Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B
D-70569 Stuttgart

Master thesis

Phonetic Representations of Speech for Human Pronunciation Feedback and Automatic Accent Transfer

Isaac Riley

Studiengang: M.Sc. Computational Linguistics

Prüfer*innen: Prof. Dr. Wolfgang Wokurek
Prof. Dr. Antje Schweitzer

Betreuer: Prof. Dr. Wolfgang Wokurek

Beginn der Arbeit: 06.04.2021

Ende der Arbeit: 06.10.2021

Erklärung (Statement of Authorship)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Sie ist weder vollständig noch in Teilen bereits veröffentlicht. Die beigefügte elektronische Version stimmt mit dem Druckexemplar überein.[1]

(Isaac Riley)

[1]Non-binding translation for convenience: This thesis is the result of my own independent work, and any material from work of others which is used either verbatim or indirectly in the text is credited to the author including details about the exact source in the text. This work has not been part of any other previous examination, neither completely nor in parts. It has neither completely nor partially been published before. The submitted electronic version is identical to this print version.

Acknowledgments
XXX

Inhaltsverzeichnis

1 Introduction [4]

In linguistics, the notion of accent refers to a pattern of pronunciation that does not change the semantic content of what is uttered, but which may carry pragmatic meaning and convey demographic information about the speaker. While some phonetic variation may be random or occur at the individual level (idiolect), accent typically varies systematically according to native language and dialect, which in turn vary geographically and demographically.

Accent is occasionally an impediment to human understanding of speech; however, among native and advanced speakers, accent is often no impediment to communication. In automatic speech recognition, accent accents underrepresented in training datasets typically have a much stronger negative effect on recognition accuracy. For this reason, accent is an active subject of research in automatic speech processing.

Related to accent transfer is the task of voice conversion. The goal of voice conversion is to generate speech containing identical linguistic information as an input sample, modifying only the timbre to match a target speaker. In some respects, voice conversion is a simpler task; namely, there is no temporal realignment and phonetic information remains the same. The absence of temporal realignment means that voice conversion models have an equal number of frames in input and output. This makes possible frame-to-frame mappings, also surrounding context must still be taken into account in order to achieve good results.

This works explores approaches to accent conversion with and without voice conversion. These approaches may be referred respectively as intra-speaker accent conversion and cross-speaker accent conversion. In intra-speaker accent conversion (henceforth simply *accent conversion*), an utterance from the source speaker is modified to match the accent of the target speaker, without changing voice quality or non-accent-carrying linguistic content. By contrast, in cross-speaker accent conversion, accent conversion is combined with voice conversion. Henceforth, this will simply be referred to as speaker conversion.

This work investigates two distinct but related approaches to speech conversion,

both accent conversion and speaker conversion. The first is a continuous approach in which speech is mapped to and synthesized from a continuous phonetic representation. The second is a discrete approach, in which speech is transcribed to and synthesized from the International Phonetic Alphabet (IPA).

Clearly, in the simplest case, speaker conversion may involve the combination of ASR and TTS, where the ASR model recognizes the speech of the source speaker and the TTS model synthesizes the speech of the target speaker from that same text. This work takes an alternative approach, in which speech is transcribed in the IPA, modified by an intermediate model to match the phonetic patterns of the target speaker, and a TTS model trained to synthesize speech from IPA is used to synthesize the desired speech.

Summary and Comparison of Speech Conversion Tasks

Speech Aspect	Accent Conversion	Voice Conversion	Speaker Conversion
content	source	source	source
accent	**target**	**target**	source
vocal quality	source	**target**	**target**

2 Background [9]

2.1 Automatic Speech Recognition [2]

2.2 Text-to-Speech Synthesis [1.5]

As its name suggests, test-to-speech synthesis involves generating waveform speech from text input. State-of-the art approaches are typically neural and involve a feature prediction network (FPN) and and audio generating network (AGN).

*** STILL NEED TO TYPESET REST OF NOTES ***

2.3 Speech Signal Processing [1.5]

2.3.1 Short-Time Fourier Transform

*** STILL NEED TO TYPESET NOTES ***

2.3.2 Mel Spectrum and Coefficients

2.3.3

2.3.4

2.4 Neural Networks [3]

2.4.1 Feedforward Neural Networks

A feedforward neural network

2.4.2 Recurrent Neural Networks

Feedforward neural networks are effective in processing vector inputs. If the input data

One powerful and popular architecture for recurrent neural networks is the long short-term memory (LSTM) network.

2.4.3 Convolutional Neural Networks

When input data have a spatial component, it is beneficial to use a network architecture that can take advantage of spatial relationships in the data.

2.4.4 Generative Adversarial Networks

Operating within the parradigm of supervised learning, neural networks can be used for classification by learning a posterior distribution over labels, given numerical

features representing features of an input sample. Given sufficient exposure to labeled training data, an appropriately designed model can update its parameters so as to assign an increasingly larger probability to the correct label.

Another branch of machine learning involves generative learning. Given a training dataset, a generative model learns to output samples resembling those from the training dataset.

