I use PPGs (and SPPGs) inferred from Mel spectrograms for the remainder of this dissertation.

2.1.4. PPGs encode acoustic pronunciation distance

Pronunciation distances are useful for automatic evaluation of language acquisition, language or accent learning, and speech pathology. They are also a useful evaluation metric for speech enhancement and speech synthesis systems. I propose an interpretable distance measure of framewise pronunciation error. Let $G \in \mathbb{R}^{|P| \times T}$ be a phonetic posteriorgram on phoneme set P and time frames T, such that $G_{p,t}$ is the inferred probability that the speech in frame t is phoneme p. By default, my PPG training is not class-balanced, and some phonemes are significantly more likely to occur in the dataset (e.g., "aa" occurs far more often than "zh"). To prevent this from imposing bias on my proposed distance, I train a class-balanced PPG model using class weights $\lambda_i = \min_j C_j / C_i$ to weight the relative contribution of each phoneme to the training loss, where C_x is the number of frames where phoneme x is ground truth. I extract from this class-balanced model an interpretable representation of similarity between phonemes $\mathcal{S} \in \mathbb{R}^{|P| \times |P|}$ (Figure 2.4) that captures the extent to which—for all pairs of phonemes $(b, c) \in P \times P$—the model assigns probability to phoneme c when it predicts b as the most likely phoneme. My proposed acoustic pronunciation distance ΔPPG can be stated as follows, where JS is the Jensen-Shannon divergence.

$$\Delta\text{PPG}(G_t, \hat{G}_t) = \text{JS}(\mathcal{S}^\gamma G_t, \mathcal{S}^\gamma \hat{G}_t) \tag{2.1}$$

Hyperparameter γ controls the extent to which phoneme similarity matrix $\mathcal{S}$ influences pronunciation distance. Intuitively, a pronunciation distance should be lower when all words are pronounced similar to the reference pronunciation and higher when this is not

the case. This is also true of word error rate (WER), a commonly used speech intelligibility evaluation metric that measures the difference in automatic speech recognition (ASR) output between synthesized audio and a ground truth transcript. WER performs a dynamic programming alignment between the words of target and hypothesis sentences to compute the minimum number of word insertions n_i, deletions n_d, and substitutions n_s required to transform the hypothesis sentence into the target. Given a ground truth transcript containing n words and transcript derived from corresponding predicted audio, WER is as follows.

$$\text{WER} = \frac{n_i + n_d + n_s}{n} \tag{2.2}$$

I tune γ on my validation partition to maximize the Pearson correlation with word error rate (WER) using Whisper-V3 [**90**] using 2,000 audio files of test data from my original pitch disentanglement experiments [**15**] that have been pitch-shifted by ± 200 cents. Using optimal hyperparameter $\gamma = 1.20$, ΔPPG demonstrates strong Pearson correlation with WER ($r = 0.697$; $n = 2000$; $p = 1.76 \times 10^{-291}$). In Section 3.3.1, I discuss some potential issues with data leakage using WER with Whisper-V3 that can be resolved by using my proposed ΔPPG.

Pronunciation distances can be utilized to develop automated software for the evaluation of language acquisition (in infancy), language learning, voice acting (e.g., learning accents) and clinical speech pathology. In such applications, a reference recording is played containing the specific phonemes or speech accent of interest. A user listens to the recording and then attempts to mimic the pronunciation. The software then scores the user's pronunciation relative to the reference using, e.g., ΔPPG. Framewise pronunciation

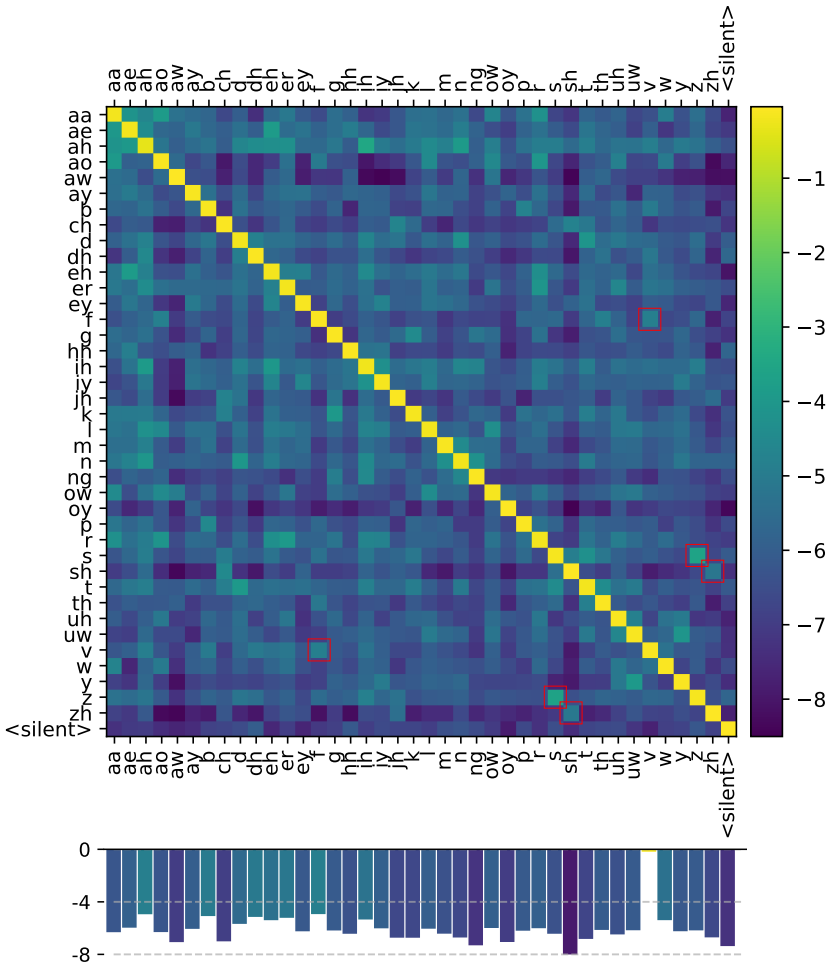

Figure 2.4. **Acoustic phoneme similarities |** (**top**) Row b column c is $\ln \mathcal{S}_{b,c} = \ln \mathbb{E}\left[\lambda_c G_{c,t}; \lambda_b G_{b,t} \geq \lambda_z G_{z,t} \, \forall z\right]$, the log of the average class-weighted probability assigned to phoneme c when phoneme b is the maximum model prediction. Averages are taken over all frames of my validation partition of Common Voice [**3**] using my PPG model trained with class-balancing on Mel spectrogram inputs. **Red** boxes show that the corresponding unvoiced fricative (/f/, /s/, /sh/) to each voiced fricative (/v/, /z/, /zh/) is assigned relatively high probability, and vice versa. Class-balanced training and class-weighting are used to remove column banding indicative of natural phoneme frequency. Remaining column banding (e.g., "ah" is brighter than "ao") is caused by differences in self-similarity (i.e., the yellow diagonal). (**bottom**) Zooming in to the row of voiced fricative "v" demonstrates that corresponding unvoiced fricative "f" is usually the next most likely prediction when "v" is most likely.

distances (such as ΔPPG) provide the additional advantage of specifying precise frames containing mispronunciations for playback and practice. However, it is not reasonable to expect the user to perfectly match the phoneme durations of the reference. My pronunciation distance can be easily extended to compare audio of different lengths and unknown alignment by first performing dynamic time warping (DTW) to find an alignment. This is similar to Bartelds et al. [**8**], who measure distance between unaligned speech recordings by computing the optimal alignment using DTW and then taking the L2 distance between aligned wav2vec 2.0 features. I compare ΔPPG to the method proposed by Bartelds et al. in Section 4.7 (Figure 4.11; bottom).

The existing state-of-the-art lexical-based pronunciation distance [**56**] does not directly compare to, e.g., wav2vec 2.0 latents [**8**], but note that "Pursuing [self-supervised] neural models to predict accent distance (from Bartelds et al., 2022 [**8**]) is another potentially worthwhile future direction, if the cost- and time-related barriers to training these models could be addressed." While I disagree with the cost- and time-related barriers—those only apply to *training* a large, self-supervised model, not performing inference with a pre-trained model as in Bartelds et al.—I note that my Mel-based PPG distance shows obvious merits over the wav2vec 2.0 method proposed by Bartelds et al. [**8**] while taking less than a week to train using one GPU and less than one Tb of data storage. In practice—as with wav2vec 2.0—the availability of my code and pre-trained models for inferring PPGs omits the need to perform such training: my proposed pronunciation distance can be run on commodity hardware. In fact, me and a research collaborator are currently developing an application for visualizing and comparing PPGs on a consumer mobile device. In summary, my proposed PPG-based pronunciation distance addresses

the entanglement of prior acoustic pronunciation distances [**8**] with attributes other than pronunciation (Figure 4.11), exhibits high correlation with a standard speech intelligibility metric (WER), and solves the open challenges of acoustic-based pronunciation distances indicated by the authors of the state-of-the-art lexically-based pronunciation distance.

2.2. Viterbi-decoded neural pitch estimation

Vibrating objects produce sound. When those vibrations are periodic (and in the range of human hearing), we perceive a *pitch*.[4] In languages such as English and Spanish, pitch is used to indicate emphases and phrase boundaries [**83**], and in tone-based languages (e.g., Mandarin Chinese) it indicates lexico-semantic content. Given that differences in pitch can fundamentally change a listener's perception of a speaker's characteristics, such as their current emotion [**21**], estimated pitch contours are widely used in speech synthesis [**75, 91, 107**], singing voice synthesis [**58**], voice conversion [**89**], voice privacy [**85**], and prosody editing [**74**]. Physiologically, pitch contours used for speech synthesis aim to capture the human perception of pitch produced by the opening and closing of the glottis at rates within the audible frequency range. The frequency of the opening and closing of the glottis is determined by the dimensions of the vocal mechanism as well as the amount of contraction in the muscles of the larynx (Section 1.1.1).

Pitch estimation using machine learning techniques—neural networks, in particular—significantly outperform prior digital signal processing (DSP) methods in accuracy and noise robustness. However, the errors of prior neural pitch estimators induced inaccuracies, entanglement, and audible artifacts in the resulting audio, and the speed of neural

[4]This pitch typically agrees with the fundamental frequency (F0) [**121**], but can occasionally disagree [**27**]. As with prior works, I assume these disagreements can be ignored and refer to both as "pitch".

pitch estimation was significantly slower than DSP-based methods. Further, while DSP-based methods are typically expected to work for any frequency range, neural pitch estimators are only capable of performing estimation within the frequency range of the training data. This makes it unclear as to how or whether neural pitch estimators can generalize across audio domains (e.g., speech and music). In this chapter, I describe my proposed, updated version of the FCNF0 neural pitch estimator [**2**] (FCNF0++) that addresses these issues. FCNF0 is itself an extension of CREPE [**35**], an early and very popular neural pitch estimator. FCNF0 and CREPE are both prior state-of-the-art neural pitch estimators. In my original paper on pitch estimation [**72**], I also compare to DeepF0 [**99**] and HarmoF0 [**117**]. DeepF0 achieves marginally better pitch estimation accuracy, but is roughly 10x slower than FCNF0. HarmoF0 jointly performs pitch estimation on all frames of a speech recording—as opposed to independently performing estimation on single frames. This prohibits generalization due to domain-specific transition probabilities being learned by the model. For these reasons, I use CREPE (Section 2.2.1) and FCNF0 (Section 2.2.2) as baselines.

My contributions to pitch estimation are as follows.

- I propose a set of techniques (Section 2.2.3) that improve the pitch estimation accuracy of multiple recent neural pitch estimators: such as CREPE, FCNF0, and DeepF0.
- I produce a high-fidelity pitch estimator with CPU inference speed approaching fast, DSP-based methods and GPU inference speed that is hundreds of times faster than real-time—making my proposed estimator amenable for pitch estimation on large speech datasets.

- I demonstrate how a neural pitch estimator can generalize well to training and inference across audio domains, such as speech and music.

2.2.1. Baseline: CREPE

Convolutional REpresentation for Pitch Estimation (CREPE) [**35**] is a neural pitch estimator composed of six convolutional blocks followed by a linear layer. Each convolutional block consists of a one-dimensional convolution, ReLU activation, batch normalization [**32**], and dropout [**104**]. CREPE takes as input 1024 samples of a 16 kHz audio waveform that has been preprocessed to have a mean of zero and standard deviation of one. The network produces logits that yield independent Bernoulli posterior probabilities over each pitch bin after sigmoid normalization.

$$p(y = f|x);\ f \in F \tag{2.3}$$

where $x \in \mathbb{R}^{1024}$ are the input audio samples, y is the pitch of the input audio, f is the center frequency of a quantized pitch bin, and F is a set of pitch bins. CREPE uses $|F| = 360$ pitch bins, spaced every 20 cents between 31 and 2006 Hz.

$$f_i = 31 \times 2^{\frac{20i}{1200}};\ i = 1, \ldots, |F| \tag{2.4}$$

CREPE uses a sigmoid activation and binary cross-entropy (BCE) loss on each pitch bin. CREPE applies a Gaussian blur to the one-hot-encoded training targets (with a standard deviation of 25 cents) followed by peak normalization. This encourages the network to increase the variance of its prediction by assigning probability mass to adjacent pitch bins.

2.2.2. Baseline: FCNF0

Fully-Convolutional Network for Pitch Estimation (FCNF0) [**2**] makes several modifications to CREPE. FCNF0 omits zero-padding from convolutional layers, as a significant amount of inference time in CREPE is spent performing computations on zero-padding. The authors replace the output linear layer with a convolutional layer to improve speed. FCNF0 omits dropout to increase the network capacity and uses an 8 kHz sampling rate for the input audio instead of 16 kHz. Otherwise, the six convolutional blocks of FCNF0 are identical to CREPE, but with a different number of convolution channels, and with pooling strides and kernel sizes hand-tuned to produce output of a desired length provided the reduction in the time dimension due to omitting zero-padding. The output categories of FCNF0 are 486 pitch bins with a minimum pitch of 30 Hz and a bin width of 12.5 cents, resulting in a frequency range of 30–1000 Hz.

2.2.3. Proposed methods for pitch estimation

I propose seven modifications to FCNF0 in order to improve pitch estimation as follows. **Increase the frequency resolution |** CREPE predicts bins with a quantization width (i.e., the spacing between the centers of two adjacent bins) of 20 cents. FCNF0 predicts bins with a width of 12.5 cents. Assuming a uniform distribution of noiseless, continuous ground truth pitch values over the range of a given pitch bin, the expected value of the error of the quantization of CREPE is five cents. To see this, note that the minimum error is zero and the maximum error is one half-bin. The expected value of a uniform distribution between real values a and b, where $b > a$, is $(b - a)/2$ (i.e., one quarter-bin). I reduce the bin width to five cents (i.e., $|F| = 1440$ pitch bins), which reduces the

expected quantization error to 1.25 cents. For comparison, human musicians can typically determine whether one sine tone is higher or lower than another when the tones are at least 3 cents apart, whereas distinguishing between higher or lower speech signals requires at least 10-20 cents [**37, 102**]. These thresholds on human perceptual discrimination are called *just noticeable differences* (JNDs). Note that my motivation for improving pitch estimation accuracy is not solely to achieve a pitch accuracy within a certain JND; improvements in pitch accuracy remove noise from the representation, which increases the quality of speech synthesis and editing (Section 4.8; Table 4.5).

Train on frames without a pitch | Prior methods were trained using only frames in which the ground truth annotations indicate that a pitch is present. However, a proper pitch estimator must behave sensibly for audio without periodic structure (i.e., providing a low confidence score to be used for accurate periodicity estimation (Section 2.3)). I include frames without pitch labels during training, setting their ground truth pitch bin to a random bin. This promotes a high-entropy uniform distribution (i.e., low confidence) in aperiodic regions, which can be seen when comparing the inferred pitch distributions of adjacent audio frames using FCNF0 and my proposed FCNF0++ (Figure 2.5).

Do not stop early | CREPE and FCNF0 are undertrained as a result of early stopping (Section 2.2.1). These baseline models all stop when the validation accuracy has not improved for 16,000 steps, where evaluation occurs every 500 steps. This leads to significantly fewer training steps than training to convergence. Running training for an indefinite period, I find that the model converges closer to (or slightly before) 250k steps. I observe an insignificant amount of overfitting at or after convergence.

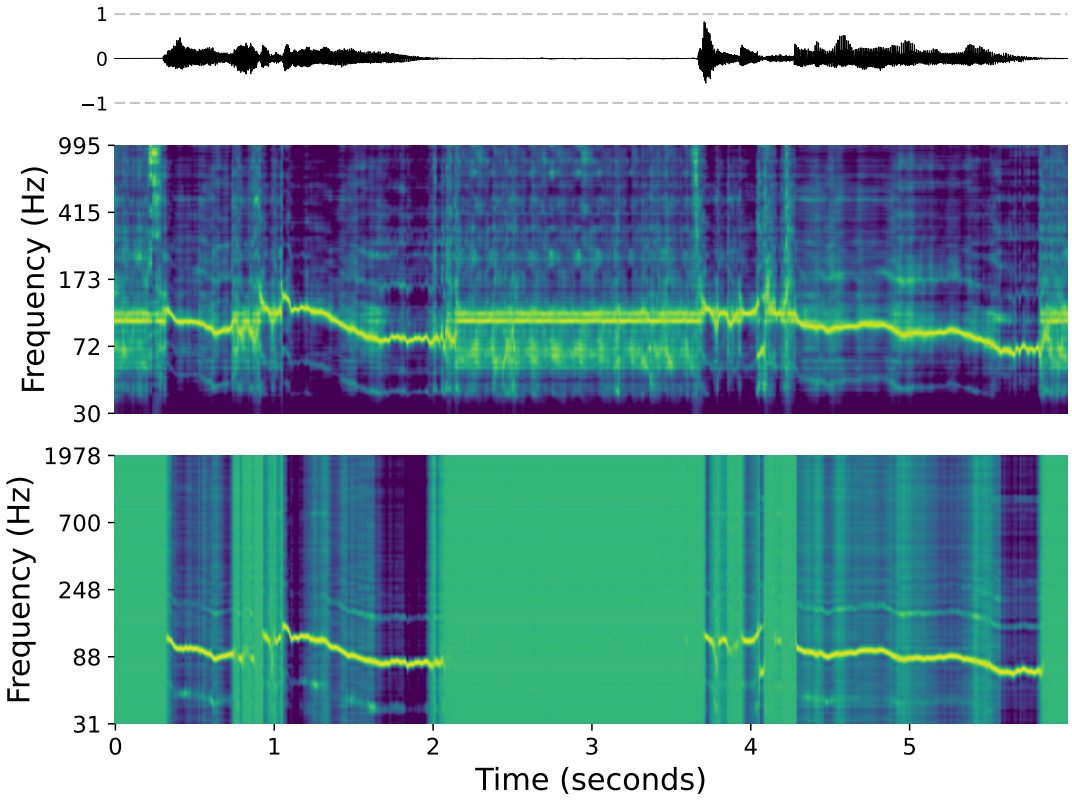

Figure 2.5. Pitch posteriorgrams $D \in \mathbb{R}^{|F| \times T}$ produced by my reimplementation of the baseline FCNF0 pitch tracker [**2**] (**middle**) and my proposed FCNF0++ (**bottom**), where $D_{f,t} = p(y_t = f|x_t)$ is the time-varying categorical distribution produced by performing inference on adjacent input audio frames $x_1, \ldots, x_T$. The input audio (**top**) is the same as in Figure 1.1: the speech utterance "I am sitting in a room, different from the one you are in now". To produce these visualizations, I apply softmax to the $|F|$-dimensional network logits inferred for each time frame to produce normalized distributions and take the natural log. Greater brightness indicates higher probability. The y-axis frequency ranges are representative of the pitch bin ranges of the baseline and proposed models. My proposed methods produce a sharper peak during pitched frames and encourage uniform probability in unpitched regions, making it easy to identify these regions algorithmically (Section 2.3).

`figure-1-1-sitting.wav`

Do not normalize the input | CREPE and FCNF0 normalize the input audio to have a mean of zero and a standard deviation of one. While this improves performance early in training, removing early stopping demonstrates that normalization slightly harms pitch and periodicity accuracy and marginally increases the amount of compute required. I omit this normalization.

Replace binary cross entropy loss | Binary cross entropy independently predicts Bernoulli distributions for each pitch bin at each time step. Categorical cross entropy (CCE) predicts a single categorical distribution over all pitch bins at each time step. CCE improves performance metrics when early stopping is omitted. It is also a more sensible choice when used with my entropy-based periodicity measure, which assumes the input is a categorical distribution. I apply the Gaussian blur used in CREPE to all models, with a standard deviation of 25 cents applied to the training targets, which improves performance for both BCE and CCE losses.

Increase the batch size | Prior methods use normalization methods such as batch and weight normalization. Batch normalization induces instabilities when the batch size is small [**5**]. Increasing the batch size can help to remove these instabilities. I verify this by increasing the batch size from 32 to 128, where a batch size of 128 is the largest multiple of two within the memory capacity of my GPU for all models.

Use layer normalization | Increasing the batch size may help, but batch normalization is still non-robust to outliers. I use layer normalization [**5**] to solve this issue and improve performance relative to batch or weight normalization. In other words, rather than normalizing the network weights or normalizing over the batch and length dimensions, I normalize over the channel and length dimensions.

2.2.4. Viterbi decoding for neural pitch estimation

Many prior works set pitch to zero [**69**] or linearly interpolate pitch [**72**] in unvoiced regions, in which the periodicity is below a threshold (Section 2.3). However, no such threshold exists such that all voiced frames are above the threshold and all unvoiced frames are below the threshold for any existing periodicity estimator (Figure 2.6). Further, a binary voiced/unvoiced threshold cannot represent partially voiced frames (e.g., during a transition between voiced and unvoiced phonemes). As with some prior pitch estimators [**35, 63**], I utilize Viterbi decoding to decode pitch in Hz from frequency-specific confidence scores. Let $D \in \mathbb{R}^{|F| \times T}$ be the pitch posteriorgram formed by concatenating posterior distributions $p(y_t = f|x_t)$ inferred by pitch estimator FCNF0++ for adjacent frames of speech $x_1, \ldots, x_T$ on pitch bins with centers $f \in F$. Viterbi decoding [**109**] finds the optimal sequence of pitch bins $f_1^*, \ldots, f_T^*$ within D given initial and transition probabilities. My initial probabilities are uniform and my transition probabilities are triangular distributions that assign maximal probability to staying on the same pitch and zero probability to pitch jumps greater than one octave between adjacent time frames.

The primary issue with utilizing Viterbi decoding for this purpose is its computational complexity: given $|F|$ pitch bins and T time frames, Viterbi decoding has time complexity $O(|F|^2 T)$. Despite the ubiquity and generality of the Viterbi decoding algorithm, I could not find an open-source Viterbi decoder fast enough to scale to large datasets. I develop and open-source[5] a Viterbi decoder that decodes time-varying categorical distribution D on the VCTK dataset [**119**] 1.62x faster than a widely-used reference [**66**] on a 16-core CPU. Using a batch size of 1, my GPU implementation on an A40 GPU is 1,760x faster

[5] `github.com/maxrmorrison/torbi`

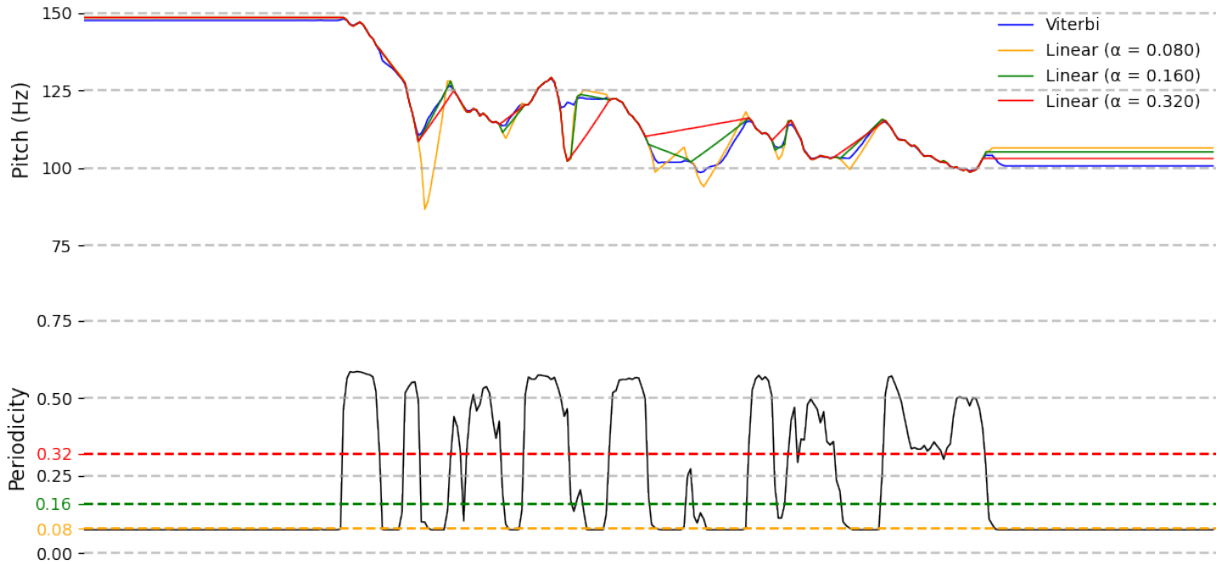

Figure 2.6. **Decoding methods for neural pitch estimation |** I compare Viterbi decoding with linear interpolation of unvoiced regions. (**top**) Pitch contours produced via each decoding method. (**bottom**) entropy-based periodicity contour (**black**) (Section 2.3) and three voiced/unvoiced periodicity thresholds. No voiced/unvoiced threshold α exists that sufficiently separates voiced and unvoiced frames to remove spurious, large pitch jumps in unvoiced regions. Viterbi decoding mitigates this issue and produces relatively smooth pitch contours within unvoiced regions while maintaining high accuracy in voiced regions.
`figure-2-6-0016-000174.wav`

than my 16-core CPU implementation. Using a batch size of 512, my GPU implementation is 309,000x faster than a 16-core CPU (501,000x faster than reference). I have begun the requisite process to have my Viterbi decoding implementation included in PyTorch [**87**].

