Modeling Timbre for Neural Singing Synthesis

Methods for Data-Efficient, Reduced Effort Voice Creation, and Fast and Stable Inference

TESI DOCTORAL UPF / 2022

Merlijn Blaauw

Directors de la tesi

Dr. Emilia Gómez Gutierrez
Dr. Jordi Bonada Sanjaume

upf. Universitat Pompeu Fabra Barcelona

Music Technology Group
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona, Spain

This document was typeset using LuaLaTeX and KOMA-script. Fonts used are Minion 3 for main text, Myriad Pro for sans serif text in figures and tables, and Latin Modern Math for math.

Acknowledgments

This thesis has been carried out at the Music Technology Group (MTG) of Universitat Pompeu Fabra (UPF) in Barcelona, Spain, from October 2018 to February 2022, with some relevant publications the year prior. I am very thankful to all the people who have been supportive of this work during this time. I am particularly grateful to my thesis supervisors, Dr. Emilia Gómez and Dr. Jordi Bonada, the latter of which has been a close collaborator on all of the research contained in this thesis. Parts of this work have been funded by the Spanish Ministry of Economy and Competitiveness under the CASAS project (TIN2015-70816-R), and the Horizon 2020 EU Programme for Research and Innovation under the TROMPA project (H2020 770376). As an industrial partner on these projects, Voctro Labs provided some of the datasets used in this work. I would also like to thank Yamaha Corporation, Japan, who were supportive throughout this time and also allowed some of their proprietary datasets to be used for research purposes. NVIDIA Corporation provided a number of hardware grants that were particularly helpful in the early stages of this work. Finally, I would like to extend my gratitude to all colleagues, friends and family who have in one way or another supported this work. In particular, to my closest family, Iciar, Annemiek and Mila, without whose time, understanding and support I could not have completed this dissertation.

Abstract

Singing synthesis has seen a notable surge in popularity in the last decade and a half. Music producers use this technology as an instrument, there is an audience for music with synthetic vocals, and an entire range of cultural phenomena surrounding singing synthesis has emerged. At the time of starting this work, the prevailing approaches for singing synthesis were concatenative synthesis on the one hand, and hidden Markov model synthesis on the other. Concatenative synthesis was state of the art in terms of quality, but lacked flexibility due to being based on signal processing, heuristics and carefully prepared data. By contrast, hidden Markov model synthesis is based on data-driven machine learning, which brings a certain degree of flexibility, but was never able to match the sound quality of concatenative synthesis. At the same time, the field of text-to-speech started to shift towards powerful new deep learning models that have shown to be able to combine high-quality results with a high degree of flexibility. In this dissertation, we try to answer whether similar models can also live up to this potential for singing synthesis. We also try to answer whether these approaches allow fast and stable synthesis, qualities important for many real-world applications. Finally, we try to answer whether the flexibility that the deep learning approaches offer allows creating new voices with smaller amounts of data, and less effort (time, expert knowledge), which is a notable bottleneck in older approaches. To this end, we propose a number of singing synthesis models, and evaluate them, principally through listening tests. The first part of this dissertation focuses on modeling timbre, via autoregressive and non-autoregressive models. The second part focuses on improving data efficiency through voice cloning, reducing the voice creation effort by using a sequence-to-sequence mechanism that requires fewer annotations, and a semi-supervised model which combines supervised pre-training with unsupervised training of a new target voice. Through our experiments, we show deep learning methods can not only outperform the previous state of the art, they can also allow for a significantly reduced voice creation effort. With our work on these elemental problems in singing synthesis, we hope that future research can advance the field further by focusing on topics such as expression, user control and non-modal voice qualities.

Resumen

La síntesis de canto ha visto un aumento notable en popularidad en la última década y media. Los productores de música utilizan esta tecnología como instrumento, existe una audiencia para la música con voces sintéticas y ha surgido toda una gama de fenómenos culturales en torno a la síntesis del canto. Al momento de comenzar este trabajo, los enfoques principales para la síntesis de canto eran la síntesis concatenativa por un lado y la síntesis basada en modelos ocultos de Márkov por el otro. La síntesis concatenativa era estado del arte en términos de calidad, pero carecía de flexibilidad debido a estar basado en el procesamiento de señales, heurísticas y datos cuidadosamente preparados. Por el contrario, la síntesis basada en modelos ocultos de Márkov se basa en el aprendizaje automático usando datos, lo que brinda cierto grado de flexibilidad, pero nunca pudo igualar la calidad de sonido de la síntesis concatenativa. Al mismo tiempo, el campo de «text-to-speech» comenzó a cambiar hacia nuevos y poderosos modelos de «deep learning» que han demostrado ser capaces de combinar resultados de alta calidad con un alto grado de flexibilidad. En esta tesis, tratamos de responder si modelos similares también pueden estar a la altura de este potencial para la síntesis de canto. También tratamos de responder si estos enfoques permiten una síntesis rápida y estable, cualidades importantes para muchas aplicaciones del mundo real. Finalmente, tratamos de responder si la flexibilidad que ofrece el «deep learning» permite crear nuevas voces con cantidades más pequeñas de datos y menos esfuerzo (tiempo, conocimiento experto), lo cual es un cuello de botella importante en los enfoques previos. Para ello, proponemos una serie de modelos de síntesis de canto y los evaluamos, principalmente a través de pruebas de escucha. La primera parte de esta tesis se centra en el modelado del timbre, a través de modelos autorregresivos y no autorregresivos. La segunda parte se centra en mejorar la eficiencia de los datos a través de la clonación de voz, reduciendo el esfuerzo de creación de voz mediante el uso de un mecanismo «sequence-to-sequence» que requiere menos anotaciones y un modelo semisupervisado que combina un entrenamiento previo supervisado con un entrenamiento no supervisado de la nueva voz deseada. A través de nuestros experimentos, mostramos que los métodos de «deep learning» no solo pueden superar el estado del arte anterior, sino que también pueden permitir un esfuerzo de creación de voz significativamente reducido. Con nuestro trabajo sobre estos problemas elementales en la síntesis del canto, esperamos que la investigación futura pueda avanzar más en el campo centrándose en temas como la expresión, el control del usuario y las cualidades de voz no modales.

Resum

La síntesi del cant ha experimentat un notable augment de popularitat en l'última dècada i mitja. Els productors musicals utilitzen aquesta tecnologia com a instrument, existeix un públic interessat en la música amb veus sintètiques i ha sorgit tota una sèrie de fenòmens culturals al voltant de la síntesi del cant. En el moment d'iniciar aquest treball, els enfocaments predominants per a la síntesi del cant eren la síntesi concatenativa d'una banda i la síntesi basada en els models ocults de Màrkov de l'altra. La síntesi concatenativa era l'estat de l'art en termes de qualitat, però mancada de flexibilitat perquè es basa en el processament del senyal, l'heurística i les dades curosament preparades. Per contra, la síntesi basada en models ocults de Màrkov es fonamenta en l'aprenentatge automàtic a partir de dades, que aporta un cert grau de flexibilitat, però que mai no ha igualat la qualitat de la síntesi concatenativa. Al mateix temps, el camp «text-to-speech» va començar a canviar cap a nous models potents de «deep learning» que han demostrat ser capaços de combinar resultats d'alta qualitat amb un alt grau de flexibilitat. En aquesta tesi, intentem respondre si models similars també poden estar a l'altura d'aquest potencial per a la síntesi del cant. També intentem respondre si aquests enfocaments permeten una síntesi ràpida i estable, qualitats importants per a moltes aplicacions del món real. Finalment, intentem respondre si la flexibilitat que ofereix el «deep learning» permet crear noves veus emprant menors quantitats de dades i menys esforç (temps, coneixement expert), que és un dels grans coll d'ampolla dels enfocaments anteriors. Per a això, proposem una sèrie de models de síntesi de cant, i els avaluem, principalment mitjançant proves auditives. La primera part d'aquesta tesi se centra en la modelització del timbre, mitjançant models autoregressius i no autoregressius. La segona part se centra a millorar l'eficiència de les dades mitjançant la clonació de veu, reduint l'esforç de creació de veu mitjançant l'ús d'un mecanisme «sequence-to-sequence» que requereix menys anotacions, i un model semisupervisat que combina un entrenament previ supervisat amb un entrenament no supervisat de la nova veu desitjada. A través dels nostres experiments, mostrem que els mètodes de «deep learning» no només poden superar l'estat de l'art anterior, sinó que alhora permeten un esforç de creació de veu significativament reduït. Amb el nostre treball sobre aquests problemes elementals en la síntesi del cant, esperem que properes investigacions puguin avançar encara més centrant-se en temes com l'expressió, el control d'usuari i les qualitats de veu no modals.

Contents

List of figures

List of tables

Introduction 1

Singing synthesis, or singing voice synthesis, can be defined in general terms as the task of artificially generating vocal sounds in the context of a musical composition or performance. For this work, however, we will utilize a more specific definition; here, we consider singing synthesis to be the task of modeling a specific singer's voice, in order to be able to generate a digital waveform corresponding to any given symbolic score with lyrics. Ideally, the model would be able to render a performance of the score that could pass as a recording of the original singer. This task thus includes aspects of expressive interpretation of a musical score, as well as the actual rendition of the acoustic signal. In this work, we will focus mostly on the latter part, sometimes referred to as timbre modeling, although expression modeling is also discussed to some extent, in order to present a full working system capable of performing our task definition.

1.1 Motivation

As a musical instrument, the human singing voice is very interesting, complex and challenging to model. Just from a signal perspective, its timbre includes linguistic content corresponding to the lyrics, the singer's identity, as well as dynamics and other expressive aspects related to voice color. The voice signal is generally continuous, but also includes obstruent phases, and can be voiced, unvoiced, have mixed voicing, or be transient-like. Additionally, there are several non-modal phonation modes, which include regular and irregular modulations, extremely low pitches, aspiration, and many other phenomena. Additionally, from a more musical perspective, there is a myriad of other factors at play, such as legatos, portamentos, ornaments, vibratos, and so on. Yet, for all its complexity, nearly everyone has some degree of familiarity with the singing voice. Even non-singers typically have a reasonable understanding of how the sounds are produced and what expression the singer is trying to convey.

There is something almost magical about a machine convincingly performing a task thought to be innately human, such as speaking emotionally or singing. While we are perhaps still quite a long way off from fully autonomous and indistinguishable-from-real synthetic singing, this is a fascinating aspect of such technology for many people.

One of the interesting things of research on singing synthesis is that it is a very multi-disciplinary field. Singing synthesis lies on the intersection of many different scientific and artistic areas; signal processing, speech processing, machine learning, music, and also incorporating knowledge from fields such as phonology, physiology of the voice, and studio engineering.

Advances in singing synthesis have the potential to have a notable cultural impact. In the last decade or so, singing synthesis technology has seen a huge surge in popularity with the advent of commercial software such as VOCALOID, and in particular the virtual singer Hatsune Miku (depicted in Figure 1.1). These products have shown that there is a demand from musicians to have access to such technology, as well as an audience willing to consume music that includes synthetic singing, especially in Japan and other East Asian countries. The cultural impact of such virtual singers has extended well beyond just producing music with synthetic vocals. For instance, there have been many highly popular "live" concerts, number one charted albums, many videos with several tens of millions of views, video games, depictions of the characters used in racing and spacecraft, and even a "marriage" to the virtual character. Arguably more importantly, this technology and these characters have sparked countless artistic endeavors and collaborations, for instance, people making music videos, writing lyrics, producing music, "programming" vocals, making 3-d models, drawing novel art, writing stories, and so on. We can speculate that any notable advancement of singing synthesis technologies, will also have a positive effect on all of these surrounding cultural aspects. Additionally, improvements in singing synthesis may also bring a more widespread acceptance of synthetic vocals in music to the Western world.

One of the principal applications of singing synthesis is as a tool for music producers, allowing them to create music with vocals using little more than a desktop computer. This opens up using vocals with any lyrics to people who do not sing themselves, want to use a voice with a timbre different from their own, or do not have access to other singers. In cases where virtual singers become well known, it allows unknown producers to have access to a famous (virtual) singer. In some cases it may even be possible to create a model using recordings of a deceased singing, allowing new material to be created. Using virtual singers allows for easier collaboration and remixing. For singers, being able to "share" their voice with other musicians, or potentially "freeze" their voice as they age, can be interesting possibilities.

Besides the application as a tool or instrument for musicians, there are many other possible applications of this kind of technology. Some such applications include social media apps, television and film, games, musical messaging, automatic sonification of

Figure 1.1: Hatsune Miku is a Japanese voicebank for Yamaha's VOCALOID software. Since its release in 2007, it has grown to one of the most famous virtual singers. Thanks to its incredible success, it provides a clear example of the potential cultural impact that this kind of application of singing synthesis technologies can have. Depicted here is the official moe anthropomorphism of Hatsune Miku. [By Crypton Future Media, Inc., licensed under CC BY-NC 3.0.]

score repositories, application in karaoke, automatic translation of songs into different languages, marketing applications, application in singer training, choir rehearsal, and so on. Of course, nearly all such applications will require additional research to adapt the technology to the specific task at hand.

Lastly, on a personal note, I have been involved in singing synthesis for almost two decades, including collaborating on some of the most prominent applications of this technology, such as the VOCALOID software. I think this gives me a good understanding of not only many of the technical aspects of this research area, but also some of the cultural context of this technology. Moreover, I have witnessed a number of different "generations" of technology, each opening up more possibilities and getting closer to the goal of truly natural synthesis. I think therefore it only makes sense for me to focus on this topic for a doctoral dissertation.

1.2 Opportunities and challenges

At the time of starting the work contained in this thesis, singing research had already been researched for several decades. The technology had matured enough to be integrated into commercial software, and the sound quality was already mostly good enough to be included in professionally produced music. At the time, the two prevailing approaches were concatenative synthesis (see §2.3.1) and hidden Markov model (HMM)-based synthesis (see §2.3.2).

Exhaustively listing all remaining challenges in singing synthesis is difficult, as applications and user expectations are constantly shifting. However, at the time of starting this work, there were a number of areas clearly lacking. Firstly, the sound quality of singing synthesizers is something that seems to remain challenging. While techniques such as concatenative synthesis are able to obtain results that are good enough for many applications, there is still a notable gap in quality compared to natural singing. This is especially true for languages that are more complicated in terms of phonetics, such as English, where concatenating diphones tends to fall short. Another area that nearly always leaves room for improvement is musical expression. For instance, pitch generation based on heuristic rules tends to lack certain imperfections that make the performance sound expressive and truly natural. Related to expression is the area of controllability. It is often not enough to generate singing voice completely automatically, but we typically require some amount of user control. Similarly, requiring so much detailed control that creating good sounding results takes an excessive effort is also not desirable, and a notable issue with some of the current commercially available singing synthesizers. Some aspects of the singing voice are arguably not essential for generating basic singing synthesis, but are still important and frequently used in real singing. These include non-modal voice types, such as breathy voices, rough voices, growls, vocal fry, and so on. Finally, one major challenge in singing synthesis is the effort required to create a new voice. For instance, for concatenative synthesis, creating a new voice may take an expert several months to complete.

Coinciding with the start of this work, text-to-speech (TTS) went through something of a paradigm shift thanks to a new wave of powerful deep learning models. In a relatively short time, these models have arguably advanced the field more than several decades of research that came before them, and have now reached a point where some systems are almost indistinguishable from human speech (see §2.4.2). Not only that, but this type of model is much more flexible than previous approaches, allowing them to be extended more easily. As, at the start of this work, these models had not yet been applied to singing synthesis, this provided a clear opportunity to try to apply similar approaches in order to advance the field of singing synthesis as well.

1.3 Scope and objectives

Working on all remaining challenges in singing synthesis within this thesis is not feasible, and therefore we must limit its scope somewhat. The first major limitation is to mostly focus on modeling timbre. This means aspects such as pitch and timing are not the principal focus of this work. One significant advantage of focusing only on a single aspect of the voice is that this greatly simplifies evaluation, e.g., comparing two examples where only timbre is different is much easier than two examples where timbre, pitch and timing are all potentially different. That said, we do provide some work on a "complete" model of the singing voice, which can synthesize vocals from a score with lyrics. For modeling timbre, we propose a number of different deep learning models, each with different weaknesses and strengths. Here, the main objective is to obtain models able to beat the then state of the art in terms of synthesis quality, and at the same time provide great flexibility. We also consider aspects important for practical application, such as efficient and stable synthesis.

Besides principally focusing on modeling timbre, out of all the remaining challenges in singing synthesis, we limit ourselves to researching ways to make voice creation require less effort and be more data efficient. Coming from a background in concatenative synthesis, these issues often formed a limiting factor for practical application. For instance, to create a new voice segmenting recordings into diphones, selecting best occurrences, editing audio, and so on, can take months of highly skilled work. Similarly, exactly what to record in order to obtain the ideal set of diphones is very non-trivial, especially for "phonetically complicated" languages that have a lot of coarticulation, such as English. Thus, if we can create a system where we can create new voices from (virtually) any kind of recording of natural singing and not have to do a lot of manual annotations, this would radically reduce the voice creation effort. Similarly, for many applications having to record several hours of a professional singer in order to create a new voice is not feasible. Thus, methods which increase the data efficiency of models are very welcome. Unlike older methods, such as concatenative synthesis, which have very rigid data requirements, we feel that these are areas where deep learning methods can potentially really shine.

1.4 Thesis outline

❧ Chapter 2 "Background": Background information to contextualize the thesis work, including some early history of singing synthesis (§2.1), a taxonomy of different kinds of singing synthesizers (§2.2), an overview of traditional and modern approaches to text-to-speech and singing synthesis (§2.3 and §2.4), and a working model of the singing voice (§2.5). This chapter does not have to be read to understand the main body of the dissertation, although it may be helpful to introduce and contextualize the topic.

❧ Chapter 3 “Methodology”: This chapter provides some basic building blocks on which the main body of the dissertation is built. In §3.1, we discuss how we can structure singing synthesizers, common signal processing blocks, and typical acoustic and control features. Datasets for singing synthesis are discussed in §3.2, including available public datasets, and how to record and annotate new ones. Evaluation of singing synthesis models is discussed in §3.3. Especially readers familiar with the subject may not need to read this chapter in its entirety. Instead, they may follow references to this section from the main body of this dissertation as is needed.

❧ Part I “Modeling timbre” (Chapters 4–6): This part of the thesis talks about different ways of modeling timbre. First, we discuss an autoregressive model with a focus on comparing with more traditional methods that were the then state of the art, in particular by using datasets that are suitable for all of these systems (§4.2). As a second step, go beyond the constraints imposed by traditional methods, and train a complete system on natural singing, which can generate singing voice from a score with lyrics (§4.3). Next, we consider a non-autoregressive model, which offers different strengths and weaknesses compared to the autoregressive models, and employs self-attention to allow more coherent modeling of timbre over time (Chapter 5). Finally, we discuss a fast, non-autoregressive method of generating waveforms from the intermediate acoustic features generated by the previously discussed timbre models (Chapter 6).

❧ Part II “Data-efficient and reduced effort voice creation” (Chapters 7–9): This part of the thesis talks about creating new voices in ways that require a relatively small amount of data and less effort, in particular in terms of manual or semi-automatic annotations. First, we consider applying voice cloning techniques to singing synthesis in order to create voices from small amounts of target data (Chapter 7). Next, we adapt the previously described non-autoregressive model based on self-attention to behave like a sequence-to-sequence model, where phoneme timings are inferred from note onsets only, thus notably reducing the annotation effort required to create a new voice (Chapter 8). Finally, we discuss a semi-supervised system that first uses a supervised pre-training step, but then allows us to create new voices in a completely unsupervised manner, that is, from audio only (Chapter 9).

❧ Chapter 10 “Conclusions”: Here we summarize and conclude our work, provide a list of the most important contributions of our work, and discuss some possible future avenues of research.

Background | 2

In order to contextualize the topic of singing synthesis, we will first provide some background for this thesis. This background includes some early history of singing synthesis (§2.1), a taxonomy of types of singing synthesizers (§2.2), an overview of some traditional and modern approaches to singing synthesis (§2.3 and §2.4), and a working model of the singing voice (§2.5).

When reviewing past and present approaches, we cannot consider singing synthesis in isolation. In particular, the field of text-to-speech (TTS) is closely related, and must also be taken into account. In fact, one could argue that many of the advances in singing synthesis have been driven by advances in TTS. Therefore, we will discuss relevant works on TTS, as well as singing synthesis.

2.1 Early history of singing synthesis

Singing synthesis has a long history, much of which in parallel with the advances in speech synthesis. Joseph Faber's mechanical speech synthesizer, Euphonia (depicted in Figure 2.1), was reported to have sung *"God Save The Queen"* at its 1846 exhibition in The Egyptian Hall, London[1]. Anecdotally, this exhibition is said to have made a great impression on Alexander Melville Bell, father of Alexander Graham Bell, inventor of the telephone and founder of the American Telephone and Telegraph Company (AT&T). Almost a century later, it is precisely AT&T's Bell Labs, who would demonstrate their Voder (Voice Operating DEmonstratoR) based on Homer Dudley's work on the vocoder, at the 1939 and 1940 New York World's Fair (depicted in Figure 2.2). During these exhibitions, a skilled operator would make the electrical machine sing *"Auld Lang Syne"*, recordings of which are still available[2]. A couple of decades later, at this same Bell Labs, an IBM 704 mainframe became the first computer to sing. This earliest known recording of digital singing synthesis was the song *"Daisy Bell (Bicycle Built for*

[1]Article *"The Speaking Machine"* in Punch, or the London Charivari, Volume 11 (July – December 1846), published at The Office, London, 1846.

[2]Smithsonian Speech Synthesis History Project (SSSHP) 1986–2002, tape 51, *"AT&T Bell Labs Voder from world's fair exhibits (New York, San Francisco) 1939–40 era"*.