In "vanilla" generative adversarial networks (GANs), the are two networks whose training takes the form of a minimax game, in which each model seeks to maximize the loss of the other. The first network is the generator, which is given random noise as input and tasked with outputting samples of the same type as the target data. The second model is the discriminator, a classification network tasked with classifying generated samples as real or generated. Throughout the (ideal) adversarial training process, the generator continually improves to generate increasingly realistic outputs, and the discriminator becomes increasingly adept at distinguishing real samples from generated samples.

2.5 International Phonetic Alphabet [1]

3 Related Work [8]

3.1 Prior Work on Accent Transfer [2]

3.2 Models Used [6]

3.2.1 Tacotron 2

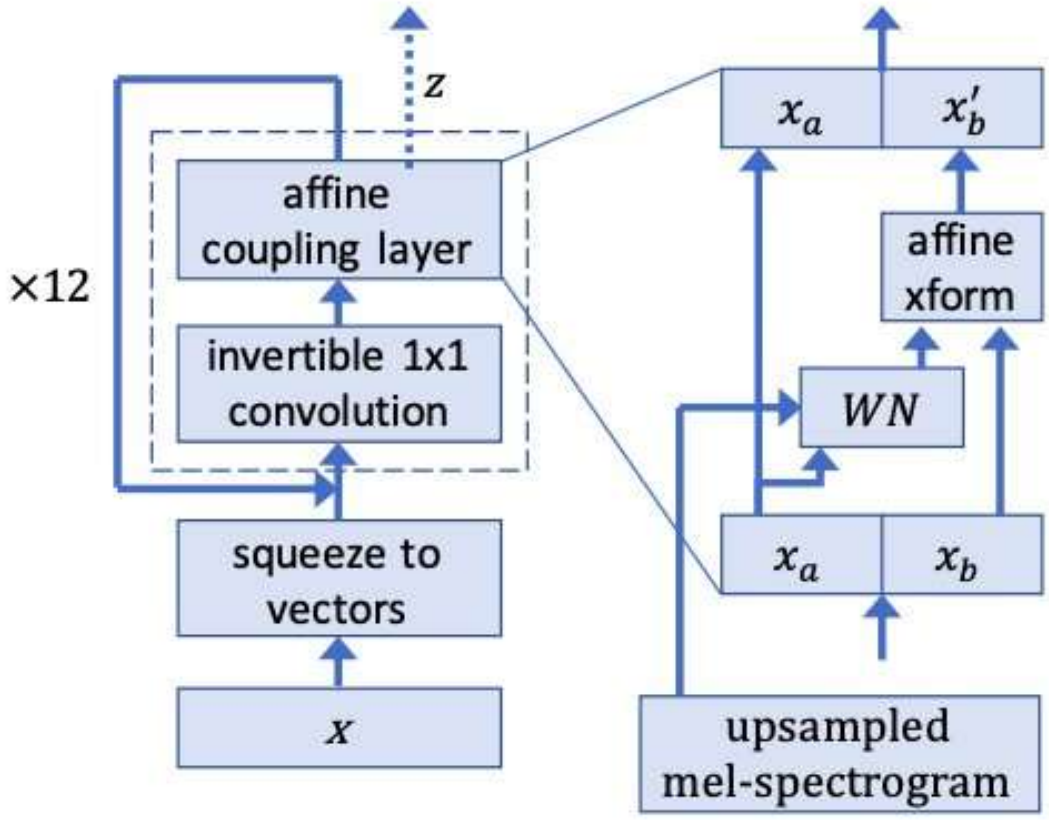
Tacotron 2 is a deep neural sequence-to-sequence model that generates mel spectrograms from text. I can be combined with a deep neural vocoder, it forms an end-to-end text-to-speech system.

3.2.2 WaveGlow

WaveGlow is a deep neural vocoder that generates waveform audio from mel spectrograms (Prenger et al., 2019). It has a flow-based architecture that learns an invertible mapping, converting a noise vector and additional conditioning input into nearly human-sounding waveform audio. Its audio generation is accomplished by sampling from a distribution, which is transformed as it passes through the model and is conditioned on mel-spectrograms representing the audio that is to be output in waveform.

It is based on the Glow architecture for image generation (Kingma and Dhariwal, 2018) and on WaveNet, the groundbreaking autoregressive waveform generation model (van den Oord et al., 2016). Its primary advantage over WaveNet is faster inference speed; because it is non-autoregressive, it can generate audio much faster than WaveNet, and 25 times faster than real time on a V100 GPU [XX].

Its basic architecture is illustrated below:



Note that much of the computational work is done repetitively, the convolutional and subsequent affine coupling layer are repeated 12 times to generate the level of fine-grained detail required in high-quality waveform.