2.2.5. Objective evaluation of pitch estimation

I design my evaluation to test the extent to which my proposed methods improve the pitch accuracy of the widely-used neural pitch estimators I build upon. I further show

that my models approach the CPU inference speed of state-of-the-art DSP-based pitch estimators.

Evaluating the impacts of improving the unvoiced regions of pitch estimation requires non-standard evaluation methods, as standard pitch estimation datasets do not contain labels for unvoiced regions. Therefore, I evaluate the impacts of Viterbi-based pitch decoding (Section 2.2.4) using downstream speech synthesis and editing tasks (Section 4.8).

2.2.5.1. Data for training and evaluating pitch estimation. Obtaining accurate ground truth pitch annotations for natural signals necessitates complex methods that can induce noise and require humans to make manual corrections. Nonetheless, existing datasets have proven sufficient for training and evaluating machine-learning-based pitch estimators that outperform the alternative DSP-based methods. I use one music and one speech dataset, each with ground-truth pitch and voicing annotations. For each dataset, I perform a random 70-15-15 data split of files into training, validation, and test partitions. For music, I use MDB-stem-synth [**94**], a music dataset used in multiple recent pitch estimation papers. MDB-stem-synth consists of 230 solo stems from MedleyDB [**9**] resynthesized from ground truth pitch for a total of 15.6 hours of music data. PTDB [**88**] is the dataset most commonly used in recent work on pitch estimation for speech. For this reason, I use PTDB as a representation of performance on speech data. PTDB consists of 4,718 English speech and corresponding laryngograph recordings (contact microphones placed on the neck) from 20 speakers (10 male and 10 female) using 2342 senteces from the TIMIT corpus for a total of 9.6 hours of speech data. Ground truth pitch contours for PTDB are extracted using RAPT [**105**] and then manually corrected.

2.2.5.2. Evaluating pitch accuracy. I measure the pitch error of voiced frames using the average error in cents. Let y and $\hat{y}$ be pitch values in Hz.

$$\Delta\text{¢}(y, \hat{y}) = \left|1200 \log_2(y/\hat{y})\right| \tag{2.5}$$

2.2.5.3. Evaluating inference speed. Fast pitch estimation is imperative for processing large datasets as well as for real-time applications. I evaluate the inference speed of each pitch estimator using both CPU and GPU compute, reporting the real-time factor (RTF), or the average number of seconds it takes to perform pitch and periodicity estimation on one second of audio. I use a hopsize of ten milliseconds. I use one Intel i9-9820X 3.30 GHz 10-core CPU for CPU inference and one NVIDIA GeForce RTX 3090 for GPU inference. I use a batch size of up to 2048 frames during GPU inference, requiring multiple forward passes for audio files greater than 2048 frames (20.48 seconds). I do not limit the batch size for CPU inference. I do not provide GPU speeds for my baseline DSP-based pitch estimators that do not provide GPU implementations. All reported speeds include loading of audio, copying between devices, and saving results.

2.2.5.4. Experiments. I apply my proposed methods (Section 2.2) to FCNF0, producing FCNF0++. I perform training and inference on both MDB-stem-synth and PTDB. I compare to my baseline models as well as three common open-source pitch estimators: PYIN [**63**], an accurate DSP-based method; DIO [**69**], a fast DSP-based method written primarily in C++; and my open-source `torchcrepe` [**70**] implementation of CREPE that directly transfers the original CREPE weights trained on six music datasets from the

original Keras model to PyTorch [**87**] and does not permit retraining.[6] I perform estimation using PYIN and DIO using a 16 kHz sampling rate, which I find outperforms an 8 kHz sampling rate for these estimators. I also perform ablations of each of my proposed methods on my FCNF0++ model to assess their relative merit.

Human speech has a pitch range of approximately 50-550 Hz. I could impose this as an inductive bias in the evaluation of PTDB by constraining the output to the pitch range of human speech. I omit this inductive bias during pitch evaluation in order to evaluate pitch estimation in the more challenging setting of an unrestricted frequency range. During pitch estimation for speech synthesis (Chapter 3) and editing (Chapter 4), I do apply this inductive bias by setting pitch posteriorgram probabilities to zero for pitch values outside the human speaking range before Viterbi decoding (Section 2.2.4).

2.2.6. Pitch estimation results

Results for pitch estimation accuracy and speed for my baseline models, my proposed models, and additional baseline neural- and DSP-based pitch estimators (Table 2.2) indicate my proposed methods improve the pitch estimation accuracy. My reimplementation of CREPE from scratch (Section 2.2.1) outperforms `torchcrepe` in speed due to automatic mixed precision (AMP) [**67**] inference and in accuracy due to being jointly trained

[6]Since its release in 2020, my open-source `torchcrepe` package has been downloaded via `pip` over 3.7 million times and included in `so-vits-svc`, a state-of-the-art open-source singing voice conversion (SVC) system that has been used in numerous viral social media posts featuring AI voice clones of celebrities, presidents, and popular characters exchanging dialogue or singing. `so-vits-svc` was listed as one of the "best inventions of 2023" by TIME magazine [**106**]. FCNF0++ was released in 2023 as part of Pitch Estimating Neural Networks (`penn`) [**72**] and has exceeded 100,000 downloads within its first year. This significantly outpaces the download rate of the first year of `torchcrepe`.

Model	Pitch Δ¢ ↓	Time RTF (GPU)↓	Time RTF (CPU)↓
Proposed (FCNF0++)	**12.72**	.0024	.0861
FCNF0 [**2**] (Section 2.2.2)	18.01	**.0019**	.0753
CREPE [**35**] (Section 2.2.1)	21.07	.0093	.3574
`torchcrepe` [**70**]	59.40	.0199	.6435
PYIN [**63**]	110.5	-	.0639
DIO [**69**]	80.10	-	**.0177**

Table 2.2. **Objective evaluation of pitch estimation |** Pitch error and speed of my baselines, my proposed model, and common open-source models on both PTDB and MDB-stem-synth datasets. Pitch error in cents (Δ¢) and real-time factor (RTF) metrics are described in Sections 2.2.5- 2.2.5.3. I consider FCNF0++ to be most useful model for most downstream applications, with competitive accuracy and speed. ↑ indicates that higher is better and ↓ indicates that lower is better.

on speech data (i.e., cross-domain). FCNF0++ exhibits highly competitive pitch estimation, CPU inference speeds approaching DSP-based methods, and GPU inference speeds that can process large audio datasets in minutes.

Table 2.3 shows the pitch and periodicity accuracy of ablations of my proposed improvements relative to my FCNF0++ model (see Section 2.2). All of my proposed methods improve pitch or periodicity accuracy. The pitch accuracy of using only voiced frames for training is comparable; however, my proposed method for training on unpitched frames by selecting a random ground truth pitch bin substantially improves periodicity accuracy, where periodicity is described in detail in Section 2.3. According to my ablation study (Table 2.3), my most notable improvements are my smaller quantization width, unvoiced training strategy, and larger batch size.

Model	Pitch Δ¢ ↓	Periodicity F1 (Entropy)↑
FCNF0++	12.72	**.9816**
Coarse quantization	17.23	.9814
Voiced only	**12.62**	.8496
Early stopping	13.32	.9804
Input normalization	13.14	.9811
Binary cross-entropy loss	13.64	.9815
Smaller batch size	13.63	.9800
Batch normalization	17.44	.9734

Table 2.3. Ablations of my proposed methods described in Section 2.2. For example, "Early stopping" is FCNF0++ trained with early stopping and "Voiced only" is FCNF0++ trained only on voiced frames. Note that rows are not cumulative: each row independently evaluates removing exactly one of my suggested improvements (see Section 2.2) relative to my proposed FCNF0++ model. All models are trained and evaluated on both PTDB and MDB-stem-synth datasets. Pitch error in cents (Δ¢) and voiced/unvoiced F1 metrics are described in Sections 2.3.2- 2.2.5. ↑ indicates that higher is better and ↓ indicates that lower is better.

2.3. Entropy-based periodicity estimation

Not all speech sounds contain a pitch. *Unvoiced* phonemes (e.g., /s/, /k/, /f/) do not produce periodic opening and closing of the glottis indicative of pitched (i.e., *voiced*) speech sounds. Instead, the resonator (i.e., the shape of the nasal and vocal tract; Section 1.1.1) perform a spectral filtering of wide-band aperiodic noise produced via air pressure from the lungs. The addition of aperiodic noise lowers the *periodicity* of the signal (i.e., the extent to which a segment of audio contains repetition at a regular interval). Thus, periodicity estimators are typically derived from some measure of confidence assigned by a statistical pitch estimator that a given frame of audio contains a pitch. Periodicity contours are widely used for speech synthesis [**107**], where they are used for

making binary decisions about whether speech is voiced (low periodicity indicates an unvoiced region) or which portion of the pitch contour calculated on input speech is likely to be meaningful (e.g., omitting unpitched regions when evaluating pitch distance).

My prior work in periodicity estimation [**75**] has made periodicity error a standard method in the literature for evaluating speech reconstruction accuracy [**52, 53, 97, 100**]. This prior work uses as periodicity the pitch bin probabilities along the Viterbi-decoded path of pitch posteriorgram D inferred using `torchcrepe` [**70**]. In this chapter, I describe my more general entropy-based periodicity estimator that utilizes my pre-trained FCNF0++ pitch estimator (Section 2.2) to perform state-of-the-art framewise voiced/unvoiced classification (Section 2.3.2). Relative to prior methods, my entropy-based periodicity estimator not only produces state-of-the-art framewise voiced/unvoiced classification, but is decoder-free and elegantly handles polyphony.

Consider a periodicity value $h \in [0, 1]$ for input audio frame x and $p(y = f|x)$, the categorical posterior distribution of a neural pitch estimator (Eq. 2.3). A periodicity near zero indicates aperiodic noise and a periodicity near one indicates a noise-free, pitched signal. I consider two methods for periodicity estimation.

Method one (*max*) is a simple, domain-agnostic baseline approach that takes the maximum posterior probability over the pitch bins in each time frame.

$$\hat{h}^{(\max)} = \max_{f \in F} p(y = f|x) \tag{2.6}$$

Method two (*entropy*) is my novel method that derives periodicity from the entropy of the categorical posterior distribution. The entropy is scaled to the range $[0, 1]$ by dividing by $\ln |F|$—the maximum entropy of a categorical distribution with $|F|$ categories—and

subtracted from 1 (i.e., low entropy indicates high periodicity).

$$\hat{h}^{(\text{entropy})} = 1 - \frac{1}{\ln |F|} \sum_{f \in F} p(y = f|x) \ln p(y = f|x) \tag{2.7}$$

Unlike the direct prediction method of Gfeller et al. [**23**], my entropy-based periodicity method requires no additional loss functions or modifications to the neural network architecture. This reduces complexity and improves training and inference speed. Unlike baseline method one (*max*), my entropy-based method can elegantly handle polyphony. Consider a peak-normalized sine wave with a frequency within the range of the pitch estimator. This signal should produce a periodicity of one. Now, add another sine wave at a different frequency also within the range of the estimator, so that the posterior distribution produced by the model has two peaks. The resulting signal should still have a periodicity of one. Method one (*max*) produces a periodicity of 0.5, while my proposed entropy-based method produces a periodicity of $1 - \frac{1}{\ln |F|}(2 \times 0.5 \times \ln 0.5)$. For my proposed $|F| = 1440$, this produces a periodicity of 0.905. As $|F|$ approaches infinity, the periodicity approaches one, as desired. Note that my prior method of using the pitch bin probabilities along the Viterbi-decoded path of pitch posteriorgram D produces the same issue as method one (*max*).

2.3.1. Binary voicing decisions

Once I have produced a periodicity estimate, I can perform thresholding to produce per-frame classifications of whether a pitch is present. This is referred to as the voiced/unvoiced decision in the context of speech. Specifically, I aim to make a sequence of binary voiced/unvoiced decisions $\hat{v}_1, \ldots, \hat{v}_T$ from inferred periodicity contour $\hat{h}_1, \ldots, \hat{h}_T$. I use

Model	Entropy (Eq. 2.7) ↑	Max (Eq. 2.6) ↑
Proposed (FCNF0++)	**.9816**	**.9813**
FCNF0 [**2**] (Section 2.2.2)	.9602	.9361
CREPE [**35**] (Section 2.2.1)	.9626	.9509
`torchcrepe` [**70**]	.9293	.9305
PYIN [**63**]	.9199†	

Table 2.4. **Objective evaluation of periodicity |** Voiced/unvoiced F1 score of periodicity estimation using my baselines, my proposed model, and common open-source models on both PTDB and MDB-stem-synth datasets. The voiced/unvoiced F1 metric is described in Section 2.3.2. Baseline models `torchcrepe` and PYIN are described in Section 2.2.5.4. ↑ indicates that higher is better and ↓ indicates that lower is better.
†PYIN uses the sum of peak-picked densities for periodicity decoding.

the following decision rule, where $\alpha \in [0, 1]$ is a voicing threshold hyperparameter.

$$\hat{v}_t = \hat{h}_t > \alpha;\ t = 1, 2, \ldots, T \tag{2.8}$$

2.3.2. Objective evaluation of periodicity estimation

I evaluate periodicity via classification F1 score of the binary voiced/unvoiced decision (Section 2.3.1). I extract voicing decisions from estimated periodicity using both the *max* and *entropy* periodicity methods (Section 2.3). I use the same data 2.2.5.1 and baseline methods 2.2.5.4 as used for evaluation of pitch estimation and report the F1 score of the binary voicing classification for each pitch estimation method that I evaluate.

For each voicing threshold hyperparameter search, I use the validation data partitions of both datasets to perform two grid searches using values $\alpha = 0.0, 0.1, \ldots, 0.9$ and $\alpha = 2^{-i};\ i = 1, 2, \ldots, 9$ to produce candidate threshold value α^*. I then assume the hyperparameter landscape is convex at α^* and perform a sequential grid search, reducing the step size from 0.05 by a factor of two at each step for eight steps. This can be thought

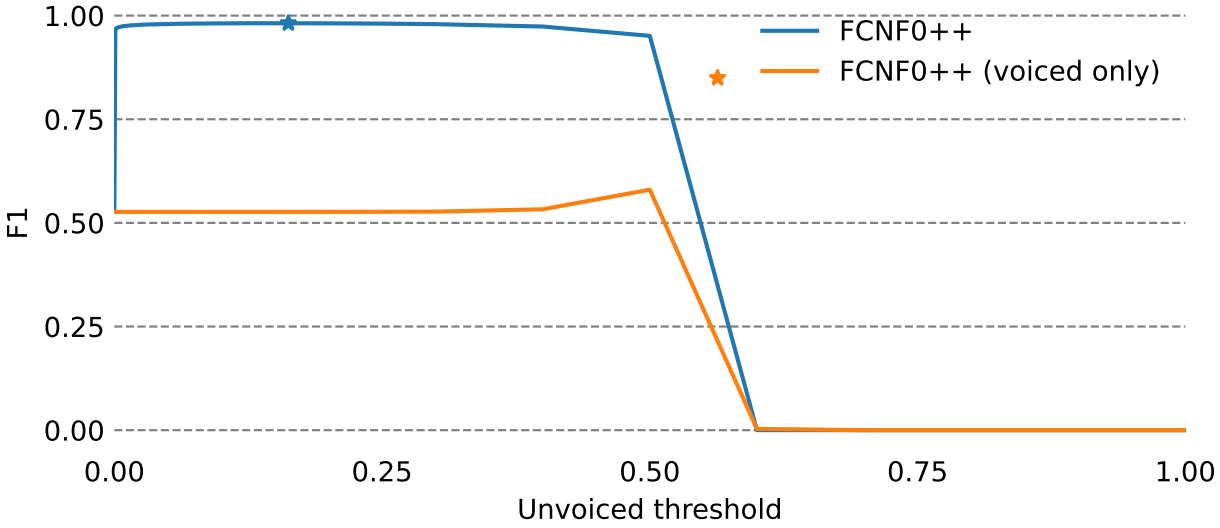

Figure 2.7. Hyperparameter landscape of the voiced/unvoiced threshold on the entropy-based periodicity estimate produced by FCNF0++ with (blue) and without (orange) my proposed unvoiced training strategy (Section 2.2.3) on PTDB and MDB-stem-synth. Stars indicate optimal F1 values found via a fine-grained binary search (Section 2.3.2). My unvoiced training strategy of selecting a random bin (Section 2.2.3) improves the optimal F1 score of the model and produces state-of-the-art voiced/unvoiced classification F1 scores across a large region of the hyperparameter space.

of as fine-tuning α^* via gradient ascent on the F1 score with an exponentially decreasing step size, where first-differences are used to measure the gradient direction. This fine-grained hyperparameter search is necessary for all models without my proposed unvoiced training strategy. Without this training strategy, models exhibit a narrow peak for the optimal hyperparameter value (Figure 2.7).

As seen in Table 2.4, training on unvoiced frames significantly improves periodicity estimation accuracy. The significance of this effect is visible in Figure 2.7. Without my unvoiced training strategy, the optimal F1 score decreases and the landscape of unvoiced thresholds with a competitive voiced/unvoiced classification F1 score narrows.

2.4. Multi-band A-weighted loudness

Human perception of sound occurs due to variations in sound pressure at the surface of the tympanic membrane within the ear. Larger variations in sound pressure cause larger displacements of the membrane. Within the audible frequency range (approximately 20 - 20,000 Hz), larger displacements are perceived as being *louder*. However, equal sound pressures are not perceived as equally loud at all frequencies. This phenomenon is captured via *equal-loudness contours* that indicate the sound pressure levels at which simple sinusoidal tones at varying frequencies are perceived as equally loud.

Audio recording devices convert physical sound pressure variations traveling through the air into variations in electrical current, which are encoded by an analog-to-digital converter into a format amenable to storage, modification, and transmission by a computer (e.g., 16-bit floating point values between -1 and 1, inclusive, at a sampling rate of 44,100 Hz). However, human loudness perception occurs at a more coarse sampling rate, and this format does not encode equal-loudness contours, necessitating a transformation from the sample resolution to a lower sample resolution (e.g., one floating-point value every ten milliseconds). Specifically, I use the A-weighted loudness (Figure 2.8; bottom) [**65**]: a frequency-average of a weighted magnitude spectrogram, with per-channel weights derived from human perceptual studies of loudness variation. Let $\Omega = \{\omega_1, \ldots, \omega_{\lfloor N/2+1 \rfloor}\}$ be the sorted, real-valued, positive FFT frequencies of an N-dimensional complex FFT: $\omega_c = \frac{s}{N}(c-1);\ c = 1, \ldots, \lfloor N/2+1 \rfloor$. Given a frame of audio x_t with sampling rate s, its A-weighted loudness $a_t = L_{\mathcal{A}}(x_t; \Omega)$ is as follows.

$$L_{\mathcal{A}}(x_t; \Omega) = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \max\left(B_L, \log_{10}\left|\mathrm{FFT}(x_t, \omega)\right| + \mathcal{A}(\omega) - B_H\right) \tag{2.9}$$

$\mathcal{A}$ is the A-weighted transformation of frequency ω based on human perceptual data, and B_H and B_L are reference A-weighted decibel (dBA) values. I use $B_H = 20$ and $B_L = -100$.

Reconstructing speech from my representation using A-weighted loudness produces worse loudness reconstruction relative to Mel spectrograms (Table 3.1; w/o multi-band), but Mel spectrogram vocoding entangles pitch and pronunciation. To address this trade-off, I propose using *multi-band A-weighted loudness* (Figure 2.8). I group frequencies into K bands $\Omega_1, \ldots, \Omega_K = \{\omega_1, \ldots, \omega_{\lfloor |\Omega|/K \rfloor}\}, \ldots, \{\omega_{(K-1)\lfloor |\Omega|/K \rfloor + 1}, \ldots, \omega_{|\Omega|}\}$ and compute the average A-weighted energy within the band $a_{k,t} = L_{\mathcal{A}}(x_t; \Omega_k);\ k = 1, \ldots, K$. A hyperparameter search over 2, 4, 8, 16, and 32 bands indicates optimal disentanglement and loudness reconstruction at 8 bands.

One interesting side effect of this representation is that speech content creators can edit loudness using a different number of bands than is used during synthesis. For example, volume modifications performed in speech content creation often specify changes (in decibels) that are applied uniformly across all bands, with magnitudes and directions of change that vary between words or phrases. This familiar interface can be achieved by having the user edit the single-band A-weighted loudness (e.g., Figure 1.2) and corresponds to a proportional scaling of each band of the underlying multi-band loudness. For efficiency, the single-band loudness can be computed directly as a channel-wise average of the multi-band loudness. As shown in Section 4.3, editing the single-band A-weighted loudness within my proposed speech editing system can produce independent control over volume and the timbral correlates of volume.

2.5. Prior work in speech representation

In this chapter, I described my proposed disentangled and interpretable speech representation that satisfies the four properties (outlined at the start of this chapter) as prerequisite for speech editing representations: *invertibility* (Properties 1 & 2), *interpretability* (Property 3), and *disentanglement* (Property 4). Next, I describe representations of speech that precede my proposed representation and the relative advantages of my representation.

2.5.1. Time-frequency representations

Time-varying representations of frequencies such as a Mel spectrogram (Section 1.1.2; Figure 1.1) are widely-used and interpretable speech representations that have been used in the context of neural networks as loss functions [**40**], output representations [**96**], and input representations [**60, 81**]. Vocoding using Mel spectrograms produces high-fidelity speech reconstruction (Section 3.4), and is commonly used as a high-anchor for text-to-speech research. However, time-frequency representations are not disentangled: there is no simple way to edit them to independently change, e.g., the pronunciation or pitch. My prior work [**74**] as well as Wang et al. [**115**] propose to disentangle speech attributes from the Mel spectrogram by tuning the number of bottleneck channels to remove information. My subsequent work omits the bottleneck by replacing the Mel spectrogram with an interpretable pronunciation representation that enables fine-grained pronunciation editing (Section 2.1) as well as an interpretable multi-band loudness representation (Section 2.4) inspired by this general intuition of constraining spectral features to be low-dimensional. I find applying the low-dimensional constraint directly to the spectral features to be the

more sensible choice, as the bottleneck encoder is trained to produce optimal speech reconstruction and is significantly more capable of using spurious correlations between imperceptible, low-energy time-frequency bins to minimize training losses—at the expense of entanglement.

Morise et al. [**69**] propose analysis-modification-synthesis of speech using (1) a time-frequency representation, (2) aperiodicity, and (3) pitch. Time-Domain Pitch-Synchronous Overlap-and-Add (TD-PSOLA) [**80**] is a non-parametric method that first segments the audio at the start of each repetition in the waveform (or at equal intervals in unvoiced regions) and uses overlap-add to combine modified (e.g., repeated) audio frames. TD-PSOLA may be the most widely-used method for pitch-shifting and time-stretching of speech, given its inclusion in the popular PRAAT software for speech analysis and manipulation [**10**]. Both WORLD and TD-PSOLA are digital signal processing (DSP) methods that modify formants (Section 4.4) in typically undesirable ways when performing pitch-shifting. My proposed speech editing system not only avoids these unnatural formants by default, but allows speech content creators to apply a high-fidelity version of this formant modification effect for creative purposes (Section 4.4). In Chapter 4, I utilize both WORLD and TD-PSOLA as baseline methods for pitch-shifting 4.1 and time-stretching 4.2.

2.5.2. Lexical representations

Lexical representations such as graphemes (characters) and phonemes (discrete units of speech sound) are inputs for text-to-speech (TTS) systems [**4**]. Ren et al. [**91**] demonstrate TTS with phoneme duration control; Łańcucki [**48**] demonstrates pitch control.

However, when used for analysis-modification-synthesis, graphemes and phonemes induce coarse discretization, causing ambiguous pronunciation. Further, TTS phoneme inputs are often produced from text, ignoring the input speech pronunciation and requiring a transcript. This makes speech audio non-invertible (or only partially invertible) relative to its corresponding lexical representation.

2.5.3. Latent representations

Latent speech representations are non-interpretable and typically not disentangled. These representations exhibit strong performance in speech reconstruction [**45**] and generation [**111**]. However, precise control of, e.g., pitch is difficult, as pitch is entangled within a non-interpretable representation. Notable exceptions are representations with partial speaker disentanglement [**41**], pitch-agnostic latents of automatic speech recognition (ASR) models [**14, 43**], and discrete factorizations [**33**]. However, lack of interpretable pronunciation representation prohibits fine-grained pronunciation control.

2.5.4. Source-filter representations

Neural source filter (NSF) methods [**73, 84, 114, 120**] represent speech as a periodic source excitation and a time-varying finite impulse-response (FIR) filter. NSF methods demonstrate fast, accurate, and high-fidelity pitch-shifting. However, the compute time required to perform analysis-modification-synthesis with existing NSF models is constrained by slow pitch estimators required for sufficient pitch accuracy. As well, no NSF model has demonstrated pronunciation or spectral balance editing.

No prior speech representation of any type has demonstrated fine-grained, disentangled control of volume from the timbral correlates of volume (Section 4.3).