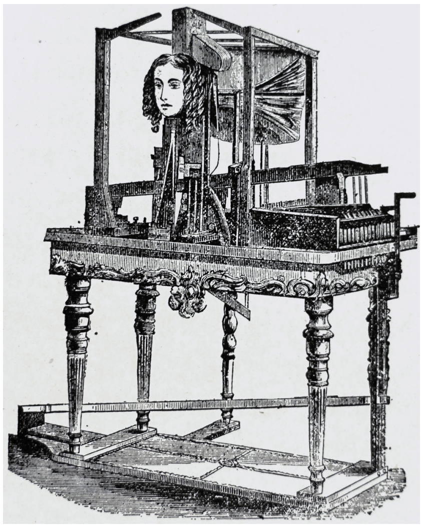

Figure 2.1: An illustration of Joseph Faber's Euphonia talking machine. Adapted from a poster advertising an exhibition by B. P. Barnum during 1873 in his museum. [Public domain.]

Two)"[3], and was programmed by Max Mathews (musical accompaniment), John Kelly and Carol Lochbaum (vocals). Science fiction author Arthur C. Clarke witnessed this remarkable demonstration and was so impressed that he famously incorporated it a climatic scene of the novel and screenplay for *"2001: A Space Odyssey"*, where the HAL 9000 computer sings *"Daisy"* during its gradual deactivation.

Up to this point in time, most of these early examples of singing synthesis have been in the context of more lighthearted demonstrations of speech synthesis systems. However, building on these works, soon also more rigorous research into synthesis of the singing voice started. In particular, the work by Sundberg and colleagues at the KTH is an important body of work in this direction (an overview of which is available in Sundberg, 2006). In the '70s, Gunnar Fant's OVE (Orator Vox Electrica) speech synthesizer was modified by Jan Gauffin to feature continuous controls, which resulted in the analog Music and Singing Synthesis Equipment (MUSSE) singing synthesizer (Larsson, 1977). A later digital version of this system, MUSSE DIG (Carlsson-Berndtsson and Sundberg, 1993), would include computer control which allowed precise studying of voice parameters, using *analysis by synthesis* methods (Sundberg, 1989). These studies resulted in a set of rules related to acoustic properties of the voice, but also the performance of the singer. Such an approach obtained excellent results for the time; a synthetic rendition of one of the Vocalise pieces by Panofka presented at a symposium[4] in 1977 is said to have been the only piece to receive spontaneous applause from the

[3]Various artists – *"Music From Mathematics"*, Decca, DL 79103, USA, 1962. Track A3, Max Mathews – *"Bicycle Built for Two"*.

[4]IRCAM symposium on the Psychoacoustics of Music, Paris, France, 11–13 July, 1977. The presented piece was a Vocalise from Heinrich Panofka's *The Art Of Singing: 24 Vocalises, Opus 81*, with live piano accompaniment.

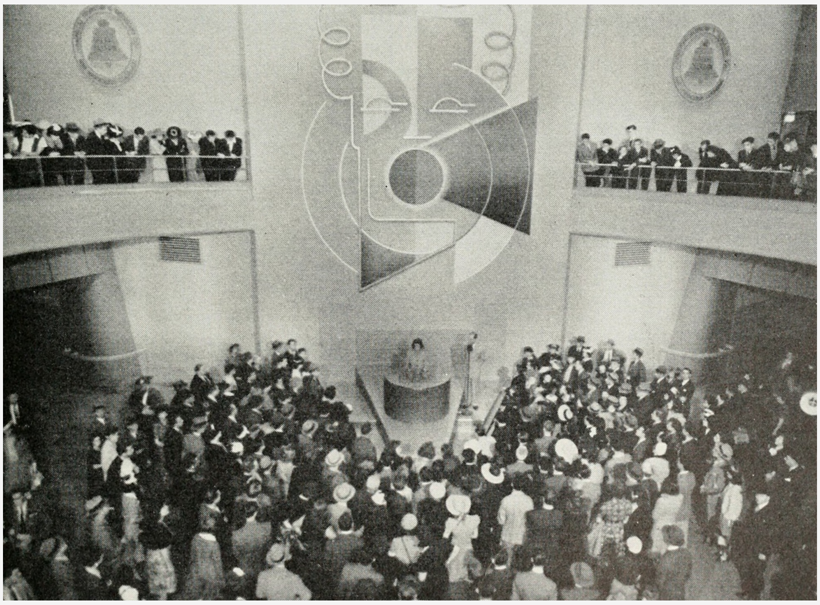

Figure 2.2: Crowds gather at the 1940 New York World's Fair to see Bell Labs' Voder exhibit. This exhibit of some of the earliest works in speech and singing synthesis is reported to have received over 5,000,000 visitors. [Taken from Bell Telephone Quarterly Vol. XIX, January 1940, p. 65. Public domain.]

audience. A notable limitation of these early works is the lack of consonants, although this was later partially addressed (Zera et al., 1984).

Some other notable early performances and pieces involving singing synthesis include "*Speech Songs*" (Dodge, 1989), a duet between Titze and his synthesizer embodied as *Pavarobotti* singing Pucccini's "*Nessun Dorma*" (Titze and Story, 1993), and the creation of a virtual Castrato singer via morphing techniques for a musical film about Farinelli[5] (Depalle et al., 1994).

2.2 Acoustic, articulatory and performative models

Defining a broad taxonomy of approaches in singing synthesis will help focus the review of existing works on the approaches most relevant to this dissertation. To this end, we classify models as either *acoustic* or *articulatory*, with an additional qualifier of

[5]The 1994 film "*Farinelli*", about the celebrated 18th century Italian castrato singer Farinelli, by French director Gérard Corbiau.

whether they are *performative*. Roughly following Cook (1996), acoustic approaches model the acoustic voice signal directly or in some perceptually relevant representation, whereas articulatory approaches model the physical (physiological) voice production system. Models that are considered performative focus on the real-time playability of the synthesizer as a musical instrument. Such models are not directly comparable to the non-performative models, because the focus on performative aspects generally has negative consequences for other aspects of the model, such as sound quality or singing lyrics exactly. In this work, we focus on non-performative, acoustic models, but will discuss some of the relevant works in each group next for completeness.

Acoustic models can be seen as descendants of Dudley's early work on the vocoder (Dudley, 1940). This class of models is sometimes also called spectral models, but this is a bit of a misnomer as these models may use any given acoustic representation of the voice signal, including time-domain and other non-spectral representations. As a rule of thumb, these models tend to model a specific speaker's voice, and often involve both an analysis and synthesis component. As with the articulatory models, these models may contain some physiological motivation or correspondence, but this is not the main focus of the model. Some early approaches of this kind include FM synthesis (Chowning, 1989), FOF (from the french *"Formes d'Ondes Formantiques"*) in the Chant synthesizer (Rodet, 1984), and other FOF-like voice pulse models (Kaegi and Tempelaars, 1978). Modern approaches to acoustic models are discussed in §2.3.1, §2.3.2 and §2.4.

Articulatory models, sometimes called physical models, can be seen as descendants of Kelly and Lochbaum's work on the acoustic tube (Kelly and Lochbaum, 1962). Some works in this area include Kob (2002), Birkholz (2007), and Kröger and Birkholz (2009). This more direct, physiological modeling of the voice production system, however, has a number of drawbacks. The (physiological) controls of this kind of system are generally not intuitive, and make generating realistic results difficult. Often there is a greater focus on vowels than consonants. Often the model is a general voice model, not modeling a specific singer's voice. Ultimately, this kind of model has difficulty competing with acoustic approaches in terms of sound quality and realism, and is thus mainly interesting for studying the voice production system itself, or in certain performative applications where sound quality is less of a concern.

Performative models, which can be either acoustic or articulatory, focus on real-time playability as a musical instrument. Often the interfaces used resemble some of the very earliest works on singing synthesis, prior to computer control, such as the Euphonia and the Voder (depicted in Figure 2.3). A distinction has to be made between performative systems and systems which are merely real-time, as these lack study of the musical interaction with the system. Perhaps the most prominent performative singing synthesizer is Cantor Digitalis by Christophe d'Alessandro's group (e.g., Feugère et al.,

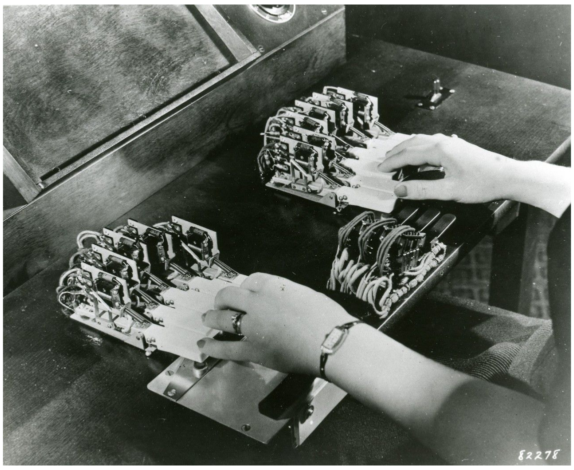

Figure 2.3: A photograph of the controls of Bell Labs' Voder speech and singing synthesizer. The operator would manually form each syllable using complex button sequences. It is reported that it takes about a year of practice to be able to produce fluid speech. [Public domain.]

2017). This system uses a two-handed, tablet-based interface. One hand controls a point on a 2-d vowel space, the other hand controls pitch and other parameters. Being a performative system, there have been many concerts involving this system, often with groups of musicians all playing the instrument.

2.3 Traditional approaches

We will first review some traditional approaches to TTS and singing synthesis. Here, we consider approaches "traditional" when they use methods predating the use of neural networks, which is the current prevailing paradigm. At the time of starting this work, traditional methods were still widely considered state of the art.

2.3.1 Concatenative synthesis

The basic idea behind concatenative synthesis is to take short samples of recorded sound from an inventory, and then generating a new sequence by rearranging, transforming and concatenating these samples. In TTS and singing synthesis there are two commonly used forms; diphone synthesis and unit selection. In diphone synthesis, each sample is a diphone, the combination of two halfphones, capturing the transition from one sound to the next. Generally, the inventory will contain only a single entry for each diphone of the language in this case. In unit selection, each sample typically is a variable-length unit, and the inventory will generally be much larger with potentially many variants of the same unit in different linguistic contexts. For TTS, unit selection quickly surpassed diphone synthesis, as the units can contain part of prosody and a wider range of phonetic contexts (Hunt and Black, 1996; Black and Taylor, 1997). For a long time, such systems have been state of the art. Especially in domain-specific applications where the inventory of units can be recorded with a specific task in mind, these methods can have excellent results. For singing synthesis, diphone synthesis has been traditionally more popular, as recording large inventories with a high degree of coherence tends to be more problematic for singing than for speech. However, some ideas of unit selection have also been incorporated in some concatenative singing synthesis systems.

The basic steps involved in concatenative singing synthesis are as follows; (i) select units from the inventory, often by scoring candidates and using Dynamic Programming techniques to find the optimal sequence, (ii) transform units to match the target (pitch shifting, time scaling, etc.), and (iii) concatenate units, by smoothing timbre discontinuities and ensuring phases are continuous. An overview of this process is depicted in Figure 2.4. In this simplified schema we assume a target, e.g., a sequence of phonemes with durations and F0, to be given as input. However, there are typically also generated from an input score with lyrics as part of the system. The performance of this kind of system tends to be highly dependent on careful tuning of various heuristics and meticulous design of the signal processing algorithms involved.

Another key element in the success of concatenative systems is the preparation of the inventory, sometimes called the database or voicebank. Typically, specialized recording scripts have to be prepared to ensure coverage of all required units, as the system has no capacity to generalize. In single inventory systems, in particular, special care must be taken from which phonetic contexts the units are extracted, as certain contexts can introduce undesirable coarticulation effects, such as nasalization of vowels. Likewise, recordings generally have to be done with great care. Once the material has been recorded, it must be annotated with highly accurate phoneme or diphone boundaries, typically requiring manual correction. All samples must be carefully inspected for things like noises or irregular pronunciations, finding replacements for those not suitable. Overall, this tends to be a lengthy, tedious process. More information on this topic can be found in §3.2.

In concatenative singing synthesis, using an inventory derived from natural singing (i.e., expressive songs) generally has poor results because the different samples will come from different musical contexts and lack coherence. In such cases, there will be very noticeable "patchwork" artifacts. One workaround for this issue is to record what we call pseudo singing. Pseudo singing is short phrases sung in a consistent song-like timbre, at a constant pitch, and often with a controlled cadence and relatively clear articulation (see §3.2.2). Recording several constant pitches of pseudo singing allows covering timbres corresponding to a wider pitch range. An important negative consequence of this approach is that pseudo singing is essentially lacking any expression, and therefore this data cannot be used to generate expressive pitch. This means that expression has

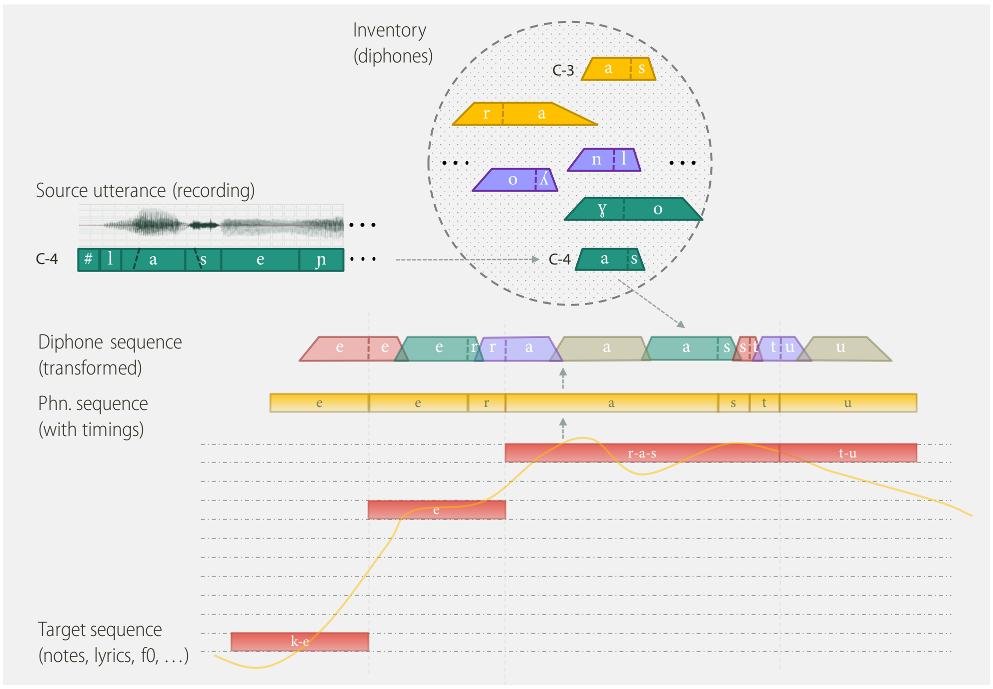

Figure 2.4: A schematic view of concatenative synthesis. The top of the figure shows (part of) the diphone inventory of the voice, with different colors indicating different pitches. To the left of the diphone inventory, a source utterance (recording with phoneme and diphone segmentation) is shown. From this utterance, the C-4 [a-s] diphone is included in the inventory. At synthesis, from the score (notes with lyrics) typically an F0 sequence is generated together with a sequence of timed phonemes. The next step is to select a sequence of diphones from the inventory following certain criteria, and to transform the samples to match target pitch and durations. Finally, diphones are concatenated to produce a continuous output waveform.

to come from another source, potentially causing a lack of coherence between timbre and expression.

Some early works on diphone concatenative singing synthesis include Bonada et al. (2001) and Uneson (2002), which were later built upon, resulting in more sophisticated systems such as Bonada and Serra (2007), and more recently Ardaillon (2017) (chapter 3, pp. 63–84). The winner of the 2016 Singing Synthesis Challenge, used a concatenative approach closer to unit selection (Bonada et al., 2016), albeit with a modest size inventory. It is also worth noting that the to-date most widely used commercial singing synthesizer, VOCALOID, is also based on concatenative synthesis (Kenmochi and Ohshita, 2007; Bonada et al., 2006). All of the listed systems tend to use external models for generating F0, for instance using unit selection (Umbert et al., 2013; Umbert et al., 2015), or parametric models (Ardaillon et al., 2016).

The main advantage of concatenative methods for singing synthesis is that quality can be very good. Thus, this approach was widely considered state of the art for almost two decades, until recent modern methods surpassed it. Especially in cases where the selected unit is close to the target in terms of pitch, duration and phonetic context, only minor transformations have to be applied and thus the sound quality will be close to that of a recording. In practice, this is mostly dependent on the quality of the recordings and their annotations, which resulted in voice creation often taking several months of work. The language of the voice also plays an important role in the quality of the results. Languages with few, highly contrasted vowels and less frequent use of consonant clusters tend to work better – for instance, Japanese or Spanish is notably easier than English. Another advantage is that artifacts in the voice can often be easily corrected, at least as far as allowed by the source material. While this can take time, it can be important for productization and something that the more "black box" machine learning approaches generally do not offer.

On the other hand, the main downside of such systems is that they are very inflexible and significantly improving them takes a huge amount of effort and can quickly become impractical. For instance, as these systems do not generalize, covering several voice qualities (e.g., a "soft" or "powerful" voice type) requires recording the full inventory of samples again, at several pitches. Even assuming that the required annotations could be obtained automatically with sufficient accuracy, just the sheer volume of recordings quickly becomes prohibitive. Some proposed approaches to widen the range of voice qualities based on signal processing or morphing, e.g., Bonada and Blaauw (2013), do allow to work around this problem to some degree, but often have difficulty delivering very realistic, dynamic results. The practical requirement of using pseudo singing recordings is also a very important limiting factor as it forces a disconnect between the kind of voice used to build the system and the kind of voice we wish to generate. For instance, capturing natural variations in timbre or coarticulation effects in their natural context becomes difficult. The single inventory diphone approach also has notable difficulty reproducing natural pronunciation of phonetically more complex languages such as English. This is largely a result of the rigidity of such systems – for instance, obtaining highly accurate phonetic transcription for English recordings can already be very difficult, but such discrepancies cause artifacts in diphone systems, while machine learning approaches can handle such issues much more elegantly.

2.3.2 Hidden Markov model synthesis

A hidden Markov model (HMM) is a probabilistic generative model for modeling time series, assuming there are a number of discrete, hidden states underlying the observed data to be modeled. For an observed time series, $\mathbf{x} = \{x_1, x_2, \dots, x_T\}$, there is a corresponding state sequence, $\mathbf{q} = \{q_1, q_2, \dots, q_T\}$, with each state index, $1 \leq q \leq N$, and N the number of discrete states. The model then defines the joint probability,

$$p(\mathbf{x}, \mathbf{q}) = p(x_1 \mid q_1)p(q_1) \prod_{t=2}^{T} p(x_t \mid q_t)p(q_t \mid q_{t-1}). \tag{2.1}$$

Here, $p(q_t \mid q_{t-1})$, is the *state transition probability* matrix, which is speech applications is often restricted to a left-to-right topology with self-loops, meaning only repeating the current state or moving to the next state index is allowed. The *output probability* for a given timestep, $p(x_t \mid q_t)$, can be defined to be any given distribution, in speech applications often a continuous distribution such as Gaussian or mixture of Gaussians (MoG). The *initial state probability*, $p(q_1)$, defines the probability of starting in a specific state, in our case simply $p(q_1 = 1) = 1$. Learning the parameters of such a model given observations is typically done using expectation-maximization. A more formal and thorough derivation of this model is given in Rabiner (1989), instead we will focus on a brief overview of the use of this kind of model in TTS and singing synthesis.

In speech applications, each phoneme is modeled using an HMM, with a fixed number of states, e.g., typically 5 states per phoneme. Conceptually, states thus correspond to different time segments within a phoneme, e.g., its stationary part, transition parts from previous and to next phoneme, etc. Phonemes are typically not modeled in isolation, as they are heavily dependent on their phonetic context, thus we use so-called *context-dependent* models. For instance, a model can correspond to the central phoneme within a specific triphone or pentaphone context. For singing synthesis, the context also includes musical information related to note pitch, duration, etc. As these contexts become richer, the number of occurrences of each context within the dataset will go towards one, making robust statistical parameter estimation difficult. To mitigate this problem and to allow the model to generalize to unseen contexts, similar contexts are clustered using a decision tree. In this tree, the decisions correspond to a set of "handcrafted" questions, e.g., "Is the previous phoneme voiced?", "Is the central phoneme a plosive?", "Is the next phoneme an [s]?", and so on. As the clustering is typically done independently for each state, it is also referred to as state tying. An overview of this model is depicted in Figure 2.5.

The observed time series $\mathbf{x}$ to be modeled typically tends to be some acoustic features, such as mel-generalized coefficient (MGC) or F0 (see §3.1.3). These are modeled using an output probability distribution, typically a MoG, also known as Gaussian mixture model (GMM). While in speech synthesis the usage of a single Gaussian is more common, the term HMM-GMM is still often used as HMMs first became prominent in speech recognition where multiple mixture components is an important aspect. Multivariate output distributions typically use diagonal covariance, thus not modeling the correlation between feature dimensions within a timestep. Likewise, the probability distribution of a given timestep is assumed to be independent of previous timesteps.

In TTS and singing synthesis, the actual model used is a hidden semi-Markov model (HSMM), where the duration of each state is explicitly modeled using a duration model. In a standard HMM, the state duration probability is implicitly modeled using a geometric distribution[6], which is a very poor fit to the duration of phonemes in speech or singing (and thus the same applies to states within a phoneme). Typically, a Gaussian state duration model is used instead. A related issue is that the statistics within a state are constant, while the acoustic features of speech are typically continuously and smoothly varying. A common approach to mitigate this disconnect between the model and the signal, is to model not only the static features, but also delta and delta-delta features. During synthesis a maximum output probability parameter generation (MOPPG) algorithm (Tokuda et al., 2000) then generates a (mostly locally) smoothed output trajectory. As a typical problem of HMM-based synthesis is oversmoothing, often a parameter generation algorithm considering global variance (GV) is used (Toda and Tokuda, 2007). Such algorithms try to artificially recover the per-utterance variance of the training set during synthesis.