3.2.3 CycleGAN

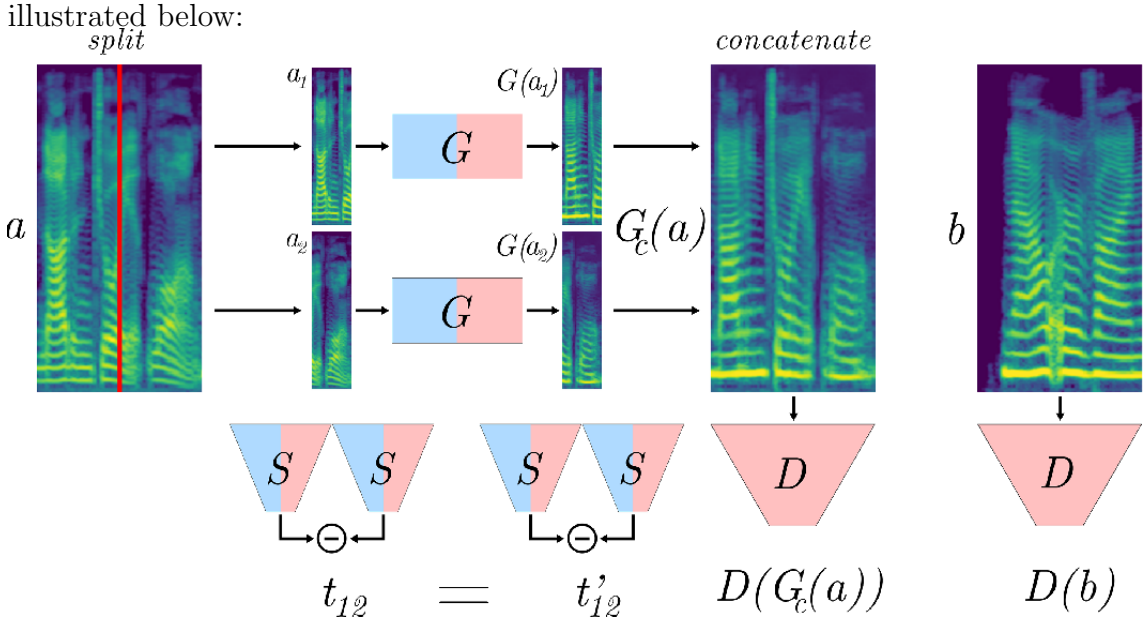
As described above, in 'vanilla' Generative Adversarial Networks, the input is typically random noise, which is mapped to some point in the target distribution. Moreover, because the generator is rewarded simply for generating some output appearing to have been drawn from the target distribution, there is no direct control over which point from the target distribution is generated. However, there are many applications in which one might wish to exert more control over the output by conditioning on some desired criteria. One common case involves the task of image translation, in which certain aspects of an image are altered while others remain

intact. A simple example of image translation is a model that transforms horses into zebras. In this case, a simple GAN may take a horse image as input and output a very dissimilar zebra image as output; this is because there is no constraint to enforce content preservation. CycleGAN () achieves this by training an inverse generator to map the transformed image back to the original. This cyclical consistency constraint requires the forward generator to preserve semantic content from the original image so that it the original can be reconstructed from the transformed image.

3.2.4 MelGAN-VC

Most work on GANs focuses on data with a fixed output shape, where input size and output shape are typically equal, such as in the case of image generation. When an input is supplied, it is for the purpose of conditioning the output and the input typically has the same shape as the output.

In settings where the data has an intrinsically sequential nature and the input size varies, it is not possible to require a fixed and shape. One such setting is voice conversion, and one clever solution has been proposed by Marco Pasini in [XX], illustrated below:

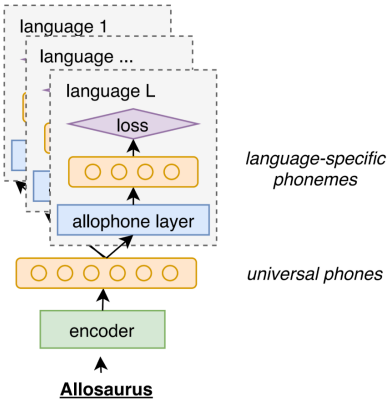


The key insight is that by taking a fixed length (T frames) of the input spectro-

gram, then splitting the input into two equal-sized blocks and passing each to the generator network. The converted spectrograms, i.e. the two generator outputs are then recombined and passed to the discriminator. Due to the adversarial loss training objective, the generator must learn to convert each block in such a way that, when reassembled, it will be indistinguishable from a real sample from the target set. Because the target set does not contain spectrograms with discontinuities, such a discontinuity at the splitting point would clearly identify a fake sample. Thus, the generator is incentivized to produce outputs which, when re-joined, do not contain such telltale discontinuities at the splitting points. Because this applies to any split point, it is possible to train a relatively small generator that can receive inputs of arbitrarily length one block at a time and generate the desired output.