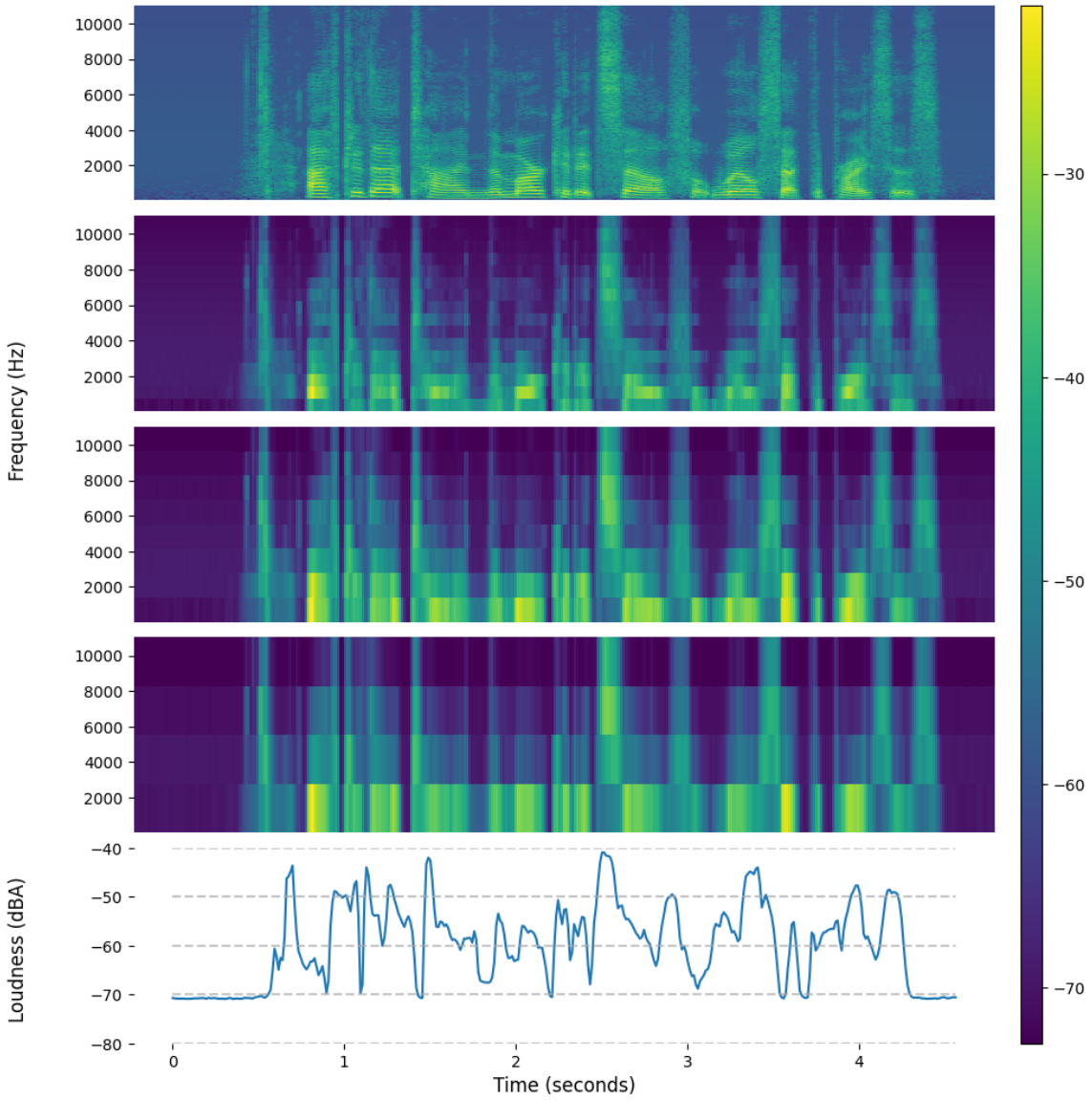

Figure 2.8. **Multi-band A-weighted loudness |** My proposed multi-band A-weighted loudness for interpolating the trade-off between disentanglement and loudness reconstruction. (**top**) A-weighted magnitude spectrogram (equivalently, a 513-band A-weighted loudness). (**bottom**) Single-band A-weighted loudness. (**middle**) From top to bottom, the 16-band, 8-band (optimal), and 4-band A-weighted loudness.
`figure-2-8-0097-000680.wav`

CHAPTER 3

High-fidelity interpretable speech reconstruction

My proposed process for speech editing consists of (1) encoding the speech in an appropriate representation (Chapter 2), (2) editing the speech representation according to one's creative and technical vision, and (3) synthesizing a new speech waveform from the edited speech representation. This chapter addresses (3) as well as methodologies for the evaluation of speech synthesis, in which no edits are performed during (2). In the following chapter (Chapter 4), I showcase the speech editing capabilities of my proposed system and (when applicable) demonstrate the efficacy of my system relative to baselines.

Speech synthesis systems typically take as input either unaligned lexical features derived from text (e.g., an ordered sequence of phonemes without timing information) or aligned acoustic features (e.g., Mel spectrograms (Section 1.1.2) or my proposed, interpretable representation (Chapter 2)). Systems that synthesize speech from lexical features are *text-to-speech (TTS)* systems, whereas systems that synthesize speech from acoustic features are *vocoders*. TTS systems are usually slower, requiring more model parameters and more expressive generative models to compensate for lexical features having higher conditional entropy with the speech signal relative to acoustic features (e.g., the prosody must be generated). However, this depends on the quality of the acoustic features: acoustic features that are not sufficiently informative necessitate using higher-capacity TTS models as opposed to vocoders. I found this to be true in my previous work [**15**]: I trained a HiFi-GAN [**40**] speech vocoder (a standard, commonly used neural vocoder)

to synthesize speech from (non-sparsified) PPGs (Section 2.1) and FCNF0++ pitch contours using linear interpolation instead of Viterbi decoding (Section 2.2; Figure 2.6). This vocoder training frequently diverges—likely due to gradient confusion [**95**], as gradient magnitudes approach zero during divergence—but succeeds when using a VITS text-to-speech model [**34**] (a standard, commonly-used TTS system) that I minimally modify by removing the duration prediction module in order to permit time-aligned, frame-resolution input features. VITS consists of a duration predictor, flow-based distribution matching with the linear spectrogram, and a jointly-trained HiFi-GAN vocoder: VITS without duration prediction is equivalent to HiFi-GAN jointly trained with flow-based distribution matching with the linear spectrogram. This small change appears vitally important when the representation contains less information (or more noise) [**57**]. In other words, **speech synthesis necessitates a minimum mutual information (or, equivalently, a maximum conditional entropy) below which training diverges.** This can be compensated for by increasing the expressivity of the generative model (e.g., using the flow-based distribution matching) in order to generate any missing information. However, increasing the expressivity of the generative model also induces stochasticity, which causes the generated speech to sound different than the original speech. This poses the question: does my proposed representation (Chapter 2) have sufficiently low conditional entropy with the synthesized speech to enable speech synthesis using efficient neural vocoders such as HiFi-GAN?

In this chapter, I demonstrate that my proposed representation (Chapter 2) can be used to perform high-fidelity speech vocoding using the standard

HiFi-GAN vocoder [40] with objective and subjective evaluation showing comparable performance to Mel spectrograms (Section 3.4). After establishing that my representation is capable of high-fidelity reconstruction, I demonstrate various existing speech edits using my proposed representation and compare objective and subjective speech editing results with edit-specific baselines.

3.1. Neural speech editing model

My high-fidelity, fine-grained neural speech editing model (Figure 1.2) is an off-the-shelf HiFi-GAN vocoder [**40**] that has been trained on speech encoded in my proposed disentangled, interpretable representation (Chapter 2). I further condition my neural speech editing model on three time-invariant features: augmentation ratios r_f and r_l (Section 3.2.1) and a jointly trained speaker embedding (Section 3.2.2). The HiFi-GAN vocoder consists of a generator neural network that performs vocoding and multiple discriminator neural networks that are jointly trained to distinguish between synthesized and real speech. Because there are typically multiple discriminators, they jointly dominate the total training time. As such, the design of the discriminator(s) is a key choice affecting both synthesis quality and speed. One discriminator used in HiFi-GAN is the multi-scale spectrogram discriminator (MSD) [**40**]. I replace the MSD with the more recent complex, multi-band spectrogram discriminator [**45**], which allows the discriminator to further utilize phase information to distinguish between real and fake speech.

I train for 400k steps on one A40 GPU, where each step corresponds to training the model on a batch of 64 speech recordings excerpts. Each excerpt is a randomly-selected, 16,384-dimensional, contiguous speech waveform with a sampling rate of 22,050 (i.e., 0.74

seconds), as well as the corresponding 64 frames of my proposed speech representation. I train my speech editing model to reconstruct the 16,384-dimensional speech excerpt from its corresponding 64 frames of my proposed representation. I use the AdamW optimizer [**61**] with a learning rate of 2×10^{-4}. All other details of neural vocoder architecture and training are unchanged from the original HiFi-GAN implementation.

3.2. Data

I use VCTK [**119**] for training and evaluation. VCTK is a standard dataset for speech synthesis that consists of 82.4 hours of clean speech from 109 speakers. I select five male and five female test speakers. I select ten utterances from each test speaker for a total of 100 test utterances. Utterances used for subjective evaluation (Section 3.3.2) should be long enough to contain salient differences, but short enough to not exhaust the attention span of the participant. Therefore, I require all 100 test utterances be between four and ten seconds in length. I reserve 64 random validation utterances. I omit test or validation data recorded with a different microphone from training. In Section 4.5, I show my proposed speech editing model is also capable of adaptation to unseen speakers from the DAPS [**82**] dataset. I perform adaptation independently on five male and five female speakers, with 10 test utterances between four and ten seconds in length for each adaptation speaker and all remaining data for each speaker used for training.

3.2.1. Augmentation

Prior works use resampling [**74**], pitch-shifting [**6**], or volume modifications [**100**] to augment a dataset speech recordings used for training neural vocoders. However, these

methods cannot be directly used in speaker-conditioned generation without causing the synthesized speech to also inherit any artifacts found in the augmented training data. For example, using resampling as data augmentation produces pitch-shifted and time-stretched audio, but preserves relative formant energies. While the frequency ratios between formants should be preserved (to maintain a harmonic relationship), energy ratios between formants should not be preserved during pitch-shifting; they should adhere to the conditional distribution of formant energies given F0 and speaker. Perceptually, these unnatural formants induce the vocal effect used by Alvin from *Alvin and the Chipmunks*.

I propose a technique that increases the range of the desired audio feature within the training distribution and enables the artifacts induced by data augmentation to be independently controlled. I apply my proposed augmentation technique to the disentanglement of pitch (F0) from spectral balance, as well as the disentanglement of volume from the timbral correlates of volume.

Disentangling pitch and spectral balance | Let $R_f(x; d, e)$ be a function that resamples speech recording x from sampling rate d to sampling rate e. Given original sampling rate s, target sampling rate q, and random pitch shift factor $r_f \sim \text{Uniform}(-1, 1)$, I augment training data with $x_f = R_f(R_f(x; 2^{r_f} s, s); s, q)$. For example, VCTK [**119**] has an original sampling rate of $s = 48,000$, while my system synthesizes speech at $q = 22,050$. I first resample from sampling rate $2^{r_f} s$ to sampling rate s, which performs corresponding pitch-shifting and time-stretching while preserving relative formant energies. As described at the start of this section, preserving relative formant energies during pitch-shifting sounds unnatural. Then, I resample the speech from sampling rate s to sampling rate q for use with my system. Resampling to target sampling rate q afterwards prevents any

loss of frequency bandwidth as long as $r_f < \frac{s}{q}$. I pass r_f to the network during training, so the model learns that low r_f indicates more energy at low-frequencies and high r_f indicates more energy at high-frequencies. I create one randomly pitch-shifted copy of each training utterance. In Section 4.4, I show that this resampling-based data augmentation enables control over the relative amount of energy in low- and high-frequencies.

Disentangling volume and the timbral correlates of volume | Let $R_l(x; g)$ be a function that increases or decreases the volume of speech recording x by g decibels. Given a randomly sampled volume shift $r_l \sim \text{Uniform}(-1, 1)$, I augment training data with $x_l = R_l(x; 12r_l)$. If any value in x_l is outside $[-1, 1]$, I draw a new sample for r_l until x_l is within $[-1, 1]$. I pass r_l to the network during training. I create one randomly volume-shifted copy of each training utterance. During generation, setting $r_l > 0$ increases volume, $r_l < 0$ decreases volume, and $r_l = 0$ maintains the current volume; r_l does not control the timbral correlates of volume. Instead, framewise edits to A-weighted loudness produce audible changes in timbre corresponding to louder or quieter speech while maintaining accurate volume control (Section 4.3). As mentioned in Section 2.4, these framewise edits can be performed on either the multi-band or the corresponding single-band loudness—with the latter typically being more intuitive. While this kind of editing is technically possible without data augmentation, it requires either significantly larger edits to the A-weighted loudness to have a similar perceptual effect—which causes clipping when too loud and out-of-distribution artifacts when too quiet—or performing multiple small edits, which is both inefficient and more likely to induce artifacts. Thus, my proposed data augmentation method can be seen as a perceptual loudness *compressor*, which modifies the scale of volume relative to the timbral correlates of volume so that smaller changes in

volume have a larger perceptual effect. Control over the timbral correlates of volume—with or without my proposed data augmentation—has not been demonstrated by any prior work.

3.2.2. Preprocessing

The ideal (e.g., most intuitive) speech representation for speech content creators performing speech editing tasks may not be the same as the ideal representation for performing high-fidelity speech synthesis. My proposed multi-band A-weighted loudness (Section 2.4) is one example of this, wherein the user typically edits only one band (the average), but multiple bands are passed to the neural network to lower conditional entropy with the speech signal and improve fidelity. Two additional examples of transforming my representation to be more apt for consumption by a neural network occur in my speech editing system: (1) the multi-band loudness is scaled from perceptually comprehensible units (dBA) to the range [-1, 1] prior to speech synthesis to alleviate the need for the network to learn the (relatively large) scale of common dBA values and (2) I encode the pitch contour in an embedding table that permits jointly learning a multi-dimensional representation for each bin in a set of quantized pitch bins. Prior works utilize embedding tables that quantize pitch conditioning into equal-width bins [**73, 107, 115**]. This leads to infrequently used bins at endpoints of the pitch range of the training distribution that cause instability and artifacts. I change the bin spacing so each pitch bin is accessed equally often during training. This produces a variable-width quantization that allocates more bins to frequently used pitch regions (e.g., 100 to 200 Hz). The effect

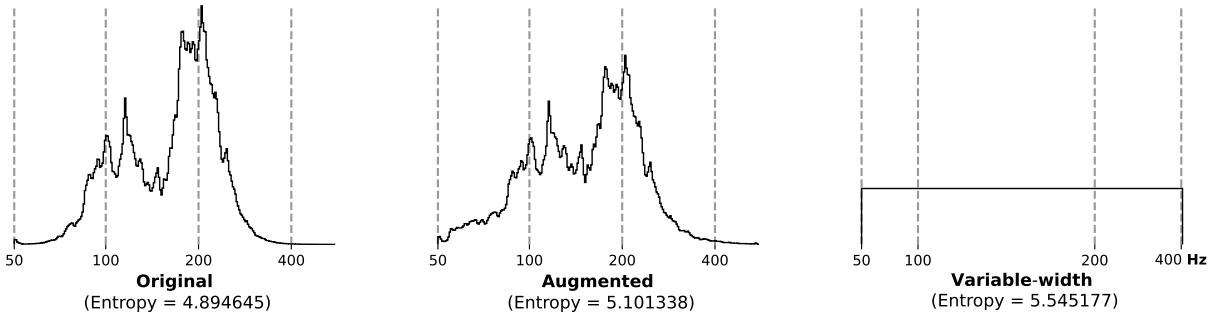

Figure 3.1. **Variable-width pitch quantization | (left)** Training distribution of pitch bins in VCTK before my proposed data augmentation (Section 3.2.1) and variable-width pitch quantization (Section 3.2.2). **(middle)** The same distribution after augmentation. **(right)** The same distribution after augmentation and variable-width quantization. My proposed variable-width pitch quantization transforms the spacing of the 256 pitch bins to produce a uniform training distribution, which corresponds to the maximum-entropy distribution ($\ln 256 = 5.545$).

of this variable-width quantization are visualized in Figure 3.1. I use 256 bins and a 64-dimensional embedding table.

I also employ an embedding table to jointly learn a multi-dimensional representation of each speaker in the training dataset. During speaker adaptation, I assign the new speaker an index of zero during training and inference, which overwrites the corresponding row in the embedding table. This is standard practice for multi-speaker and speaker adaptive speech synthesis systems. More recent systems demonstrate that scaling to tens of thousands of hours of training data can enable few-shot generalization (i.e., only using a few seconds of audio from a new speaker) with high perceptual naturalness and speaker similarity. My current system requires significantly more speech (at least 5-10 minutes) to achieve high perceptual fidelity and speaker similarity. Provided sufficient compute and data resources, it is straightforward to combine the efficacies of these few-shot systems with those of my proposed representation in future work.

3.3. Evaluating speech synthesis and editing

Consider a speech recording synthesized using my proposed system. How can one evaluate this speech to determine the efficacy of my system relative to baseline systems? In general, a system for reconstructing and editing speech should be *accurate*—in that it closely reflects the original audio as well as any edits performed by a speech content creator–and synthesize *perceptually high-quality* speech. Using the invertibility of my speech representation, I evaluate accuracy by encoding the edited speech in my interpretable representation and comparing the reconstructed representation with the representation used as input to produce the edited audio (Section 3.3.1; Figure 3.2). I further design and perform subjective evaluations (Section 3.3.2) that evaluate the perceptual quality of speech synthesis and editing relative to baseline systems.

3.3.1. Objective evaluation of speech synthesis and editing

I design my objective evaluation of speech synthesis and editing to determine the extent to which the synthesized speech accurately reflects speech attributes specified by the input speech representation. In other words, I propose to evaluate the extent to which the synthesized speech exhibits the specified pronunciation (Section 3.3.1.1), pitch (Section 3.3.1.2), periodicity (Section 3.3.1.3), and loudness (Section 3.3.1.4).

Prior works also perform objective evaluation of speaker "similarity" and speech "quality". However, I find that prior objective metrics for speaker similarity are not disentangled: edits such as pitch-shifting and time-stretching cause large dissimilarities in the Resemblyzer [**110**] and WeSpeaker [**112**] speaker representations irrespective of pitch-shifting or time-stretching method. This indicates that standard, widely-used speaker

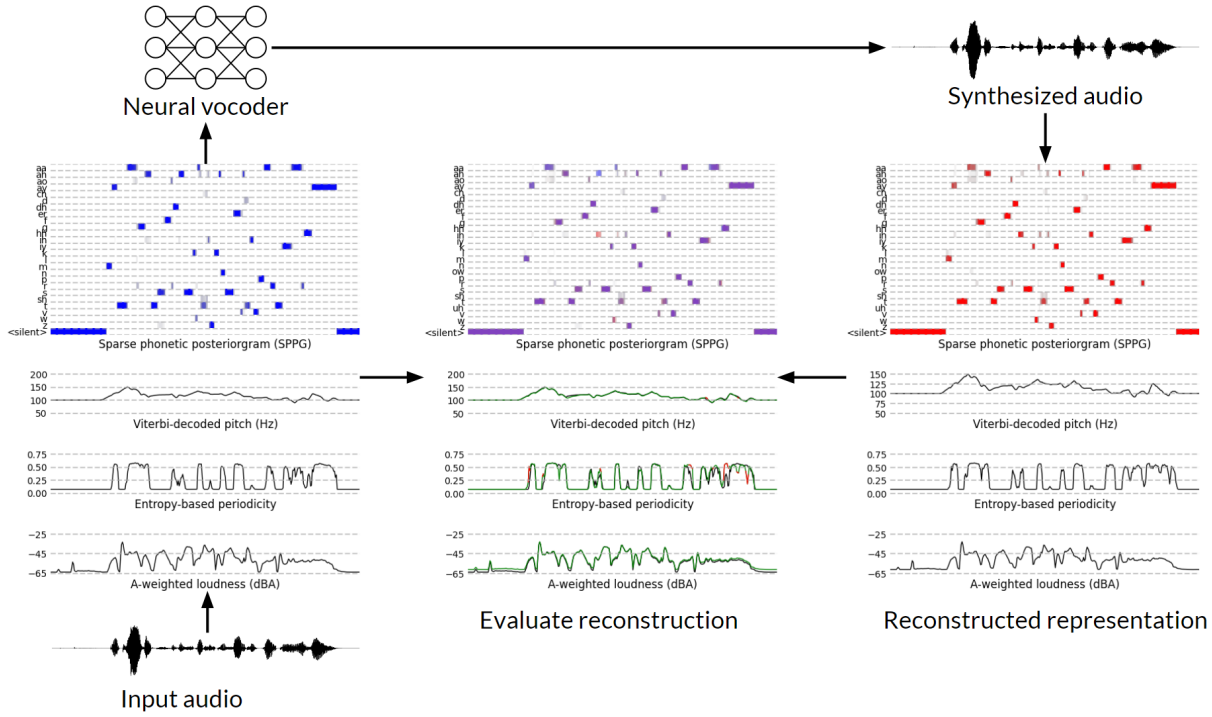

Figure 3.2. **Speech reconstruction overview** | I (1) encode an example speech utterance in my proposed representation (**left**), (2) perform speech synthesis (**top**), (3) encode the synthesized speech in my representation (**right**), and (4) overlay the speech representation inferred from synthesized speech on the input representation to demonstrate the reconstruction accuracy of my speech representation (**center**).
Blue SPPGs are inputs, **red** SPPGs are inferred from synthesized speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
`figure-3-2-0016-000442-original.wav`
`figure-3-2-0016-000442-reconstructed.wav`

representations such as Resemblyzer and WeSpeaker entangle speaker information with at least speaking rate and mean F0. If such a representation existed that fully disentangled speaker information from my proposed representation, it could be used for not

only evaluating speaker similarity but zero-shot voice conversion. I believe this is possible using jointly-learned discrete features [**41**].

Speech quality metrics such as Mel-cepstral distortion (MCD) [**44**] and PESQ [**31**] rely heavily on accurate reconstruction of precise spectral features. This makes both of these metrics impossible to use for evaluating the editing of pitch, duration, loudness, spectral balance, or pronunication, as no ground-truth spectral features exist for the edited speech. Further, these methods prioritize the accurate reconstruction of spectral features over perceptual quality, which induces bias in favor of spectral input features that more closely represent, e.g., background noise, channel effects, and *noise floor* reconstruction (i.e., the decibel level of the sum of all unwanted sounds in a signal).

3.3.1.1. Evaluating pronunciation accuracy. I use my proposed pronunciation distance (ΔPPG) between input SPPGs and SPPGs inferred from edited audio. As described in Section 2.1.4, ΔPPG demonstrates high correlation with word error rate (WER), which measures the number of errors made using an automatic speech recognition (ASR) system such as Whisper [**90**]. Whisper-based WER is commonly used as an evaluation metric for speech intelligibility. The most recent version of Whisper (Whisper-V3), is trained using roughly five million hours of data—including four million hours of unsupervised data gathered via web scraping. Because Whisper is trained using such large data scraped from the web, it is highly likely to contain not only my exact test utterances, but various common transformations such as Mel spectrogram reconstruction and DSP-based modifications—even though data augmentation was not explicitly used by the authors. My experience using WER to evaluate speech reconstruction and editing in this work has indicated problematic behaviors that can be explained by overfitting to noise in the speech

signal not reconstructed by my proposed representation. For example, the WER of speech reconstruction using Mel spectrograms in this chapter is 1.3%, while the WER of performing pitch-shifting and time-stretching using baseline DSP methods TD-PSOLA [**80**] or WORLD [**69**] to the point of clearly negatively impacting speech intelligibility produces a WER of 0.6%—less than half of the error of high-fidelity Mel reconstruction. I hypothesize that Whisper-V3 is highly overfit to imperceptible details in the speech waveform that are preserved during DSP-based editing using, e.g., TD-PSOLA or WORLD. This would make WER an inappropriate choice of evaluation metric for evaluation of, e.g., speech editing. This hypothesis was independently validated by Wang et al. [**113**] less than a month prior to my dissertation defense. This highlights a significant advantage for using my proposed pronunciation distance that is trained on a single, standard dataset (Common Voice [**3**]) and is relatively straightforward to train on other datasets compared to, e.g., Whisper [**90**]—making it easy to avoid data leakage during evaluation.

3.3.1.2. Evaluating pitch accuracy. I measure pitch error as the average framewise error in cents.

$$\Delta\text{¢}\left(y, \hat{y}\right) = \frac{1200}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} \left|\log_2(y_t / \hat{y}_t)\right| \tag{3.1}$$

$y = y_1, \ldots, y_T$ is the ground truth frame resolution pitch contour in Hz inferred from the original speech recording; $\hat{y} = \hat{y}_1, \ldots, \hat{y}_T$ is the pitch contour in Hz inferred from synthesized speech; and $\mathcal{V}$ is the set of voiced time frame indices where both the original and re-synthesized speech contain a pitch (i.e., when the entropy-based periodicity exceeds 0.1625; Figure 2.7). Evaluating pitch reconstruction in this manner is not novel; however, the efficacies afforded by improved pitch estimation (Section 2.2) as well as

voiced/unvoiced classification (Section 2.3.1) enable more accurate evaluation of pitch accuracy relative to prior works.

3.3.1.3. Evaluating periodicity accuracy. I propose to measure the average frame-wise RMSE of the entropy-based periodicity of the FCNF0++ pitch estimator (Section 2.3). My periodicity error is the average RMSE between ground truth frame resolution periodicity contour h inferred from the original speech recording and the predicted periodicity contour $\hat{h}$ inferred from synthesized speech.

$$\Delta\phi\left(h, \hat{h}\right) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(h_t - \hat{h}_t\right)^2} \tag{3.2}$$

My previous iteration of this periodicity evaluation using my previous periodicity estimator [**75**] has become a standard method in the literature for evaluating speech reconstruction accuracy [**52, 53, 97, 100**]. As discussed in Section 2.3, using my proposed entropy-based periodicity instead of my previous, Viterbi-based periodicity estimator has further advantages of being decoder-free (i.e., faster in some cases) and as well as elegantly handling polyphonic audio.