The acoustic (vocoder) features modeled generally consist of multiple components, such as harmonic, aperiodic and F0 components (see §3.1.3). To model these components, a multi-stream model is used, where each component is modeled using a separate model. Typically, these models are completely independent. The F0 stream has a particular issue as it has to model both voiced and unvoiced regions. One solution to this issue is to use a so-called multi-space density that combines a Bernoulli distribution for the

[6]While the probability of a *state transition* is constant w.r.t. the timestep, the probability of a specific *state duration* depends on the duration and thus timestep. E.g., in a coin flip, the probability of getting heads twice in a row is higher than the probability of getting heads twelve times in a row, while the probability of getting heads or tails is constant at each flip.

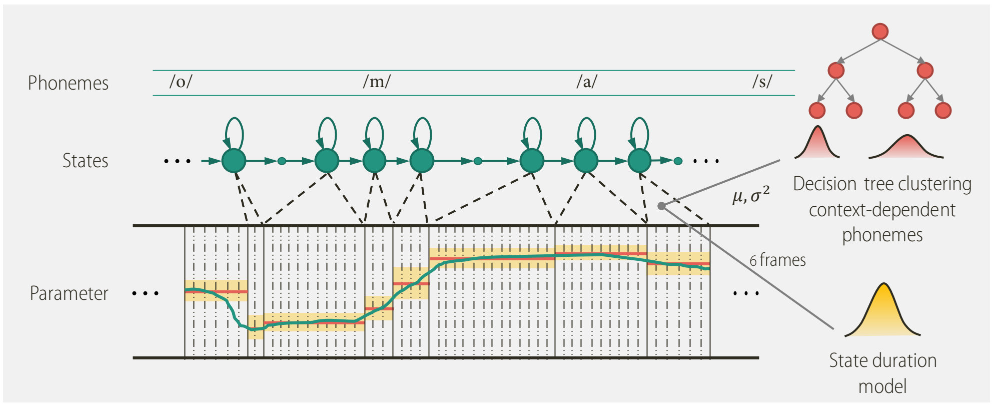

Figure 2.5: Schematic overview of hidden semi-Markov model (HSMM)-based singing synthesis. On the top is an input sequence of phonemes. Each phoneme is modeled by a fixed number of states (here three), with a certain duration obtained from the Gaussian state duration model (bottom-right). The parameters for each state, here Gaussian mean and variance (μ and σ^2), are obtained from a decision tree (top-right), which considers phonetic context, e.g., here it may be the third state of the [a] phoneme in the triphone context [m-a-s]. Finally, from the predicted per-state static and delta, delta-delta parameters, the output continuous parameter is generated (bottom-left). [Adapted from HTS Slides 2.3 by the HTS Working Group, licensed under CC BY 3.0.]

probability of a state being voiced or unvoiced, with a Gaussian distribution to model the probability of the log F0 value within a state when voiced (Tokuda et al., 1999).

The training process of HMMs tends to be rather convoluted, as direct learning of the model parameters with good results is generally not possible. Typically, an iterative approach is used, briefly; (i) start with context-independent models, initialized from scratch or using some initial phonetic segmentation, (ii) re-estimate context-independent models embedded in the sequence of phoneme models, (iii) clone context-independent models to context-dependent models, (iv) embedded re-estimation of context-dependent models, (v) context clustering, (vi) embedded re-estimation, (vii) untying context clusters, (viii) embedded re-estimation, (ix) second context clustering, and finally (x) embedded re-estimation.

Using HMMs for singing synthesis was first proposed by Saino et al. (2006). The Sinsy system (Oura et al., 2010) developed at the Nagoya Institute of Technology (Nitech), is a complete and refined singing synthesis system, which includes data augmentation using pitch shifting (Mase et al., 2010), and later pitch-adaptive training (Oura et al., 2012). Initially designed for the Japanese language, it was later also extended to English (Nakamura et al., 2014). Multi-speaker models have been also been explored (Shirota et al., 2014). Other groups also worked on similar systems, such as Pucher et al. (2016) for German operatic singing, and Li and Wang (2016) for Mandarin Chinese singing.

The main advantage of this approach is that it can jointly model timbre and expression from natural singing. Unlike concatenative methods, which typically require specialized pseudo singing recordings (see §2.3.1). It is a very flexible method, allowing things such multi-speaker models and speaker adaptation. There is no requirement to use diphones, but wider phonetic contexts are considered. In general, the HMM approach is more tolerant w.r.t. small noises or irregularities in the training data, and annotations such as phonetic segmentation is less critical. In fact, the model can be trained with unaligned phonetic and acoustic sequences, although this generally does not achieve the best results. As the model can generalize to unseen phonetic contexts, the design of the recordings scripts to exhaustively cover all common diphones is less critical. Finally, for productization, the synthesis is very fast and the models tend to be very compact.

The main disadvantage of the HMM approach is that the quality of its results has always been a little sub-par compared to concatenative methods. This fundamentally is the result of several of the many assumptions the model makes not holding true. The principal symptom of this is excessively smooth predictions, so-called oversmoothing. One issue is that phonemes do not consist of a small number of pseudo-static segments (states). One example where this becomes apparent in rapidly varying phonemes such as plosives or trills, that will lack definition. In general, the excessive averaging over time within a state will cause overly flat spectra and result in what is often called "buzziness". For long vowels in singing, the small number of states can also cause audible state transitions. Context clustering also tends to result in having to make a tradeoff between a high degree of clustering, resulting in coherent but overly smooth synthesis, and a low degree of clustering, resulting in less averaging, but also potentially more discontinuous synthesis (see §2.4.1 for further discussion). Lack of dependence between streams, particularly the lack of dependence of timbre on frame-wise F0, is also an issue in singing. Finally, while very flexible in theory, in practice this kind of system can be difficult to extend or improve. Changing the model typically involves deriving update rules by hand and implementing modifications in many different points.

2.4 Modern approaches

We differentiate between two groups of neural network approaches, which we call the first and second *waves*. The first wave of approaches coincided with the initial boom in interests in deep learning around 2012. These approaches delivered promising results, but were not clearly better than the state of the art at the time, unit selection synthesis. The second wave, which is still ongoing at the time of writing, we consider to have started with the publication of WaveNet (van den Oord et al., 2016a). This model not only catapulted neural networks for TTS as state of the art in terms of quality and naturalness, but also really pushed the envelope as to the expectation of the capability of neural networks. Prior to this work, direct modeling of the time domain waveform with good results was long thought to be nearly impossible. As a consequence, a renewed interest in applying neural networks to speech synthesis resulted in many recent publications, which as a whole can be considered the current state of the art.

As a historical note, there have been attempts to use neural networks for speech synthesis prior to what we call the first wave, such as those by Tuerk and Robinson (1993) and Karaali et al. (1996). However, these early approaches were heavily limited by the lack of commodities associated with modern deep learning, such as graphics processing unit (GPU) processing, large datasets and modern learning algorithms that allow deeper networks.

2.4.1 First wave of neural network approaches

The early work on the application of neural networks was very much rooted in the preceding work on HMMs. As discussed in §2.3.2, the approach based on HMM, has a number of important inherent limitations. Arguably, the most important of these are (i) general difficulties robustly estimating parameters of a GMM output distribution, (ii) fragmentation of the input feature space due to using a decision tree to cluster contexts, and (iii) modeling variations in time as a sequence of few discrete states per phoneme.

Robustly estimating parameters of the GMM output distribution in the HMM-GMM approach can be difficult. In particular when modeling high-dimensional, highly correlated acoustic features, and when using several mixture components. To alleviate these issues, one early approach combines a decision tree-clustered HMM with deep belief network (DBN) state-output distributions (Ling et al., 2013), essentially replacing the GMM in HMM-GMM. While this mitigates the first issue with the HMM-GMM approach, the latter two remain.

Using a decision tree means that its size is usually a hyperparameter to be tuned. A small tree with few leaf nodes, may result in excessive averaging. A large tree with many leaf nodes will reduce averaging, at the cost of less data per leaf node and poorer generalization (Bengio et al., 2010). In general, such a fragmented representation is not suited to model the complex dependencies between linguistic and acoustic features efficiently. The idea of using a distributed representation offers an attractive alternative and is closely tied to the theory behind restricted Boltzmann machines (RBMs) and

DBNs, where each layer extracts an increasingly higher-level binary (Bernoulli) representation of its input. In Kang et al. (2013) a DBN is used to model the joint distribution of linguistic and acoustic features, thus replacing both decision trees and GMMs.

Similarly, in Zen et al. (2013) a deep neural network (DNN) (see Figure 2.6a) is used to model the conditional distribution of acoustic features given linguistic features, replacing both decision trees and GMMs. Here, the hidden units are deterministic and the outputs can be deterministic or stochastic by predicting parameters of a probability density function. When the output layer represents a mixture density, such as a GMM, the network is often referred to as a mixture density network (MDN) (Zen and Senior, 2014) (see Figure 2.6b). This approach is the closest to the modern approaches; a simple feed-forward network deterministically computes the hidden layers by connecting many simple non-linear operations and is trained with backpropagation. This connectionist approach has very few limitations, unlike the approaches based on undirected probabilistic networks such as RBMs and DBNs, which are plagued with intractabilities and have to be trained with approximate algorithms such as contrastive divergence. The main reason why this simpler approach was not used earlier, was due to initial difficulties in training networks with many hidden layers, which was later resolved with the advent of bigger datasets, better initialization schemes, optimizers, non-linearities, and so on. This approach was later also proposed for singing synthesis (Nishimura et al., 2016), with results exceeding those of the baseline HMM-GMM approach.

In the above methods, the models generally learn a frame-wise mapping from linguistic features that are constant throughout the duration of a phoneme, to acoustic features produced by a vocoder. By appending information about the position of the frame within the phoneme to the input features, avoid the issue of a small number of discrete states over time present in HMM approaches. However, there may still be discontinuities from frame to frame or a general lack of coherence over time. A common approach to mitigate such issues, taken from HMM methods, is to also model delta features and use an MOPPG algorithm during synthesis (see §2.3.2).

A more principled approach to this problem is to use a neural network with connections between timesteps (see Figure 2.6c). Recurrent neural networks (RNNs) are commonly used for this kind of architecture. In this case, the activation of a hidden unit is not only a function of its inputs, but also of its activation at the previous timestep. Occasionally, recurrent connections are also used in the output layer (see Figure 2.6d). To mitigate problems with vanishing or exploding gradients during training, often RNNs with gated units, such as long short-term memory (LSTM), are used (Fan et al., 2014; Zen and Sak, 2015). In a large-scale comparison between production-level LSTM-RNN and unit selection systems, both systems are comparable overall, with the best method depending on the language (Zen et al., 2016).

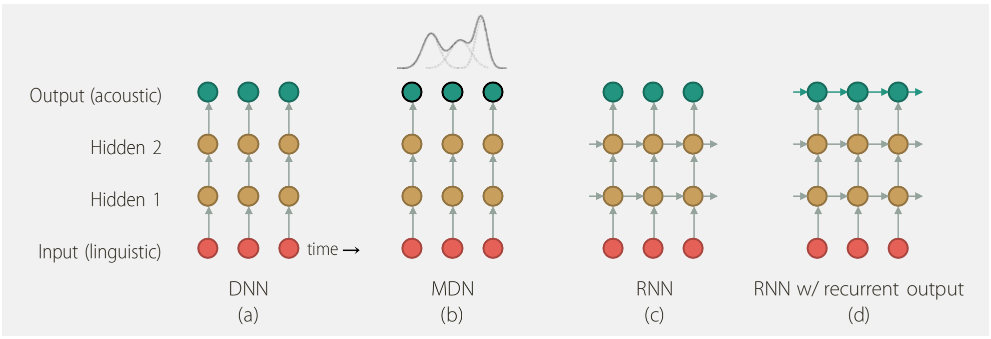

Figure 2.6: An overview of different first wave neural network speech synthesis models. Here, a simplified network is depicted with an input layer, two hidden layers, an output layer, and is unrolled for three timesteps. The first column shows a deep neural network (DNN) (e.g., Zen et al., 2013; Nishimura et al., 2016). The second column shows a mixture density network (MDN) (e.g., Zen and Senior, 2014). The third column shows a recurrent neural network (RNN), here unidirectional, but may also be bidirectional (e.g., Fan et al., 2014). And finally, the fourth column shows an RNN with recurrent output layer (e.g., Zen and Sak, 2015).

2.4.2 Second wave of neural network approaches

What we consider the second wave of approaches based on neural networks started with the publication of WaveNet (van den Oord et al., 2016a) and is still ongoing at the time of writing this dissertation. Since this publication, using powerful neural networks has become the prevailing paradigm for TTS and singing synthesis, and the number of publications on this topic has exploded. In this section, we will try to discuss some of the most important developments, in chronological order, grouping works that share a lot of similarities. That said, as the number of works is so great, it is not feasible to exhaustively discuss all of them. Similarly, as the field developed organically, it is sometimes impossible to simultaneously group similar approaches and follow a strict chronological order.

WaveNet

What makes WaveNet (van den Oord et al., 2016a) remarkable is that it is the first time a neural network clearly outperforms the then state of the art, concatenative synthesis. In a mean opinion score (MOS) listening test, WaveNet was rated 4.21 (with a 4.55 hidden reference), whereas a state of the art concatenative system was rated 3.86. The model is able to achieve this performance by modeling the speech waveform directly, rather than predicting some intermediate acoustic features and using a vocoder to produce the final waveform. One of the key aspects of the model that make this possible is that it is an autoregressive model, meaning that prediction of a timestep depends on the predictions of previous timesteps as well. This is somewhat similar to the recurrent output layers discussed in §2.4.1, but in this case the feedback is much *deeper*, feeding back from the output to the input of the network, instead of being contained to the output layer only. The model contains several other architectural innovations[7] which undoubtedly also play an important role in its success; (i) the model is fully convolutional, which allows very efficient, parallelized training compared to sequential RNN training, (ii) using gated unit, gives the model similar capacity to RNN models with gated units, such as LSTM or gated recurrent unit (GRU), (iii) using dilated convolutions allows the model's receptive field to grow exponentially with the number of layers, and to extract multi-scale representations (Yu and Koltun, 2016), (iv) using residual and skip connections facilitate optimization with deeper architectures (He et al., 2016), (v) using a simple conditioning mechanism using per-layer additive linear projections, and (vi) using a categorical output distribution to model the μ-law encoded 8 bit quantized waveform, essentially changing the model's task from regression to classification. One notable downside of this model is that inference (generation) tends to be several orders of magnitude slower than the models discussed in §2.4.1, due to the sequential autoregressive sampling step at inference, needing many steps to produce some duration of speech (due to the relatively high sampling rate), and each step consisting of the forward pass through a high complexity model (i.e., many layers, many channels per layer). The model is also still based on a traditional TTS pipeline, meaning that for instance prosody is predicted by external models which can cause compounded errors, and input to the model are "handcrafted" features. The original WaveNet is conditioned on many linguistic features (similar to those of the handcrafted "questions" of the decision tree in HMM-based synthesis, see §2.3.2). These features additionally have to be time-aligned to the audio, prior to training, i.e., phonetic segmentation needs to be available. The model is also conditioned on F0 predicted by an external, more traditional (LSTM) model. Finally, while sound quality is very good it can be considered somewhat "lo-fi", that is, the output waveform has a 16 kHz sample rate, and is 8 bit μ-law, notably lower than e.g., the 44.1 kHz, 16 bit PCM of CD quality audio.

After the publication of WaveNet, interest in this type of model spiked, prompting many other groups to work on similar topics. The Deep Voice model (Arik et al., 2017a) is heavily based on WaveNet, but addresses some of its shortcomings. This work still uses a traditional TTS pipeline structure, but all components are now entirely based on neural networks. In particular, the model is trained without "handcrafted"

[7] Most of these innovations were first introduced in earlier models proposed for modeling images, such as (in chronological order) PixelRNN and PixelCNN (van den Oord et al., 2016b), and Gated PixelCNN (van den Oord et al., 2016c). Similar models were later also successfully applied to video (Video Pixel Networks) (Kalchbrenner et al., 2016b) and text (ByteNet) (Kalchbrenner et al., 2016a). After WaveNet, several improvements to the base image model have also been proposed, e.g., PixelCNN++ (Salimans et al., 2017).

linguistic input features, but rather text is converted to a sequence of phonemes (by dictionary lookup), and then passed through a stack of bidirectional quasi-recurrent neural network (QRNN) (Bradbury et al., 2017) to obtain context-dependent features on which the waveform generator is conditioned. Similarly, the model can be trained on <text, audio> pairs without alignment (i.e., phonetic segmentation), as this alignment is first learned in an unsupervised manner using a network with connectionist temporal classification (CTC) loss (Graves et al., 2006). While a pipeline with neural components is potentially more powerful than its traditional counterpart, they are still trained independently, hence the issue of compounding errors remains. By using a fast inference algorithm based on caching computations (similar to Ramachandran et al., 2017) and a deeper-but-narrower architecture, the authors obtain real-time inference speeds. The successor to the Deep Voice model, Deep Voice 2 (Arik et al., 2017b), further improves the components of the neural TTS pipeline, and applies additional optimizations to the modified WaveNet architecture. This latter model also introduces so-called multi-speaker models that are trained on data from multiple speakers and conditioned on a learned embedding representing the speaker.

Models and architectures inspired by WaveNet have also been applied widely in singing synthesis (e.g., Blaauw and Bonada, 2017b; Wada et al., 2018; Bous and Roebel, 2019; Yi et al., 2019).

Sequence-to-sequence models with separate neural vocoders

One long-standing goal neural TTS research is to have a truly end-to-end[8] system that can generate the speech waveform from input text using a single model, rather than multiple independently trained components. A principal problem with this is that the alignment between input text and output audio has to be inferred during training and synthesis. One possible solution to this problem is to use a so-called sequence-to-sequence (Seq2Seq) architecture, which can learn mappings between unaligned sequences. In the case of TTS, these typically do this using an attention mechanism. In practice, it turns out to be very difficult to learn the alignment between a text sequence and the corresponding high sample rate waveform. As a compromise, typically two separate models are used; one Seq2Seq model learns the mapping from text to mel-spectrogram, and a second model, called a neural vocoder, learns the mapping from mel-spectrogram to waveform. This approach is fairly convenient, as the first model can be learned from <text, audio> pairs with requiring alignment, and the neural vocoder can be trained from audio only, which makes gathering big datasets easier. As a result, this approach has become the de facto standard.

[8] Most current systems are what is considered in the strictest sense of the word *nearly* end-to-end, as they often rely on external text normalization. Also, while using orthographic text input is possible, current best results are generally obtained using phonetic text input.

Two such systems were developed simultaneously and independently. The first system is Char2Wav (Sotelo et al., 2017), which consists of a Seq2Seq model which converts text to WORLD vocoder features, and a neural vocoder. The neural vocoder in this case is based on a multi-scale RNN-based architecture called SampleRNN (Mehri et al., 2017). The second system, Tacotron (Wang et al., 2017) consists of a recurrent Seq2Seq model which predicts a mel-spectrogram from text or phonemes, a post-processing step that converts the mel-spectrogram to a linear spectrogram, and finally the Griffin-Lim algorithm (GLA) to produce the final output waveform. In this case, all components can be trained jointly[9]. However, the use of the GLA notably degrades performance compared to using a fully neural vocoder. For instance, Arik et al. (2017b) use the Tacotron model together with a neural vocoder based on WaveNet as a baseline. Especially the naturalness of prosody produced by the Tacotron system is significantly higher compared to previous systems using a discrete TTS pipeline.

The next significant milestone in TTS came with the successor of the Tacotron model, Tacotron 2 (Shen et al., 2018). This model is particularly significant as it is the first model that achieved a MOS rating that's almost identical to the ground truth hidden reference, 4.53 and 4.58 respectively. This means that listeners have trouble telling real and synthetic speech apart. Tacotron 2 uses a simplified and improved version of the recurrent Seq2Seq model of Tacotron, combined with a neural vocoder. In this case, the neural vocoder is based on WaveNet with a mixture of logistic distributions (MoL) output layer similar to Salimans et al. (2017). This allows predicting a 24 kHz, 16 bit PCM waveform, rather than the 16 kHz, 8 bit μ-law output of the original WaveNet. While nothing fundamentally new is proposed in this work, the execution and results are remarkable. This model was later also adapted to singing synthesis (Angelini et al., 2020).

Convolutional sequence-to-sequence models

One downside of recurrent Seq2Seq models is that training cannot be fully parallelized. As models such as WaveNet are already fully convolutional, having a Seq2Seq mechanism that is also convolutional is thus also desirable to speed up training. One such mechanism is provided by Gehring et al. (2017) and applied in several TTS systems, including Deep Voice 3 (Ping et al., 2018) (not a direct continuation of Deep Voice 2), DCTTS (Tachibana et al., 2018) and Fast DCTTS (Kang et al., 2021). Training is reported to be an order of magnitude faster than the recurrent Seq2Seq-based Tacotron. While no direct comparison between these models and state of the art Tacotron 2 has been done in terms of MOS (except for the latter, all these models came before Tacotron

[9] The Griffin-Lim algorithm (GLA) (Griffin and Lim, 1984) is a signal processing algorithm, and therefore does not need to be trained.

2), we generally understand them to perform slightly worse. This approach has also been applied to singing synthesis (e.g., Lee et al., 2019, based on DCTTS).