3.2.5 Allosaurus

Allosaurus is a Python library containing tools for automatic neural phonetic transcriptions using IPA. Released by a team of researchers at Carnegie Mellon University, it performs phone recognition, rather than the more typical task of phoneme recognition, and its main innovation is its so-called "allophone layer". This is a phone-to-phoneme mapping that restricts the phonetic invntory to that of a single language. This enable Allosaurus to capture the benefits both of shared phoneme models and private phoneme models. While all three architectures use an encoder, shared phoneme models use a common phone layer across languages, while in the private phoneme model, each language has a separate model following the shared encoder. Allosaurus strikes a balance by using a shared decoder or "universal phone layer" and leaving the per-language customization to the allophone layer. This allows the decoder to benefit from larger amounts of training data and greater generality. These differences in architecture are shown below:


language 1
language ...
language L
loss
language-specific phonemes
allophone layer
universal phones
encoder
Allosaurus

4 Methods

4.1 Phonetic Continuous Representation-Based Speech Synthesis and Accent Transfer [3]

4.1.1 Phonetic Posteriorgrams

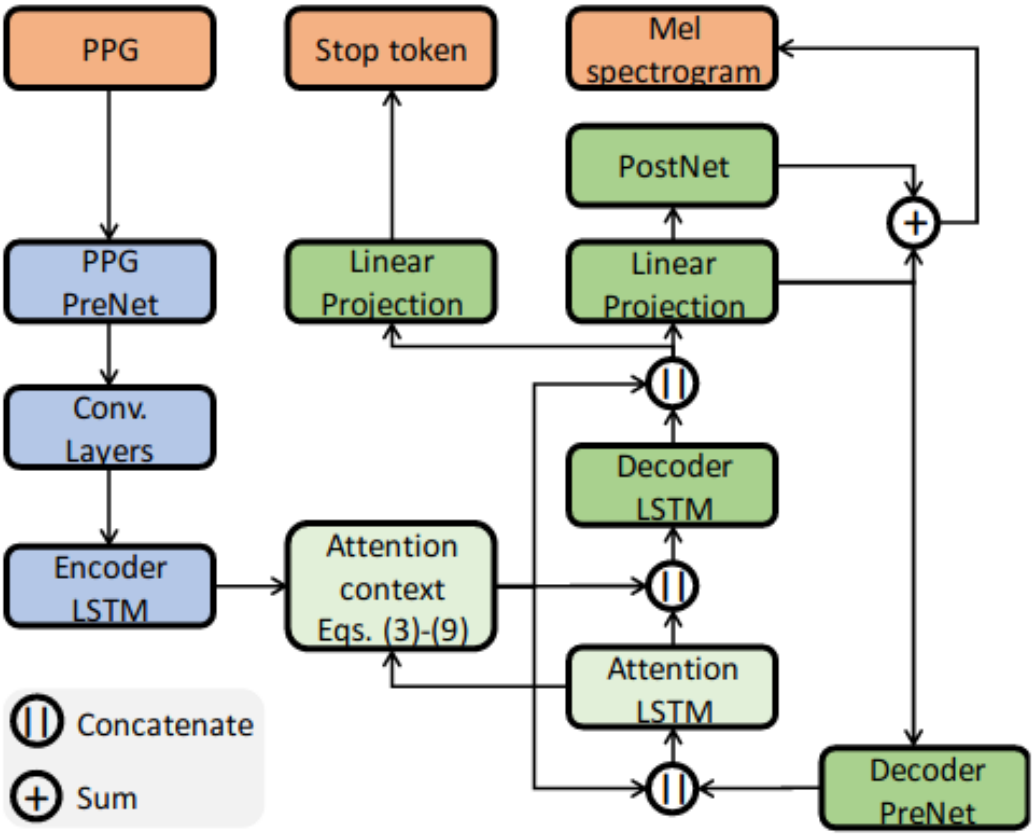
A phonetic posteriorgram represents predicted framewise phoneme probabilities from a speech sample. In this work, a pretrained acoustic model from [XX] was used, due to the high quality of the model, which had been trained on the extremely large and varied LibriSpeech corpus.

The acoustic model predicts 5816 senones; the dimensionality is greatly reduced by a linear discriminant analysis matrix, yielding the posterior probability for the 40 phonemes in the TIMIT reduced phoneme set, which include a silence phoneme (not a true phoneme, but essentially treated as one). Refer to the Data section for samples.

4.1.2 Speech Synthesis from PPGs [1]

To generate a mel spectrogram representation from a PPG, a modified version of Tacotron2 was used. This worked followed NVIDIA's open-source implementation with a few modifications. As in [XX], and as shown below, the key differences were the addition of a localized attention constraint, which decreases the computational complexity. This is especially attractive in this case because the expected alignment between PPG and mel spectogram is tighter than the alignment between text and mel spectrogram (e.g. in the standard Tacotron2 model). Because of this, it is superfluous to query all encoder states.

The other important difference is that instead of an embedding layer for the input, a linear projection is used, since the input is not sparse, i.e. not one-hot encoded. Thus, each input dimension must be taken into account by the model. The following image summarizes the architecture [XX]:



4.1.3 Accent Transfer in PPG Space [2]

Inspired by the basic architecture of MelGAN-VC, the PPG accent transfer model uses a generative adversarial network with a cyclical consistency constraint. There are two different variants.