3.3.1.4. Evaluating loudness accuracy. I measure loudness error as the average frame-wise error of the single-band A-weighted loudness [**65**] in decibels. My loudness error is the average RMSE between ground truth frame resolution loudness contour $a = a_1, \ldots, a_T$ computed from the original speech and predicted loudness contour $\hat{a} = \hat{a}_1, \ldots, \hat{a}_T$ computed from synthesized speech.

$$\Delta\text{dBA}\left(a, \hat{a}\right) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(a_t - \hat{a}_t\right)^2} \tag{3.3}$$

3.3.2. Crowdsourced perceptual evaluation

Existing metrics for objective evaluation of speech synthesis are not sufficient to guarantee a corresponding improvement in human perception of synthesis quality, naturalness, or accuracy. Therefore, it is standard practice to include subjective evaluations of the proposed speech synthesis system relative to reference files (e.g., the original audio) or baseline speech synthesis systems. **Two primary challenges exist in performing robust subjective evaluation relative to objective evaluation.**

- **Challenge 1: Finding attentive participants |** Acquiring high-quality subjective evaluation data requires participants who are attentive and engaged in the evaluation task. As well, producing statistically significant results necessitates a sufficiently large (task-specific) number of participants. Crowdsourced survey platforms offer a solution: researchers upload their evaluation task (e.g., as a web page) to a third-party service (e.g., Amazon Mechanical Turk (MTurk) or Prolific) and specify criteria that participants must satisfy to be allowed to perform the evaluation in exchange for payment. For example, I perform my subjective evaluations on MTurk and require three participant qualifications: (1) participants must be located in the US, (2) participants must have completed at least 1,000 assignments, and (3) participants must have a minimum approval rating of 99%. I further perform my own prescreening listening test, which ensures participants are in a listening environment suitable for discerning auditory features of interest. These rather strict measures for filtering participants are designed to handle the MTurk ecosystem that includes bots, VPN usage, and black market sales of MTurk worker accounts that emerged due to the relatively

high compensation of performing surveys on MTurk compared to the median salary in many nations. Such accounts have been linked to poor data quality on some tasks [**116**]. However, recent policy changes such as the European Union's directive on administrative cooperation 7 (DAC 7)—which instates tax reporting requirements on all digital platforms that do business in the EU—may have limited the utility of black market account sales on MTurk. In short, while third-party services like MTurk and Prolific offer a solution to recruiting large numbers of participants for subjective evaluation, these services are themselves complex and time-varying systems with parameters that must be carefully and periodically tuned to the specific evaluation being performed.

- **Challenge 2: Developing the evaluation |** High-quality subjective evaluation requires the evaluation design to adequately represent the underlying experimental hypothesis being tested, while being simple and easy-to-follow to minimize participant confusion. This requires significant expertise in full-stack web development and user experience design, as well as knowledge of the trade-offs of common subjective evaluation designs such as A/B, ABX, Mean Opinion Score (MOS), and MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) [**30**] in order to select the correct design for the specific evaluation being performed. It is rare for machine learning and audio researchers to have sufficient background in these skills to quickly produce a high-quality subjective evaluation system. Instead, one-off subjective evaluation systems of questionable design are commonplace. Crowdsourcing platforms such as MTurk attempt to provide templates for common tasks (e.g., sentiment analysis, toxicity detection,

and image captioning). However, the support for audio tasks is insufficient, and does not support, e.g., a prescreening listening test to verify that the participant can hear the audio.

In my early work [**74, 75, 78**], I relied on closed-source subjective evaluation systems. In my experience, every company doing generative machine learning research (and many labs) spends considerable resources independently developing closed-source subjective evaluation systems for evaluating both research and commercial products. In some cases, these systems are mature enough that high-quality subjective evaluation becomes a "coin-operated machine": it is as easy to perform as objective evaluation (e.g., calling a Python function) with the notable difference of incurring a direct financial cost. I found these efficient subjective evaluation systems to be instrumental in quickly evaluating my speech synthesis and speech editing research. As such, I developed and open-sourced my own subjective evaluation system called Reproducible Subjective Evaluation (ReSEval) [**79**].

ReSEval (Figure 3.3) lets researchers launch A/B, ABX, Mean Opinion Score (MOS), MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) tests, and more on audio, image, text, or video data from a command-line interface or using one line of Python, making it as easy to run as objective evaluation. ReSEval was accepted as a workshop paper at the 2022 International Conference on Learning Representations (ICLR) [**79**].

I use ReSEval (v0.1.6) for all subjective evaluations performed in my dissertation. I pay participants according to the minimum wage in Evanston, Illinois, US, which was raised on January 1st, 2024 from $13.35 per hour to $14.00 per hour (USD). I use all 100 test utterances from VCTK described in Section 3.2 for all subjective evaluations. I determine which utterances are seen by which participant via uniform random sampling

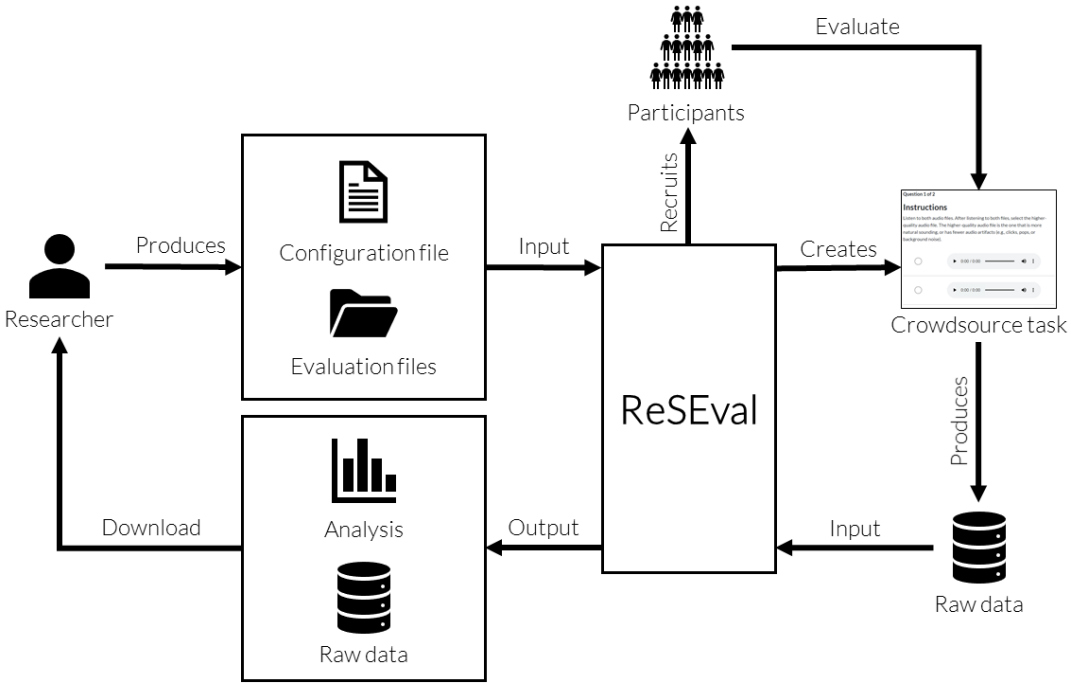

Figure 3.3. **Reproducible subjective evaluation (ReSEval) system flow** | A researcher creates a subjective evaluation by providing a configuration file and a directory of evaluation files as input. ReSEval creates a crowdsource task and recruits participants via, e.g., Amazon Mechanical Turk (MTurk). Participants complete the task, producing evaluation data for analysis. ReSEval analyzes the evaluation data and presents the researcher with a statistical analysis. With ReSEval, the researcher does not have to perform any web development. As well, aside from a one-time acquisition of API keys, the researcher does not have to interact with any third-party services (e.g., MTurk, Heroku, or Amazon Web Services). Instead, ReSEval performs all of the necessary interactions with third-party services to configure and manage databases, servers, file storage, and crowdsourcing on behalf of the researcher.

without replacement. Further details of subjective evaluation designs corresponding to specific evaluations are provided in their respective sections.

Representation	Δ¢ ↓	$\Delta\phi$ ↓	ΔdBA↓	ΔPPG↓	Subjective↑
Proposed	**17.1**	**.055**	.959	.109	.471 ± .046
Mel spectrogram	21.5	.061	**.512**	**.041**	.529 ± .046

Table 3.1. **Speech reconstruction results |** Objective and subjective evaluation of speech reconstruction using Mel spectrograms (Section 1.1.2) or my disentangled, interpretable representation (Chapter 2; Figure 1.2). Objective evaluation metrics are defined in Section 3.3.1 and my subjective evaluation is described in Sections 3.3.2 and 3.4. ↑ indicates that higher is better and ↓ indicates that lower is better.

3.4. Speech reconstruction

In speech reconstruction, a speech recording is encoded in a speech representation—such as a Mel spectrogram (Figure 1.1; bottom) or my proposed speech representation (Chapter 2; Figure 1.2)—and then decoded via a speech synthesis system to reproduce the original speech recording. While speech reconstruction is not itself a useful operation (the input and output are—at best—the same), it is a standard method for evaluating the accuracy and perceptual quality of speech representations.

I compare speech reconstruction using my proposed representation to Mel spectrogram vocoding. While Mel spectrograms do not disentangle pronunciation from prosody or voicing—and therefore cannot be used for corresponding speech editing tasks—vocoding with HiFi-GAN is a common high-anchor for text-to-speech systems: comparable perceptual reconstruction accuracy with Mel spectrogram vocoding indicates reconstruction that often requires a trained ear and a suitable listening environment to distinguish. I include speaker conditioning (Section 3.2.2), data augmentation (Section 3.2.1), and the complex, multi-band discriminator [**45**] in our baseline Mel spectrogram model. This is for fair comparison and because these techniques improve the baseline. I use objective metrics described in Section 3.3.1.

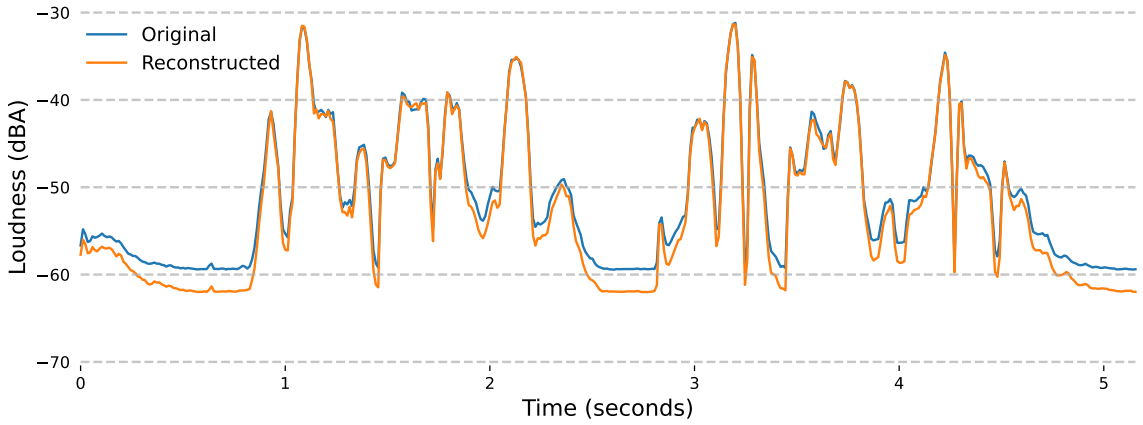

Figure 3.4. **Noise floor reconstruction** | Single-band A-weighted loudness of original speech and speech reconstruction using my proposed speech representation (Chapter 2). The majority of loudness reconstruction error (Table 3.1) can be attributed to my proposed representation not reconstructing the exact noise floor of the speech recording. Note that it is straightforward to copy/paste the silent regions of the original audio into the edited audio and apply crossfades to produce the exact noise floor.
`figure-3-4-0016-000321.wav`

For subjective evaluation, I recruit 35 participants to each perform 15 ABX comparisons of reconstruction accuracy. In an ABX comparison, a participant selects which of two speech recordings ("A" or "B") sounds more similar to a reference recording ("X"). I use as reference recording the original audio and have participants select between corresponding reconstructions using Mel spectrograms or my proposed representation (Chapter 2). All participants passed the listening test and five participants left early, giving us 450 ABX comparisons.

Table 3.1 shows that participants rated our representation more similar to ground truth audio in 212 of 450 ABX comparisons (47.1%). A two-sided Binomial test indicates no significant preference among raters ($p = 0.23$); the perceptual reconstruction accuracy of our representation is roughly as good as Mel spectrograms. Table 3.1 further shows that

my representation improves pitch and periodicity reconstruction relative to Mel spectrograms. Note that my PPGs are computed from Mel spectrograms, which biases ΔPPG to favor Mel vocoding. This is because both the PPG neural network and the speech synthesizer neural network learn to utilize low-energy bins in the Mel spectrogram that do not significantly contribute to auditory perception to overfit and reduce training loss. Further, Mel spectrograms are in essence a fine-grained (frequency-dependent) representation of energy, while my proposed representation can only coarsely capture the noise floor (i.e., the decibel level of the sum of all unwanted sounds in a signal); the primary source of loudness error (ΔdBA) is due to our model producing silence with a different noise floor (Figure 3.4). This can further be seen by splitting the evaluation of ΔdBA at a silence threshold of −60 dBA, such that ΔdBA_+ evaluates loudness reconstruction accuracy in frames t where both $a_t > -60$ and $\hat{a}_t > -60$ and ΔdBA_- evaluates loudness reconstruction accuracy in all other frames. My proposed representation produces $\Delta\text{dBA}_- = 1.30$ dBA and $\Delta\text{dBA}_+ = 0.57$ dBA, whereas Mel spectrogram reconstruction produces $\Delta\text{dBA}_- = 0.63$ dBA and $\Delta\text{dBA}_+ = 0.40$ dBA. Thus, the improvements in loudness reconstruction of Mel spectrogram vocoding relative to my proposed representation are primarily due to better reconstruction of low-energy time-frequency bins that are largely irrelevant to human perception. Figure 3.5 overlays an example utterance in my proposed representation with my representation inferred from reconstructed speech to provide a complete depiction of the reconstruction accuracy using my proposed representation.

In this chapter, I described my system for producing synthesized speech from my proposed representation (Section 3.1) established a methodology for objective and subjective

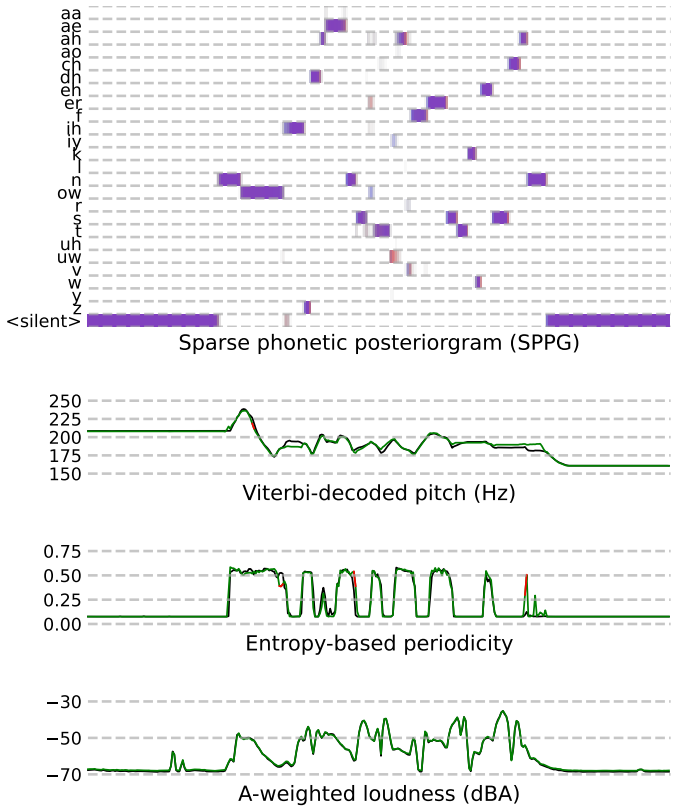

Figure 3.5. **Speech reconstruction example |** I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) perform speech synthesis, (3) encode the synthesized speech in my representation, and (4) overlay the speech representation inferred from synthesized speech on the input representation to demonstrate the reconstruction accuracy of my speech representation.
Blue SPPGs (**top**) are inputs, **red** SPPGs are inferred from synthesized speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
`figure-3-5-0108-000345-original.wav`
`figure-3-5-0108-000345-reconstructed.wav`

evaluation of speech synthesis (Section 3.3) and demonstrated that my proposed representation is capable of performing speech reconstruction with objective and subjective

performance comparable to the widely-used Mel spectrogram (Section 3.4). However, speech reconstruction is not useful it itself: at best, the output is the same as the original audio. In the next chapter, I utilize my speech synthesis system and evaluation methodologies to showcase a variety of high-fidelity, state-of-the-art, and novel speech editing capabilities afforded by my representation that are difficult or impossible to perform using a Mel spectrogram.

CHAPTER 4

High-fidelity interpretable speech editing

Now that I have shown that my speech representation (Chapter 2) can be used as an input representation to perform high-fidelity speech reconstruction (Chapter 3), it remains to demonstrate that my speech synthesis system trained on my representation can be used to perform interpretable and disentangled editing of speech attributes of interest. In this chapter, I demonstrate that my speech editing system is capable of performing fine-grained (i.e., frame-resolution) editing of pitch (Section 4.1), phoneme durations (Section 4.2), the timbral correlates of volume (Section 4.3) and pronunciation (Section 4.7). I further demonstrate global (i.e., applied to the entire speech recording) spectral balance editing (Section 4.4) and speaker identity via both voice conversion (Section 4.6) and speaker adaptation (Section 4.5). Finally, I perform ablations of key design decisions that I have proposed throughout my dissertation (Section 4.8).

Editing the pitch contour of a speech recording requires specifying a target pitch for each frame. Likewise, editing phoneme durations necessitates target phoneme durations, editing loudness requires a target loudness contour, and editing the spectral balance requires specifying a value for r_f, the relative spectral energy augmentation parameter described in Section 3.2.1. Therefore, to demonstrate editing of each of these parameters, I must specify target values. In practice, many appropriate target values exist—so long as the edits are perceptible and applied uniformly to all speech editing models being compared. I perform pitch-shifting by editing the pitch inferred from original speech by ±600

cents (i.e., one tritone), time-stretching by changing the total speech duration in seconds by $\sqrt{2}$ or $\sqrt{2}/2$, loudness editing by ± 10 dBA, and editing of relative formant energies using $r_f = \sqrt{2}$ and $r_f = \sqrt{2}/2$. A pitch shift of ± 600 is equivalent to multiplying the frequency in Hz by $\sqrt{2}$ or $\sqrt{2}/2$. This decision of using $\sqrt{2}$ and $\sqrt{2}/2$ is mostly arbitrary. One benefits of using these factors is that pitch-shifting by ± 600 cents is equivalent to pitch-shifting by a musical tritone. Speech formants (Figure 1.1) place high amounts of energy at integer multiples of F0 in Hz (i.e., 1200 cents, 2400 cents, ...). Pitch-shifting by a tritone maximally removes any possible bias induced by overlapping formant energies before and after shifting. However, I found that the corresponding loudness edit of ± 5 dBA was only subtly perceptible, and instead use ± 10 dBA. On the companion website[1] of my corresponding paper on speech editing [**71**], I demonstrate editing pitch, duration, loudness, and spectral balance using a range of values.

4.1. Editing pitch

I demonstrate the disentanglement of pitch by performing pitch-shifting using my proposed speech editing system and performing objective and subjective evaluation. For both objective and subjective evaluation, I use pitch-shifting by ± 600 cents (i.e., one tritone) while keeping all other features the same. I use both TD-PSOLA [**80**] and WORLD [**69**] as baseline pitch-shifting methods. Despite being over thirty years old, TD-PSOLA represents the state-of-the-art in pitch-shifting accuracy, with only my prior work [**73**] and Wang et al. [**115**] demonstrating comparable pitch-shifting accuracy and subjective quality. I use the objective metrics described in Section 3.3.1. I also perform a subjective evaluation, in which participants listen to speech recordings from each condition and rank

[1]Temporary link during review period: https://master.d17fxebptfqpit.amplifyapp.com/

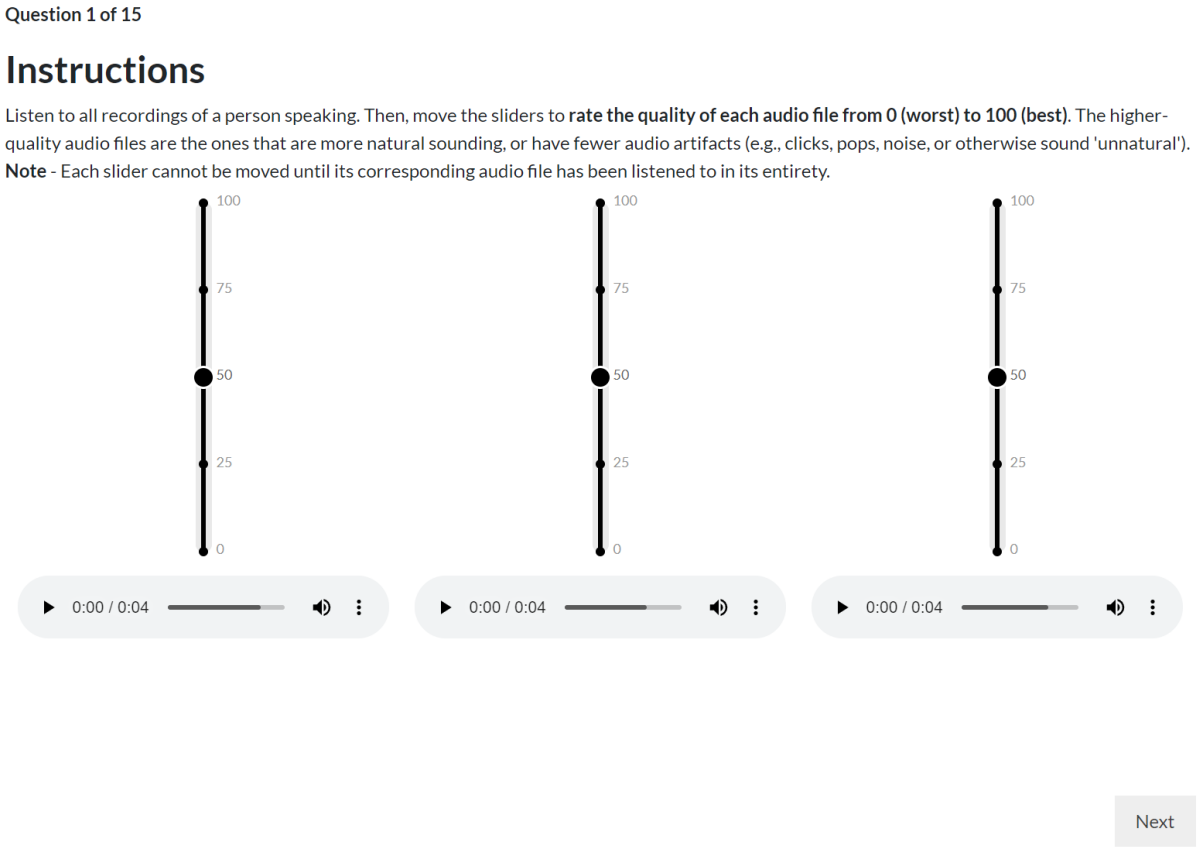

Figure 4.1. **Subjective evaluation interface for pitch-shifting and time-stretching |** Participants recruited on Amazon Mechanical Turk listen to 15 sets of three audio files, where each set consists of the same edit (a pitch-shift or time-stretch) applied to an original audio recording using three methods: (1) my proposed method for pitch-shifting (Section 4.1) or time-stretching (Section 4.2) as well as DSP baselines (2) TD-PSOLA [**80**] and (3) WORLD [**69**].

their relative quality from 0 (worst) to 100 (best) using a slider (Figure 4.1). Note that this is strictly a test of perceptual quality and not accuracy; both my baseline and proposed methods for pitch-shifting exhibit sufficient objective accuracy to be likely within the just noticeable difference (JND) of human pitch discrimination of speech signals [**37**].

I recruit 35 participants. Five participants failed the prescreening listening test for the pitch-shifting and one participant left early, so I received 435 three-way comparisons.

Method	Δ¢ ↓	$\Delta\phi$ ↓	ΔdBA↓	ΔPPG↓	Subjective↑
Proposed	22.5	**.090**	2.17	.137	**68.9**
TD-PSOLA [**80**]	21.6	.115	**1.66**	**.109**	61.1 ± 3.13
WORLD [**69**]	**17.7**	.113	1.76	.286	45.0 ± 3.25

Table 4.1. **Pitch-shifting results** | Results of pitch-shifting by ±600 cents using my proposed system and two DSP baselines. Objective metrics are defined in Section 3.3.1). ↑ indicates that higher is better and ↓ indicates that lower is better.