Probability density distillation

Besides obtaining more end-to-end models, another important research goal is to obtain models with efficient inference. While the issue of slow sequential inference of autoregressive models, such as WaveNet, can be mitigated to some degree by clever implementations and architectural optimizations, it can still be an issue for taking such models into production. A more fundamental solution to this problem was proposed in Parallel WaveNet (van den Oord et al., 2018). This approach uses a pre-trained WaveNet teacher network to distill knowledge into a non-autoregressive student network. In this case, the student network is based on Inverse Autoregressive Flows (IAF) (Kingma et al., 2016), which offers non-autoregressive inference, but slow, sequential traditional maximum likelihood learning (hence the distillation). An extension of Deep Voice 3, called ClariNet (Ping et al., 2019) takes a similar approach, with some differences. In general, these models are able to achieve relatively fast inference, e.g., 20 times real-time on GPU for Parallel WaveNet, while obtaining good results, e.g., a MOS 4.40 against 4.54 ground truth hidden reference (just slightly below Tacotron 2) for ClariNet. The main downsides of this approach are that a two-step training process is rather cumbersome, and that getting the distillation process to converge is non-trivial, e.g., needing auxiliary losses and regularization.

Normalizing flows

To avoid this two-step process of probability density distillation, somewhat similar normalizing flow-based models have been proposed that can be trained using maximum likelihood directly. Briefly, a normalizing flow is a series of invertible functions (bijections). During training, the flow is used in one direction to transform the complex data distribution into something simple, e.g., an isotropic Gaussian. Then using the change of variables formula, we can then compute the negative log-likelihood used to optimize the network parameters (via the determinant of the Jacobian of the flow). During inference, we use the flow in the other direction, transforming a sample from the simple distribution to the complex data distribution. In the case of a neural vocoder, we can condition the flow on some input features, such as a mel-spectrogram. This approach has been used in FloWaveNet (Kim et al., 2019) based on RealNVP (Dinh et al., 2017), and WaveGlow (Prenger et al., 2019) based on Glow (Kingma and Dhariwal, 2018). One of the major disadvantages of this approach is that because each layer that implements a bijective function has a lot of requirements and constraints, it tends to not be very powerful compared to a typical layer in a neural network. Thus, flow-based networks may require over 100 convolutional layers, resulting in important computational and memory requirements. Thus, while inference is done in a single step, it will require very powerful hardware and training will be slow. Some work has been done to mitigate these issues (e.g., Zhai et al., 2020; Wu and Ling, 2020; Ping et al., 2020; Luong and Tran, 2021).

More recent works involving normalizing flows, include Glow-TTS (Kim et al., 2020), Flow-TTS (Miao et al., 2020) and Wave-Tacotron (Weiss et al., 2021), which notable learn alignment between text and audio in these non-autoregressive models.

Generative adversarial networks

One issue with simple feed-forward models, like a convolutional neural network (CNN), is that defining a suitable loss function that considers all of the generated timesteps jointly is not straightforward. In autoregressive models, this is done implicitly by conditioning on past timesteps, and thus we can compute a loss between individual timesteps and average the results for instance. If we do the same with a non-autoregressive model, we get poor results, e.g., oversmoothing, lack of high-frequency details, etc. Approaches based on generative adversarial networks (GANs) try to circumvent these issues by using an adversarial loss by using a discriminator which determines whether a sequence (or part of a sequence) is real or fake (generated). The generator network and the discriminator network are trained jointly, one trying to get samples from the generator to be classifier as real, the other samples from the dataset as real, and samples from the generator as fake. The adversarial loss provides a kind of implicit, learned loss between the generated samples and the target, considering multiple timesteps at once. This approach is quite attractive as inference is a single feed-forward operation, there is no need for a teacher network, and there are no real constraints on the generator and discriminator networks. There have been many works using this approach for neural vocoders (e.g., Yamamoto et al., 2020; Kumar et al., 2019; Yang et al., 2021; Wu et al., 2021; Kong et al., 2020; Tian et al., 2020a; Yamamoto et al., 2021; Hono et al., 2021b; Yoneyama et al., 2021), and also a few works that use this approach for complete TTS systems (e.g., Bińkowski et al., 2020; Gritsenko et al., 2020).

For singing synthesis, using GANs has also been very popular. Some works use an adversarial loss to reduce oversmoothing in intermediate acoustic features, typically a mel-spectrogram (e.g., Chandna et al., 2019; Hono et al., 2019; Choi et al., 2020; Sankaran et al., 2021). More recently, this is done in combination with Seq2Seq architectures (e.g., Wu and Luan, 2020; Chen et al., 2020; Lee et al., 2021). Some works also use GAN-based neural vocoders (e.g., Chen et al., 2020; Chen et al., 2021a; Huang et al., 2021; Liu et al., 2021b). In a few cases GANs are also applied in super-resolution networks that predict linear spectrograms from mel-spectrograms (e.g., Lee et al., 2019; Lee et al., 2020).

Diffusion probabilistic models

A very recent group of works are based on a new type of generative model called diffusion probabilistic models (e.g., Ho et al., 2020). Very briefly, this model defines a Markov chain that, starting from some data, gradually adds Gaussian noise at each step, until it becomes isotropic Gaussian noise. A neural network is trained to learn (a variational lower bound) on the reverse step, the denoising process. Once this network is trained, we can sample from an isotropic Gaussian, perform the denoising process for a number of steps, until recovering a sample from the data distribution. This type of model is very flexible and training is straightforward and efficient. In image modeling, this type of model is state of the art in certain metrics, and perceptually sample quality is similar to that of the best GAN models. A downside of this model is that inference is still sequential, and although the number of steps does not depend on the sequence length like with autoregressive models, the number of steps required for good quality samples can still be relatively high. That said, much current research focuses on improving quality further and reducing the number of steps required at inference.

In TTS, this approach has been applied to neural vocoders (e.g., Chen et al., 2021b; Kong et al., 2021). It also has been used in Seq2Seq TTS models to improve mel-spectrogram prediction which is then combined with a GAN-based neural vocoder to produce the waveform (e.g., Popov et al., 2021; Jeong et al., 2021). Diffusion probabilistic models can even be used to predict waveform directly in Seq2Seq models (e.g., Chen et al., 2021c). In singing synthesis, diffusion probabilistic models have been applied to improve mel-spectrogram generation, combined with a GAN-based neural vocoder (Liu et al., 2021a).

Architecture-based models

Another group of works, which we call architecture-based models, are based mostly on innovations in architecture, rather than the underlying model.

For instance, WaveRNN is a fairly widely used neural vocoder (Kalchbrenner et al., 2018). This model is recurrent, thus inference is sequential, like with autoregressive models. However, this model aims to reduce the number of operations per sequential step, in order to achieve efficient inference. In this case, it uses a compact, single-layer RNN, which is shallow-but-wide (e.g., 2048 hidden units). Then, weight pruning is applied to reduce the size of the model with a sparsity of 96% or more. Additionally, by using two 8 bit softmax outputs, it is able to generate high fidelity waveforms. The reduced model can then be run in real-time on a mobile central processing unit (CPU). In singing synthesis, for instance Gu et al. (2020) use the WaveRNN vocoder.

One interesting, but unusual model is VoiceLoop (Taigman et al., 2018), which uses an architecture that is neither recurrent nor convolutional, but instead uses a *shifting buffer working memory*. It is an end-to-end model, using an attention-based Seq2Seq architecture, like many of its preceding models. In listening tests, a MOS of 3.69 compared to a 4.60 ground truth is obtained. An interesting addition to the model is the ability to fit a new speaker given a trained multi-speaker model, by optimizing a new speaker embedding given few, possibly "in-the-wild" recordings of the target voice. This concept was later expanded upon by combining by Nachmani et al. (2018) the VoiceLoop model with a network that predicts a new speaker embedding from a few recordings, thus avoiding the need for iterative optimization and orthographic or phonetic transcription of the target voice.

Another widely used architecture is the Transformer network (Vaswani et al., 2017), which uses self-attention rather than RNNs or CNNs to model interaction between timesteps. For TTS, these models may be autoregressive (e.g., Li et al., 2019; Ihm et al., 2020), but in particular non-autoregressive variants are popular (e.g., Peng et al., 2019; Ren et al., 2019). In singing synthesis, this architecture is also used in several works for mel-spectrogram generation (e.g., Lu et al., 2020; Shi et al., 2020; Chen et al., 2020).

A few works on singing synthesis also use more conventional architectures for mel-spectrogram generation. These include the U-Net architecture, which uses downsampling and upsampling operations to model time series at different scales (e.g., Nakamura et al., 2019; Nakamura et al., 2020), or LSTM RNNs (e.g., Kim et al., 2018; Hono et al., 2021a).

End-to-end sequence-to-sequence models

As mentioned, previously, simultaneously learning alignment between text and audio, and directly output waveforms, was considered excessively difficult. However, recently, thanks to advances in new alignment mechanisms and generative models, this is now possible. Some are based on normalizing flows (e.g., Kim et al., 2020; Miao et al., 2020; Weiss et al., 2021), and others are GAN-based (e.g., Donahue et al., 2021).

2.5 A working model of the singing voice

While machine learning models can mostly treat data "blindly", having some understanding of the underlying properties of the data can often be beneficial. Here, we try to provide a rough "working model" of the singing voice and its properties. This model is not intended to be a model for analyzing or synthesizing singing voice signals directly, but rather provide some context. This contextual information can be used in various ways, e.g., analyzing the system's performance by ear, introducing some forms of domain knowledge – which can be useful in practice as training data can be limited, interacting with singers in recording sessions, and so on.

2.5.1 The source-filter model

As is standard in speech research, our working model is based on the source-filter theory of speech production (Fant, 1960). The more traditional version of this model is derived from studies of the phonological process of speech production for vowels: Air from the lungs is forced through the glottis where the elastic muscular forces of the vocal folds cause it the open and close in a cyclical manner, generating a so-called glottal flow. This acoustic wave then travels up through the vocal tract where it is filtered by the various resonating cavities, finally radiating from the mouth. Typically, models will consider the glottal flow (glottal volume-velocity waveform) to generate a harmonic series with a −12 dB/octave spectral slope, the vocal tract to be a time-varying all-pole (i.e., minimum-phase) filter, and the mouth radiation to be a filter with a 6 dB/octave spectral slope. Many models will combine the slopes of the mouth radiation filter and the glottal flow, and instead model the voice source as the derivative glottal flow, with a −6 dB/octave spectral slope.

While the above model works reasonably well for modal vowels, there are many simplifications that make it difficult to apply this model to all types of vocal sounds that occur in real speech or singing. For instance, the voice source is often not purely harmonic, but also contains noisy components. The slope of the voice source is not fixed, but may depend on whether phonation is tense or lax. The vocal tract does not only produce resonances, but can also produce anti-resonances in nasals. Certain phonemes, such as plosives, fricatives and affricates, have different production mechanisms altogether. For these reasons and many others, we opt to use a more relaxed source-filter model, depicted in Figure 2.7. In this model, the source signal is assumed to have a flat spectral slope, but is allowed to be a non-trivial combination of harmonics, noisy components, and other components such as transients. The filter is allowed to have the response needed to transform the source into the final speech signal, including resonances and anti-resonances. In this model, the key concept is that the source is responsible for pitch and timbrical texture (e.g., a breathy voice or rough voice), while the filter is responsible for the timbre itself, which we consider to include phonetic content, speaker identity, voice color, intensity, and so on.

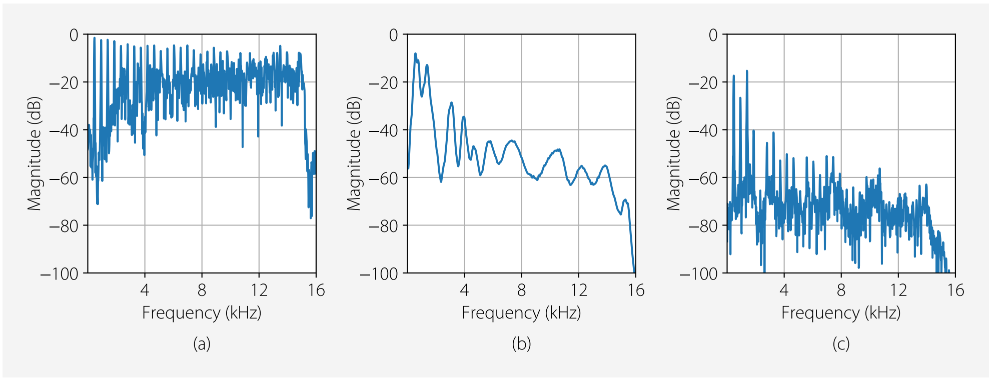

Figure 2.7: Example of a voice signal decomposed in source and filter. Here, the source, (a), is approximately flat, and consists of harmonics and an aperiodic component. The filter, (b) includes formants from the vocal tract, and the spectral tilt corresponding to the glottal voice source and lip radiation. The final output, (c), is produced by filtering the source signal with the filter.

2.5.2 The timbre component

In our working model, we consider timbre to be determined by on the one hand the spectral envelope, and other the other hand a timbrical texture that is provided by the voice source. The spectral envelope generally follows the harmonic peaks in the voice signal, and consists of a number of resonances, called formants. The lower formants are the most significant in providing the phonetic content of the voice, e.g., all vowels can generally be uniquely identified by looking at the first two formants only. There are also anti-resonances, called anti-formants, which are typically produced by the nasal tract. There is a glottal formant, which together with spectral slope (or spectral tilt), is an important property of the voice source (in the traditional source-filter model). This formant below the first vocal tract formant affects the tenseness of the voice (Doval and d'Alessandro, 1997). There also is a singer's formant, a prominent peak around 3 kHz, that is associated with classical operatic singers (Sundberg, 1989; Sundberg, 2001).

We consider the spectral envelope to be the determining factor in several important aspects of the voice, such as phonetic content and speaker identity. These two aspects of the voice normally cannot be separated, as they are a product of complex interactions between formant properties (i.e., formant positions, widths, and amplitudes), which all vary over time. While principally determined by timbre, pitch also has some influence on these aspects, speaker identity more so than phonetic content. Somewhat unique to the singing voice is the concept of voice registers. While generally any change in pitch affects the corresponding spectral envelope, we can often define a number of discrete ranges of pitch, called registers, between which a bigger, abrupt change in timbre can occur. These registers typically include chest voice, middle voice, head voice, and falsetto (sometimes included in the head register). Besides these more abrupt and significant changes in timbre according to the pitch range, even relatively small changes in pitch, such as a vibrato, can have a noticeable effect on the spectral envelope of the signal. Many singers will also intentionally apply vowel modification (aggiustamento) according to pitch in order to obtain the desired vocal tone. Another significant aspect of timbre is the concept of tenseness (often described as a continuum between lax and tense voice) which typically also is related to intensity, dynamics, or loudness.

Besides the spectral envelope, we also consider the timbre to contain some kind of timbrical texture. Perhaps the most important aspect of this texture is the amount of noisy components in the voice. Typically, the voice source will contain a mixture of harmonic and noisy components. These noisy components can either be generated at the glottis, e.g., as part of the glottal flow in breathy voice or aspirated phonation, or produced in the mouth when forcing air through a narrow channel made by placing two articulators close together, e.g., in dental fricatives. In our working model, we consider the voice source to be the sum of harmonics and filtered noise. This gives rise to the concept of aperiodicity which for each frequency or band of frequencies determines the degree to which it is harmonic or noisy. Noisy components generated at the glottis are generally considered to consist of a combination of constant noise and so-called pulsatile noise, which is noise with a cyclic amplitude modulation, synchronized to the glottal flow (e.g., Mehta and Quatieri, 2005).

As a general rule of thumb, we consider the phase of the voice signal to be of lesser perceptual importance. As the vocal tract can be approximated as a minimum-phase all-pole filter, we generally consider a minimum-phase response of the spectral envelope of the signal to be a good fit. However, many more sophisticated models of the voice source include causal as well as anti-causal components [e.g., CALM model]. It should also be noted that the perceptual importance of phase of the voice signal depends a lot on the pitch of the signal. In particular for low pitch voices, phase becomes of much greater perceptual relevance.

2.5.3 The pitch component

The pitch component of a singing voice signal contains the melodic content, as well as expression. That is, any given melody can be sung in a large variety of different ways. There will be some minimum constraints that ensure the pitch sequence can be recognized as the melody in question, but there are many other expressive aspects of the pitch sequence in which the singer is pretty much free to do as he or she wants. Besides melodic content and expression, there typically also tends to be some natural variation in pitch as a function of the phonetic content, this is typically referred to as microprosody (Taylor, 2009, Chap. 9.1.4, "Micro-prosody", p. 229).

The expressive component of pitch, involves many different things, perhaps too many to list exhaustively here. Some basic aspects of pitch expression can be grouped into attacks and releases of notes, transitions between notes, ornaments, and vibratos. Attacks and releases may include things such as scooping attacks, creaky or vocal fry attacks, glottal attacks and releases, the so-called "gospel" release, and so on. Note transitions include staccato, legato, portamento, and so on.

There typically also is a natural correlation between pitch and dynamics or loudness. That is, low pitches within a singer's tessitura tend to have low dynamics, while high pitches tend to have high dynamics. Most singers can vary dynamics to some degree at high pitches, singing low pitches with high dynamics seems especially challenging.

2.5.4 The timing component

Besides timing, another major component of expression is timing. In our working model we consider expressive timing to exist on two levels; note timing and phonetic timing. Note timing is normally highly constrained by the score that is being sung. However, there typically can be a significant deviation in the sung note timings compared to the nominal note timings as written in the score. Once the note timings are established, the phonetic timings within a note must be determined. We consider this to be generally the lesser of the two when it comes to expression, as the phonetic timing is in part determined by the natural duration of phonetics. That said, there can be some significant variability in phoneme durations, especially in the case of certain phoneme types such as fricatives and nasals. As a general rule of thumb, we assume that each note will correspond to a single syllable, but there are exceptions, notably with melismatic singing. The onset of the vowel (or syllabic consonant) in the syllable will typically coincide with the note onset. The remainder of the note will be occupied by the syllable's coda consonants, as well as the onset consonants of the next adjacent note (if any).

2.5.5 Other components

There are other components of the singing voice that are not fully covered by the above. Many of these are related to non-modal voice types, and are somewhat beyond the scope of this work. That said, non-modal vocal resources are used frequently in expressive singing. Some non-vocal voice types, such as growls or rough voice, are produced by modulation of the voice pulses (e.g., Bonada and Blaauw, 2013). Growls tend to have a clear macro period to the modulation, although this may vary over time, and be accompanied by changes in formants. In the spectral domain, these modulations are often visible as sub-harmonics, with specific phase patterns. Rough voices, on the other hand, are characterized by more irregular modulations. The breathiness of the voice can also be used as an expressive resource, although partially covered by the concepts of aperiodicity and tenseness of timbre. Vocal fry, typically as very low pitch scoop attacks, is also a common expressive resource in singing. In this case, the voice pulses are perceived more like a sequence of transient pulses, rather than an actual pitch.

Methodology 3

Before presenting the main contributions of this work, we should preface it with some of the methodology typical of singing synthesis research. In particular, we should know how singing synthesizers are typically constructed, what kind of datasets are typically used, and how to evaluate such systems.

3.1 A framework for singing synthesizers

In the main body of this work we consider a number of different singing synthesis models. All of these models, however, share the same basic framework. This framework consists of how we approach the task, certain simplifications we employ to make the task more manageable, how we structure the model, signal processing components we utilize besides the principal machine learning models, and so on.

3.1.1 The task of singing synthesis

The most canonical way to define the task of singing synthesis is arguably "converting a musical score with lyrics into a waveform of a corresponding sung vocal". This means that we effectively want to model what a singer does when presented with a score, including interpreting the score and the actual act of producing the vocal sounds. Even if we limit ourselves to modeling a specific singer singing in a specific style, any given score may be rendered as one of many plausible waveforms. Thus, it makes sense to define our model in probabilistic terms (as is commonly done in deep learning),

$$p_\theta(\mathbf{x} \mid \mathbf{c}) := \dots, \tag{3.1}$$

that is, we model the (conditional joint) probability of a waveform, $\mathbf{x} = [x_1, x_2, \dots, x_T]$, given a note sequence score, $\mathbf{c} = [n_1, n_2, \dots, n_N]$. Without going into much detail, we will assume that a note n at least encodes information about note pitch, duration and corresponding lyric (syllable). In the above definition, our model is parametrized by θ.

To train this model we can optimize θ to maximize likelihood on the training data,

$$\theta^* = \arg\max_\theta \sum_i \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{c}^{(i)}), \tag{3.2}$$

where $\mathbf{x}^{(i)}, \mathbf{c}^{(i)}$ are pairs of corresponding waveform and score examples that make up the training set. In practice, we perform this optimization using minibatch gradient ascent, computing the sum in Equation (3.2) at each step over a small number of randomly sampled examples from the dataset, rather than over the whole dataset (e.g., Goodfellow et al., 2016, Chap. 8). Similarly, examples $\mathbf{x}^{(i)}$ in the dataset often have varying lengths in practice, in which case we can weigh the sum in Equation (3.2) by sequence length.

Once the model is trained, we may synthesize waveform $\mathbf{x}^*$ given an unseen score $\mathbf{c}^*$, typically by drawing a sample from the model,

$$\mathbf{x}^* \sim p_{\theta^*}(\mathbf{x} \mid \mathbf{c}^*). \tag{3.3}$$

Issues with the canonical task definition

While this seems fairly straightforward, there are several issues that make this a difficult problem. Some of these issues include:

- The exact alignment between waveform and score is not given. While the scores will contain some information on the nominal timing of notes, we cannot derive the exact timing of the vocal performance from it directly. For instance, the timing of phonemes[1] will have a big influence on the waveform, but this timing cannot be uniquely derived given only a musical score. Inferring this latent alignment during training and synthesis, while possible, makes the learning problem more difficult.
- Encoding score information, in particular lyrics, in a way that is easily processable by a neural network is not straightforward. The lyrics corresponding to a note, usually a syllable, tends to either be a sequence of a variable number of phonemes, or an entry into a large syllable space.
- Waveform data is generally difficult to work with. While only 1-dimensional, it has a very high temporal resolution (generally at least 32,000 samples per second for high-quality singing voice). Unlike other acoustic representations, waveforms are not smoothly varying over time, combining periodic and aperiodic components,

[1] For the sake of argument, we will assume the timing of phonemes, i.e., their begin and end times, can be determined exactly. In reality, this tends to be somewhat ambiguous as many phonemes are continuous in nature, with one phoneme smoothly transitioning into the next.

transients, and so on. Big differences in waveform signals can have very small perceptual effects. For instance phase inversion, small variations in timing, small variations in pitch, and so on, can heavily affect the waveform signal, while being perceptually unnoticeable. Somewhat paradoxically, relatively small errors in waveform predictions, such as discontinuities, irregularities, oversmoothing, and so on, can have big perceptual effects.