The first architecture is convolutional, using one-dimensional convolutions across time. Two-dimensional convolutions were not used because there are no local spatial relationships across phonemes in a PPG; phoneme oder is not meaningful. The spatial relationships of interest are exclusively temporal.

The second architecture is sequential, a bidirectional LSTM. Each of these architectures, when used, is used in both the discriminator and the generator.

*** ILLUSTRATION TO COME ***

4.2 Phonetic Transcription-Based Speech Synthesis and Accent Transfer

4.2.1 Relational Mappings for Segmental Accent Transfer [1]

Because accents tend to be characterized by a number of salient features differing from the 'standard' dialect or simply from other dialects, it is possible to formulate mapping rules that define which phonetic changes must be made to a source accent to make it sound more like a target accent. For example, a speaker of Received Pronunciation English wishing to imitate a speaker of American English will typically pronounced syllable-final "r" as [ɹ] and will voice intervocalic occurrences of "t". An American English speaker wishing to imitate RP English might apply the inverse of these rules.

The following section seeks to investigate and formalize approaches to learning phonetic transformation rules from non-parallel data.

A phonetic sequence consists of a number of IPA segments s_i:

$$S^{(n)} = \langle \text{start} \rangle \, , s_1^{(n)}, s_2^{(n)}, ...s_L^{(n)}, \langle \text{stop} \rangle$$

Given the data, it is straightforward to calculate the empirical conditional probabilities. The probability that segment s_j follows segment s_i is calculated as follows:

$$\hat{p}(s_j|s_i) = \frac{c_{ij} + 1}{c_i}$$

where c_{ij} represents the corpus frequency of the bigram s_i s_j and c_i represents the corpus frequency of s_i. This is analogous to a simple language model with smoothing, but it would be more accurate to refer to the full set of conditional probabilities as a phonetic model.

Another necessary component is a matrix of segment similarities. This can be written as

$$A = [a_{ij}]$$

where

$$a_{ij} = \mathrm{sim}(s_i, s_j) = \frac{\mathbf{f}_i \cdot \mathbf{f}_j + 1}{||\mathbf{f}_i||\ ||\mathbf{f}_j||},$$

i.e. each element of this matrix represent the smoothed cosine similarity product of the appropriate articulatory feature vectors. For the special characters _ (word boundary), ⟨start⟩, and ⟨stop⟩, define self-similarity $\mathrm{sim}(x, x) = 1000$, since each of these characters will invariably be mapped to itself.

The following table lists and describes the articulatory features considered.

Articulatory Features

Feature	Relevant Segments
voiced	a, b, etc.
voiceless	f, h, etc.
vowel	a, æ, etc.
approximant	l, l̪, ɹ, ɻ, w
consonant	b, d, ð, etc.
front vowel	a, æ
central vowel	ə, ɨ
back vowel	ɑ, ɒ, o, ɔ, u, ʊ, ʌ
open vowel	a, æ, ɑ, ɒ, ɛ, ɔ, ʌ
mid vowel	e, ə, ɛ, o, ɔ, ʌ
close vowel	e, i, ɪ, ɨ, o, u, ʊ
unrounded vowel	a, æ, ɑ, e, ɛ, i, ɪ, ɨ, ʌ
rounded vowel	ɒ, o, ɔ, u, ʊ
bilabial	b, m, p, w
labiodental	f, v, w
dental	ð, l̪, θ
alveolar	d, ʤ, l, n, ɹ, ɻ, ɾ, s, ʃ, t, tʰ, ʧ
postalveolar	ʃ, ʒ
retroflex	ɖ, ɳ, ʂ, ʈ, ʐ
palatal	j, ɟ, ɟ
velar	g, k, kʰ, ŋ, x
uvular	none
pharyngeal	none
glottal	h

Articulatory Features (cont.)

Feature	Relevant Segments
plosive	b, d, ɖ, ɟ, k, k^h, p, t, t^h, ʈ
nasal	m, n, ɳ, ŋ
trill	r
tap/flap	ɾ
fricative	ð, f, h, j, s, ʂ, ʃ, v, x, z, ʐ, ʒ
lateral	l, ɭ
affricate	ʤ, ʧ
rhotic	r, ɹ, ɻ, ɾ

The core problem of segmental accent transfer is to learn a set of segmental mapping rules. In the simplest case, such a mapping is injective.When an injective mapping is assumed, the mapping can be defined as follows:

$$f_1(s_i^{(n)}) = t_i^{(n)} = \arg\max_{t_j} \alpha \cdot p(t_j|t_{j-1}) + \text{sim}(t_j, s_i^{(n)}).$$

Here α controls the weighting of the target phonetic model relative to similarity between source and target segments. It is to be chosen empirically.