Table 4.1 indicates that pitch-shifting using my proposed speech editing system demonstrates statistically significant improvements in perceptual quality over both TD-PSOLA [**80**] ($p = 5.41 \times 10^{-6}$) and WORLD [**69**] ($p = 3.56 \times 10^{-38}$) using Wilcoxon signed-rank tests. Note that as periodicity error ($\Delta\phi$) improves, the number of voiced frames increases. However, these additional voiced frames have relatively low periodicity. This makes accurate pitch estimation (Δ¢) more difficult. Thus, pitch and periodicity error must be jointly evaluated, and no single system stands out as "best" in terms of pitch and periodicity; however, my proposed method is clearly superior in perceptual quality. Figure 4.2 overlays an example utterance in my proposed representation where pitch is edited −600 cents (left) or +600 cents (right) with my representation inferred from the corresponding synthesized speech. Focus on the yellow boxes in this figure, which contain the desired pitch contours for pitch-shifting down (left) and up (right) overlaid with corresponding pitch contours inferred from speech that was synthesized using the desired pitch contours. Within voiced regions (i.e., when the periodicity exceeds 0.1625), more overlap between

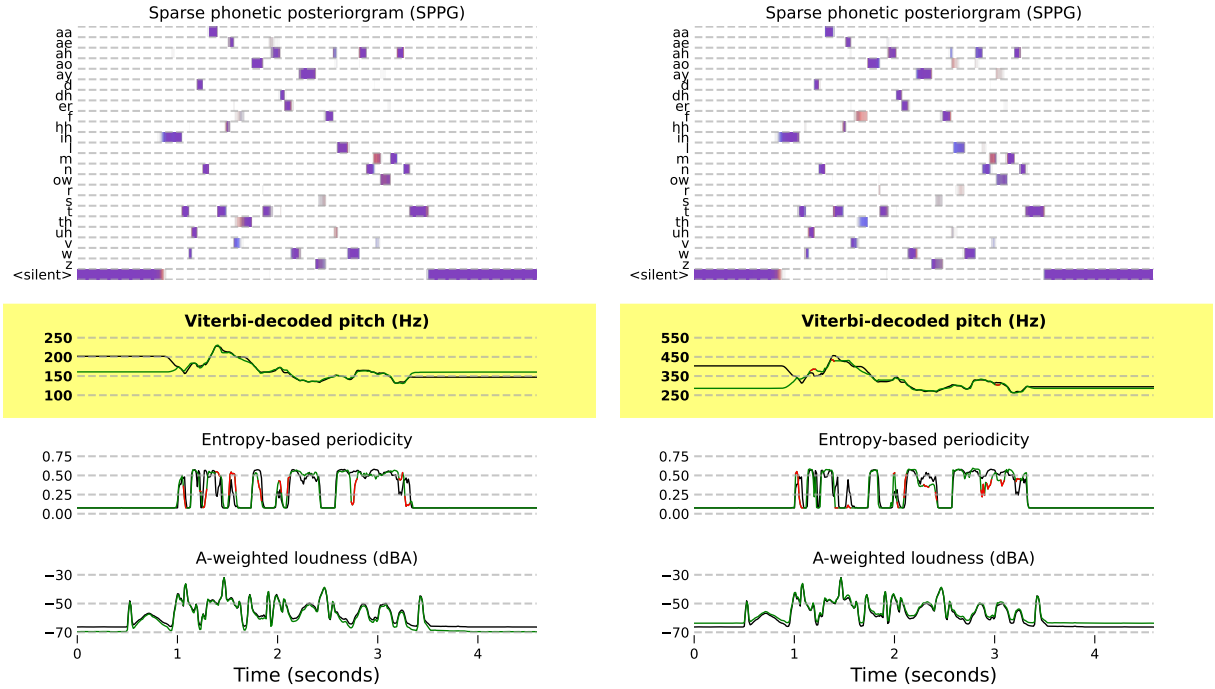

Figure 4.2. **Pitch-shifting example |** I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) modify the pitch by −600 (**left**) and +600 (**right**) cents, (3) perform speech synthesis using the modified pitch contours to produce pitch-shifted speech, (4) encode the pitch-shifted speech in my representation, and (5) overlay the speech representation inferred from pitch-shifted speech on the input representation (*after the* ±600 *cent shift has been applied*) to demonstrate pitch disentanglement. **N.B., the pitch range (i.e., the y-axis) varies between the left and right figure.**
Blue SPPGs (**top**) are inputs, **red** SPPGs are inferred from pitch-shifted speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from pitch-shifted speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
🔊 `figure-4-2-0108-000430-original.wav`
🔊 `figure-4-2-0108-000430-(-600¢).wav`
🔊 `figure-4-2-0108-000430-(+600¢).wav`

pitch contours indicates a more successful edit. As you can see, the pitch contours are strongly overlapped.

4.2. Editing duration

I demonstrate the ability to perform variable-rate time-stretching using my proposed speech editing system. Variable-rate time-stretching can be accomplished using my proposed speech editing system by performing interpolation on my speech representation (Chapter 2) and then using my speech synthesizer (Section 3.1) to synthesize speech from my interpolated speech representation. Pitch (in base-two log-Hz), periodicity, and loudness (in dBA) are all amenable to linear interpolation. PPGs are constrained to be a valid, $|P|$-dimensional categorical distribution (i.e., non-negative and sums to one). This is equivalent to constraining PPG frames to be points on the $|P|$-dimensional simplex. Performing linear interpolation between two points on a simplex (i.e., non-negative and sums to one) also produces a point on the simplex. Having empirically compared both linear interpolation and SLERP but not previously worked out this equivalence, my current system utilizes spherical linear interpolation (SLERP) [**98**]. SLERP is a standard method for interpolation on the surface of the hypersphere, which intersects the simplex when the PPG places all probability on one phoneme. In practice, objective evaluation (Section 3.3.1) on my validation data using either interpolation method for interpolating PPGs produces equivalent results during time-stretching. This indicates that the network learns to compensate for this (linear) transformation during training due to the interpolation from the sample rate used during PPG inference (16 kHz; Section 2.1) and the sample rate used for speech synthesis (22.05 kHz; Section 3.1). I use SLERP when interpolating PPGs (and SPPGs) and linear interpolation for pitch (in base-two log-Hz), periodicity, and loudness.

Human speakers change speaking rate in a phoneme-dependent manner [**46**]. This corresponds to variable-rate time-stretching. I perform a coarse generalization of this phoneme-dependent behavior using two rates: one (i.e., 1.0) for unvoiced phonemes, and another (d) for all other frames. I perform time-stretching to increase or decrease the total duration of speech by a factor of $\sqrt{2}$, such that $d = \sqrt{2}\left(1 - \frac{|\mathcal{U}|}{T}\right)$ to increase the total duration and $d = \frac{\sqrt{2}}{2}\left(1 - \frac{|\mathcal{U}|}{T}\right)$ to decrease the total duration. $\mathcal{U}$ is the set of frames in which the sum of unvoiced phoneme probabilities in the SPPG exceeds 50%. Note that one could also determine the set of voiced and unvoiced frames by thresholding the entropy-based periodicity (Section 2.3.1). However, periodicity is more strongly influenced by per-frame variations in background noise and hum, making it more descriptive of the overall audio signal—as opposed to just the salient speech content. In contrast, PPGs are trained to detect the spoken phoneme regardless of background noise. This produces clearly improved alignment with the voiced and unvoiced phonemes, detecting entire voiced phonemes missed by the periodicity-based voiced/unvoiced method (Figure 4.3).

I evaluate variable-rate time-stretching using the objective metrics described in Section 3.3.1. I also perform a subjective evaluation using the same design and baseline methods as was used for evaluating pitch-shifting (Section 4.1). No participants failed my listening test or left early, so I received perceptual results from 525 three-way comparisons.

Method	**Δ¢ ↓**	**Δϕ ↓**	**ΔdBA↓**	**ΔPPG↓**	**Subjective↑**
Proposed	20.4	.066	**1.29**	.195	64.0
TD-PSOLA [**80**]	22.0	**.062**	1.65	**.189**	63.3 ± 1.71
WORLD [**69**]	**18.2**	.103	4.48	.473	46.5 ± 2.20

Table 4.2. **Time-stretching results** | Objective and subjective results of time-stretching by factors of $\sqrt{2}$ and$\sqrt{2}/2$. ↑ indicates that higher is better and ↓ indicates that lower is better.

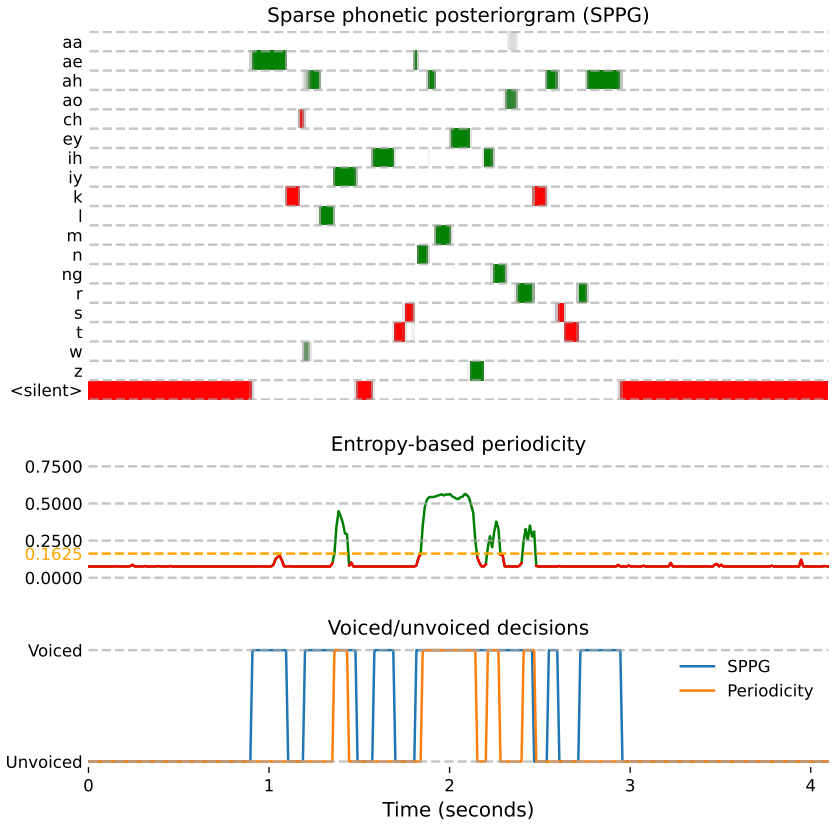

Figure 4.3. **Comparing voiced/unvoiced decisions from periodicity and PPGs |** (**top**) SPPG (Section 2.1) with voiced phonemes in **green** and unvoiced phonemes in **red**. (**middle**) Entropy-based periodicity (Section 2.3) with voiced frames in **green**, unvoiced frames in **red**, and voicing threshold α in **orange**. (**bottom**) Voicing decisions derived from the SPPG (**blue**) and periodicity (**orange**). The SPPG framewise voiced/unvoiced decisions exhibit clearly improved alignment with voiced and unvoiced phonemes.
`figure-4-3-0083-000365.wav`

Table 4.2 demonstrates that my proposed system performs variable-rate time-stretching with perceptual quality at least as good as high-fidelity DSP-based baselines. A Wilcoxon signed-rank test between my proposed system and DSP-based method TD-PSOLA indicates an insignificant preference for my system ($p = 0.45$). Further analysis indicates a

small, insignificant preference for my method relative to baselines when decreasing the duration (increasing speed; $p = 0.35$) relative to increasing the duration ($p = 0.87$). Table 4.2 shows that my system achieves competitive values for all objective metrics and top performance for loudness reconstruction. Figure 4.4 overlays an example utterance in my proposed representation that has been interpolated to be slowed down (left) or sped up (right) by a total factor of $\sqrt{2}$ with my representation inferred from the corresponding synthesized speech. Focus on the difference in duration of the left and right plots, as indicated by the x-axis at the bottom. Successful time-stretching should produce left and right plots that look as close as possible (except the difference in duration) and overlap precisely with the corresponding interpolated representation used as input. Clearly, the left and right plots look similar and exhibit strong overlap with the input representation. The largest difference is small deviations in periodicity that differentiate the voiced fricative "z" and the corresponding unvoiced fricative "s". My prior work indicates that this is a known drawback of non-autoregressive speech synthesizers with relatively small receptive fields [**75**]. Figure 4.4 also indicates that there is a small amount of misalignment in the SPPGs. I hypothesize this may be due to learning some properties of speaking rate variation from the volume data augmentation, which applies time-stretching to the entire recording (instead of only specific phonemes) at a constant rate.

4.3. Editing the timbral correlates of volume

Human speech possesses different timbral qualities correlated and associated with speaking louder or quieter. At the extremes, quiet speech is associated with whispering—wherein vocal fold engagement is not sufficient to produce voiced speech—and loud speech

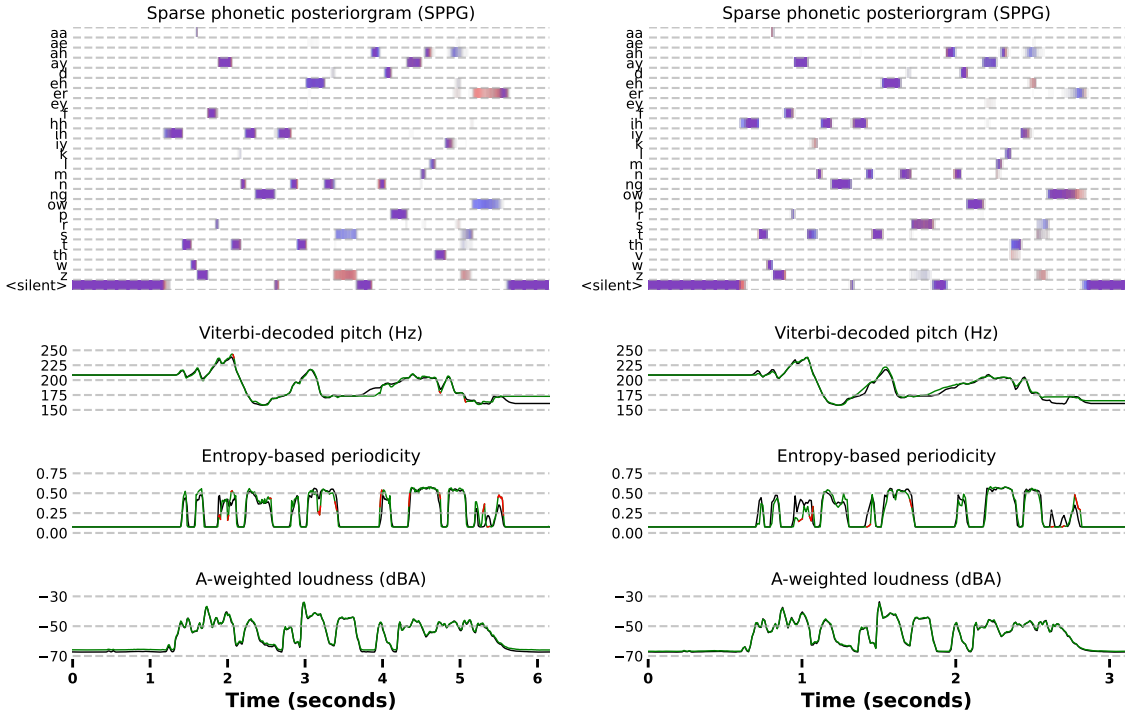

Figure 4.4. **Time-stretching example** | I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) interpolate the voiced and silent frames of my proposed representation to increase (**left**) or decrease (**right**) the total duration by a factor of $\sqrt{2}$, (3) perform speech synthesis using the interpolated representation to produce time-stretched speech, (4) encode the time-stretched speech in my representation, and (5) overlay the speech representation inferred from time-stretched speech on the input representation (*after interpolation*) to demonstrate the preservation of speech features during time-stretching. **N.B., the duration (i.e., the x-axis) varies between the left and right figure.**
Blue SPPGs (**top**) are inputs, **red** SPPGs are inferred from time-stretched speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
`figure-4-4-0082-000568-original.wav`
`figure-4-4-0082-000568-(0.71x).wav`
`figure-4-4-0082-000568-(1.41x).wav`

with shouting. Between these extremes, variations in volume of airflow across the glottis as well as variation in vocal fold tension produce differences in dynamic range (e.g., sharper or flatter signal transients at the beginnings of plosives) as well as differences in high-frequency content. This effect can also be observed in any musical instrument: a guitar string being plucked harder or a woodwind or brass instrument with more input airflow will have a "brighter" timbre, consisting of relatively more high-frequency content. Audio playback devices and volume editing via standard DSP-based methods decorrelate volume from the timbral correlates of volume: whispers can be made arbitrarily large and shouts arbitrarily quiet.

Standard audio effects plug-ins (e.g., multi-band compressors) can theoretically be used to manually model the effect of varying vocal engagement. However, the correct settings are dependent on the speaker and F0, meaning that one would also have to perform fine-grained manual automation. This is a highly labor-intensive editing workflow not typically performed in practice. Given the option, speech content creators would be more likely to have the speaker rerecord the audio. In professional studios, this requires additional cost and time. Further, generated speech content cannot be rerecorded in this manner, and regeneration may cause changes throughout the speech recording as opposed to a precise, fine-grained edit of a few frames (e.g., one word) of the speech recording.

My research is the first to demonstrate fine-grained disentanglement and control of volume and the timbral correlates of volume, where volume editing is performed using standard DSP-based gain scaling (as is used during data augmentation of volume in Section 3.2.1) and jointly editing volume and the timbral correlates of volume is performed by editing the frame-resolution A-weighted loudness (Section 2.4). My system can also

perform volume editing independent of the timbral correlates of volume by editing the gain scaling augmentation ratio r_l; however, I find this to be less accurate and without advantage over standard DSP-based volume editing.

I perform editing of the timbral correlates of volume using my proposed system by editing the single-band A-weighted loudness contour (i.e., the average over all bands of the multi-band A-weighted loudness), applying the edit uniformly to all bands of the multi-band loudness, and then using my proposed speech synthesizer (Section 3.1) to synthesize speech with the desired edit. I perform both subjective and objective evaluation. I use the objective metrics described in Section 3.3.1. For subjective evaluation, I perform an A/B subjective evaluation between original speech recordings and corresponding recordings synthesized single-band A-weighted loudness contours modified to be 10 dBA louder or quieter throughout. I then volume match the synthesized speech with the original speech at the frame resolution using DSP-based gain scaling between the original A-weighted loudness and the A-weighted loudness inferred from synthesized speech. This is the same DSP-based gain scaling used during data augmentation (Section 3.2.1). This produces pairs of original and perceptual-loudness-edited speech recordings with equal A-weighted loudness. I ask 35 participants to select which of two speech recordings "sounds like the speaker is speaking **louder**". Six participants failed the prescreening listening test, so I receive 435 perceptual A/B comparisons.

Table 4.3 indicates that jointly editing the volume and the timbral correlates of volume using my proposed system produces objective metrics (Section 3.3.1) comparable to performing speech reconstruction without edits (Section 3.4). In 290 subjective A/B comparisons ($66.7 \pm 4.6\%$), participants' selection matches my intended modification of

Representation	Δ¢ ↓	$\Delta\phi$ ↓	ΔdBA↓	ΔPPG↓	Subjective↑
Proposed (reconstruction)	17.1	.055	.959	.109	33.3 ± 4.6%
Proposed (±10 dBA)	17.9	.065	1.91	.141	**66.7 ± 4.6%**

Table 4.3. **Results for editing volume and timbral correlates of volume** | Objective results of jointly editing volume and its timbral correlates, and subjective evaluation of disentangled editing of the timbral correlates of volume by first performing joint editing and then DSP-based A-weighted loudness matching at the frame resolution. Reconstruction objective metrics included to highlight the accuracy of editing relative to reconstruction. Objective evaluation metrics are defined in Section 3.3.1. ↑ indicates that higher is better and ↓ indicates that lower is better.

the timbral correlates of volume on volume-matched audio ($p = 3.20 \times 10^{-12}$). Figure 4.5 plots the frequency spectra of reconstruction and volume-matched editing of the timbral correlates of volume to demonstrate that my proposed editing method has the expected behavior of performing a corresponding increase or decrease in high-frequency speech content. Figure 4.6 overlays an example utterance in my proposed representation where the single-band A-weighted loudness has been modified by ±10 dBA with my representation inferred from the corresponding synthesized speech. Focus on the yellow boxes in this figure, which contain the desired A-weighted loudness contours for jointly decreasing (left) and increasing (right) the volume and its timbral correlates, overlaid with corresponding A-weighted loudness contours inferred from speech that was synthesized using the desired loudness contours. More overlap between A-weighted loudness contours indicates a more successful edit. As you can see, the A-weighted loudness contours are strongly overlapped. Noise floor reconstruction in the right plot is not perfectly overlapping, but—as discussed in Section 3.4)—this is a nearly imperceptible difference. Note that when the change in spectral balance is large enough, the inferred phoneme in the SPPG can also change.

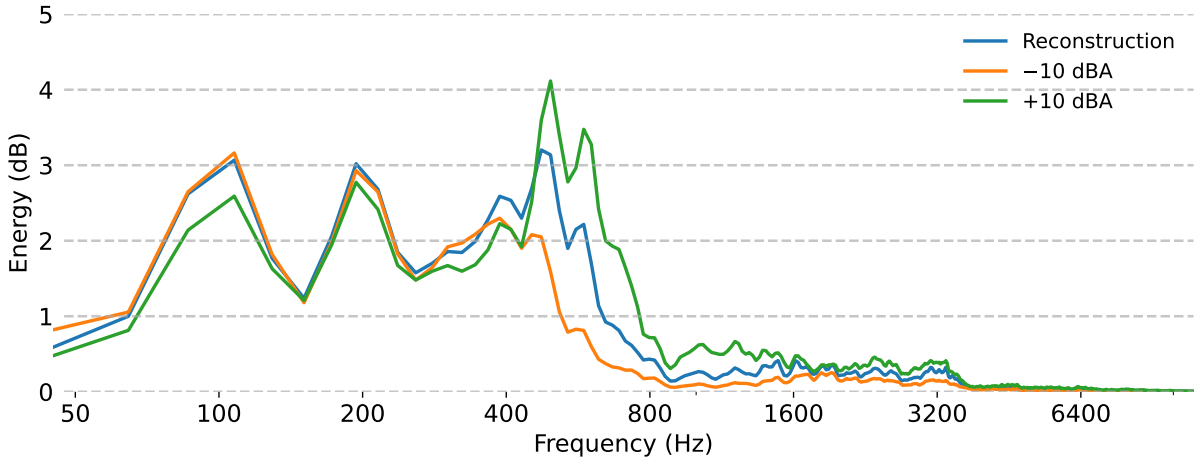

Figure 4.5. **Frequency spectra of speech reconstruction and volume-matched editing of timbral correlates of volume** | I modify the single-band A-weighted loudness (Section 2.4) of an unedited speech recording by 0 (i.e., reconstruction; **blue**), −10 (**orange**), and +10 dBA (**green**) using my proposed method for jointly editing volume and its timbral correlates. I then perform frame-resolution A-weighted loudness matching using DSP-based gain scaling (Section 3.2.1) with the original audio. This produces reconstructed and perceptual-loudness-edited speech recordings with equal volume. I plot the frequency spectra of speech reconstruction as well as volume-matched editing of the timbral correlates of volume to show that my proposed method captures the expected behavior of perceptually louder speech having relatively more high-frequency content. In voice quality literature, this increase in the relative amount of high-frequency content has been found to explain over 20% of intraspeaker acoustic variations [**54**]. Likewise, most musical instruments produce relatively more high frequency content when played with more input energy (e.g., hitting a drum harder or bowing a violin string faster or with more pressure). My proposed system learns the timbral correlates of volume within a speech dataset and enables disentangled control.

`figure-4-5-0073-000047-original.wav`
`figure-4-5-0073-000047-reconstruction.wav`
`figure-4-5-0073-000047-(-10dBA).wav`
`figure-4-5-0073-000047-(+10dBA).wav`

This is most evident in Figure 4.6 in the phonemes /aa/, /ao/, and /er/, where /aa/ has relatively more energy in high-frequency harmonics and /er/ has relatively less.

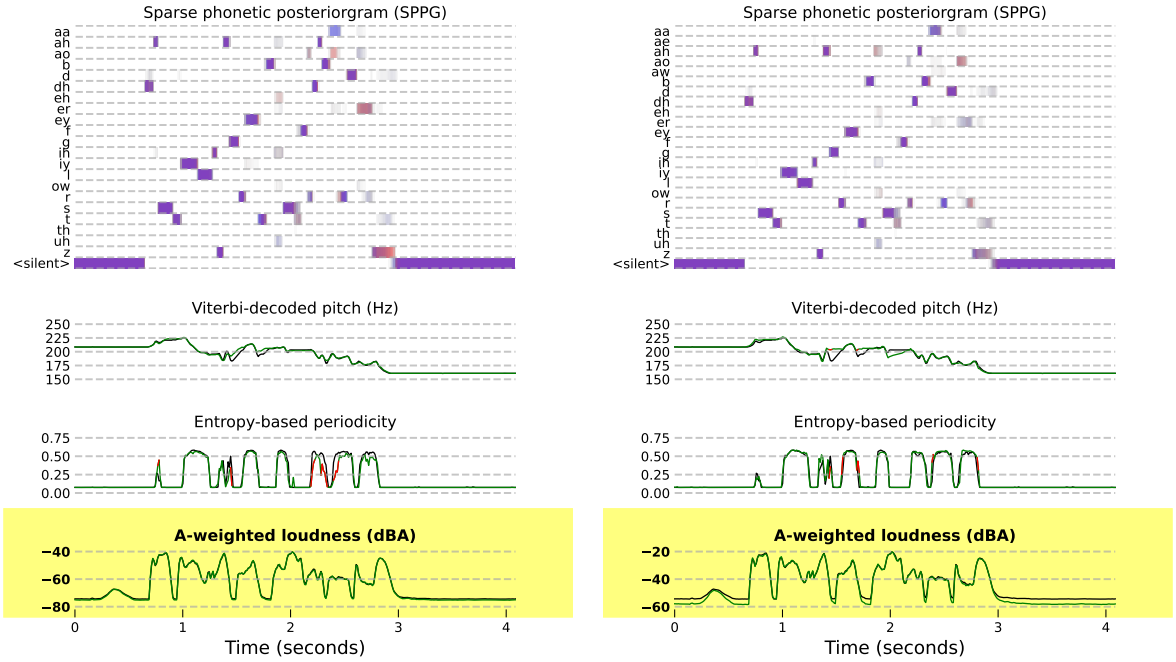

Figure 4.6. **Example of jointly editing volume and the timbral correlates of volume |** I (1) encode a speech utterance in my proposed representation (Chapter 2), (2) modify the single-band A-weighted loudness by −10 (**left**) and +10 (**right**) dBA, (3) perform speech synthesis using modified A-weighted loudness to produce speech in which volume and its timbral correlates are jointly edited, (4) encode the synthesized speech in my representation, and (5) overlay the speech representation inferred from synthesized speech on the input representation (*after ±10 dBA edits have been applied to the A-weighted loudness*) to demonstrate accurate reconstruction of input features during loudness and volume editing.
Blue SPPGs (**top**) are inputs, **red** SPPGs are inferred from synthesized speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume. **N.B., the loudness range (i.e., the y-axis) varies between the left and right figure.**

`figure-4-6-0037-000659-original.wav`
`figure-4-6-0037-000659-(-10dBA).wav`
`figure-4-6-0037-000659-(+10dBA).wav`

4.4. Editing spectral balance

I demonstrate that my proposed resampling-based data augmentation (Section 3.2.1) permits disentangled control over the relative energy of high- and low-frequencies. I perform speech synthesis using two values ($r_f = \sqrt{2}$ and $r_f = \sqrt{2}/2$) for the relative spectral balance parameter described in Section 3.2.1. I perform estimation of F0 and its harmonics and measure the displacement of F0 reconstruction and its first two harmonics H1, H2 in cents in voiced regions. I also measure the change in spectral centroid between ground-truth and edited audio in voiced regions. Low displacement error with a change in spectral centroid in the direction of r_f indicates disentangled control of the spectral balance.