❧ For most existing machine learning models modeling the joint probability over thousands of random variables, while also allowing efficient and exact evaluation of log-likelihood (and corresponding gradients), and efficient and exact sampling from this model, is a very challenging problem.

Simplified task definition

One way to obtain a more workable solution is to simplify the task by imposing additional constraints on the conditioning data. Instead of conditioning the model on a score with lyrics, where the note is the unit with which we work, we could condition on a sequence of timed phonemes to which note properties (such as pitch and duration) are associated. This avoids the issue of having to infer the time-alignment between input and output. Additionally, it avoids issues with encoding variable-length syllables associated with notes; as the phoneme space is generally small, representing each phoneme by an embedding is a practical solution.

In order to condition the model on a timed phonetic sequence, we need to phonetically segment the training data. Typically, this is done by an automatic process, often with manual corrections to reduce errors (see §3.2). For synthesis, we typically either use a timing model (see §4.3.3) or a phonetically segmented reference recording (see §3.1.5).

It should be noted that this simplification is not strictly necessary, as so-called sequence-to-sequence (Seq2Seq) models can infer the alignment between unaligned input and output data during training and inference. While more challenging for the model, alleviating the need for phonetic segmentation of the training data can greatly speed up the creation of voices (see Chapter 5). That said, these models generally still tend to be controlled by inputs on a phoneme-level, rather than note-level.

3.1.2 Splitting task up into smaller sub-tasks

Another way to simplify the model is to break the task up into smaller, easier-to-model sub-tasks. A separate model can then be trained for each sub-task, and these models can then be combined to form a complete synthesizer (e.g., as in §4.3). It should be noted, that, at least in theory, having a single "end-to-end" model may have a performance advantage compared to combining multiple discrete models that are trained separately, rather than jointly optimized. Prediction errors in upstream models may cause errors to compound in predictions by downstream models relying on predictions of the former. That said, at the time of writing, the approach of multiple discrete sub-models tends to be at least competitive to or occasionally even outperforming end-to-end methods (e.g., Weiss et al., 2021; Donahue et al., 2021).

As a rule of thumb, whenever the approach of multiple models is used, they are generally trained using so-called "teacher forcing". That is, say we have a cascade of two models, rather than training the second model on predictions by the first model, we train it on a separate ground truth obtained from the training data. The principal reason to do it this way is to aid the models converging, especially early on in training. However, a potential downside is the so-called "exposure bias" issue; there will be a discrepancy between training, where teacher forcing is used, and inference, where only predictions will be used (see "Regularization to mitigate exposure bias" in §4.1.3).

There are also some practical issues related to using multiple models. In general, this approach complicates to some degree coding, training and evaluation. When using teach forcing multiple models can be trained sequentially (one after the other), but occasionally they can also be trained simultaneously, graphics processing unit (GPU) memory permitting.

The generator-vocoder split

While there are many ways to break up the singing synthesis task into small sub-tasks, the most common approach is to alleviate the issues associated with predicting the waveform signal directly by instead predicting some higher-level intermediate acoustic features that have been derived from the waveform. A separate second model can then learn only the inversion of these acoustic features back to the final waveform(e.g., Shen et al., 2018; Sotelo et al., 2017; Ping et al., 2018). The intermediate acoustic features are typically the mel-spectrogram or parametric vocoder features, explained in depth in §3.1.3. We will call the first model which converts the input score or timed phonetic sequence to intermediate acoustic features the "generator"[2], and the model which converts these features to the output waveform the "vocoder". This type of model is depicted in Figure 3.1.

Important advantages of this approach are that the intermediate acoustic features tend to have a much lower rate than the waveform, typically around 100 or 200 Hz, and that there is a much more direct relationship between such acoustic features and perception of the sound. That is, generally, small perceptual differences will also correspond to

[2]There's no standard term for what we call the "generator" model; other common names include the "decoder", "spectrogram prediction network", "character to spectrogram model", or "acoustic model".

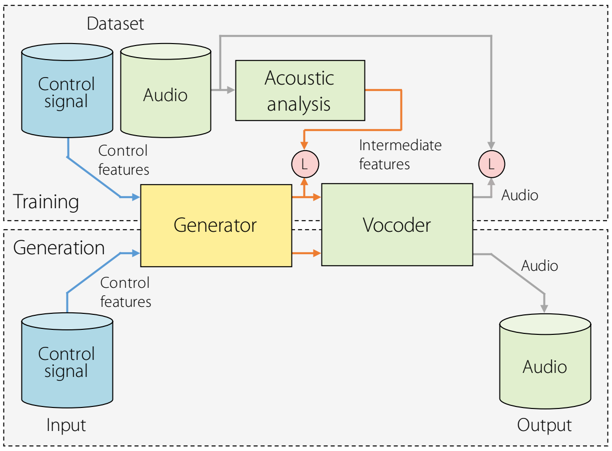

Figure 3.1: Diagram depicting a singing synthesizer with generator-vocoder split. Here, the "L" blocks indicate a loss used to train the relevant module, generator or vocoder. These modules are trained separately.

small differences in intermediate acoustic features, but may potentially correspond to big differences in the waveform domain. Another advantage is that as the acoustic features are obtained directly from the waveform via signal processing analysis, there are no issues related to time-alignment, and obtaining large datasets for training is relatively easy, as no annotations are needed. The main disadvantage of this approach is that there is no way to determine which intermediate acoustic features, or related hyperparameters are optimal for a given problem (see §3.1.3), so we typically have to resort to reasonable defaults or do time-consuming experiments.

It should also be noted that in cases where input and targets are not time-aligned (i.e., the simplification of "Simplified task definition" in §3.1.1 is not applied), and this issue is solved using a Seq2Seq model, predicting higher-level acoustic features rather than waveform becomes even more important. Up until very recently, Seq2Seq models generally would not converge when predicting waveform directly. With the advent of non-autoregressive waveform generation models and alternative Seq2Seq mechanisms, this seems to be slowly changing (e.g., Weiss et al., 2021; Donahue et al., 2021). While this fully end-to-end approach may provide a pathway towards improved results eventually, current models still seem to perform slightly below the generator-vocoder split approach.

The pitch-timbre-vocoder split

A further simplification is to split the generator into two components, as depicted in Figure 3.2; a pitch model which predicts F0 from a note score and a timbre model which

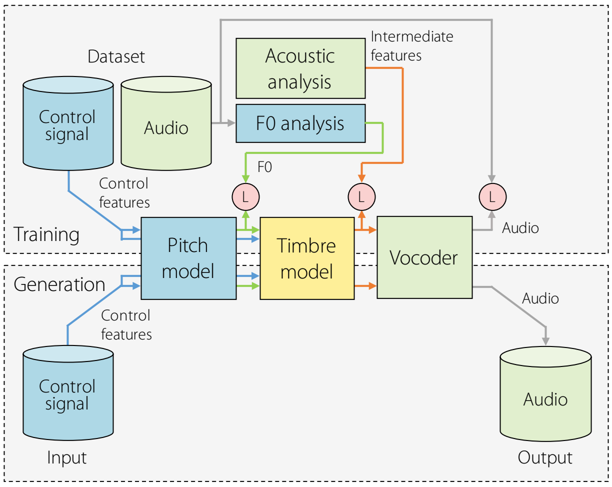

Figure 3.2: Diagram depicting a singing synthesizer with pitch-timbre-vocoder split. Here, the "L" blocks indicate a loss used to train the relevant module, pitch model, timbre model or vocoder. These modules are trained separately.

is conditioned on F0 and predicts the intermediate acoustic features. One reason we may want to do this, is that pitch and timbre are notably different, and the possibility to use different models to tackle each may be beneficial. Having an intermediate pitch prediction, also allows some adjustments, such as tuning correction (see §4.3.2). Finally, arguably the most important reason might be that it simplifies evaluation to some degree. That is, we will be able to compare different timbre models while keeping pitch identical, e.g., by using pitch extracted from a reference recording (see §3.1.5), and vice-versa.

Other splits

We can, of course, take the idea of splitting the generator into different sub-models even further. This works especially well when we do this for aspects that can be easily extracted from the audio signal. For instance, we fairly easily extract features related to loudness from the audio signal and then condition the timbre model on such features (e.g., Bous and Roebel, 2019). We could then have an additional dynamics model that predicts loudness features, e.g., considering long-term properties of the input musical score and corresponding dynamics annotations. Similar approaches may be possible for non-modal voice resources a singer may utilize, such as growls, vocal fry, and so on. That said, in this work, we do not extend our models beyond the approach described in "The pitch-timbre-vocoder split".

3.1.3 Intermediate acoustic features

Mel-spectrogram

Mel-spectrogram features are by far the commonly used intermediate acoustic features in text-to-speech (TTS), and by extension, singing synthesis. The reason why this is, is probably because they offer a fairly compact representation, with high perceptual relevance. Additionally, they only involve a moderate amount of signal processing, heuristics and hyperparameters.

While there are a few slight variations of the mel-spectrogram algorithm, we assume that for our purposes all of these should perform similarly. We will discuss briefly how we compute such features, which should provide a good default for most applications.

First, we use the short-time Fourier transform (STFT) to compute the magnitude spectrum of the input waveform $\mathbf{x}$,

$$\mathbf{X} = |\mathrm{STFT}(\mathbf{x})|. \tag{3.4}$$

It should be noted here that while the STFT is an invertible operation, the $|\cdot|$ operation discards phase and is thus lossy. Relevant hyperparameters here include the window function (e.g., Hann), the window time (e.g., 32 ms), the hop time (e.g., 5 ms), and the fast Fourier transform (FFT) size (e.g., 2048, which determines zero-padding and resulting dimensionality).

The second step is to apply a filterbank of half-overlapped triangular filters that are equally spaced on the mel scale. The mel scale is a frequency scale that has been inspired by human perception. We use what is often referred to as the "Slaney" definition of the mel scale. This definition linearly maps frequencies below 1000 Hz so that 0 Hz corresponds to 0 mel and 1000 Hz corresponds to 15 mel. Frequencies above 1000 Hz follow a logarithmic scale where 6400 Hz corresponds to 42 mel.

$$m = \begin{cases} f/66.67 & \text{if } f < 1000 \\ 1000/66.67 + 27\log(f/1000)/\log(6.4) & \text{otherwise.} \end{cases} \tag{3.5}$$

The filters are additionally normalized to have approximately constant energy per channel. Relevant hyperparameters here include the edge frequencies of the lowest and highest filters of the filterbank (e.g., 50–15,000 Hz), and the number of filters (e.g., 80). It should be clear that this step is also lossy, in particular losing details in the higher end of the spectrum. An example of such a filterbank is shown in Figure 3.3.

Finally, the mel-spectrogram is often converted to a decibel scale, often clipping below a certain threshold (e.g., −140 dB), which also avoids numerical problems with the log operation. An example of a mel-spectrogram is shown in Figure 3.4.

As the mel-spectrogram is a lossy representation of the underlying waveform, an exact inversion is not possible when we want to synthesize. The traditional approach is to first invert the mel filterbank, usually by simply performing the transposed operation, and then reconstructing phase from the linear magnitude spectrogram using the Griffin-Lim algorithm (GLA) (Griffin and Lim, 1984). The GLA iteratively tries to optimize the missing phase component, ensuring that it is consistent with the magnitude spectrum. The downsides of this approach include that it is rather computationally expensive, due to requiring many steps, and that the resulting quality often is not acceptable. While applying the GLA to a linear spectrogram with a high frame rate can result in a fairly high-quality signal, when applied to a spectrogram reconstructed from a mel-spectrogram, the results often include very heavy "modulation" artifacts.

To overcome these issues, recently much effort has gone into researching so-called neural vocoders (e.g., Tamamori et al., 2017; Prenger et al., 2019; Wang et al., 2019; Yamamoto et al., 2020). These vocoders are trained to invert the mel-spectrogram to waveform using a neural network. The main advantage of this approach is that unlike with purely signal processing methods, like the GLA, neural vocoders can recover some approximation to the information lost during the lossy mel-spetrogram calculation. While these often can generate very high-quality results, their main issue is that this often comes at the cost of high computational complexity and/or lack of parallelization. Much of the current research in this area is focused on improving exactly these aspects.

Alternatively, some works employ so-called super-resolution networks that reconstruct linear frequency spectrograms from lower resolution mel-spectrograms (e.g., Lee et al., 2019). Once the full resolution spectrogram has been reconstructed, we can employ the GLA to generate the output waveform. Currently, this approach is less popular than using a neural vocoder, possibly because the iterative GLA is still relatively slow compared to modern neural vocoders, and ultimately less powerful.

Parametric vocoder features

The mel-spectrogram can be seen as a lossy transformation of the input signal. Parametric vocoders on the other hand try to analyze a given signal to estimate its underlying parameters, while simultaneously being able to reconstruct an approximation of the original signal from these higher-level parameters. While there are a number of such parametric vocoders, in this work we mainly focus on the WORLD vocoder (Morise et al., 2016) (D4C edition, Morise, 2016). It should be noted that a highly related and

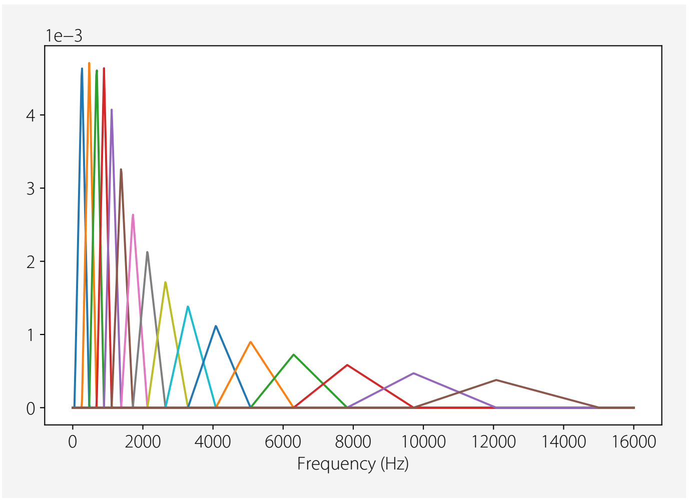

Figure 3.3: Example of a mel scale filterbank. In this case using 16 filters for the sake of visualization.

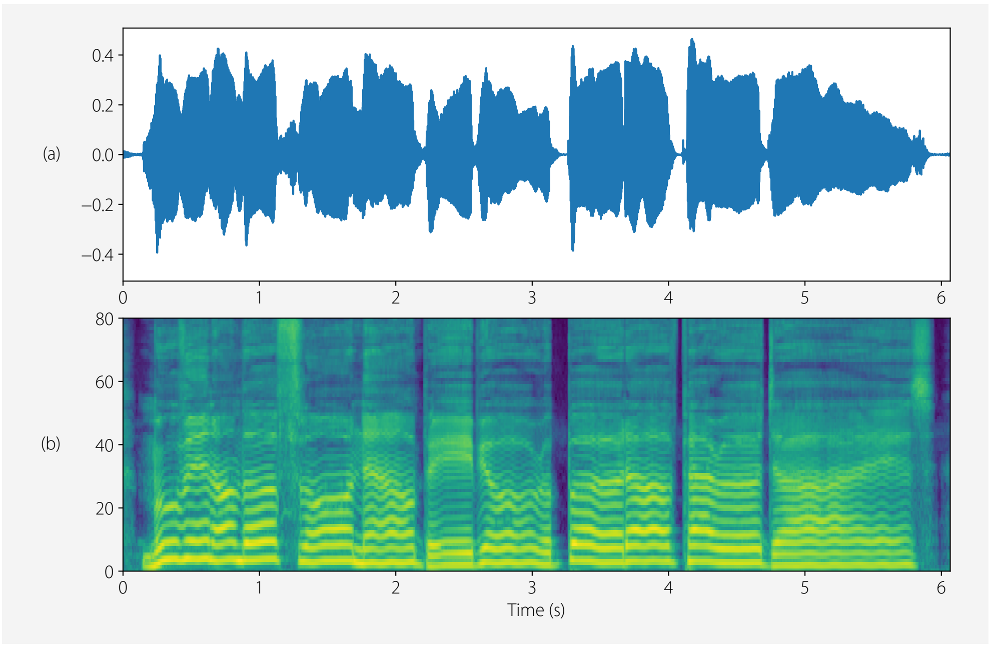

Figure 3.4: Example of a mel-spectrogram. Shown here are; (a) the waveform, and (b) the log mel-spectrogram (dB). This example uses 80 bands over a 50–15,000 Hz frequency range, a hop time of 5 ms, and a 32 ms Hann window.

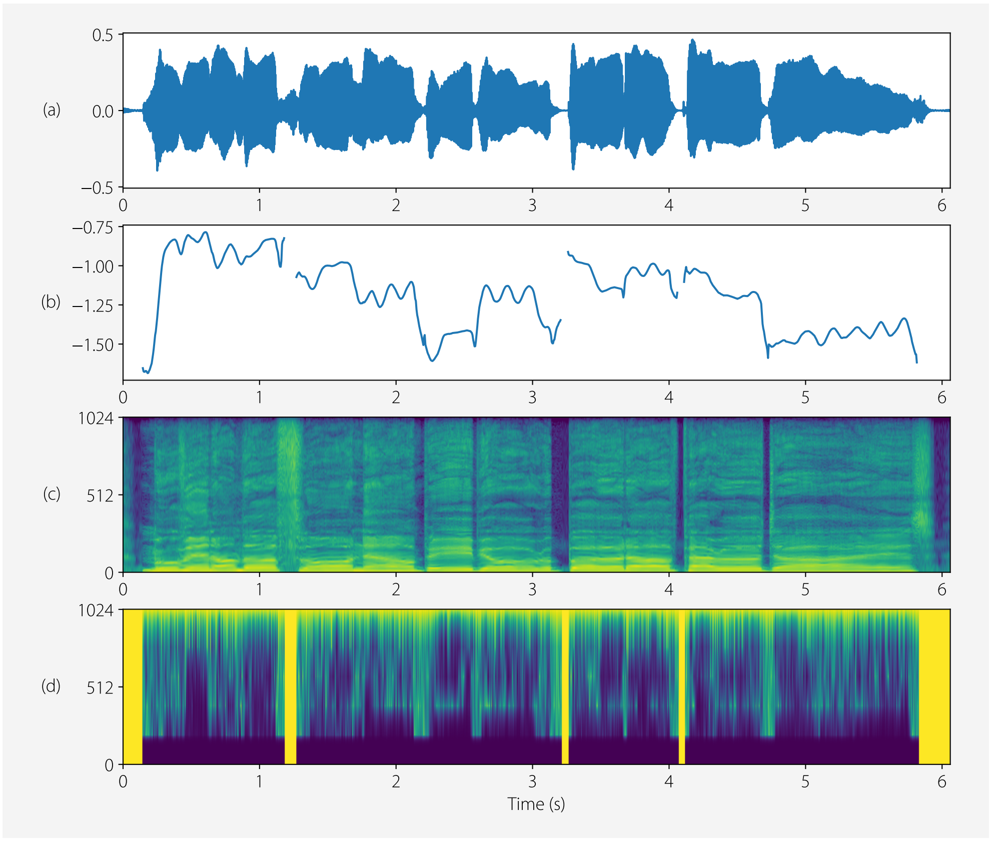

Figure 3.5: Example of WORLD features over time. Shown here are; (a) the waveform, (b) log F0 (octaves), (c) spectral envelope (dB), without dimensionality reduction, and (d) (squared) aperiodicity, interpolated to full dimensionality.

similar vocoder called STRAIGHT also exists (Kawahara et al., 1999; Kawahara, 2006). Both WORLD and STRAIGHT decompose the signal into three basic components as depicted in Figure 3.5; F0, a spectral envelope, and an aperiodicity feature. One key goal of these algorithms is to have as little cross-interference between the components as possible.

While WORLD includes several F0 estimators, in this work we usually rely on an in-house estimator especially optimized for singing voice, called SAC (Villavicencio et al., 2015). The main advantage of this estimator is that the default hyperparameters are suitable for a wide range of singing voices, resulting in few occurrences of common problems such as octave errors and voiced/unvoiced errors. Before modeling the F0 feature, we interpolate unvoiced regions, as modeling a continuous feature is easier than modeling a feature that combines continuous (voiced) properties and discrete (unvoiced) properties (Tokuda et al., 1999). We use linear interpolation as it is simple and avoids overshoot that can happen with higher-order interpolation, albeit being a somewhat unnatural pitch behavior and not continuously differentiable.

In some experiments we have also used modern data-driven neural networks such as that proposed by Gfeller et al. (2020) to perform F0 estimation. This approach is especially interesting as it self-supervised, that is, no F0 annotations are required to train the estimator, just audio. We have empirically found the performance of such an estimator to be on par with that of SAC. Being data-driven, in theory, allows the estimator to generalize to a wider range of voices, e.g., non-modal voices, without having to manually derive suitable heuristics. In practice, we have found performance on non-modal voices not to be particularly better than SAC, possibly due to data scarcity in those domains. That said, there are some practical benefits to SPICE; e.g., it has a continuous F0 output, as well as a continuous confidence output, which can be used as a continuous measure of voicedness. It is also more easily integrable in deep learning pipelines, whereas SAC generally needs to be pre-computed prior to training.