Equivalently, in pseudocode notation:

Algorithm 1 Injective Mapping

input: list S, function sim(), phone model p, target phone inventory I, $\alpha \in [0, 1]$
output: list T
$i \leftarrow 1$
$T[0] \leftarrow \langle \text{start} \rangle$
$N \leftarrow \text{length}(S)$
while $i < N$ **do**
    best $= 0$
    for phone in I **do**
        score $= \alpha * p(\text{phone} \,|\, T[i-1]) + \text{sim}(\text{phone}|S[i])$
        if score $>$ best **then**
            best $\leftarrow$ score
            phone* $\leftarrow$ phone
        end if
    end for
    $T[i] \leftarrow$ phone*
    $i \leftarrow i + 1$
end while
$T[i] \leftarrow \langle \text{stop} \rangle$

Note that this belongs to the family of greedy search algorithms.

However, it may be desirable to increase flexibility by allowing one-to-two and two-to-one mappings. To accomplish this, it is necessary to add to the target character set the empty string. As in the formal language theory literature, the character "ϵ" will denote the empty string. To allow 2-to-1 mappings, define

$$\text{sim}(\epsilon, s) := \bar{a} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}}{N^2} \quad \forall s \in S : s \neq \epsilon$$

and

$$\text{sim}(\epsilon, \epsilon) = 1$$

To allow for more general alignments, we define a more flexible algorithm as follows:

Algorithm 2 Bigram-Compatible Mapping

input: list S, function sim(), phone model p, target phone inventory I, $\alpha \in [0, 1]$
output: list T
$i \leftarrow 1$
$T[0] \leftarrow \langle \text{start} \rangle$
$N \leftarrow \text{length}(S)$
while $i < N$ **do**
    best $= 0$
    for phone in I **do**
        score1 $= \alpha * p(\text{phone} \,|\, T[i-1]) + \text{sim}(\text{phone}|S[i])$
        if score1 $>$ best **then**
            best $\leftarrow$ score1
            phone* $\leftarrow$ phone
            $i \leftarrow i + 1$
            outcome $\leftarrow 1$
        end if
        score2 $= \alpha * p(\text{phone} \,|\, T[i-1]) + \text{sim}(\text{phone}|S[i-1])$
        if score2 $>$ best **then**
            best $\leftarrow$ score2
            phone* $\leftarrow$ phone
            outcome $\leftarrow 2$
        end if
        score3 $= \alpha * p(\text{phone} \,|\, T[i-1]) + \text{sim}(\text{phone}|S[i+1])$
        if score3 $>$ best **then**
            best $\leftarrow$ score3
            phone* $\leftarrow$ phone
            outcome $\leftarrow 3$
        end if
    end for
    if outcome < 3 **then**
        $T[j] \leftarrow$ phone*
        $j \leftarrow j + 1$
    end if
end while
$T[i] \leftarrow \langle \text{stop} \rangle$

Note that the modifications relative to Algorithm 1 allow for both 1:2 and 2:1 mappings. For example, if in the relation R it is the case that $s_{i-1}rt_{j-1}$, it is possible at alignment step (i, j) to have

$a)$ $s_{i-1}Rt_{j-1}, s_iRt_j$ (outcome1)
$b)$ $s_{i-1}Rt_{j-1}, s_{i-1}Rt_j$ (outcome2)
$c)$ $s_{i-1}Rt_{j-1}, s_iRt_{j-1}$ (outcome3)

Thus, it is possible to model systematic insertions and deletions, without increasing computational complexity, which remains $\mathcal{O}(n)$.

4.2.2 Segment-Based Speech Synthesis

Once again, a modified version of Tacotron2 is used to generate mel spectrograms, and Waveglow is used for waveform generation. The input encoding is different from the classic English speech synthesis implementation, but otherwise no modifications were required.

5 Data

5.1 Speech Corpora

To apply these methods, a novel dataset was created, containing speech recordings of two speakers. All data were downloaded from YouTube using the open-source tool youtube-dl. The first speaker is Grant Sanderson, a well-known educator and popularizer of mathematics. His dialect is standard American English. The second is Rajamunickam Antonimuthu, a YouTuber operating a popular channel about science and technology news. His native language is Tamil, and is has a srong influence on his speech. The speech samples of each speaker are relatively clean spontaneous speech. Depending on the exact process each uses for video creation, it may be at least partially scripted, but the nature of the speech samples is closest to spontaneous speech, especially compared to professional-quality read speech such as the LJSpeech corpus. In particular, there are more pauses and filler words.

For recognition and especially for synthesis, it is crucial to divide the longer speech files into smaller segments. To do this for the Antonimuthu corpus, subtitles were also scraped from the videos. Because the subtitles are time-aligned to the video, and hence to the audio files, it was possible to divide on subtitle time stamps, combining segments up to a maximum length of 10 seconds.

For the Sanderson corpus, no subtitles were available, and so it was necessary to cut on silence. To accomplish this, the Python library `pydub` was used, with a silence threshold of -30 and a minimum silence length of 300 miliseconds. These values were found empirically to work best. As in the Atonimuthu corpus, smaller segments were combined as long as their combined length was less than 10 seconds.

The final corpus sizes were 10500 utterances for Sanderson and 2100 utterances for Antonimuthu, with average utterance length 4.49 and 6.34 seconds, respectively.