Qualitatively, increases to r_f produces a similar effect as, e.g., Alvin from *Alvin and the Chipmunks*, but without requiring voice actors to sing/speak unnaturally slowly—as was done in the original recording process. A closed-source effect similar to this is available in some commercial audio plug-ins, and has been used in, e.g., a popular hip-hop song "Swimming Pools" by Kendrick Lamar. Note that the lyrics of this song contain explicit content. In both cases, the efficacy of the use of this type of editing is highly subjective—some listeners may consider the effect appropriate for the context and some may not. For this reason, I do not perform a subjective evaluation of spectral balance editing. Further refinement and evaluation of my proposed method—as well as including a mechanism for fine-grained, framewise edits—may produce useful facilities for voice quality editing, which is associated with semantic attributes such as speaker emotion [**24**] and performed gender [**38**].

My experiences so far in using existing formant estimation methods such as peak-picking [**11**], Viterbi decoding [**42**], or neural methods [**20**] have found them to be too noisy to be considered for evaluating the accuracy of harmonic or formant reconstruction. As well, no prior work has combined the efficacies of neural networks and Viterbi-based formant estimation. I propose using my pitch representation (Section 2.2) as F0 and performing Viterbi decoding on a high-resolution log magnitude spectrogram in bands and its harmonics are restricted to bands $(i+w)\times \mathrm{F0} < \mathrm{H}i < (i+1/w)\times \mathrm{F0}$ and $w = 4/5$ is tuned by visual inspection on training data to prevent octave errors. Note that after F0 inference, all harmonics can be decoded in parallel. My high-bandwidth batched Viterbi decoding implementation (Section 2.2.4) is well-suited for performing decoding of all harmonics in parallel.

My proposed harmonic estimation method (Figure 1.1; bottom) reconstructs F0 with an average error of 18.73 cents and H1 and H2 with an average error of 5.60 cents. The change in framewise spectral centroid has Pearson correlation of .853 with r_f, indicating strong, disentangled control of spectral balance. This method achieves lower pitch and harmonic reconstruction errors relative to other methods I tried that decode from LPC coefficients or a pitch posteriorgram. However, I have not performed extensive evaluation of my estimation method relative to prior methods—or thorough hyperparameter searches to further improve my proposed method. Therefore, I refrain from calling my harmonic estimation method state-of-the-art at this time. However, my method does demonstrate impressive visual agreement with harmonics shown on a spectrogram (Figure 1.1) and is sufficient to demonstrate that my proposed method maintains harmonic positions while varying the spectral balance.

Figure 4.7 overlays an example utterance in my proposed representation with my representation inferred from the corresponding speech synthesized using relative spectral balance parameters r_f that have been modified from the default value of $r_f = 1.0$ to $r_f = \sqrt{2}/2$ (**left**) and $r_f = \sqrt{2}$ (**right**). Focus on the SPPG, which demonstrates a clear change before and after editing. This effect is more clearly visualized in Figure 4.8, in which only the salient differences between SPPGs are provided. Certain phoneme pairs (e.g., "t" and "d", or "aa" and "ao") can be distinguished from one another by their relative amounts of high-frequency energy. Thus, spectral balance editing with $r_f < 1$ (**left**) should produce corresponding low-frequency phonemes, and $r_f > 1$ (**right**) should produce corresponding high-frequency phonemes. Figure 4.8 clearly demonstrates that my system exhibits this expected behavior. Note as well that there is a small amount of change in the A-weighted loudness in Figure 4.7. This is because the input A-weighted loudness corresponds to A-weighting of the ground truth frequency distribution during training. This indicates that perhaps an interpolation of the original A-weighted loudness should be used during training—as opposed to the A-weighted loudness of data augmented using my resampling augmentation. However, this can be easily compensated for using DSP-based gain scaling to perform A-weighted volume matching, as is used in my gain scaling data augmentation (Section 3.2.1).

4.5. Speaker adaptation

Speech content creators require speech editing systems that can be utilized with an arbitrary speaker. The process of fine-tuning a speech synthesis system on training data from a single speaker not seen during training to enable synthesis in the voice of that

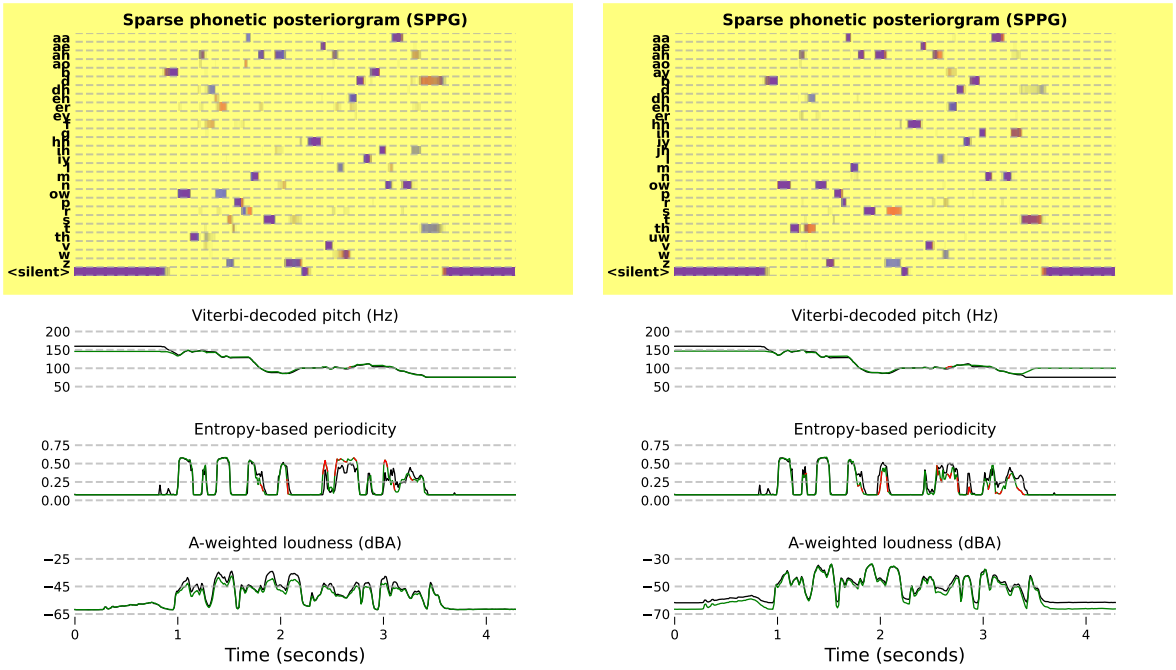

Figure 4.7. **Spectral balance editing example** | I (1) encode a speech utterance in my proposed representation (Chapter 2), (2) perform speech synthesis using the encoded representation with resampling augmentation factors (Section 3.2.1) of $r_f = \sqrt{2}/2$ (**left**) and $r_f = \sqrt{2}$ (**right**), (3) encode the synthesized speech in my representation, and (4) overlay my speech representation inferred from synthesized speech on the input representation to demonstrate accurate reconstruction while editing the spectral balance. My editing method produces corresponding changes in vowels in the inferred SPPG, such as predicting "ao" instead of "aa" (**right**) and "ay" instead of "eh" (**left**). To further demonstrate this behavior, Figure 4.8 zooms in on just the phonemes that were changed in the SPPG.
Blue SPPGs (**top**) are inferred from a recording of the source speaker, **red** SPPGs are inferred from speech synthesized with modified spectral balance, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
`figure-4-7-0047-000757-original.wav`
`figure-4-7-0047-000757-(r_f=0.71).wav`
`figure-4-7-0047-000757-(r_f=1.41).wav`

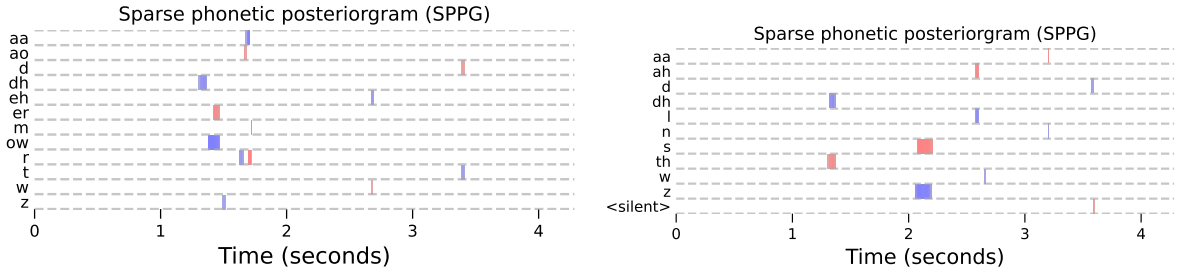

Figure 4.8. **Spectral balance editing produces corresponding changes in phoneme probabilities |** I visualize all frames in which the absolute difference between SPPGs inferred from ground truth audio and audio with my proposed spectral balance editing exceeds 35%, using the same audio as in Figure 4.7. **Blue** indicates where SPPGs inferred from original audio are assigned at least 35% more probability than SPPGs inferred from edited audio. **Red** indicates where SPPGs inferred from audio with spectral balance editing are assigned at least 35% more probability than SPPGs inferred from ground truth audio. On the **left** ($r_f = 0.71$), we see that "er" is replaced by "ow" (1.4 seconds), "aa" is replaced with "ao" (1.7 seconds), "eh" is replaced with "w" (2.8 seconds), and "t" is replaced with "d" (3.4 seconds). These replacements all correspond with less energy in the high-frequencies. On the **right** ($r_f = 1.41$), we see that "dh" is replaced with "th" (1.3 seconds), "z" is replaced with "s" (2.1 seconds), "l" is replaced with "ah" (2.7 seconds). These replacements all correspond with more energy in high frequencies. Interestingly, "s" (an unvoiced fricative) replaces "z" (the corresponding voiced fricative), but periodicity remains the same (Figure 4.7). This can be explained by the fact that English pronunciations of, e.g., pluralizations, are commonly "z", but are labeled as "s" in the lexically-derived PPG training data (Section 2.1.3.1).

speaker is called *speaker adaptation*. **In this section, I demonstrate that my proposed speech editing system is capable of performing speaker adaptation. I note that this is simply a proof of concept to demonstrate this useful capability, and does not reflect the state-of-the-art practices in speaker adaptation.** Very recent systems demonstrate the ability to completely forego speaker adaptation in place of high-fidelity discrete speaker embeddings [**41**] or few-shot learning [**33, 49**]. In future work, I aim to use such a method in combination with my proposed speech representation

Dataset	Δ¢ ↓	$\Delta\phi$ ↓	ΔdBA↓	ΔPPG↓
VCTK (multispeaker) [**119**]	**19.7**	**.068**	**1.48**	.144
DAPS (adaptation) [**82**]	26.7	.080	1.56	.197

Table 4.4. **Objective evaluation of speaker adaptation** | Speech editing accuracy of fine-tuning for 10,000 steps on 10 speakers (5 male; 5 female) speakers from the DAPS [**82**] dataset compared to multispeaker performance of the base model on held-out data from speaker seen during training. Reported results are averages over pitch-shifting (by ±600 cents), time-stretching (by factors $\sqrt{2}$ and $\sqrt{2}/2$), loudness edits (by ±5 dBA), and reconstruction. ↑ indicates that higher is better and ↓ indicates that lower is better.

(Chapter 2) to further improve speech editing accuracy and fidelity while omitting the need for speaker adaptation.

I perform speaker adaptation on the "clean" partition of the DAPS dataset.[2] I select five male and five female test speakers. I select ten utterances from each test speaker for a total of 100 test utterances. I require all test utterances be between four and ten seconds in length. During adaptation training and synthesis, I set the speaker index to zero and train only on speech recordings from a target speaker (and data-augmented variations 3.2.1) for 10,000 steps. I report the average objective metrics (Section 3.3.1) over a set of modifications: pitch-shifting (Section 4.1), time-stretching (Section 4.2), editing the timbral correlates of volume (Section 4.3), and reconstruction (Section 3.4). Note that this set of modifications does not fully capture all intraspeaker acoustic variability (e.g., it

[2]Attentive readers may note that speaker adaptation on a different dataset induces a confounding variable: the relative amount of noise in each dataset. My prior work on speech vocoding from Mel spectrograms [**75**] demonstrates that a vocoder without speaker conditioning trained on only VCTK produces equivalent or superior performance on DAPS relative to VCTK without any training or fine-tuning using DAPS. This is because the "clean" partition of DAPS that I use contains less noise than VCTK. The alternative of withholding speakers from VCTK for adaptation would reduce training data and negatively impact all of my other results in this dissertation. As mentioned at the start of Section 4.5, I am simply showing a proof-of-concept that my system can generalize to new speakers—an important consideration for real-world users.

does not take into account variations in formant position and energy [**54**]). However, existing alternatives for measuring speaker similarity within the speech synthesis literature are too sensitive to noise floor (i.e., speaker and noise are entangled) and any out-of-distribution prosody (e.g., pitch-shifting). This is further discussed in Section 3.3.1). Table 4.4 demonstrates that fine-tuning on individual speaker from DAPS does not yet reproduce the objective performance of multispeaker training on VCTK. However, existing research in speaker adaptation proposes methods that would likely address this [**103**] performance gap.

4.6. Voice conversion

The efficacy of representations similar to my proposed representation for voice conversion is well-studied [**125, 115**]; voice conversion is the task that brought renewed attention and interest to PPGs after their original utilization in query-by-example for speech databases. As well, the current state-of-the-art in voice conversion tends to rely on highly entangled discrete self-supervised methods trained on tens of thousands of hours of speech recordings [**33, 49**]. It is likely that such methods can be used in conjunction with my proposed representation; for example, converting between a discrete, self-supervised representation and my proposed representation (Section 5.1.1) or jointly learning residual time-varying features (Section 5.1.5). I leave these directions as future work. For now, I show examples of utilizing my proposed speech editing system to convert between male and female speakers to demonstrate a proof-of-concept that—as with all prior voice conversion methods using some variant of PPGs and acoustic features [**125**]—my speech editing system is capable of *any-to-many* voice conversion, wherein a speech recording of

an arbitrary speaker can be coverted to speech with the same (or similar) prosody and pronunciation but in the voice of a speaker utilized during speech synthesis training (Section 3.1). In the next section, I demonstrate that my system is also capable of speaker adaptation, which allows a pretrained speech synthesizer to be fine-tuned to synthesize speech in a new voice. Speaker adaptation combined with any-to-many voice conversion produces *any-to-any* voice conversion, in which the prosody and pronunciation of any input speech can be transferred to any target speaker. Figure 4.9 demonstrates *many-to-many* voice conversion, wherein a held-out speech recording from a speaker utilized during training is encoded in my proposed representation and used as input to synthesize speech in the voice of another speaker utilized during training. I modify the using the mean and standard deviation of the source and target speakers to ensure that the pitch used as input is (with high likelihood) within the distribution of the target speaker. I provide examples of converting from a male to a female voice (**left**) and vice versa (**right**). Successful voice conversion should produce maximal overlap between all features. This is true for all features except periodicity, which does deviate slightly. This is due to entanglement between speaker identity and background noise: if a certain speaker recorded in a noisier environment, the average periodicity for that speaker will be lower. Prior work [**29**] addresses this by utilizing a data augmentation method similar to my general augmentation framework (Section 3.2.1), but specifically for adding parameterized noise to the data to reduce the mutual information between speaker and noise. This also provides a global control over the amount of noise in the generated speech.

Figure 4.11 demonstrates any-to-many voice conversion, in which a speech utterance spoken by my co-author is used to perform voice conversion in the voice of a speaker

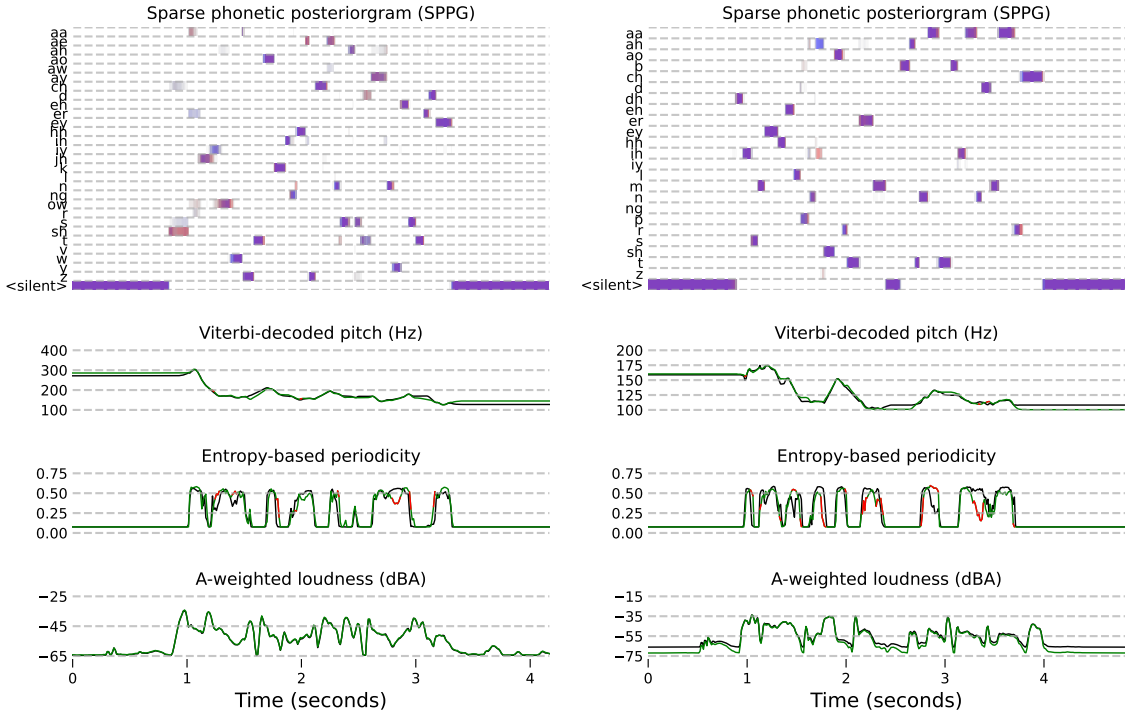

Figure 4.9. **Voice conversion example** | I (1) encode speech utterances from a male (left) and a female (right) source speaker in my representation (Chapter 2), (2) shift the pitch by the difference of the mean base-two log-F0 in voiced regions between the source speaker and a target speaker of the opposite gender, (3) perform speech synthesis using the target speaker index and the source speaker representation (with mean-corrected pitch) to synthesize the source speech content using the speaker embedding associated with the target speaker, (4) encode the synthesized speech in my representation, and (5) overlay my representation inferred from synthesized speech on the input representation to demonstrate accurate reconstruction during voice conversion.
Blue SPPGs (**top**) are inferred from a recording of the source speaker, **red** SPPGs are inferred from speech synthesized in the voice of the target speaker, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
`figure-4-9-0073-000053-0082-000741-*.wav`[3]
`figure-4-9-0108-000684-0032-000686-*.wav`

utilized during training as well as jointly performing voice conversion with novel speech editing capabilities of fine-grained phoneme interpolation (middle) or manual phoneme editing (right). I further discuss these pronunciation editing capabilities in Section 4.7.

My system is also capable of reconstructing or transferring some aspects of voice quality (VQ), such as breathy or creaky voice. My representation was not designed with these aspects in mind; future work may refine my representation for VQ control. However, the information extracted via my representation is sufficient to inform the neural vocoder of VQ attributes. Figure 4.10 provides two examples: a man yawning (i.e., breathy voice; left) and a woman sustaining an /ah/ with and without vocal fry (right). The example on the right also indicates that a sustained 50 Hz hum in the training dataset causes sustained pitches to have an increased probability of silence.

4.7. Editing pronunciation

While prior works have demonstrated that PPGs enable conversion between accents [**124**], my work is the first to demonstrate that interpretable PPGs permit interpretable, fine-grained user control of speech pronunciation. Prior to my work, no previous speech editing system of any kind had demonstrated any type of fine-grained pronunciation control. In Figure 4.11, I visualize my proposed speech representation before and after performing voice conversion with my proposed speech editing system (**left**), as well as jointly performing voice conversion and phoneme interpolation (**center**) or manual phoneme editing (**right**). I use spherical linear interpolation (SLERP) [**98**] for interpolating PPGs. As described in Section 2.1.4, I use as pronunciation reconstruction error the JS divergence

[3]Includes the `source` utterance, an example `target` speaker utterance, and voice conversion results using my `proposed` system.

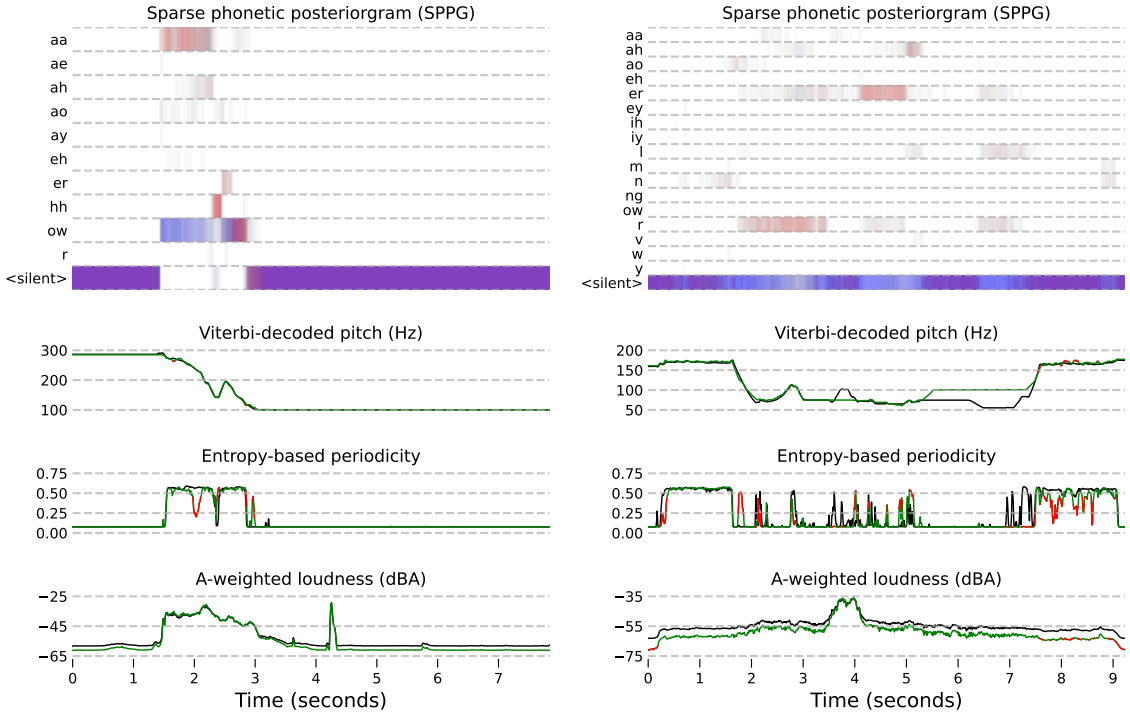

Figure 4.10. **Voice quality conversion example |** I (1) encode speech utterances from a male yawning (left) and a female sustaining an /ah/ sound with and without vocal fry (right) in my representation (Chapter 2), (2) perform speech synthesis using the speaker embedding associated with a target speaker seen during training, (4) encode the synthesized speech in my representation, and (5) overlay my representation inferred from synthesized speech on the input representation.
Blue SPPGs (**top**) are inferred from a recording of the source speaker, **red** SPPGs are inferred from speech synthesized in the voice of the target speaker, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
`figure-4-10-0032-000780-*.wav`
`figure-4-10-0013-000531-*.wav`

between the input, interpolated PPG and the corresponding PPG inferred from the generated audio (Figure 4.11; **top**).

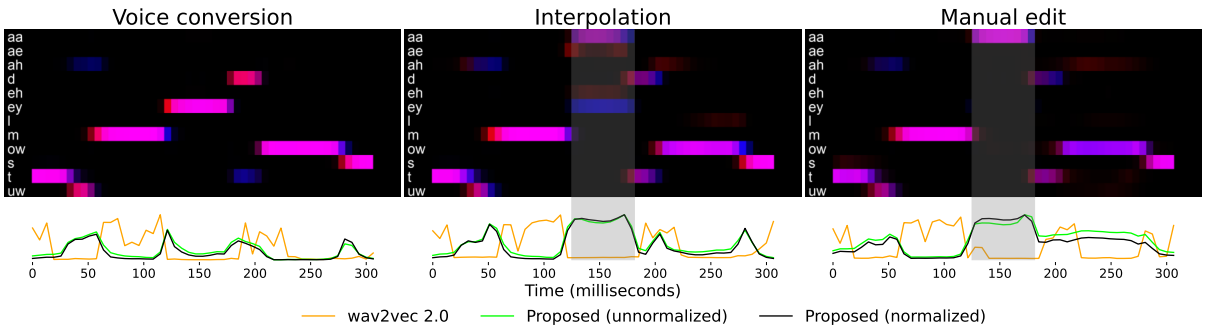

Figure 4.11. **Pronunciation interpolation and distance** | Examples of using my proposed speech representation (Chapter 2) for **(left)** voice conversion, **(center)** pronunciation interpolation, and **(right)** manual phoneme editing. **(top)** I visualize overlapping PPGs of a recording of the word "tomato" (**blue**) and inferred from the synthesized speech (**red**). For readability, phoneme rows in the PPGs with maximum probability $< 10\%$ are omitted. The accurate reconstruction of PPGs (**magenta**) indicates preservation of (potentially edited) phonetic content in the generated speech. In the center, the input (**blue**) PPG is interpolated only within the (**gray**) edit region to be halfway (i.e., 50%) between the /eh/ in the left PPG and the /aa/ in the right PPG using SLERP [**98**]. Note that the reconstruction of interpolating "ey" (**left**) and "aa" (**right**) is "ae" or "eh" (**center**). This is consistent with interpolating vowels in formant space (F1, F2 - F1) [**47**] and indicates that one pronunciation can be represented more than one way in a PPG. **(bottom)** Pronunciation distances between synthesized speech and the original audio. My proposed distance (Section 2.1.4) is more robust to resynthesis artifacts and accurately captures pronunciation interpolation without a transcript.