The spectral envelope is the "outline" of the spectrum, typically a smooth function that passes through the harmonics at lower frequencies and noise at higher frequencies. It corresponds to the vocal tract, as well as the spectral "shape" of the voice source. In WORLD, the so-called CheapTrick algorithm (Morise et al., 2016) estimates the spectral envelope. Without going into too much detail, it is based on spectral analysis with a pitch-adaptive window, followed by different steps that stabilize and smooth the spectral envelope, including cepstral liftering. The result is a kind of envelope spectrogram, which has similar dimensionality to a normal spectrogram, but is much smoother and does not contain any harmonic peaks.

The aperiodicity feature is a value between zero and one which determines how aperiodic (or harmonic) a given frequency is. This is an important feature of speech and singing voice signals, as these signals naturally have varying amounts of aperiodicity. For instance, there is a natural tendency for higher frequencies to be more aperiodic than lower frequencies. This is speaker dependent, varies depending on phoneme, may vary depending on the amount of breathiness of the voice, and so on. In WORLD, the so-called D4C algorithm (Morise, 2016) estimates aperiodicity. The resulting aperiodicity from this algorithm is a so-called band aperiodicity, which only predicts a relatively small number of aperiodicity values to cover the whole frequency axis. In this case, the range up to 15 kHz is split into 3 kHz intervals (e.g., for a 32 kHz sample rate there will be 4 bands). Frequencies at DC and Nyquist are fixed to 0.001 (−60 dB) and 1 (0 dB) respectively. Whenever a full bin-wise aperiodicity spectral envelope is required, the bands are simply linearly interpolated (in the log domain). Whenever the F0 estimator considers a frame to be unvoiced, aperiodicity is set to one for all bands.

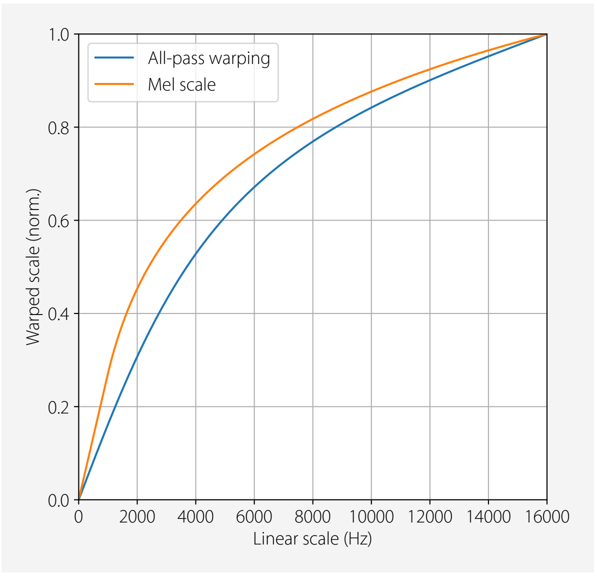

Figure 3.6: Comparing the frequency warping of the mel scale to the frequency warping of the all-pass filter used in the computation of mel-generalized coefficients (MGCs). In this case a sample rate of 32 kHz is used and a warping factor $\alpha = 0.45$.

It should be noted that while these algorithms include many heuristics involving certain constants, there are no real user-facing hyperparameters, except for the hop time (e.g., 5 ms). Very small hop times (e.g., 1 ms) generally improve quality, but this tends to complicate modeling these features, so this is not commonly done.

Similar to the mel-spectrogram, the spectral envelope features in parametric vocoders are often reduced in dimensionality to aid modeling, following a perceptually motivated frequency scale. While recent versions of WORLD include functionality to perform this operation (Morise et al., 2017), in our work we use cepstral domain frequency warping using an all-pass filter (Tokuda et al., 1994), followed by truncation to reduce dimensionality. This operation essentially serves the same function as the filterbank and discrete cosine transform (DCT) traditionally used in mel-frequency cepstral coefficient (MFCC) computation (see Figure 3.6 for a comparison of the two frequency scales). We then convert this cepstral representation back to the spectral domain by mirroring the cepstrum and computing the discrete Fourier transform (DFT), which results in a real spectrum. We refer to this representation as mel-frequency spectral coefficients (MFSCs), which are equivalent to MFCCs (or mel-generalized coefficients (MGCs)) without the final transform. We omit this transform because it makes the features easier to interpret, and the additional decorrelation effects of the transform, while potentially useful for traditional models, offer few benefits to modern deep learning methods. To take the inverse of this dimensionality reduction operation during synthesis, we simply cubic spline interpolate the log MFSCs. This is a fairly accurate approximation of the much slower cepstral inverse.

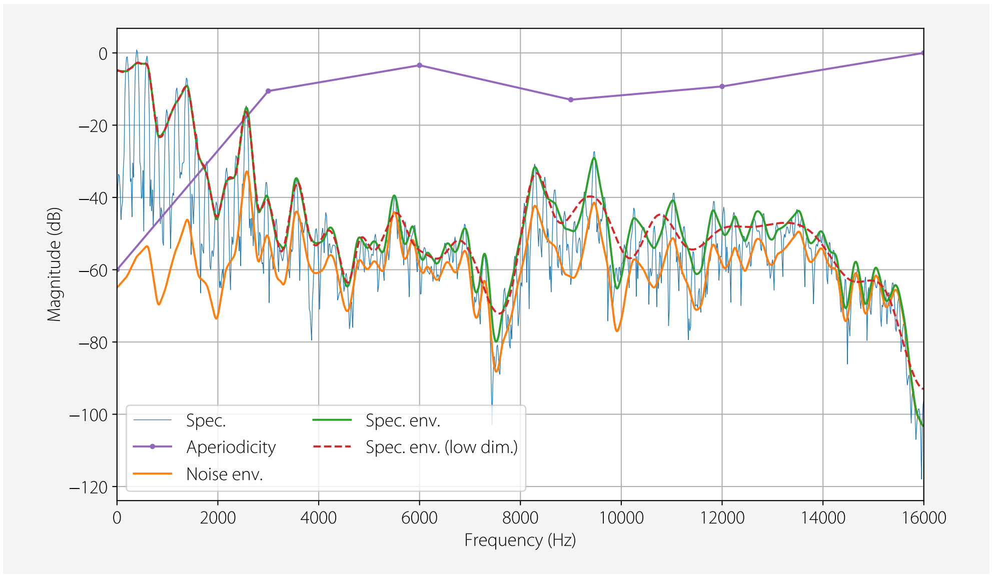

Figure 3.7: Example of a single frame of WORLD features with dimensionality reduction. Shown here are the spectrum (with a fixed 32 ms window), spectral envelope, spectral envelope reconstructed from reduced dimensionality mel-frequency spectral coefficient (MFSC) features, low-dimensional aperiodicity (in dB), and aperiodic envelope (aperiodicity added to spectral envelope).

Synthesis of WORLD features is quite close to traditional speech synthesis algorithms. The harmonic component is generated using voice pulses given F0. The aperiodic component is generated using a noise generator. Then, the two are mixed according to the aperiodicity feature. Finally, this voice source signal is filtered by the spectral envelope, ensuring a minimum-phase response (Smith, 2011, Chap. 4.9, "Minimum-Phase and Causal Cepstra", p. 147). Figure 3.7 shows the WORLD features corresponding to a single frame, as well as the spectral envelope after reconstruction from its low-dimensional representation.

Compared to mel-spectrogram features, parametric vocoder features have a number of potential advantages. First, one goal of these vocoders is to produce features that vary smoothly in time, thus modeling these should be easier than the mel-spectrogram with all its irregularities and high-frequency detail. Also, especially for singing voice, separating pitch and timbre is very interesting. The model may generalize better, without the need to apply data augmentation by pitch shifting for instance, thus speeding up training. Similarly, the model should be able to synthesize any given pitch to some degree, even if it was not seen in the training data. A synthesis algorithm based on a heuristic vocoder will typically be faster than a neural vocoder, and will be easier to get up and running. Evaluation of systems is also more straightforward compared to the mel-spectrogram and neural vocoder option, where determining whether an artifact is produced by the mel-spectrogram predictor or the neural vocoder can often be somewhat muddied.

It should be noted that copy synthesis (i.e., analysis-synthesis, without any modifications of the parameters) tends to be fairly transparent. The principal issues tend to be related to the lack of resolution in the aperiodicity feature (e.g., in voiced fricatives and such) and similarly voiced/unvoiced estimation errors. There may be some issues related to time resolution in plosives and such. Whenever dimensionality reduction is applied, the lack of resolution in high frequencies of the spectral envelope also results in some degradation. Regardless, with a model that makes good enough predictions, we may still obtain acceptable results with a parametric vocoder. In practice, often the spectral envelope estimated by parametric vocoders has notable local variance over time, thus encoding part of the vocal sound in these frame-to-frame variations. Even with powerful models, there will be some oversmoothing in both time and frequency in the predicted parameters. This in turn causes some of the most notable artifacts associated with the use of parametric vocoders; a certain "buziness" and "lack of detail" in the output sound.

One issue that arises with parametric vocoder features is that there can be multiple feature "streams", e.g., the spectral envelope and aperiodicity features. The simplest approach to this issue would be to concatenate features and model them with a single model. Alternatively, separate models could model each stream individually, allowing the models to be optimized for their specific task. However, this approach has the potential downside that the correlation between streams is lost in the predicted features. A cascade of separate models can work around this, by conditioning each model on its predecessor.

Finally, some of the concepts behind parametric vocoders, such as the separation of the signal into source and filter components can also be applied in neural network models. This is the case because much of these operations can be formulated in such a way that they are differentiable and thus can be optimized using standard gradient descent. This kind of approach has been successfully applied in several works (e.g., Lee et al., 2020; Wang et al., 2019; Wang et al., 2020b; Wang and Yamagishi, 2019; Wang and Yamagishi, 2020).

3.1.4 Control input features

One issue of special importance in the general design of singing synthesizers is deciding which features to condition the model on and how to present these to the network in a suitable fashion. We could consider this part of the model or network design, but it deserves some special attention that can carry over between models and networks. In particular, we may think of the network consisting of an encoder part, which takes the control inputs and encodes them in a suitable way, and a decoder part, which takes the encoded inputs and produces the final output. The encoder can be either a neural network or some heuristic process. The former approach is arguably the most powerful, as a data-driven neural network may be able to extract higher-level information, beyond what is reasonably possible with a heuristic process designed by hand. On the other hand, the latter approach offers a way of introducing some inductive bias, which may be beneficial to improve generalization, or help the model converge during training for instance.

Assuming we are using a system that predicts intermediate acoustic features (see §3.1.2) and an input sequence based on timed phonemes (see "Simplified task definition" in §3.1.1), we will have mainly phoneme-level input and frame-level output. There may be some additional frame-level inputs as well, such as F0 or positional features. At some point, phoneme-level features have to be converted to frame-level features, typically by repeating phoneme-level feature vectors by the corresponding phoneme duration in frames. We can either apply an encoder to the phoneme-level features and then repeat the resulting vectors to frame-level features, or first repeat the phoneme-level features to frame-level features and then apply the encoder. If the encoder integrates information over several timesteps, e.g., using a convolutional neural network (CNN), there are significant differences in both approaches. In particular in the latter approach, the encoder may not be able to effectively integrate information across phonemes if the durations are significant with respect to the encoder's receptive field. In this case, it can be beneficial to include information of consecutive phonemes within the feature corresponding to a frame (e.g., a "window" of 5 phonemes around the phoneme of the timestep in question).

Categorical features, such as phoneme identity, can be embedded into an n-dimensional space, or one-hot encoded as a one-hot vector and then passed through a 1×1 convolution, which generally will be equivalent.

Real-valued, continuous features can either be kept as is, or occasionally can be encoded as a coarse vector. When kept as is, the values should generally have a suitable range, such as zero mean and unit variance, although often a very exact normalization is not necessary. Coarse coding of continuous features is mainly interesting under the assumption that certain features have an effect on the data to be modeled, but that this effect does not depend on the exact value of the feature. For instance, an octave difference in F0 may affect timbre notably, but one or two semitones may not. A long or very long duration phoneme may behave similarly, but notably different from a short duration phoneme. Coarse coding typically involves mapping a scalar value to a low-dimensional vector of values ranging between $[0, 1]$. One approach is a linear

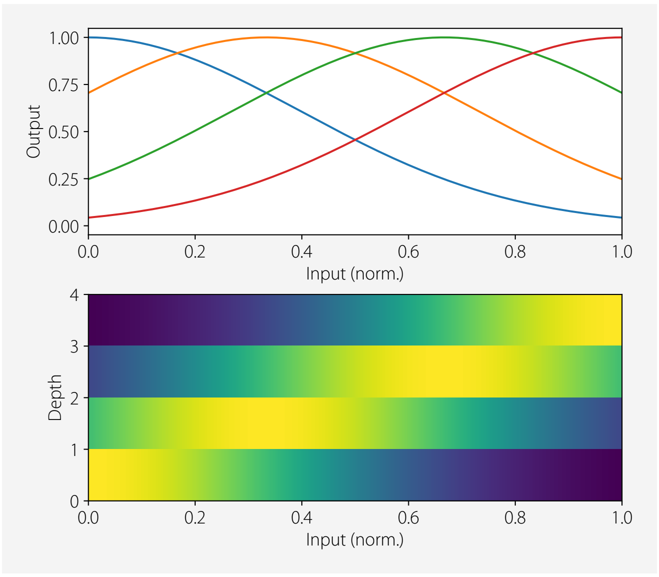

Figure 3.8: Example of a control encoding using equidistant Gaussians. Shown here are equivalent 1-d and 2-d views of a Gaussian encoding with $K = 4$ and $\sigma = \sqrt{1/(2\pi)}$.

coarse coding, where we pick K values for a K-dimensional coarse coding and linearly interpolate between these values depending on the continuous input feature (see Figure 3.9 for an example). Another approach is Gaussian coarse coding, which typically places K uniformly spaced Gaussians along a normalized input range $x \in [0, 1]$,

$$v_{i,k} = \mathcal{N}(x_i;\ k/(K-1), \sigma^2), \tag{3.6}$$

for $k \in \{0, \dots, K-1\}$ and $i \in \{0, \dots, T-1\}$, with K being the dimensionality of the encoding, and T are the number of timesteps. Here, we can use $\sigma^2 = 1/(2\pi)$ so the response at the mean of the Gaussians is one, and K is generally small (e.g., 3 or 4). An example of this type of control encoding is depicted in Figure 3.8.

A set of ascending frequencies $y = [y_0, \dots, y_{K-1}]$,

$$v_{i,k} = \begin{cases} 1 & \text{if } k = 0 \text{ and } x_i < y_k \\ 1 & \text{if } k = K-1 \text{ and } x_i \le y_k \\ \max(\dfrac{x_i - y_{k-1}}{y_k - y_{k-1}}, 0) & \text{if } x_i < y_k \\ \max(1 - \dfrac{x_i - y_k}{y_{k+1} - y_k}, 0) & \text{if } x_i \ge y_k, \end{cases} \tag{3.7}$$

for $k \in \{0, \dots, K-1\}$ and $i \in \{0, \dots, T-1\}$, with K being the dimensionality of the encoding, and T are the number of timesteps.

One continuous feature of particular importance is F0. One approach to encoding it is to use the linear coarse coding method above, using log F0 as input. We can define some fixed frequency bands derived from the singer's pitch range, e.g., by looking at a histogram of the dataset's log F0 and listening to changes in timbre, as depicted in Figure 3.9. While this will introduce some inductive bias which may be beneficial in terms of generalization and convergence of the model, it requires more manual intervention when creating a voice. Additionally, when combining data from multiple speakers, it becomes less clear what would be a good set of frequency bands. An alternative, data-driven approach is to use a neural encoder, such as a small CNN. The input to such as network could be log F0 in octaves relative to some central frequency such as 440 Hz, such that,

$$v_{i,\cdot} = F_\theta(\log_2 \frac{f_i}{440}), \tag{3.8}$$

where $F_\theta(\cdot)$ is the encoder CNN parametrized by θ, producing a K-dimensional vector. One advantage of using a CNN is that it is easy to integrate information over multiple timesteps, thus hopefully making the encoder more robust to things like single-frame outliers, being able to use information about the direction of pitch, and so on. Using a CNN with a small number of layers (e.g., one 9×1 layer), a small number of output channels (e.g., 5), and an output activate that limits the output range (e.g., $\mathrm{sigm}(\cdot)$ or $\tanh(\cdot)$), we should hopefully end up with something that also has most of the benefits of the "handcrafted" heuristic encoder. At the same time, this approach does not require specifying any frequency bands, and combining data from multiple speakers is straightforward.

When using frame-level control features, it can be beneficial to include information about the frame's timestep. Such positional features can be either relative scale, e.g., the normalized position within a phoneme or note, or absolute scale, i.e., the absolute frame index relative to phoneme or note boundaries. Either or both of these can be used, typically in combination with features derived from phoneme and/or note durations. One encoding scheme is to use the Gaussian coarse coding, e.g., with 3 states corresponding to begin/middle/end, following Equation (3.6). Another scheme is a K-dimensional cyclical encoding of the normalized relative scale frame position, $p \in [0, 1] \subset \mathbb{R}$,

$$v_{i,k} = \frac{1}{2} \cos\left(2\pi p_i - 2\pi \frac{k}{K}\right) + \frac{1}{2}, \tag{3.9}$$

for $k \in \{0, \ldots, K-1\}$ and $i \in \{0, \ldots, T-1\}$, with K being the dimensionality of the encoding, and T are the number of timesteps. This approach can be argued to better represents the continuous nature of phoneme boundaries compared to e.g., Gaussian positional encoding. An example of this type of encoding is depicted in Figure 3.10. Finally, sinusoidal encoding (Vaswani et al., 2017) is also frequently used for encoding

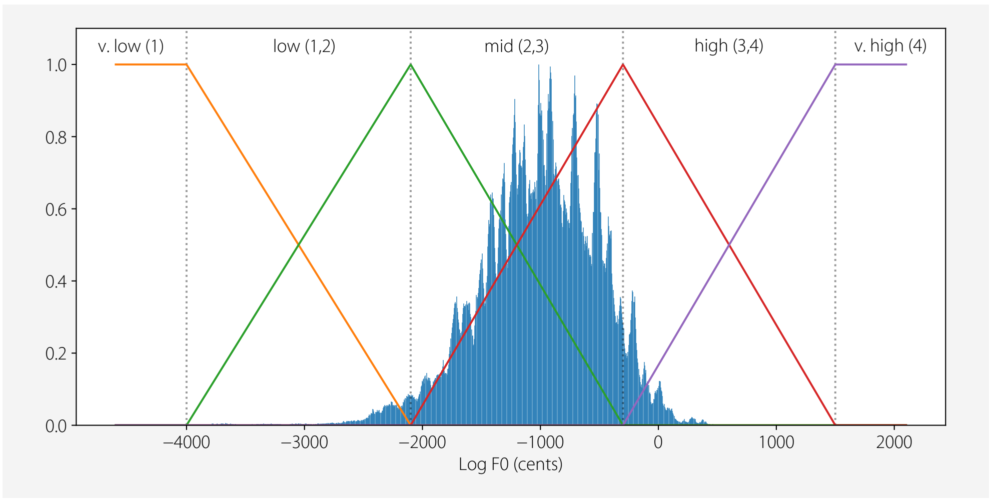

Figure 3.9: Example of a linear coarse coding of log F0 (cents). We show the histogram of log F0 for a given (male) singer and an example linear coarse coding with dimensionality $K = 4$, and approximately equidistant center frequencies $y = [-4000.0, -2100.0, -300.0, 1500.0]$. At the top of the plot, each pitch range is denoted, with the activated dimensions in parenthesis.

position; often absolute scale position in models that include self-attention layers (see Chapter 5),

$$\begin{aligned} v_{i,2j} &= \sin(i/10000^{2j/K}) \\ v_{i,2j+1} &= \cos(i/10000^{2j/K}), \end{aligned} \tag{3.10}$$

for $j \in \{0, \dots, K/2-1\}$ and $i \in \{0, \dots, T-1\}$, with K being the dimensionality of the encoding, and T are the number of timesteps. An example of this type of encoding is depicted in Figure 3.11.

The way in which conditioning is integrated into the network architecture is also something that may have an effect on the design of the input control encoding. One common approach is to have the encoded control inputs enter the network at the bottom and work their way up the layers through a series of non-linear transformations. Another approach, typically used in autoregressive models (e.g., van den Oord et al., 2016a), is to integrate the conditioning signal directly (often only transformed by a 1×1 convolution) into each layer of the network. While the former can technically be seen as a specific case of the latter (i.e., all but the first layer could ignore the conditioning signal), in practice the latter case will probably favor a much more direct use of the encoded conditioning signal. Thus, especially in the latter case, a more powerful (neural) encoder may be warranted, whereas in the former case the decoder itself may partially make up for a simpler encoder (e.g., relying more on heuristics).

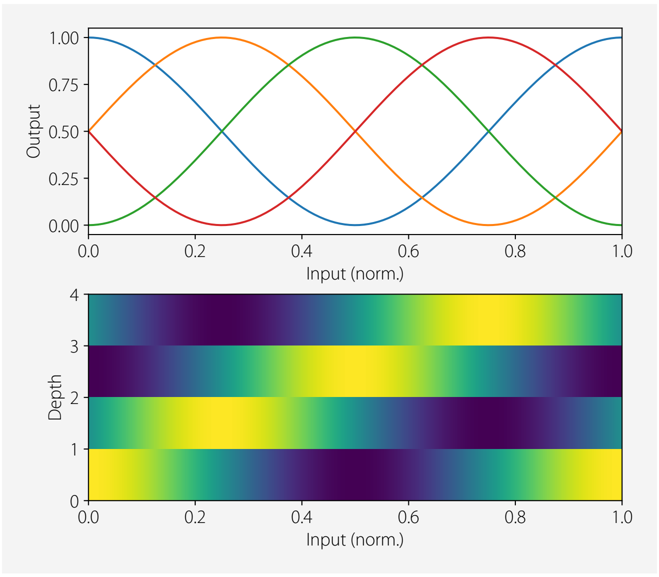

Figure 3.10: Example of a cyclical control encoding. Shown here are equivalent 1-d and 2-d views of a cyclical encoding with $K = 4$.