5.2 Mel Spectrograms

The mel spectrograms were generated in PyTorch as in the original Tacotron2 and Waveglow open-source implementations by NVIDIA [XX]. The hyperparameters were as follows:

- input sampling rate: 22050 Hz
- number of mels: 80
- window length: 1024 (samples) $\approx$ 46.44ms
- filter length: 1024 (samples) $\approx$ 46.44ms
- shift (hop length): 160 (samples) $\approx$ 7.26ms
- minimum mel frequency: 0 Hz
- maximum mel frequency: 8000 Hz
- audio bit depth: 32768 (15-bit)

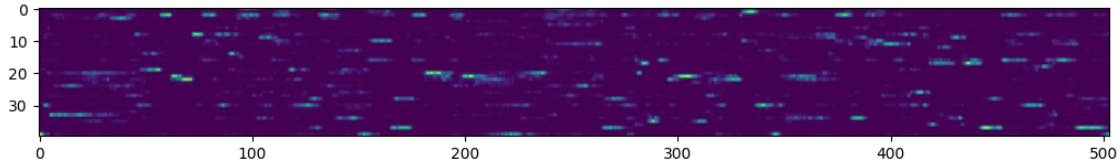
5.3 Phonetic Posteriorgrams

Using the acoustic model described earlier and introduced in [XX], phonetic posteriorgrams were generated for each utterance, using the following hyperparameters:

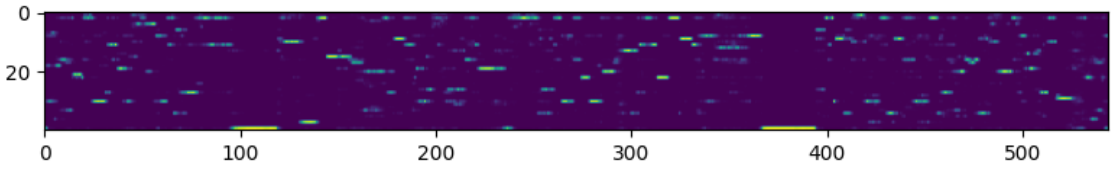
- input parameters:
  - number of mels (for input): 80
  - sampling rate: 22050 Hz
  - window length: 64ms
  - filter length: 64ms
  - shift (hop length): 10ms
- number of senones: 5816
- number of output phonemes: 40
- phoneme set: TIMIT reduced (40-phoneme) set

Examples:

- An Antonimuthu PPG representing the utterance *"To overcome the drawbacks of SLM, Lawrence Livermore National University researchers"*:



- A Sanderson PPG representing the utterance *"Geometric algebra, that is a whole other topic, uh, potentially for another day, but that's a - that's a really valid qustion"*:



5.4 Transcriptions [1]

The transcription process made use of a hybrid process. Analogously to the gain in speed reported in machine-assisted translation, it is faster to correct automatically-generated transcriptions than it is to manually create all transcriptions from scratch.

First, transcriptions were generated automatically by the Python library Allosaurus. These were only moderately accurate, and due to the nature of the Connectionist Temporal Classification loss which which the model is trained, the output sequence cannot have repeating tokens. Additionally, word boundaries were not included. This lack of word boundaries was hypothesized to be disadvantageous for speech synthesis, especially because the standard Tacotron 2 model input includes spaces and (standardized) punctuation.

Because of this, transcriptions were corrected by hand. Word boundaries were added, denoted by the underscore character (_). Additionally, the removal of repeated phones in different words was corrected. For example, the phrase "*that takes some more rice*" would be transcribed as [ð æ t e k s ʌ m o ɹ a ɪ s]. Following the manual correction, the transcription would be [ð æ t _ t e k s _ s ʌ m _ m o ɹ _ ɹ a ɪ s]. Intuitively, it appears clear that the second transcription contains more information that would be relevant for IPA-to-speech synthesis.

Because successful automatic synthesis requires many examples of each input token, for each speaker the rarest IPA segments were removed and replaced with the best more common alternative, leaving only those segments occurring at least 20 times in the fine-tuning set of 200 utterances.

To improve the automatically-generated transcriptions, the pretrained models released by the authors of XX were fine-tuned on 200 manually corrected transcriptions each. Subsequently, an additional XX utterances were transcribed for each speaker, creating two datasets with XX utterances each.

Phonetic Inventories

Phone	Definition	Antonimuthu	Sanderson
a	open front unrounded vowel	✓	✓
æ	near-open front unrounded vowel	✓	✓
ɑ	open back unrounded vowel	✓	✓
ɒ	open back rounded vowel	✓	
b	voiced bilabial plosive	✓	✓
d	voiced alveolar plosive	✓	✓
ð	voiced dental fricative	✓	✓
ʤ	voiced post-alveolar affricate	✓	✓
ɖ	voiced retroflex plosive	✓	
e	close-mid front unrounded vowel	✓	✓
ə	mid central vowel (schwa)	✓	✓
ɛ	open-mid front unrounded vowel	✓	✓
f	voiceless labiodental fricative	✓	✓
g	voiced velar stop	✓	✓
h	voiceless glottal fricative	✓	✓
i	close front unrounded vowel	✓	✓
ɪ	near-close front unrounded vowel	✓	✓
ɨ	close central unrounded vowel	✓	
j	voiced palatal approximant (yod)	✓	✓
ʝ	voiced palatal fricative	✓	
ɟ	voiced palatal plosive	✓	
k	voiceless velar plosive	✓	✓
k^h	aspirated voiceless velar plosive	✓	
l	voiced alveolar lateral approximant	✓	✓
l̪	voiced dental approximant	✓	
m	voiced bilabial nasal	✓	✓

Phonetic Inventories (cont.)