`figure-4-11-source.wav`
`figure-4-11-conversion.wav`
`figure-4-11-interpolation.wav`
`figure-4-11-manual.wav`

Fine-grained pronunciation editing is also a useful task for further inspecting the behavior of my proposed pronunciation distance (ΔPPG; Section 2.1.4) to capture frame-level pronunciation differences during pronunciation editing. I use as a baseline the speech distance proposed by Bartelds et al. [**8**], which uses the L2 distance between wav2vec 2.0 latents and outperforms spectral-based and transcript-based speech variation distances [**8**].

My audio is already aligned, so I omit the dynamic time warping (DTW) step used to perform alignment. Figure 4.11 (**bottom**) demonstrates the behavior of each pronunciation distance during voice conversion (**left**), pronunciation interpolation (**center**), and manual pronunciation editing (**right**) relative to the original pronunciation (and not the PPGs used as input; in that case, no corresponding wav2vec 2.0 distance exists for comparison). Focus on the edit region around 150 milliseconds (i.e., the transparent **gray** rectangle). When no edit to the PPG is performed, (**left**), all the measures of pronunciation difference indicate a low difference. When edits were done to the PPGs (**middle** and **right** panels), only my proposed distance increases to indicate a high pronunciation distance, while the wav2vec 2.0 baseline remains low. Wav2vec 2.0 further assigns high pronunciation distance to unedited regions, such as the phoneme "m" that occurs between 50 and 120 milliseconds. Wav2vec 2.0 [**7**] fails to detect clear pronunciation differences captured by my proposed, interpretable pronunciation distance based on the JS-divergence between PPGs.

In my prior work [**15**], **I exemplify further editing capabilities that can be built utilizing my proposed fine-grained pronunciation editing: (1) interpretable, regex-based accent conversion and (2) automatic onomatopoeia.** In (1), an accent is encoded as a set of sequential substitution rules that map monophone, diphone, triphone, etc. sequences in the source accent to corresponding phoneme sequences in the target accent. Then, a speech recording is encoded in my proposed pronunciation and the set of phoneme substitution rules is applied to the SPPG, where substitutions are implemented by swapping and reallocating the inferred probabilities of corresponding phonemes. The challenge with this method is the need for either an expert to hand-craft a

general rule set for conversion between two accents or an automated method for learning a rule set between two accents. However, the benefit is interpretability: speech accents are the product of a shared cultural background, and an interpretable representation of pronunciation enables accent conversion with a high degree of transparency that provides an improved means for verification by experts. I currently create a separate set of sequential substitution rules catered to each utterance.

Figure 4.12 demonstrates an example, wherein I manually edit the South African (Cape Town) accent of a female speaker to align with a American Midwestern accent using my proposed, interpretable regex-based accent conversion. I provide my manually-specified sequential substitution rules in the figure caption. Focus on the SPPGs, which overlay the edited SPPGs (**blue**) and SPPGs inferred from accent-edited speech (**red**). **Violet** indicates successful editing of the pronunciation. Interestingly, I find that—while the synthesized audio does sound significantly closer to the target accent—the SPPG inferred from accent-edited speech is a mix between the original and target accent. This indicates that the speech synthesis model has—to some extent—learned speaker-specific transition probabilities between phonemes. This can be addressed via few-shot learning on a significantly larger dataset with many more speakers—which also removes the need for speaker adaptation (Section 4.5). I further discuss this direction for future work in Section 5.1.5.

In Figure 4.13, I provide an example of performing automatic onomatopoeia, in which a non-speech source sound is encoded in my proposed representation and synthesis is performed in the voice of a target speaker. This could be used, e.g., to produce large datasets of vocal imitations suitable for improving query-by-voice systems [**122, 76**], in

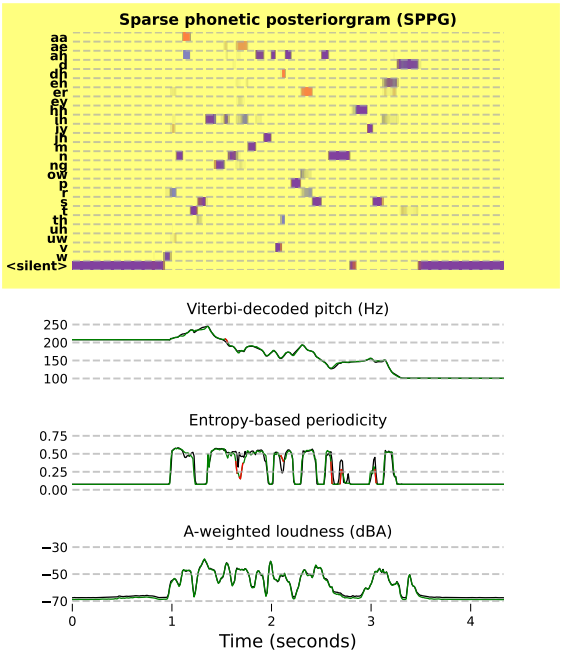

Figure 4.12. **Accent conversion example |** I (1) encode an example speech utterance in my proposed representation (Chapter 2), (2) perform my proposed regex-based accent conversion to manually convert from the original, South African accent to an American Midwestern accent, (3) perform speech synthesis to produce accent-edited speech, (4) encode the accent-edited speech in my representation, and (6) overlay the speech representation inferred from synthesized speech on the input representation to demonstrate reconstruction of both prosody and target pronunciation during accent conversion. My sequential rule set for this example is as follows: (1) `reallocate`([“dh”, “ah”], [“th”, “ah”]), (2) `reallocate`([“n”, “aa”, “t”], [“n”, “ah”, “t”]), (3) `reallocate`(“er”, “r”), (4) `reallocate`(“ae”, “eh”), where `reallocate`(b, c) reallocates all probability in regex matches for phoneme sequence b to corresponding phonemes in phoneme sequence c.

Blue SPPGs (**top**) are inputs, **red** SPPGs are inferred from accent-edited speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.

`figure-4-12-0082-000322-original.wav`

`figure-4-12-0082-000322-midwestern.wav`

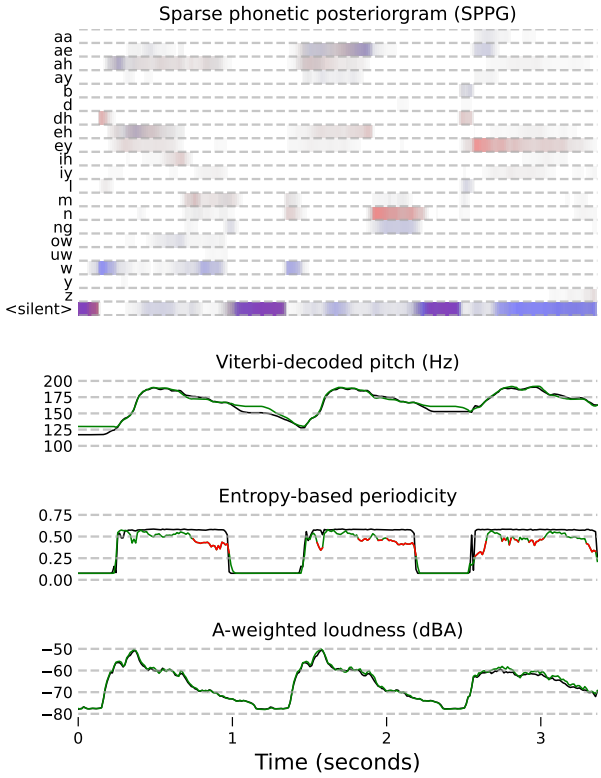

Figure 4.13. **Automatic onomatopoeia example** | I (1) encode an example recording of cat vocalizations in my proposed representation (Chapter 2), (2) perform speech synthesis using the speaker index of a target speaker to produce a vocal imitation, (3) encode the synthesized vocal imitation in my representation, and (4) overlay the speech representation inferred from synthesized speech on the input representation to demonstrate reconstruction of my speech representation during vocal imitation.
Blue SPPGs (**top**) are inputs, **red** SPPGs are inferred from synthesized speech, and **violet** indicates accurate reconstruction. For pitch, periodicity, and A-weighted loudness, the features inferred from synthesized speech are shown in **black**, while input features are **green** when the inferred feature is within an error threshold and **red** when outside that threshold. I use a threshold of 50 cents for pitch, 0.1 for periodicity, and 6 dBA for volume.
`figure-4-13-source.wav`
`figure-4-13-0082-000322-target.wav`
`figure-4-13-0082-000322-imitation.wav`

which a user searches a large database of audio recordings via vocal imitation of the desired sound.

4.8. Ablations

I demonstrate that design choices presented throughout my dissertation improve speech editing accuracy via ablations. I report the average objective metrics (Section 3.3.1) over a set of modifications: pitch-shifting (Section 4.1), time-stretching (Section 4.2), editing the timbral correlates of volume (Section 4.3), and reconstruction (Section 3.4).

Table 4.5 (**Ablations**) shows that each of my design decisions contributes to the efficacy of my proposed system, with Viterbi-decoded pitch (Section 2.2) and multi-band A-weighted loudness (Section 2.4) being particularly impactful. The most telling sign of the efficacy of my design decisions is the concurrent improvement of objective metrics not immediately impacted by the design decision. For example, Viterbi-decoded pitch and multi-band loudness improve all metrics—not just pitch metrics or loudness metrics, respectively. This is a strong indicator that my methods are improving speech synthesis and editing rather than just providing a cleaner representation. This also validates my hypotheses about the existing presence of entanglement between features, as well as that noise in unvoiced pitch frames is not being ignored during training: unvoiced pitch frames must be carefully handled to prevent overfitting. Likewise, SPPGs remove noise in low-probability phoneme categories and improve all metrics—not just pronunciation metrics.

While my proposed data augmentation and variable-width bins do not improve all metrics, they both improve pitch and loudness accuracies—and my data augmentation demonstrates further utility in enabling and improving the editing of both the timbral correlates of volume (Section 4.3) and spectral balance (Section 4.4). The slight drop in PPG performance caused by my data augmentation can be explained due to changes in spectral balance and loudness induced by data augmentation. For example, vowels

are distinguished by variations in relative formant energies and positions, and the "silent" phoneme category indicates audio below a certain threshold. This induces a slightly harder learning problem, in which the model must learn the conditional distribution of the audio given slightly entangled SPPG and augmentation parameters r_f and r_l (Section 3.2.1)—as opposed to only the SPPG. In practice, my model seems highly capable of learning this more challenging conditional distribution, and any slight entanglement is offset by improvements in other objective metrics as well as novel editing capabilities.

One additional advantage of my proposed variable-width pitch bins is training stability: while my speech synthesis model can successfully train to convergence with or without my proposed variable-width pitch bins when using 256 bins, increasing the bin count to 512 induces divergence only when not using my variable-width pitch bins due to the increase in the number of infrequently used bins. While one could add more data to attempt to increase the utilization of infrequently used bins, the pitch distribution of natural speech is inherently bimodal (Figure 3.1): increasing utilization of infrequently used bins by adding more data also increases the support of the distribution, requiring either new, infrequently used bins or truncation of the pitch distribution. My proposed variable-width pitch bins addresses these issues, making them well-suited for concurrently scaling up the amount of training data and number of pitch bins in future work.

Comparing to objective results of only performing speech reconstruction (Table 3.1), my proposed representation and design decisions demonstrate a diverse set of speech synthesis and editing capabilities that are approaching the comparable objective metrics to speech reconstruction. The difference between objective metrics for speech editing and objective metrics for speech reconstruction quantifies the remaining amount of entanglement

	Section	Δ¢ ↓	$\Delta\phi$ ↓	ΔdBA↓	ΔPPG↓
Proposed	–	**19.7**	**.068**	**1.48**	.144
w/o SPPG	2.1.2	19.9	.070	1.62	.149
w/o Viterbi decoding	2.2.4	31.9	.071	1.64	.147
w/o multi-band loudness	2.4	23.5	.074	1.96	.161
w/o augmentation	3.2.1	23.6	**.068**	1.69	**.143**
w/o variable-width bins	3.2.2	20.7	**.068**	1.61	.144
w/o all (cumulative)	–	39.2	.075	1.88	.166

Table 4.5. **Objective evaluation of ablations** | Non-cumulative ablations of the speech editing accuracy of methods proposed throughout my dissertation as well as the section in which they are described. Reported results are averages over pitch-shifting (by ±600 cents), time-stretching (by factors $\sqrt{2}$ and $\sqrt{2}/2$), loudness edits (by ±5 dBA), and reconstruction. ↑ indicates that higher is better and ↓ indicates that lower is better.

in my proposed representation. Clearly, my design decisions have significantly reduced entanglement; but a small amount of entanglement remains. I hypothesize this remaining entanglement can be addressed by scaling up the data and model capacity, improving the quality of PPGs, performing joint self-supervised discrete representation learning of residual features, and investigating multi-band periodicity.

CHAPTER 5

Conclusion

Content creators of film, video games, podcasts, and social media necessitate a diverse set of high-fidelity tools to analyze, enhance, tune, and otherwise modify their speech content according to their creative and technical vision. In this dissertation, I have described my research on the topic of speech representation and editing and my resulting speech editing system that addresses needs of these content creators. In Chapter 2, I propose a novel speech representation that addresses the need for an interpretable representation of speech suitable for disentangling and visualizing the speech attributes of interest. In Chapter 3, I describe how off-the-shelf neural text-to-speech and vocoding systems can be utilized in conjunction with my proposed representation to perform high-fidelity neural speech synthesis. In Chapter 4, I demonstrate new or improved speech editing capabilities made possible by my proposed speech representation and speech editing system. Together, my contributions set the foundations for speech editing software that expedited and advances the workflows of speech content creators responsible for the development of many of the primary means of communication (e.g., film, podcasts, and social media) used today. The remainder of this chapter describes novel research directions and speech technologies made possible by my research contributions (Section 5.1), as well as steps I have taken to ensure the reproducibility of all experimental results presented in my dissertation (Section 5.3) and considerations for ethical use of the technology I develop. I have been consistently delighted by the creative and technical utilizations of my proposed

speech technologies by research labs, industry labs, and individuals. I look forward to continuing to advance the technologies I have contributed and further uncovering their potential.

5.1. Future work

My contributions presented in this dissertation advance the state-of-the-art of neural speech editing. They also pave the way for further contributions both within the domain of neural speech editing as well as other research domains. In this section, I discuss avenues for future work relevant to my contributions.

5.1.1. Text-to-speech and hierarchical speech editing

Unlike the Mel spectrogram or self-supervised learning (SSL) representations such as wav2vec 2.0 [**7, 101**] or EnCodec [**18**], my proposed representation (Chapter 2) has not yet been used as an intermediary within a text-to-speech (TTS) system. This would create a two-tiered editing hierarchy in which speech content creators can edit speech by either changing the words in the speech transcript or performing any of the edits afforded by my system. Two options exist for incorporating my representation in a TTS system: (1) jointly train a representation generator and a duration predictor that generate speech in my proposed representation from lexical inputs or (2) train two "conversion" neural networks that convert my proposed representation to a state-of-the-art SSL representation and vice versa. It is not yet clear to me which will produce superior results.

Consider a TTS system that utilizes my proposed representation as an intermediary between text and speech audio. Such a system takes lexical inputs at, e.g., the phoneme

or word resolution. This makes such systems amenable to word-level conditioning and control. Provided an alignment with the lexical features, my proposed representation can be used to perform phoneme- or word-resolution editing of pitch, duration, loudness, and pronunciation. However, there are additional word- and phoneme-level attributes that are either not captured by this factorization (e.g., vocal fry or falsetto) or are entangled abstractions of one or more of these attributes (e.g., prominence). I have developed a streamlined system for producing the crowdsourced or automatic annotations of such word- and phoneme-level attributes that are prerequisite to fine-grained control of, e.g., prominence, vocal fry, falsetto, and mispronunciations [**77**]. More generally, my representation can be thought of a low-level, fine-grained intermediary representation on top of which more abstract representations of speech and speech editing capabilities can be built. This provides the advantage of enabling an interpretable analysis of abstract speech representations. For example, I can increase the prominence of a random word in each utterance in a speech dataset, measure the average impact on prosodic features, and compare this to the prosodic variations of speech with ground-truth human prominence annotations to evaluate the efficacy of prominence control.

5.1.2. User interfaces for speech representation and editing

While my work develops the underlying technology for a speech editing system, my command-line interfaces and Python application programming interfaces are not the most efficient or intuitive workflow for most creative professionals. However, because my proposed technology introduces fundamentally new ways to edit speech, the optimal graphical user interface is not yet known. For example, when editing pronunciation, do users prefer

mimicking the desired pronunciation, drawing on a PPG, or a mix of both? Does such an interface offer blind audio engineers a simpler experience in navigating or editing speech or music? What is a more intuitive phoneme set for a PPG, the international phonetic alphabet (IPA) or CMU pronunciation dictionary? Does this depend on the user (e.g., voice actors or clinical speech pathologists)? Many such design considerations can be explored to further the utility of this technology in the hands of creative and medical professionals.

5.1.3. Real-time accent and pronunciation coaching

Voice actors commonly undergo training to learn to mimic accents indicative of particular regions, times, ethnicities, or social classes. Emigrating L2 language learners often strive to adopt the native accent to facilitate intelligibility. Patients with speech and language disorders undergo significant physical therapy to rehabilitate the vocal mechanism and corresponding neurological pathways. Accurate representations of the pronunciation of speech can help these individuals towards their goal of speaking confidently. Figure 4.11 (top) demonstrates two PPGs overlaid on top of one another, showing the differences between two pronunciations of the same speech. One could imagine one of these pronunciations being a randomly selected speech utterance spoken by a reference speaker, while the other is an attempt by the trainee to replicate the pronunciation. Using dynamic time warping to align the pronunciations, it is possible to both visualize pronunciation errors and produce a pronunciation distance score. In other words, it is now possible to build a "Guitar Hero" for pronunciation training for voice actors, singers, language learners, and patients with speech and language disorders. Technical challenges for producing such

an application include high-fidelity, causal, low-latency SPPG inference suitable for real-time streaming and a robust method for performing online dynamic time warping [**55**] of partial sequences [**12**].

5.1.4. Query-by-onomatopoeia

My proposed PPG representation enables automatic onomatopoeia, in which an arbitrary audio recording can be rendered in a person's voice, with phonemes and pitch computed from the arbitrary recording using my proposed PPG (Section 2.1) and pitch (Section 2.2) estimators. For example, this can be used to have a human mimic the sound of a cat or a car alarm. This can be used to acquire data for the inverse problem, in which a human mimics a sound with their voice and a computer retrieves corresponding real sounds. Perhaps more generally, PPGs offer an interpretable way to encode and transfer non-written languages, such as, e.g., animal vocalizations. Thus, such representations may be useful not only for transferring animal sounds to human speech, but also representing and further understanding animal language. This is potentially a useful tool for sound designers of film, video games, and music, who need intuitive methods to traverse large sound libraries to select an appropriate audio recording for a given segment of film or music or a particular event in a video game.

5.1.5. Scaling up for performance and few-shot generalization

One current drawback of my system is the need for a few minutes of audio from the target speaker. Prior research demonstrates that high-fidelity generalization to new speakers is possible with a minute of audio or less [**49**]. However, high-fidelity speech synthesis models

that generalize to new speakers given limited context are typically the result of discrete, self-supervised speech representations trained using large servers of recent GPU models on tens of thousands of hours of speech data for days to weeks. Larger datasets also typically increase model performance across all metrics—especially when, e.g., the model capacity is also appropriately scaled. Provided access to more compute resources, this is a viable direction for future work.

5.1.6. Novel use cases for Viterbi decoding

My GPU implementation of Viterbi decoding (Section 2.2.4) is fast enough to enable novel applications. For example, non-autoregressive speech synthesizers such as HiFi-GAN [**40**] infer single parameters for each sample, which means each sample is being modeled by a univariate mean of a normal distribution. However, the conditional distribution of the audio waveform given input acoustic features is multi-modal due to stochasticity in unvoiced regions, unspecified starting phases at the beginning of each voiced region, and background noise; the success of HiFi-GAN is only due to the high mutual information (low conditional entropy) of the input representation (e.g., Mels or our proposed representation) relative to the speech waveform and the mode-selecting capabilities of GANs. My fast Viterbi decoding implementation makes it feasible to train a multivariate model for each sample and only optimize along the optimal path as determined by Viterbi decoding. For example, the generator could predict a categorical distribution over quantized waveform sample values, Viterbi decoding could be used to decode the generated waveform as the optimal path, and the optimal waveform could be passed into a discriminator to

further leverage the efficacies of adversarial losses. Whether or not this improved performance, this would allow us to visualize the joint distribution of a non-autoregressive sample-resolution generative model to, e.g., explicitly visualize the multi-modality induced by ambiguity in starting phase.

In addition, Viterbi decoding could be applied to my SPPG representation (Section 2.1) to perform transcript-free phoneme alignment—or combined with a transcript to further improve phoneme alignment performance.

5.1.7. Interpretable speech coding

Speech coding is the task of compressing the storage requirements of a speech signal for efficient transmission. Recent state-of-the-art methods for speech and audio coding utilize two deep neural networks: one (the *encoder*) that learns a compressed representation of the audio and another (the *decoder*) that learns to invert the representation and reconstruct the audio waveform. The current state-of-the-art audio coding method is the Descript audio codec (DAC) [**45**], which compresses 16-bit 22.05 kHz audio at a compression rate of approximately 45x (1 kbps).

My proposed representation currently encodes speech using 50 16-bit floating point numbers per frame, resulting in a compression rate of approximately 5.0x (68.75 kbps). However, no effort has been made to efficiently encode the sparsity in the SPPGs (Section 2.1.2). I predict an average of less than five phonemes need to be represented at each frame, which would produce a compression rate of approximately 15.7x (22.44 kbps). As well, the number of bands in the multi-band loudness can be tuned based on storage

requirements (at the cost of some performance). Finally, variable-rate temporal compression [**19**] can be used to compress the pitch, periodicity, and loudness. This would produce a speech coding representation that not only significantly outperforms the current state-of-the-art audio coding representation, but is interpretable, editable, and invariant to sampling rate. To see that my representation is invariant to sampling rate, note that SPPGs (Section 2.1) are computed at a sampling rate of 16 kHz, pitch (Section 2.2) and periodicity (Section 2.3) are computed at 8 kHz, A-weighted loudness (Section 2.4) is computed at 22.05 kHz, and my speech synthesizer currently produces speech with a 22.05 kHz sampling rate; the same techniques used to interpolate my representation during time-stretching (Section 4.2) are also used to align the features of my speech representation prior to training or synthesis. This can be used to synthesize high-fidelity audio from low-fidelity audio features, or to allow a client receiving encoded speech to choose playback fidelity (i.e., the speech vocoder) best suited for their device's capabilities. This also allows the flexibility of either (1) incurring an O(1) cost of sending speaker-adapted (Section 4.5) model weights corresponding to a target speaker or (2) not sending speaker-related features at all, which could be used for low-bandwidth, high-fidelity, real-time speaker anonymization if combined with the causal SPPGs also required for real-time accent and pronunciation training using my proposed representation (Section 5.1.3).

5.2. Ethics statement

Speech editing should require the informed consent of the person's voice being edited. By informed consent, I refer to three factors: (1) consent that one's voice is to be edited, (2) consent for each of the specific types of edits being performed, and (3) consent to the

manner of distribution of the edited speech content. Consent for each specific type of edit (2) addresses the concerns of voice actors who may be comfortable with, e.g., some fine-tuning of prosody or pronunciation for intelligibility, emphasis, or contextual naturalness, but may object to the editing of their voice to contain certain emotions or words (e.g., profanity), or may outright object to any edits to the lexical speech content and allow edits to only non-lexical speech attributes. Consent to the manner of distribution (3) addresses the placement of consensually edited speech in inappropriate contexts that may alter listeners' perceptions.

Enforcing that speech editing technology is utilized for only consensual voice editing requires both regulatory measures as well as methods for detecting edited speech. Given the rapid advancement of the quality of generative machine learning, I believe it is inevitable that the differences between real and edited speech will be imperceptible to both humans and automated detection methods. Instead, regulatory measures should require speech editing software to perform multiple safeguards to protect against bad actors with varying levels of technical expertise. For example, a naïve bad actor unfamiliar with file metadata could be caught simply by encoding metadata that says what software was used for editing and when. More experienced bad actors could be detected using content authenticity measures such as *watermarking*, wherein edited speech is rendered with an inaudible fingerprint that can be used to recover what software was used for editing and when [**86**]. The Content Authenticity Initiative [**1**] demonstrates another method that utilizes blockchain to trace the set of transactions originating from a recording device that does not permit editing (e.g., a camera). Actualizing the societal value of this technology requires collaboration between academia, industry, and government.

5.3. Reproducibility

Reproducibility is imperative for efficient and reliable research progress. To that end, all of the code and model weights produced in this work are not only open-source and documented, but have been packaged and distributed via PyPi to be `pip`-installable. Each of these repositories contains all of the relevant code and configurations to reproduce all experimental results presented Information on the relevant code libraries is provided in Table 5.1.

Name	Description	URL
`penn` [**72**]	Pitch and periodicity estimation (Sections 2.2-2.3)	`github.com/interactiveaudiolab/penn`
`ppgs` [**15**]	Sparse phonetic posteriorgram inference (Section 2.1)	`github.com/interactiveaudiolab/ppgs`
`promonet` [**71**]	Speech prosody and pronunciation editing (Chapters 3-4)	`github.com/maxrmorrison/promonet`
`reseval` [**79**]	Reproducible subjective evaluation (Section 3.3.2)	`github.com/reseval/reseval`

Table 5.1. Open-source, `pip`-installable code repositories containing my work described in my dissertation. Does not include my support code libraries (e.g., `torchutil`), my libraries containing baseline models (e.g., `torchcrepe` or `psola`), or my fast Viterbi decoding implementation (`torbi`; Section 2.2.4), which I plan to release within PyTorch [**87**] instead of distributing via PyPi.