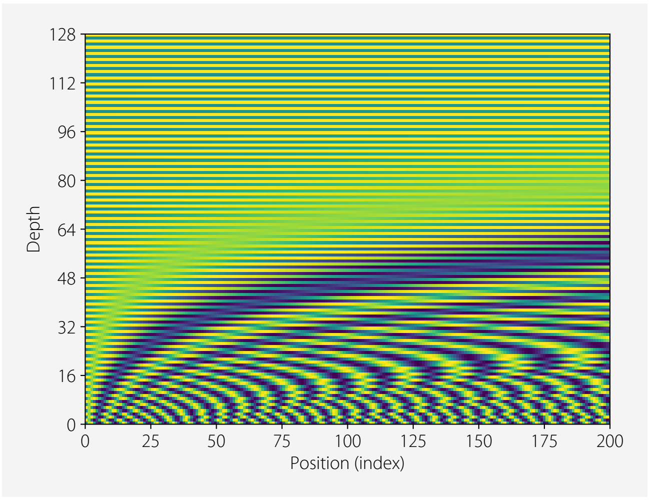

Figure 3.11: Example of a sinusoidal positional encoding. Shown here with $T = 200$ timesteps and dimensionality $K = 128$.

Grapheme-to-phoneme conversion

Generally, this is done using either of two methods; rule based, or dictionary based. This depends on the language. Some issues are out-of-dictionary words (typically rule-based fallback), foreign words, names, heteronyms (words that are written the same, but pronounced differently depending on context).

Issues specific to the singing voice, such as elongated schwas converting to other vowels in English. Maybe melismas and elations, etc. (Ríos Mestre, 1999)

3.1.5 Performance-driven synthesis

A useful technique in singing synthesis research is so-called performance-driven synthesis (Janer et al., 2006; Goto et al., 2012). In this approach, rather than synthesizing from a musical score input, a reference recording is used to provide some of the features needed to produce the output waveform. These features typically will always include phonetic timings, as this ensures the timing of the synthesis will closely match that of the reference recording. The other commonly used feature is F0, but could also include other features such as loudness.

The main reason why we use performance-driven synthesis in some parts of this work is to aid evaluation. This approach allows us to compare certain parts of the singing synthesizer in isolation, by keeping some of the features used in the synthesis constant. For instance, we can use realistic pitch and timing features extracted from a reference recording and only focus on evaluating different timbre models. Other reasons why performance-driven synthesis can be useful are typically more related to practical applications: Sometimes it is simply the fastest way to obtain realistic sounding results. Even in these cases, while the melody will be generally fixed, we can still change speaker identity (perhaps with some octave pitch shifting). Lyrics can often also be changed to some extent, even modified quite freely if care is taken while creating the reference recording. For instance, if the reference recording is just alternating vowels, it will be mostly free of microprosody (Taylor, 2009, Chap. 9.1.4, "Micro-prosody", p. 229). In this case, lyrics can be assigned freely, as long as the syllable structure fits well into the note melody. Note that in this case we would typically use note timings extracted from the reference recording, combined with a phoneme duration model (see §4.3.3).

In practice, obtaining F0 features from a reference recording can be done mostly automatically using a suitable estimator. In some cases, results can be slightly improved by some manual correction (mostly around voiced/unvoiced boundaries), or occasionally by moderate smoothing of the extracted features if they exhibit excessive frame-to-frame variations. Like when training the model, we linearly interpolate unvoiced regions to have a continuous feature. This way we can rely on the predicted voiced/unvoiced decision to ensure that timbre and voicing of the excitation match. If we do not ensure this, synthesizing voiced timbres with unvoiced excitation can cause noise bursts, and synthesizing unvoiced timbres with voiced excitation can cause a "buzzy" sound. Obtaining phonetic timings tends to be slightly more problematic: One approach is to use forced alignment between the reference recording and a hidden Markov model (HMM) corresponding to the phonetic sequence. The HMM will often either be a pre-trained speaker-independent model designed for automatic speech recognition (ASR), or the reference recording can be self-segmented as part of a larger set of recordings (see "Self-segmentation using deterministic annealing expectation-maximization" in §3.2.7). Depending on the language, the results of this approach are often not accurate enough, and will need some manual correction.

3.2 Datasets

As is generally the case with data-driven machine learning approaches, datasets are very important to the success of singing synthesis models. However, a standout aspect compared to other domains is that singing datasets involve humans, musicality and expressive performance. Even in the most closely related domain, TTS, datasets are often recordings of neutral speech, often without too many requirements with respect to the recorded speakers or text.

The research group within which this work was conducted has a fairly extensive experience creating singing datasets that was amassed over the years. This experience does not only involve datasets that were specifically designed for the now ubiquitous machine learning approaches around which this thesis is centered, but also datasets that were intended for older approaches, in particular concatenative synthesis.

Up until recently, there have not been many publicly available datasets suitable for singing synthesis. As such, much research on this topic uses proprietary datasets, where many of the details of how the datasets were collected are not discussed in depth in the associated publications. Therefore, we consider sharing some of our personal experience, thoughts and practical know-how on the dataset creation process a valuable contribution. That said, we have not conducted any rigorous scientific research on this topic.

3.2.1 Public datasets

To preface our own experience with singing dataset creation, we will discuss some of the available public datasets. This is not an exhaustive listing, but rather to give an overview for context.

While many public datasets include singing vocals, not all of them are suited for singing synthesis research. Particular requirements of the dataset of course vary a little depending on the model that will be trained using it. However, most models will have the same basic requirements. Recordings should generally cover a lot of phonetic contexts. This may exclude some datasets that consist of vocal exercises, scales, single vowels, single syllables, single words, and so on. Recordings should generally be high-quality studio recordings. This excludes some datasets that include processed vocals (i.e., "stems"), amateur "user" recordings (e.g., recordings collected from karaoke apps), low sample rate recordings, and so on. The amount of material recorded of a singer should also be sufficiently large. In some cases, a small number of recordings per singer can be compensated by having a large number of singers in a dataset. The quality of the singing itself is also of importance. In particular whether or not the singers are professional-level singers, and whether or not they are native speakers of the language. Finally, for many models, the annotations provided by the dataset are important. Of particular interest are phonetic transcription, phonetic segmentation, note transcription, and possibly other expressive annotations.

When we only consider datasets with at least some form of annotations, we are aware of only a few publicly available datasets; one for English, three for Japanese, and very recently two for Mandarin Chinese. That said, there does seem to be an uptick in datasets especially designed for singing synthesis research being publicly released by research groups.

For English there is NUS-48E (Duan et al., 2013), this is a multi-speaker dataset with 12 speakers and 4 songs per speaker. It should be noted that the singers recorded for this dataset are amateur singers, and non-native English speakers (Malaysian).

For Japanese, there is NIT-SONG070-F001, JSUT-song and the Tohoku Kiritan Singing Database. NIT-SONG070-F001 is part of the demo datasets and recipes included with HMM Speech Synthesis System (HTS) (Zen et al., 2007). This is a professional female Japanese singer singing nursery rhymes. Out of the 70 songs recorded, the public dataset includes 31 songs, totaling 31 min. JSUT-song is similar to NIT-SONG070-F001, also a (different) female Japanese singer singing the same nursery rhymes. In this case 27 songs, totaling 25 min. The Tohoku Kiritan Singing Database consists of recordings of a female Japanese singer, singing pop songs (J-pop and anime songs). This dataset includes 50 songs, totaling 3 h 31.

For Mandarin Chinese, there is OpenSinger (Huang et al., 2021) and Opencpop (Wang et al., 2022). OpenSinger is a multi-singer dataset of 93 professional singers, totaling around 50 h, and sampled at 24 kHz. Opencpop on the other hand is a single-singer dataset, of a professional female singer, singing 100 pop songs, sampled at 44.1 kHz, and including phonetic and note annotations.

3.2.2 Types of datasets

When creating a vocal dataset, some different approaches can be taken. These mainly affect what to record and how to instruct the singer. In particular, the amount of expression that the singer utilizes may be controlled to reduce the variability of the data. The idea behind this is that more coherent data may help improve the model's performance, given that in practice data is often limited. In this work, we focus mainly on two types of data; what we call pseudo singing, and natural singing.

Pseudo singing

So-called pseudo singing is a constrained type of singing, that can be considered to be in between natural speech and natural singing. This allows to reduce the variability in the data, but comes at a cost of also reducing naturalness compared to natural singing. These kinds of datasets were initially designed as a practical solution for recoding diphone inventories for concatenative synthesis. In this case, we want to "cover" all diphones in a certain pitch, and make sure phonetic context and timbre in general are as coherent as possible. Additionally, using flat pitches tends to be beneficial to many signal processing techniques applied in concatenative synthesis. Thus, pseudo singing generally asks the singer to sing a sentence as a single pitch, with constant dynamics and cadence, and a clear and coherent pronunciation. Once all sentences are recorded at a given pitch, the whole process is repeated at different pitches (usually a total of 3 or 4), to cover a range of typically 1–2 octaves. In theory, different tempos and dynamics can also be recorded, but this was not done often in practice.

This approach has a number of advantages compared to natural singing. It allows for very controlled recordings, where we can ensure that certain criteria are met, e.g., covering a certain set of diphones in a certain set of pitches, ensuring (mostly) clear pronunciation, etc. Also, the amount of recorded data tends to be less, as phonetic contexts are covered more efficiently, there are no long notes, no long rests, no repetition, etc. Finally, automatic annotation, such as phonetic segmentation, tends to work much better on pseudo singing than the much more variable and expressive natural singing.

That said, there are also a number of very significant downsides to this approach. Arguably, the biggest downside is that if our goal is to synthesize natural singing, the training data should not deviate significantly from that kind of data. That is, while in practice in certain cases pseudo singing can outperform natural singing, ultimately this should not be the case, e.g., given sufficient data, powerful models, and so on. The second-largest downside of this approach is arguably that performing pseudo singing tends to be significantly more difficult for singers than natural singing. Sentences can be unfamiliar and occasionally constructed in unusual ways, and very monotone singing for extended periods can be taxing. Especially when this technique is used for diphone concatenative synthesis, the singer has to be very careful to perform the diphone in question as required, e.g., avoiding pauses in diphones extracted from word boundaries. All of this, often causes the recording times of pseudo singing to exceed that of natural singing, even though the final dataset may be smaller. Additionally, singers may become frustrated they cannot show off their abilities as skilled singers.

There are some ways to relax the constraints of pseudo singing a little. These especially make sense when these kinds of datasets are used with modern synthesis techniques that rely less heavily on manipulation by signal processing. For instance, we may allow the singer to pick a free melody and rhythmic pattern, especially in the cases where note transcription is not required. Another approach is to construct sentences with certain properties, and then compose corresponding melodies. This latter approach is used by Koguchi et al. (2020) for their PJS dataset, which in their case allows for compact, phonetically balanced scripts, as well as avoid copyright-related issues which can be especially important when making datasets public.

Natural singing

Natural singing as the name suggests is simply recording a singer naturally singing songs. The main issue here is that we have little control over what we "cover", e.g., combinations of phonetic contexts and pitches. The most straightforward strategy to combat this is to record a lot of data, which depending on the model used, could lead to requiring a lot of annotations such as note transcription or phonetic segmentation. These annotations often have to be corrected manually for sufficient accuracy, thus making the voice creating process costly in terms of time and skills required. Furthermore, using natural singing can lead to cases where the singer produces excessive expressive variability with respect to the annotations available. For instance, a singer may use a drastically different timbre or style of singing within a song, while a model would not be able to infer this by just looking at the score. Additionally, there can be a high variability between different songs. In our experience, this can be mostly mitigated by a deliberate song selection, as well as carefully instructing the singer.

3.2.3 Creating recording scripts

Before recording, we need to establish a recording script that must be performed by the singer. In the case of pseudo singing, this can be a set of sentences and a list of pitches, and in the case of natural singing a set of songs.

In the case of pseudo singing, typically we can select sentences from a large corpus, following some optimization criterion, e.g., the least amount of sentences that covers a set of diphones. This will often involve additional constraints to make the recording as easy as possible for the singer. For instance, using all sentences with a fixed number of syllables can really help establish a rhythm while recording with a constant cadence. There often also is some final manual revision and correction of sentences selected. These techniques are very similar to those employed in TTS corpus generation (e.g., Bozkurt et al., 2003).

In the case of natural singing, ideally the singer we record can provide a unique repertoire of original songs in a coherent style, while also being sufficiently large. In practice, this is often not the case. We could compose new songs especially for this purpose, but this is typically not done due to the effort required (both to compose the songs, and for the singers to learn these new songs). Thus, the most practical solution is typically to select existing songs. Some criteria for these songs can include how familiar the singer is with the songs, and how well they fit the target style. Especially when the dataset is made public, copyright issues should be considered here. These issues are likely to depend on the applicable jurisdiction and intended use of the data. One common solution, although non-ideal, is to use public domain songs, such as nursery rhymes (e.g., Takamichi et al., 2018). We expect laws regarding using copyrighted data for training machine learning models to become better defined in the future and this seems to already be changing in some jurisdictions (e.g., Ogawa and Morise, 2021).

3.2.4 Selecting singers

When it comes to selecting singers, there are no hard and fast rules. One of the things we tend to look for is whether the singing style is what we want and whether the timbre is attractive according to our subjective opinions. Typically, we favor singers with a large tessitura as this results in a more versatile synthetic voice. Another important issue is whether the singer is a native speaker of the target language (and dialect), as singers may be able to sound convincingly in a non-native language, but pronunciation tends to be more variable in our experience. Finally, one of the more important aspects is availability and interest in collaborating in the project.

In the case of pseudo singing, one additional requirement is that the singer has to be able to follow scripts very accurately, which is a skill quite unrelated to being a good singer. We find that singers who also work as voice actors tend to be a good fit in these cases.

3.2.5 Recording datasets

Here we will discuss some practical know-how related to recording singing datasets.

Studio setup

Ideally, we will have a silent, low reverberation room for recording the singer, and a separate "control room" where the studio engineer can be during the recording. In practice, many compromises can be made, e.g., acoustically treating a room with curtains, using a microphone isolation shield, simply being quiet during recordings when a separate control room is not available, and so on.

We typically use a large-diaphragm condenser microphone such as the Neumann U87, which performs well for male and female vocals, and can handle high sound pressure levels without distortion. We typically use a mic stand with a pop shield at around 2.5–5 cm from the microphone to attenuate plosives, sibilants and affricates a little. We use a closed headphone for playback of accompaniment and talkback from the control room. When needed, we also use a large screen monitor at a distance to display lyrics, or occasionally a stand to support paper scores and lyrics.

We initially place the singer at a distance of around 30 cm, then instruct the singer to sing some phrases in lowest and highest dynamics, and adjust the distance from there if needed to ensure proper recording levels and comfort for the singer. We consider it important to keep the distance to the microphone as constant as possible, so we instruct the singer to do so. Additionally, especially when the recording session is split over several days, we may take photos and mark the positions on the floor.

Generally, we record without any dynamics processing, ensuring that we record at a high bit depth (e.g., 24 bit) to avoid quantization noise and low enough level to avoid clipping. While most of our models currently use a lower sample rate, we tend to "future proof" recordings by using a high sample rate such as 96 kHz.

Reference recordings

We may conduct some special recordings at the beginning of a session that are used as a reference to keep timbre coherent (i.e., played back to the singer later). This can also be a good time to set levels, in particular singing high pitches at high dynamics, to get a feel of the highest loudness, in order to avoid clipping. In some cases, recording some duration of silence can also be a good idea, for possible future noise reduction, if the recording environment is less than ideal.

Accompaniment

Accompaniment or background music is very important for the singer to keep in tune to some master tuning (typically 440 Hz), and not use a different tuning, drift over time, or vary from song to song. Additionally, the accompaniment can ensure the singer is on-time with respect to a score. Thus important things to consider when selecting existing accompaniment audios is whether the tuning is actually what we expect, and whether the timing is not variable. In some cases, we should also ensure accompaniment does not bleed through from the singer's headphones, and possibly apply some EQ to attenuate high-hats and such.

For natural songs, karaoke tracks can be a practical solution, although the quality may vary. For pseudo singing, some tools for automatic composition may be used. In some cases, accompaniment can even be controlled live (e.g., playing chords on a fixed tempo).

Managing singer fatigue

In our experience, typically it is best to keep sessions under 4 h per day to avoid the singer becoming excessively fatigued. Obviously, this will depend on many factors, such as the available time, the singer itself, and whether additional recording days are acceptable.

In general, we have found that it is often beneficial to record more material rather than getting the recordings "perfect". In particular, a singer may be accustomed to redoing a part many times in order to get the perfect take for an album recording, while in this case a single take may do even if there are minor imperfections. In some cases, minor imperfections in lyrics can be edited later. However, things such as singing out of tune are not acceptable (nor expected from a professional singer).

3.2.6 Post-processing

What, if any, post-processing to apply to the recorded audios depends a lot on the recording itself and the target voice. As a general rule, we apply as little processing as possible to the voice, with the idea that any desired post-processing can also be applied to the synthesis output.

Typical post-processing includes loudness normalization, using standard loudness measures like ITU-R BS.1771-3, ensuring that natural relative differences in loudness due to varying dynamics and pitch are maintained. Occasionally, some light dynamics compression, including de-essing, is also applied. In some cases, we apply some global EQ correction. Very low frequencies, e.g., sub-sonic rumble well below the fundamental, will also typically be cut.

3.2.7 Annotating datasets

To annotate datasets we generally use a semi-automatic process that we will briefly describe here. In this case, we will describe just annotating data with a timed phonetic sequence (phonetic segmentation), rather than e.g., a note score, because this is the type of annotations most frequently used throughout this work.

1. Split the larger recordings, e.g., whole songs, into phrases. Here, we consider a phrase a continuous utterance without significant pauses. That is, we split on longer silences, which often coincide with a longer aspiration. This step can generally be done automatically, but splitting manually also tends to not require too much effort.
2. Split the lyrics according to the above audio split. At this point, we can also correct gross errors in the lyrics, e.g., where the singer deviated from the lyrics.
3. We perform automatic phonetic transcription of lyrics. See "Grapheme-to-phoneme conversion" in §3.1.4.
4. Do an initial automatic phonetic segmentation using the process described in "Self-segmentation using deterministic annealing expectation-maximization" below.
5. Refine phonetic segmentation by hand. At this point, we generally refrain to correct phonetic segmentation as this tends to be hard to do consistently. That is, we prefer to keep the automatic phonetic transcription which is always coherent, even if this may not always be strictly correct, assuming that many models will be able to compensate for these kinds of discrepancies.

Self-segmentation using deterministic annealing expectation-maximization

Perhaps the most common traditional method of performing phonetic segmentation is taking a pre-trained speaker-independent HMM and performing forced alignment. Generally, the pre-trained models are speech models designed for ASR, rather than singing models. We found that this approach tends to perform rather poorly, resulting in many gross errors, e.g., where the left-to-right forced alignment at some point deviates from the true segmentation and never recovers. Even in cases where there are no gross errors, the accuracy of this approach can still be fairly low, i.e., due to systematic and no systematic errors.

Instead, we prefer an approach where we segment data using forced alignment of an HMM model trained from scratch on the same data. In order for this self-segmentation approach to be effective, we train the HMM model using the deterministic annealing expectation maximization (DAEM) algorithm (Ueda and Nakano, 1998) rather than the more common expectation maximization (EM) algorithm. In this case, the model will be speaker-dependent, and thus we need to have sufficient data to train a model. That is, for multi-speaker models with a relatively small amount of data per speaker, this method is less suited.

We basically follow the recipe included with HTS (Zen et al., 2007) to train the model, but we will summarize the most important aspects below. We use 13-dimensional MFCC features extracted from audios downsampled to 16 kHz, with a 10 ms Hamming windows, a 5 ms hop, and delta and delta-delta features. We use 5 state left-to-right monophone hidden semi-Markov models (HSMMs), that is, an HMM with each state duration modeled with a Gaussian. The model is trained from random initialization (flat-start), then the individual monophone models are re-estimated, and finally the embedded monophone models are re-estimated (i.e., all models jointly).

We found that context-dependent, e.g., triphone, models are not beneficial for determining phoneme boundaries. Likewise, we generally use a single Gaussian component in the HMMs.

In some cases, there may be some systematic errors in the phonetic boundaries. In such cases, it can be beneficial to perform the above process, manually correct some of such boundaries, and re-train a model, initializing the model from the manually corrected segmentation.

3.3 Evaluation

The evaluation of singing synthesis systems is generally not straightforward as many of the aspects of the output audio that we are interested in, such as sound quality, naturalness, expressiveness, intelligibility, and so on, are *perceptual* quantities. We first discuss some of the issues surrounding evaluation using objective metrics based on signal processing or statistics. Given these issues, we then discuss what we consider to be the current “gold standard” method of evaluating TTS or singing synthesis systems; qualitative evaluation using listening tests. While this method is not without its own share of difficulties, it is the method we use most widely in this work. That said, quantitative metrics, particularly based on signal processing, can offer some insight and is therefore discussed next. Finally, we briefly discuss non-intrusive speech quality assessment, which does not require reference audios and, more importantly, are based on powerful data-driven neural networks. While we do not use these approaches in this work, they deserve special mention as they seem like a potential way forward beyond the current issues related to evaluation and are very actively researched at the time of writing.

3.3.1 Issues related to evaluation

First, we will discuss some of the issues related to the evaluation of singing synthesis systems. These issues relate mostly to objective evaluation, and highlight the reasons why subjective evaluation is generally preferred.