Phone	Definition	Antonimuthu	Sanderson
n	voiced alveolar nasal	✓	✓
ɳ	voiced retroflex nasal	✓	
ŋ	voiced velar nasal	✓	
o	close-mid back rounded vowel	✓	✓
ɔ	open-mid back rounded vowel	✓	✓
p	voiceless bilabial plosive	✓	✓
r	voiced alveolar trill	✓	
ɹ	voiced alveolar approximant		✓
ɻ̩	syllabic voiced alveolar approximant	✓	
ɾ	voiced alveolar tap/flap	✓	
s	voiceless alveolar fricative	✓	✓
ʂ	voiceless retroflex fricative	✓	
ʃ	voiceless postalveolar fricative	✓	✓
t	voiceless alveolar plosive	✓	✓
tʰ	aspirated voiceless alveolar plosive	✓	
ʧ	voiceless palato-alveolar affricate	✓	✓
ʈ	voiceless retroflex plosive	✓	
u	close back rounded vowel	✓	✓
ʊ	near-close back rounded vowel	✓	✓
v	voiced labiodental fricative	✓	✓
ʌ	open-mid back unrounded vowel	✓	✓
w	voiced labial-velar approximant	✓	✓
x	voiceless velar fricative	✓	
z	voiced alveolar fricative	✓	✓
ʐ	voiced retroflex sibilant fricative	✓	✓
ʒ	voiced postalveolar fricative		✓
θ	voiceless dental (non-sibilant) fricative		✓

6 Results [4]

6.1 Continuous Representation-Based Speech Synthesis and Accent Transfer

The usefulness of pre-synthesis transformation of PPGs requires that there be a systematic difference between the PPGs of speaker A and speaker B. Empirically, this was not the case, as the loss of each classifier converged around 0.69 with only small and apparently random fluctuations even after hundreds of epochs. This value is approximately the expected loss of binary cross entropy on random data, or $-\ln\frac{1}{2} \approx 0.693147$. Thus, we can confidently conclude that there is no systematic difference between the PPGs of Sanderson and Antonimuthu.

While disappointing, this finding was not entirely uninteresting. It suggests that, at least at the resolution of 40 phonemes, even speakers speaking the same language with starkly contrasting accents will have their speech mapped into a common phonetic space, in which the respective phonetic representations are indistinguishable.

A further disappointing finding was that speech synthesis from 40-phone PPGs was not able to generate coherent speech for the Sanderson and Antonimuthu corpora. Because Waveglow generated comprehensible speech from mel spectrograms, this was clearly due to the generation of mel spectrograms from PPGs. This falsifies the original hypothesis that high-quality speech synthesis from 40-phone PPGs is possible.

While the "black box" nature of neural networks makes it impossible at this point to draw conclusions about the causes of this failure with any certainty, there are two factors that seem most likely. First, it may be that the 40-phone PPGs simply do not carry sufficient information. As described above [XXX] succeeded in generating high-quality speech from 5860-dimensional PPGs. The reduction by more than two orders of magnitude may simply too large a reduction.

Secondly, the nature of the data in the respective speech corpora is, despite the data cleaning process, noisier and less uniform than standard corpora used in speech

synthesis, such as LJSpeech. This is primarily due to the spontaneous nature of the speech, which is is more difficult to re-create with a neural network than speech read by a professional voice actor.

6.2 Transcription-Based Speech Synthesis and Accent Transfer

7 Conclusions

It should be noted that this work does not make any normative prescriptions about a "correct" dialect of English, nor does it condone such prescriptions. In the author's view, different dialects are equally valid. Applications of accent transfer, such as accent reduction, may be motivated by practical concerns, such as the desire to decrease the difficulty of understanding or being understood by an interlocutor. However, these considerations do not imply the superiority of any accent over another. The focus on one of the two theoretically possible directions of transfer, namely Indian English to American English, was motivated by considerations regarding the phones available in each.

This work, unfortunately, was not able to improve upon the results of XXX regarding speech synthesized from PPGs. However, an interesting direction for future work might be to work with multi-accent or even multi-language acoustic models. Conceivably, speakers with different accents would be distinguishable in such a phonetic space, rendering the transfer models more useful.

The two most interesting novel contributions in this work are the demonstration of speech synthesis from IPA, and the presentation of a method for transcription-to-transcription mappings.

*** TO BE ADDED - WAITING FOR RESULTS ***

Literatur

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, 2018.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

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