References

[1] Adobe. Content authenticity initiative. `https://contentauthenticity.org/`, 2024.

[2] Ardaillon, L., and Roebel, A. Fully-convolutional network for pitch estimation of speech signals. In *Interspeech 2019* (2019).

[3] Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., and Weber, G. Common Voice: A massively-multilingual speech corpus. In *International Conference on Language Resources and Evaluation* (2020).

[4] Arik, S. Ö., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Ng, A., Raiman, J., et al. Deep Voice: Real-time neural text-to-speech. In *International Conference on Machine Learning* (2017).

[5] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *Neural Information Processing Systems* (2016).

[6] Bae, H., and Joo, Y.-S. Enhancement of pitch controllability using timbre-preserving pitch augmentation in FastPitch. *Interspeech* (2022).

[7] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Neural Information Processing Systems* (2020).

[8] Bartelds, M., de Vries, W., Sanal, F., Richter, C., Liberman, M., and Wieling, M. Neural representations for modeling variation in speech. *Journal of Phonetics* (2022).

[9] Bittner, R. M., Salamon, J., Tierney, M., Mauch, M., Cannam, C., and Bello, J. P. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *International Society for Music Information Retrieval* (2014).

[10] Boersma, P., and Van Heuven, V. Speak and unspeak with praat. *Glot International 5*, 9/10 (2001), 341–347.

[11] Broad, D. J., and Clermont, F. Formant estimation by linear transformation of the LPC cepstrum. *Journal of the Acoustical Society of America 86*, 5 (1989), 2013–2017.

[12] Bukey, I., Zhang, J., and Tsai, T. FlexDTW: Dynamic time warping with flexible boundary conditions. In *International Society for Music Information Retrieval* (2023).

[13] Cao, H., Beňuš, Š., Gur, R. C., Verma, R., and Nenkova, A. Prosodic cues for emotion: analysis with discrete characterization of intonation. *Speech prosody (Urbana, Ill.) 2014* (2014), 130.

[14] Choi, H.-S., Yang, J., Lee, J., and Kim, H. Nansy++: Unified voice synthesis with neural analysis and synthesis. *International Conference on Learning Representations* (2023).

[15] Churchwell, C., Morrison, M., and Pardo, B. High-fidelity neural phonetic posteriorgrams. In *ICASSP Workshop on Explainable Machine Learning for Speech and Audio* (2024).

[16] Cole, J., Hualde, J. I., Smith, C. L., Eager, C., Mahrt, T., and Napoleão de Souza, R. Sound, structure and meaning: The bases of prominence ratings in english, french and spanish. *Journal of Phonetics* (2019).

[17] Company, T. B. R. Global market report. `https://www.thebusinessresearchcompany.com/global-market-reports`, 2024.

[18] Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. High fidelity neural audio compression. *Transactions on Machine Learning Research* (2023).

[19] Dieleman, S., Nash, C., Engel, J., and Simonyan, K. Variable-rate discrete representation learning. *arXiv preprint arXiv:2103.06089* (2021).

[20] Dissen, Y., Goldberger, J., and Keshet, J. Formant estimation and tracking: A deep learning approach. *Journal of the Acoustical Society of America 145*, 2 (2019), 642–653.

[21] Frick, R. W. Communicating emotion: The role of prosodic features. *Psychological bulletin 97*, 3 (1985), 412.

[22] Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon Technical Report* (1993).

[23] Gfeller, B., Frank, C., Roblek, D., Sharifi, M., Tagliasacchi, M., and Velimirović, M. SPICE: Self-supervised pitch estimation. *Transactions on Audio, Speech, and Language Processing* (2020).

[24] Gobl, C., and Chasaide, A. N. The role of voice quality in communicating emotion, mood and attitude. *Speech Communication 40*, 1-2 (2003), 189–212.

[25] Gonzalez, P., Alstrøm, T. S., and May, T. On batching variable size inputs for training end-to-end speech enhancement systems. In *International Conference on Acoustics, Speech and Signal Processing* (2023).

[26] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM* (2020).

[27] Hartmann, W. M. *Signals, sound, and sensation*. Springer Science & Business Media, 2004.

[28] Hazen, T. J., Shen, W., and White, C. Query-by-example spoken term detection using phonetic posteriorgram templates. In *IEEE Workshop on Automatic Speech Recognition & Understanding* (2009).

[29] Hsu, W.-N., Zhang, Y., Weiss, R. J., Chung, Y.-A., Wang, Y., Wu, Y., and Glass, J. Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization. In *International Conference on Acoustics, Speech and Signal Processing* (2019).

[30] International Telecommunication Union. Method for the subjective assessment of intermediate sound quality, 2001.

[31] International Telecommunication Union. Perceptual evaluation of speech quality (PESQ), 2001.

[32] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (2015).

[33] Ju, Z., Wang, Y., Shen, K., Tan, X., Xin, D., Yang, D., Liu, Y., Leng, Y., Song, K., Tang, S., Wu, Z., Qin, T., Li, X.-Y., Ye, W., Zhang, S., Bian,

J., He, L., Li, J., and Zhao, S. NaturalSpeech 3: Zero-shot speech synthesis with factorized codec and diffusion models. *arXiv preprint arXiv:2403.03100* (2024).

[34] Kim, J., Kong, J., and Son, J. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning* (2021).

[35] Kim, J. W., Salamon, J., Li, P., and Bello, J. P. Crepe: A convolutional representation for pitch estimation. In *International Conference on Acoustics, Speech and Signal Processing* (2018).

[36] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations* (2015).

[37] Klatt, D. H. Discrimination of fundamental frequency contours in synthetic speech: implications for models of pitch perception. *The Journal of the Acoustical Society of America 53*, 1 (1973), 8–16.

[38] Klatt, D. H., and Klatt, L. C. Analysis, synthesis, and perception of voice quality variations among female and male talkers. *The Journal of the Acoustical Society of America 87*, 2 (1990), 820–857.

[39] Kominek, J., and Black, A. The CMU Arctic speech databases. In *ISCA Workshop on Speech Synthesis* (2004).

[40] Kong, J., Kim, J., and Bae, J. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Neural Information Processing Systems* (2020).

[41] Kong, J., Lee, J., Kim, J., Kim, B., Park, J., Kong, D., Lee, C., and Kim, S. Encoding speaker-specific latent speech feature for speech synthesis. *arXiv preprint arXiv:2311.11745* (2023).

[42] Kopec, G. Formant tracking using hidden Markov models. In *International Conference on Acoustics, Speech, and Signal Processing* (1985).

[43] Kovela, S., Valle, R., Dantrey, A., and Catanzaro, B. Any-to-any voice conversion with F0 and timbre disentanglement and novel timbre conditioning. In *International Conference on Acoustics, Speech and Signal Processing* (2023).

[44] Kubichek, R. Mel-cepstral distance measure for objective speech quality assessment. In *Pacific Rim Conference on Communications Computers and Signal Processing* (1993).

[45] Kumar, R., Seetharaman, P., Luebs, A., Kumar, I., and Kumar, K. High-fidelity audio compression with improved RVQGAN. *Neural Information Processing Systems* (2023).

[46] Kuwabara, H. Acoustic properties of phonemes in continuous speech for different speaking rate. In *International Conference on Spoken Language Processing* (1996).

[47] Ladefoged, P., and Johnson, K. *A course in phonetics*. Cengage learning, 2014.

[48] Łańcucki, A. Fastpitch: Parallel text-to-speech with pitch prediction. In *International Conference on Acoustics, Speech and Signal Processing* (2021).

[49] Le, M., Vyas, A., Shi, B., Karrer, B., Sari, L., Moritz, R., Williamson, M., Manohar, V., Adi, Y., Mahadeokar, J., et al. Voicebox: Text-guided multilingual universal speech generation at scale. *Neural Information Processing Systems* (2024).

[50] LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*, 10 (1995), 1995.

[51] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[52] Lee, S.-g., Ping, W., Ginsburg, B., Catanzaro, B., and Yoon, S. BigVGAN: A universal neural vocoder with large-scale training. *International Conference on Learning Representations* (2023).

[53] Lee, S.-H., Choi, H.-Y., Kim, S.-B., and Lee, S.-W. Hierspeech++: Bridging the gap between semantic and acoustic representation of speech by hierarchical variational inference for zero-shot speech synthesis. *arXiv preprint arXiv:2311.12454* (2023).

[54] Lee, Y., Keating, P., and Kreiman, J. Acoustic voice variation within and between speakers. *The Journal of the Acoustical Society of America 146*, 3 (2019), 1568–1579.

[55] Li, H. On-line and dynamic time warping for time series data mining. *International Journal of Machine Learning and Cybernetics 6* (2015), 145–153.

[56] Lind-Combs, H. C., Bent, T., Holt, R. F., Clopper, C. G., and Brown, E. Comparing Levenshtein distance and dynamic time warping in predicting listeners' judgments of accent distance. *Speech Communication 155* (2023).

[57] Liu, A. H., Yeh, S.-L., and Glass, J. R. Revisiting self-supervised learning of speech representation from a mutual information perspective. In *International Conference on Acoustics, Speech and Signal Processing* (2024).

[58] Liu, J., Li, C., Ren, Y., Chen, F., and Zhao, Z. DiffSinger: Singing voice synthesis via shallow diffusion mechanism. In *AAAI Conference on Artificial Intelligence* (2022).

[59] Liu, S., Cao, Y., Wang, D., Wu, X., Liu, X., and Meng, H. Any-to-many voice conversion with location-relative sequence-to-sequence modeling. *Transactions on Audio, Speech, and Language Processing* (2021).

[60] Lorenzo-Trueba, J., Drugman, T., Latorre, J., Merritt, T., Putrycz, B., Barra-Chicote, R., Moinet, A., and Aggarwal, V. Towards achieving robust universal neural vocoding. *Interspeech* (2019).

[61] Loshchilov, I., and Hutter, F. Decoupled weight decay regularization. *International Conference on Learning Representations* (2019).

[62] Mairesse, F., Polifroni, J., and Di Fabbrizio, G. Can prosody inform sentiment analysis? experiments on short spoken reviews. In *International Conference on Acoustics, Speech and Signal Processing* (2012).

[63] Mauch, M., and Dixon, S. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *International Conference on Acoustics, Speech and Signal Processing* (2014).

[64] McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. Montreal forced aligner: Trainable text-speech alignment using Kaldi. In *Interspeech* (2017).

[65] McCurdy, R. Tentative standards for sound level meters. *Electrical Engineering* (1936).

[66] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. librosa: Audio and music signal analysis in Python. In *Python in Science Conference* (2015).

[67] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. Mixed precision training. *International Conference on Learning Representations* (2018).

[68] Mohammed, M. A., Abdulkareem, K. H., Mostafa, S. A., Khanapi Abd Ghani, M., Maashi, M. S., Garcia-Zapirain, B., Oleagordia, I., Alhakami, H., and Al-Dhief, F. T. Voice pathology detection and classification using convolutional neural network model. *Applied Sciences 10*, 11 (2020), 3723.

[69] Morise, M., Yokomori, F., and Ozawa, K. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems* (2016).

[70] Morrison, M. torchcrepe. `https://github.com/maxrmorrison/torchcrepe`, 2022.

[71] Morrison, M., Churchwell, C., Pruyne, N., and Pardo, B. Fine-grained and interpretable neural speech editing. In *Submitted to Interspeech 2024* (March 2024).

[72] Morrison, M., Hsieh, C., Pruyne, N., and Pardo, B. Cross-domain neural pitch and periodicity estimation. *arXiv preprint arXiv:2301.12258* (2023).

[73] Morrison, M., Jin, Z., Bryan, N. J., Caceres, J.-P., and Pardo, B. Neural pitch-shifting and time-stretching with controllable LPCNet. In *arXiv preprint arXiv:2110.02360* (2022).

[74] Morrison, M., Jin, Z., Salamon, J., Bryan, N. J., and Mysore, G. J. Controllable neural prosody synthesis. *Interspeech* (2020).

[75] Morrison, M., Kumar, R., Kumar, K., Seetharaman, P., Courville, A., and Bengio, Y. Chunked autoregressive gan for conditional waveform synthesis. In *International Conference on Learning Representations* (2022).

[76] Morrison, M., and Pardo, B. OtoMechanic: Auditory automobile diagnostics via query-by-example. *Detection and Classification of Acoustic Scenes and Events* (2019).

[77] Morrison, M., Pawar, P., Pruyne, N., Cole, J., and Pardo, B. Crowdsourced and automatic speech prominence estimation. In *International Conference on Acoustics, Speech, and Signal Processing* (2024).

[78] Morrison, M., Rencker, L., Jin, Z., Bryan, N. J., Caceres, J.-P., and Pardo, B. Context-aware prosody correction for text-based speech editing. In *International Conference on Acoustics, Speech and Signal Processing* (2021).

[79] Morrison, M., Tang, B., Tan, G., and Pardo, B. Reproducible subjective evaluation. In *ICLR Workshop on ML Evaluation Standards* (2022).

[80] Moulines, E., and Charpentier, F. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication* (1990).

[81] Mustafa, A., Pia, N., and Fuchs, G. StyleMelGAN: An efficient high-fidelity adversarial vocoder with temporal adaptive normalization. In *International Conference on Acoustics, Speech and Signal Processing* (2021).

[82] Mysore, G. J. Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech?—a dataset, insights, and challenges. *IEEE Signal Processing Letters* (2014).

[83] Nooteboom, S. The prosody of speech: melody and rhythm. *The handbook of phonetic sciences* (1997).

[84] Ohtani, Y., Okamoto, T., Toda, T., and Kawai, H. FIRNet: Fundamental frequency controllable fast neural vocoder with trainable finite impulse response filter. In *International Conference on Acoustics, Speech and Signal Processing* (2024).

[85] O'Reilly, P., Bugler, A., Bhandari, K., Morrison, M., and Pardo, B. VoiceBlock: Privacy through real-time adversarial attacks with audio-to-audio models. In *Neural Information Processing Systems* (2022).

[86] O'Reilly, P., Jin, Z., Su, J., and Pardo, B. Maskmark: Robust neuralwatermarking for real and synthetic speech. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2024), IEEE, pp. 4650–4654.

[87] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. PyTorch: An imperative style, high-performance deep learning library. *Neural Information Processing Systems* (2019).

[88] Pirker, G., Wohlmayr, M., Petrik, S., and Pernkopf, F. A pitch tracking corpus with evaluation on multipitch tracking scenario. In *Twelfth Annual Conference of the International Speech Communication Association* (2011).

[89] Qian, K., Jin, Z., Hasegawa-Johnson, M., and Mysore, G. J. F0-consistent many-to-many non-parallel voice conversion via conditional autoencoder. In *International Conference on Acoustics, Speech and Signal Processing* (2020).

[90] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning* (2023).

[91] Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. Fastspeech 2: Fast and high-quality end-to-end text to speech. *International Conference on Learning Representations* (2021).

[92] Rindal, U. E. *Meaning in English: L2 attitudes, choices and pronunciation in Norway*. PhD thesis, University of Oslo, 2013.

[93] Rush, A. M. The annotated transformer. In *Workshop for NLP Open Source Software* (2018), pp. 52–60.

[94] Salamon, J., Bittner, R. M., Bonada, J., Bosch, J. J., Gómez Gutiérrez, E., and Bello, J. P. An analysis/synthesis framework for automatic F0 annotation of multitrack datasets. In *International Society for Music Information Retrieval Conference* (2017).

[95] Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., and Goldstein, T. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. In *International Conference on Machine Learning* (2020).

[96] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. In *International Conference on Acoustics, Speech and Signal Processing* (2018).

[97] Shibuya, T., Takida, Y., and Mitsufuji, Y. BigVSAN: Enhancing GAN-based neural vocoders with slicing adversarial network. *International Conference on Acoustics, Speech and Signal Processing* (2023).

[98] Shoemake, K. Animating rotation with quaternion curves. In *SIGGRAPH* (1985).

[99] Singh, S., Wang, R., and Qiu, Y. DEEPF0: End-to-end fundamental frequency estimation for music and speech signals. In *International Conference on Acoustics, Speech and Signal Processing* (2021).

[100] Siuzdak, H. Vocos: Closing the gap between time-domain and Fourier-based neural vocoders for high-quality audio synthesis. *International Conference on Learning Representations* (2024).

[101] Siuzdak, H., Dura, P., van Rijn, P., and Jacoby, N. WavThruVec: Latent speech representation as intermediate features for neural speech synthesis. *Interspeech* (2022).

[102] Smith, L. M., Bartholomew, A. J., Burnham, L. E., Tillmann, B., and Cirulli, E. T. Factors affecting pitch discrimination performance in a cohort of extensively phenotyped healthy volunteers. *Scientific Reports 7*, 1 (2017), 16480.

[103] Song, K., Xue, H., Wang, X., Cong, J., Zhang, Y., Xie, L., Yang, B., Zhang, X., and Su, D. Adavits: Tiny vits for low computing resource speaker adaptation. *International Symposium on Chinese Spoken Language Processing* (2022).

[104] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[105] Talkin, D., and Kleijn, W. B. A robust algorithm for pitch tracking (rapt). *Speech Coding and Synthesis 495* (1995), 518.

[106] TIME. Best inventions of 2023. `https://time.com/collection/best-inventions-2023/`, 2024.

[107] Valin, J.-M., and S., J. LPCNet: Improving neural speech synthesis through linear prediction. In *International Conference on Acoustics, Speech and Signal Processing* (2019).

[108] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems* (2017).

[109] Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory 13*, 2 (1967), 260–269.

[110] Wan, L., Wang, Q., Papir, A., and Moreno, I. L. Generalized end-to-end loss for speaker verification. In *International Conference on Acoustics, Speech and Signal Processing* (2018).

[111] Wang, C., Chen, S., Wu, Y., Zhang, Z., Zhou, L., Liu, S., Chen, Z., Liu, Y., Wang, H., Li, J., et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111* (2023).

[112] Wang, H., Liang, C., Wang, S., Chen, Z., Zhang, B., Xiang, X., Deng, Y., and Qian, Y. Wespeaker: A research and production oriented speaker embedding learning toolkit. In *International Conference on Acoustics, Speech and Signal Processing* (2023).

[113] Wang, L., Thakkar, O., and Mathews, R. Unintended memorization in large asr models, and how to mitigate it. In *International Conference on Acoustics, Speech and Signal Processing* (2024).

[114] Wang, X., Takaki, S., and Yamagishi, J. Neural source-filter-based waveform model for statistical parametric speech synthesis. In *International Conference on Acoustics, Speech and Signal Processing* (2019).

[115] Wang, Y., Su, J., Finkelstein, A., and Jin, Z. Controllable speech representation learning via voice conversion and aic loss. In *International Conference on Acoustics, Speech and Signal Processing* (2022).

[116] Webb, M. A., and Tangney, J. P. Too good to be true: Bots and bad data from mechanical turk. *Perspectives on Psychological Science* (2022), 17456916221120027.

[117] Wei, W., Li, P., Yu, Y., and Li, W. HarmoF0: Logarithmic scale dilated convolution for pitch estimation. *International Conference on Multimedia and Expo* (2022).

[118] Wu, P., Li, T., Lu, Y., Zhang, Y., Lian, J., Black, A. W., Goldstein, L., Watanabe, S., and Anumanchipalli, G. K. Deep speech synthesis from MRI-based articulatory representations. *Interspeech* (2023).

[119] Yamagishi, J., Veaux, C., MacDonald, K., et al. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92). *University of Edinburgh Centre for Speech Technology Research* (2019).

[120] Yoneyama, R., Wu, Y.-C., and Toda, T. Source-filter hifi-gan: Fast and pitch controllable high-fidelity neural vocoder. In *International Conference on Acoustics, Speech and Signal Processing* (2023).

[121] Yost, W. A. *Fundamentals of hearing: an introduction*. Acoustical Society of America, 2001.

[122] Zhang, Y., Pardo, B., and Duan, Z. Siamese style convolutional neural networks for sound search by vocal imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 27*, 2 (2018), 429–441.

[123] Zhang, Z. Mechanics of human voice production and control. *The Journal of the Acoustical Society of America 140*, 4 (2016), 2614–2635.

[124] Zhao, G., Ding, S., and Gutierrez-Osuna, R. Foreign accent conversion by synthesizing speech from phonetic posteriorgrams. In *Interspeech* (2019).

[125] Zhao, Y., Huang, W.-C., Tian, X., Yamagishi, J., Das, R. K., Kinnunen, T., Ling, Z., and Toda, T. Voice conversion challenge 2020: Intra-lingual semi-parallel and cross-lingual voice conversion. *Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge* (2020).

[126] Zhou, Y., Tian, X., Xu, H., Das, R. K., and Li, H. Cross-lingual voice conversion with bilingual phonetic posteriorgram and average modeling. In *International Conference on Acoustics, Speech and Signal Processing* (2019).

[127] Zhu, J., Zhang, C., and Jurgens, D. Phone-to-audio alignment without text: A semi-supervised approach. In *International Conference on Acoustics, Speech and Signal Processing* (2022).

APPENDIX A

List of symbols

Unlisted symbols are not used outside the context they are introduced

Symbol	Description
$a \in \mathbb{R}$	The A-weighted loudness (in dBA) corresponding to a frame of audio (Section 2.4)
$a_1, \ldots, a_T \in \mathbb{R}^T$	T adjacent frames of A-weighted loudness; also called a "loudness contour"
$f \in \mathbb{R}^+$	A center frequency of a pitch bin used when performing pitch estimation (Section 2.2)
$h \in [0, 1]$	One frame of periodicity, which measures the extent to which the corresponding frame of audio contains pitch (Section 2.3)
$h_1, \ldots, h_T \in \mathbb{R}^T$	T adjacent frames of periodicity; also called a "periodicity contour"
$r_f \in \mathbb{R}^+$	A ratio produced via resampling-based data augmentation (Section 3.2.1) that enables coarse-grained editing of the relative energy between high- and low-formants (Section 4.4)

Symbol		Description
r_l	$\in \mathbb{R}^+$	A ratio produced via volume-scaling data augmentation (Section 3.2.1) that improves fine-grained editing of perceptual loudness (Section 4.3)
v	$\in \{0, 1\}$	A binary indicator of whether a frame of speech is voiced (i.e., $h > \alpha$) (Section 2.3.1)
$v_1, \ldots, v_T$	$\in \mathbb{R}^T$	T adjacent frames of binary voicing indicators
x	$\in \mathbb{R}^W$	One frame of an audio waveform, consisting of W samples
$x_1, \ldots, x_T$	$\in \mathbb{R}^{W \times T}$	T adjacent frames of an audio waveform, each consisting of W samples and offset from the previous frame by H samples
y	$\in \mathbb{R}$	One frame of pitch in units of Hz (Section 2.2)
$y_1, \ldots, y_T$	$\in \mathbb{R}^T$	T adjacent frames of pitch; also called a "pitch contour"
D	$\in \mathbb{R}^{\lvert F \rvert \times T}$	The pitch posteriorgram formed by concatenating posterior distributions inferred by a neural pitch estimator for adjacent frames of audio $x_1, \ldots, x_T$

Symbol		Description
F	$= \{f_1, \ldots, f_{\lvert F \rvert}\}$	A set of pitch bins, where each pitch bin represents a unique frequency range and have center frequencies (Section 2.2)
G	$\in \mathbb{R}^{\lvert P \rvert \times T}$	The phonetic posteriorgram on phoneme set P corresponding to adjacent frames of audio $x_1, \ldots, x_T$
Hi	$\in \mathbb{R}^T$	The time-varying frequency contour of harmonic i
P	$= \{$"aa", "ae",..., "zh"$\}$	The set of phonemes used for phonetic posteriorgram inference (Section 2.1)
S	$\in \mathbb{R}^{\lvert \Omega \rvert \times T}$	The magnitude spectrogram (Section 1.1.2)
T	$\in \mathbb{N}$	The number of time frames in an audio signal
W	$\in \mathbb{N}$	The window size, or the number of audio samples in one frame
α	$\in [0, 1]$	A voiced/unvoiced threshold, at or below which a periodicity value h is considered to be unvoiced
γ	$\in \mathbb{R}^+$	Hyperparameter controlling the relative contribution of phoneme similarity matrix $\mathcal{S}$ to the PPG pronunciation distance (Section 2.1.4)
λ	$\in \mathbb{R}^+$	Weight used to class-balance a phonetic posteriorgram (PPG)(Section 2.1.4)

Symbol		Description
ω	$\in \mathbb{R}$	A discrete analysis frequency used when performing short-time fast Fourier transforms (FFTs) (Section 1.1.2)
Ω	$= \{\omega_1, \ldots, \omega_{\lvert\Omega\rvert}\}$	The discrete analysis frequencies used when performing short-time fast Fourier transforms (FFTs) (Section 1.1.2)
$\mathcal{A}$	$\in \mathbb{R} \to \mathbb{R}$	The A-weighted transformation of the frequency set based on human perceptual data (Section 2.4)
$\mathcal{S}$	$\in \mathbb{R}^{\lvert P\rvert \times \lvert P\rvert}$	The acoustic phoneme similarity matrix inferred by a phonetic posteriorgram model (Section 2.1.4; Figure 2.4)
$\mathcal{V}$	$= \{t : v_t == 1\}$	The set of time frame indices $t \in 1, \ldots, T$ during which the speech is voiced (i.e., $h_t > \alpha$) (Section 2.3.1)
ΔdBA	$\in \mathbb{R} \times \mathbb{R} \to \mathbb{R}$	The RMSE between two A-weighted loudness contours (Section 3.3.1)
ΔPPG	$\in \mathbb{R} \times \mathbb{R} \to \mathbb{R}$	My proposed PPG distance (Section 2.1.4) based on a weighted JS divergence between sparse phonetic posteriorgrams
Δ¢	$\in \mathbb{R} \times \mathbb{R} \to \mathbb{R}$	The distance in cents between two pitch values $y, \hat{y}$; $\Delta¢(y, \hat{y}) = \left\lvert 1200 \log_2(y/\hat{y}) \right\rvert$ (Section 3.3.1)

Symbol	Description
$\Delta\phi \quad \in \mathbb{R} \times \mathbb{R} \to \mathbb{R}$	My proposed RMSE periodicity metric between entropy-based periodicity (Section 3.3.1)