Metrics with reference

Arguably, one of the most straightforward methods of evaluating something like a TTS or singing synthesis model is to use what is called intrusive evaluation, that is compare a sample from the model to a reference, using some metric that can be easily computed. For our task, this tends to be a feasible approach as we can condition the model in such as way as to ensure that the output will be at least close to the reference, e.g., using performance-driven synthesis (see §3.1.5). Even in such a case, one issue is that TTS and singing synthesis tends to be a one-to-many mapping, i.e., one given input control sequence may map to many possible output sequences. Thus, it is not reasonable to expect the model's output to be exactly identical to a given reference signal. At the same time, there are many output sequences that we cannot consider correct; for instance, we may synthesize a sequence that matches the target melody and lyrics, but sounds like a different singer. This leads to a kind of catch-22 situation where in order to evaluate whether a sample of a model is within the acceptable range of outputs, we would ideally need a similar model to do so.

Another important issue is that the metrics used to compare signals should ideally be highly correlated to how humans perceive signals to be close or not. A particular issue is that signals may have significant differences, while still being perceived as identical. This leads to many metrics being somewhat unreliable. Typical signal differences that are less perceptually relevant include small differences in timing, differences in low energy bands masked by higher energy bands, things like vibratos being out of phase, differences in phase in general, and so on. As the metrics used tend to be designed by hand, it is difficult to account for all such possible issues.

Likelihood and sample quality

As many TTS and singing synthesis models use probabilistic formulations, using the average log-likelihood on a test set may seem like a good way to evaluate and compare models. However, Theis et al. (2016) has shown that a model may have a good average log-likelihood, but produce low-quality samples, and vice-versa. Furthermore, this approach is limited to models where computing log-likelihood is tractable, which is not the case for many models. In those cases, approximations to log-likelihood may be used, but these approximations, such as Parzen window estimates, can introduce additional issues.

Issues with waveform models

While many models may produce some kind of intermediate acoustic features (see §3.1.3), the final output we are interested in is a waveform. Unfortunately, evaluating waveforms is especially difficult compared to many other kinds of signals. While only one-dimensional, audio waveforms tend to have high sample rates (e.g., 32 kHz), resulting in a high total dimensionality. More importantly, the relation between a waveform and how it is perceived by a human is very non-linear and indirect. To mitigate these issues, most metrics for evaluating waveforms are based on spectral analysis. However, such analysis tends to use windowing that trades off time resolution for frequency resolution. Furthermore, such analysis tends to be lossy to some degree, e.g., phase information is generally discarded, or a mel-frequency scale may be used for better correlation to perception, at the cost of losing resolution at high frequencies. As a result, many ways of objectively evaluating waveforms tend to miss many of the more subtle aspects of sound quality.

3.3.2 Qualitative evaluation

Here we will discuss some of the qualitative, or subjective, evaluation metrics we use in this work. We also mention some evaluation metrics that are not directly used in this work, but are still common. That said, this list is far from exhaustive.

As said, listening tests are currently considered the "gold standard" for evaluating TTS and singing synthesis models. That said, they are not without problems. One issue is that obtaining coherent results across subjects can be difficult, e.g., one subject's "good" can be equivalent to another subject's "above average". Additionally, there are many factors that may affect judgment, such as familiarity with a certain style of singing, background music (if any), timbre of the voice, language, and so on. Finding suitable subjects to participate in listening tests can also be difficult. Some may not be used to listening for subtle differences in sound quality. Or in the case of singing voice, we may have to evaluate things such as the naturalness and quality of singing expression. These issues can be mitigated by careful design of listening tests, e.g., ranking tends to be easier than rating on an absolute scale, and by careful selection of subjects. Both of these things tend to be limited by practical constraints, e.g., the duration of the test, or whether "expert" subjects are available in sufficient quantity. That said, in this work, we try to find a balance between these things. For instance, we generally try to find at least somewhat "expert" subjects (music technology researchers, singing voice researchers, singers, musicians), even if this comes at the cost of having fewer subjects.

When aggregating the results of listening tests, typically by some kind of averaging operation, we may also apply some heuristics that try to filter out invalid ratings or subjects. These tend to include things like spending a very short time on tests, i.e., much shorter than the time needed to listen to all stimuli, or not moving any of the sliders (which can happen when accidentally skipping a test for instance). In some cases, we may even exclude some outlier ratings, but this requires a sufficient amount of ratings and detecting outliers with a sufficiently large margin in order not to bias the results. These techniques are especially important when using online listening tests, where we cannot fully control which subjects participate.

Mean opinion score (MOS) By far the most common qualitative evaluation method for TTS and singing synthesis is a mean opinion score (MOS) listening test (e.g., Ribeiro et al., 2011). Here, several subjects are asked to listen to a series of stimuli and rate each on a (discrete) 1–5 absolute category rating (ACR) scale with respect to the quality of interest, such as sound quality or naturalness. Some drawbacks of this method are that the interpretation of the rating scale may vary from subject to subject, and that the coarse rating scale does not allow expressing more subtle differences the subjects may perceive. The exact implementation of a MOS listening test can vary depending on the exact requirements, although ITU-R P.800 (ITU-R Recommendation P.800, 1996) gives some recommendations. In its most canonical form, the MOS is calculated as the arithmetic mean over single ratings for a given stimulus,

$$\text{MOS} = \frac{\sum_{n=1}^{N} R_n}{N}, \tag{3.11}$$

where R are the individual ratings for a given stimulus by N subjects. In practice, a MOS test typically compares the output of multiple systems, and as such may present these stimuli simultaneously to the subjects. Similarly, there may be a reference audio to be considered in the rating, and typically there will be multiple stimuli for each system to be compared. In these cases, the test is actually more similar to a multiple stimuli with hidden reference and anchor (MUSHRA) test described below.

Multiple stimuli with hidden reference and anchor (MUSHRA) A variant of the MOS listening tests, the MUSHRA test (ITU-R Recommendation BS.1534-3, 2015), asks the listener to rate multiple stimuli derived from a single reference at once (e.g., outputs produced by different models to be compared, where the reference is a recording of the speaker or singer). Additionally, a finer-grained rating scale 0–100 is used, albeit typically still displayed with a 5 segment ACR scale. Among these stimuli a hidden copy of the reference is included in order to get an upper bound score (ideally close to 100). Similarly, one or more anchors, distorted copies of the reference, are included to get a lower bound score (ideally close to 0). While for things like evaluating lossy audio compression, how such anchors can be created is reasonably well defined, this is not the case for TTS or singing synthesis. The main problem is that it is difficult to artificially generate artifacts that realistically simulate the kind of degradations that occur in the voice modeling process. Another drawback of this method is that in TTS research, results are typically presented as a MOS test (i.e., on a 1–5 scale), although, as mentioned, the actual test may be implemented much more closely to a MUSHRA test than a traditional MOS test. In our work, typically, we loosely follow a MUSHRA test, with reference, hidden reference, but typically no anchor. Similarly, we present stimuli corresponding to multiple systems at once, and use a fine-grained 0–100 scale. For the final results, we first linearly map ratings on a 0–100 scale, $\hat{R}$, to ratings on a (continuous) 1–5 scale, R,

$$R = 1 + 4\frac{\hat{R}}{100}. \tag{3.12}$$

Then, we compute the final score for a given system as the arithmetic mean over the ratings of all stimuli corresponding to that system, by all subjects, following Equation (3.11),

$$\mathrm{MOS}_s = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K} R_{s,n,k}}{NK}, \tag{3.13}$$

where s corresponds to the system in question, N the number of subjects and K the number of stimuli per system. The mean opinion score (MOS) is typically presented together with the 95% confidence interval, typically printed as the confidence interval relative to the mean. This confidence interval is calculated as,

$$\mathrm{CI}_s = \mathrm{MOS}_s \pm t_{NK-1}\frac{\sigma(R_s)}{\sqrt{NK}}, \tag{3.14}$$

where t_{NK-1} is the 95-th percentile of a Student's t-distribution with $NK - 1$ degrees of freedom, and variance $\sigma^2(R_s) = \frac{1}{NK-1} \sum_{n=1}^{N} \sum_{k=1}^{K} (R_{s,n,k} - \text{MOS}_s)^2$. One way to interpret this is that if the listening test were to be repeated many times, and the 95% confidence intervals were calculated for each listening test, the proportion of confidence intervals that would encompass the true MOS would tend towards 95%. Another interpretation is that the 95% confidence interval represents values that are not statistically significantly different from the point estimate at the 0.05 level.

Comparison mean opinion score (CMOS) A comparison mean opinion score (CMOS) test is similar to a MOS test, except that stimuli are rated with respect to a reference. In this case, a (discrete) -3–+3 scale is used (much worse, worse, slightly worse, about the same, slightly better better, much better). This approach is most useful when we have a clear single reference, which could potentially be better than the systems to be evaluated, e.g., in ablation studies, or when evaluating an improved version of a synthesis model to its predecessor. However, when comparing multiple systems to each other, other tests are arguably better suited.

AB preference test A preference test, or AB preference test, is a type of paired test that is also frequently used in the evaluation of TTS and singing synthesis. These tests ask subjects to listen to two stimuli ("A" and "B") and select the preferred stimuli, or "no preference". The main advantage of this type of test is that it tends to be a relatively easy and unambiguous task for the subject. While the results of such a test do not give ratings on an absolute scale, they are very easily interpretable. The main other drawback is that when comparing many different systems, the number of pairs may become impractical.

ABX test Another type of paired test, the ABX test, is less widely applicable in TTS, but may also be used in some cases. In this type of test, the subjects are given three stimuli, "A", "B", and "X" which is randomly chosen from "A" or "B". The task is then to select whether "X" is identical to "A" or to "B". This type of test is used to see if subjects can consistently identify detectable differences between the stimuli. This kind of test makes the most sense in cases where compared systems produce very subtle differences, such as lossy audio compression. However, in TTS and singing synthesis, we can typically assume that differences between the output of different systems tend to be detectable, even if how these differences relate to the rating can be ambiguous.

3.3.3 Quantitative evaluation

Here we will discuss the quantitative, or objective, evaluation metrics used in this work. We use different metrics for different aspects of the synthesized sound, such as timbre, pitch and timing.

Dealing with time mis-alignment

All of the metrics discussed here compare the output of a model to a reference audio. In certain cases, such as when using performance-driven synthesis (see §3.1.5) using phonetic timings corresponding to the reference, we assume the model output and target to be approximately aligned in time. In certain other cases, such as using note timings and a phoneme duration model, there may be some mis-alignments in time. In the latter case, when both reference phonetic timings and synthesis phonetic timings are available, we apply a simple linear mapping between (discrete) timestep indexes. In other cases, we can apply dynamic time warping (DTW) as a more generalized solution to time-aligning the signals[3]. In cases where we expect approximate alignment, we can use DTW with certain constraints allowing only small deviations to further improve alignment.

Even when we use performance-driven synthesis, small timing deviations with respect to a reference can still quite commonly occur. For instance, we may control a synthesizer with a timed phonetic sequence extracted from a reference signal, and then compute some metrics using that same reference signal. In this case, the overall timing of the synthesis output and reference should match, but fine details such as the exact time position of the burst of a plosive may deviate one or two frames. Besides applying DTW to try to perfectly align the sequences, we can also use simpler approaches to try to mitigate such small timing differences overly affecting the final metrics. For instance, using simple heuristics (e.g., Iglewicz and Hoaglin, 1993) we can estimate which timesteps correspond to outliers in the overall error distribution, and exclude them when aggregating errors over all timesteps.

Frame-wise vs. sequence-wise metrics

Most metrics discussed here are frame-wise metrics that only consider a single timestep in a sequence. The values can then be aggregated over the whole sequence by simple operations such as averaging. An important drawback of this type of metric is that

[3]In a recent publication, Kang et al. (2021) describe in detail how to apply dynamic time warping (DTW) prior to computing the mel-cepstral distortion (MCD) metric, calling it "Elastic-MCD". However, this approach has been used quite widely since much longer (including in this work). That said, many publications do not mention this very explicitly or in much detail (exact hyperparameters and such).

the issue of oversmoothing in time is not reflected well. Oversmoothing is a common artifact of synthesis models and perceptually very relevant issue, where the predicted parameters are overly averaged in time and frequency, producing static sounds that often sound “buzzy” or “muffled”. While frame-wise metrics do reflect oversmoothing in frequency to some degree, they do not consider the importance of natural frame-to-frame variations. What is worse, such overly averaged predictions can actually produce better averaged scores than perceptually more natural predictions with more variations.

To mitigate these problems, we also employ some sequence-wise metrics. These metrics take the whole sequence into account, and especially focus on variation over time. A common metric of this type is the global variance (GV) (Toda and Tokuda, 2007). Here, the variance over an entire utterance is computed, and the results are typically presented as separate values for predicted and ground truth GV, averaged over all utterances. The values can be averaged over all dimensions of some feature, e.g., the coefficients of the timbre representation, or displayed as a plot for each dimension separately. The downside of this metric is that it is sensitive to outliers, and not very informative (i.e., two signals can be vastly different but have similar variance). The underlying assumption that the signal is Gaussian, does also not hold typically.

A more recent metric is the modulation spectrum (MS) (Takamichi et al., 2016). The MS is equivalent to the magnitude spectrum of a speech parameter time series. Typically, a long window is used, and results may be averaged between windows and utterances. Typically, this metric is presented as a plot comparing predicted and ground truth spectra. In particular, the MS gives insight into the frame-to-frame variations in the signal. Oversmoothed predictions tend to lack high-frequency content, compared to natural speech or singing. In the case of singing, the MS of F0 can even give insight into vibrato frequencies. The main drawback of this method is that the results tend to be quite noisy, and may require some additional heuristics for best results.

Timbre metrics

The metrics we use to evaluate timbre and timbre models depend to some degree on the intermediate acoustic features used (see §3.1.3). For instance, if we have separate harmonic and aperiodicity features, these can be evaluated separately.

Log-spectral distortion (LSD) Log-spectral distortion (LSD) is a commonly used distance in dB between two spectra. While this metric thus essentially is frame-wise, for time series, that is, spectrograms, we simply aggregate frame-wise errors using a simple mean. We can define the (mean) log-spectral distortion (LSD) between two $T \times K$-dimensional spectrograms X and $\hat{X}$ as,

$$D_{\mathrm{LSD}} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\left[20\log_{10}\frac{X(t,k)}{\hat{X}(t,k)}\right]^2}. \tag{3.15}$$

Mel-cepstral distortion (MCD) Mel-cepstral distortion (MCD) is the equivalent of the LSD between two mel-frequency cepstra, rather than two (linear frequency) spectra. The main difference is that it uses a mel frequency scale rather than a linear frequency scale, which is arguably closer to human perception. Note that the mel-frequency cepstrum is a linear transform of the mel-frequency spectrum, thus this metric is equally valid between two mel-frequency spectra. Traditionally, this metric derives low-dimensional mel-cepstra from the full spectra. This reduces the interference of pitch on the spectra compared (i.e., we are comparing something closer to the spectral envelope, rather than all spectral details), which tends to be considered desirable as timbre is closely related to this envelope. However, in our case, we tend to generally compute the mel-cepstral distortion (MCD) on features that are already on a mel scale, e.g., a model which outputs mel-spectrograms, or the features may inherently follow the spectral envelope, e.g., when using harmonic vocoder features. We can define the (mean) MCD between two $T \times K$-dimensional cepstrograms C and $\hat{C}$ as,

$$D_{\mathrm{MCD}} = \frac{10}{\log 10}\sqrt{2}\frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\left[C(t,k) - \hat{C}(t,k)\right]^2}. \tag{3.16}$$

A more robust mel-cepstral distortion (MCD) In this work we use some additional heuristics for computing MCD metrics, with the main to make the results more reliable, especially when used in the context of singing voice. First, we extract mel-cepstral parameters from WORLD spectra (Morise et al., 2016) rather than STFT spectra, in order to better handle high pitches, where the distance between harmonics is more pronounced. Additionally, to reduce the effect of pitch mismatches between reference and prediction, we filter pairs of frames with a pitch difference exceeding ±200 cents. Similarly, to increase robustness to small misalignments in time around rapidly changing phonemes such as plosives, frames with a modified z-score exceeding 3.5 are not considered (Iglewicz and Hoaglin, 1993). In this case, the robust MCD is computed for harmonic components, using 33 (0–13.6 kHz) coefficients.

Band aperiodicity distortion (BAPD) Identical to MCD or LSD, except computed over linearly spaced band aperiodicity coefficients. In the case of WORLD intermediate acoustic features, we compute band aperiodicity distortion (BAPD) over the fixed number of bands the algorithm provides, e.g., 4 aperiodicity coefficients (3–12 kHz) at a sampling rate of 32 kHz.

Modulation spectrum (MS) of harmonic features As discussed above, one issue with frame-wise metrics, like MCD, is that these do not consider the behavior of the predicted parameter sequences over time. In particular, predictions that are oversmoothed in time are not penalized, and sometimes even favored over predictions with natural frame-to-frame variation. As such, this aspect of these metrics does not correlate well to perceptual qualities. In order to compliment the frame-wise metrics, we propose to use the modulation spectrum (MS) (Takamichi et al., 2016), which considers variations of parameters over time. In order to handle variable-length sequences, we use a 1-d STFT with a window of 512 timesteps (each timestep corresponding to 5 ms) and a 16 timestep shift, for each dimension of the harmonic features. We use a Tukey window with shape parameter $\alpha = 0.05$. We then compute the magnitude spectra in dB and average across all dimensions. The resulting (average) modulation spectrum (MS) can be visualized to get an idea of how the predicted parameters behave. For instance, showing oversmoothing as a rolloff of higher modulation frequencies. We are mainly interested in the lower band of the MS (e.g., <25 Hz), because the higher band of the reference (natural singing) can be overly affected by noise in the parameter estimation. We can also compare the MS of the predicted parameters to that of a reference signal. To obtain a single scalar metric, we use the modulation spectrum log-spectral distortion (MS-LSD), the log-spectral distortion (LSD) between modulation spectra (MS) of a predicted parameter sequence and that corresponding to a reference recording.

Pitch metrics

We separate pitch into metrics considering continuous pitch, and metrics considering voiced/unvoiced (V/UV) decision separately. Alternatively, we can also consider V/UV decision to be part of aperiodicity features (if any).

F0 root mean squared error (RMSE) A typical metric to evaluate F0 would be root mean squared error (RMSE). While for TTS this metric is sometimes given in Hz, for singing it is more appropriate to express it as a function of log F0, e.g., using cents. Thus, we can compute this metric as,

$$D_{\mathrm{RMSE},¢} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left[1200\log_2\left(\frac{\hat{f}_n}{f_n}\right)\right]^2}, \tag{3.17}$$

where $\hat{f}$ and f are the N predicted and target F0 values (in Hz) respectively. It should be noted that these metrics are often not very correlated to perceptual metrics in singing (Umbert et al., 2015). For instance, starting a vibrato slightly early or late compared to the reference may be equally valid musically, but can the cause the two F0 contours to become out of phase, resulting in high distances.

F0 correlation The Pearson correlation coefficient between the predicted (log) F0 and target (log) F0 can also give some indication of the performance of an F0 model. This metric is a real number in the range $[-1, 1]$, where higher values indicate a high degree of correlation. This metric can be calculated as,

$$r = \frac{\sum_{n=1}^{N} (¢_n - m_{¢})(\hat{¢}_n - m_{\hat{¢}})}{\sqrt{\sum_{n=1}^{N} (¢_n - m_{¢})^2 \sum_{n=1}^{N} (\hat{¢}_n - m_{\hat{¢}})^2}}, \tag{3.18}$$

where $\hat{¢}$ and $¢$ are the N predicted and target F0 values (in cents), i.e., $¢ = 1200 \log_2(f/440)$. Again, it should be noted that this metric is not perceptually motivated and generally should only be considered in conjunction with other metrics and possibly qualitative evaluation.

Modulation spectrum (MS) of log F0 Similar to timbre, we use MS-based metrics to get a sense of how close the generated F0 contours are in terms of variability over time. The MS of F0 is computed by first segmenting the score into sequences of continuous notes, without rests. Then, for each sequence, the remaining unvoiced regions in the log F0 curve are filled using cubic spline interpolation. We apply a Tukey window corresponding to a 50 frame fade in and fade out, and subtract the per-sequence mean. Then, the modulation spectra are computed using a DFT size 4096, and averaged over all sequences.

Voiced/unvoiced (V/UV) decision metrics In singing voice, there is a notable imbalance between voiced and unvoiced frames due to having many long, sustained vowels. As both false positives (unvoiced frames predicted as voiced) and false negatives (voiced frames predicted as unvoiced) can result in highly noticeable artifacts, we list both false positive rate (FPR) and false negative rate (FNR) for this estimator. All silences are excluded. These metrics can be calculated as,

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.19}$$

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}, \tag{3.20}$$

where TP, FP, TN, and FN are the number of true positives, false positives, true negatives and false negatives respectively.

Timing metrics

In order to get more informative results, we split the timing metrics up into a few components, rather than just "blindly" compute the error between predicted and target phoneme durations. First, we compute note timing errors, split into note onset error and note offset error. Here, we consider note onsets to be much more critical for perceived timing, while note offsets allow for much more variation without affecting timing too much. Note offsets, in general, tend to be more ambiguous and more difficult to pinpoint to a single time instance. Next, we compute error metrics between predicted and target consonant durations. We explicitly exclude vowel durations as these essentially are a function of note timings and consonant durations (i.e., vowels make up the remainder of the duration), and as such, could possibly skew the phoneme duration results due to errors in note timing.

Timing root mean squared error (RMSE) We calculate RMSE for durations, onsets and offsets as,

$$D_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left[\hat{d}_n - d_n\right]^2}, \tag{3.21}$$

where $\hat{d}$ and d are the N predicted and target durations respectively.

Timing mean absolute error (MAE) We calculate mean absolute error (MAE) for durations, onsets and offsets as,

$$D_{\text{MAE}} = \frac{1}{N}\sum_{n=1}^{N}|\hat{d}_n - d_n|, \tag{3.22}$$

where $\hat{d}$ and d are the N predicted and target durations respectively.

Timing correlation The Pearson correlation coefficient between the predicted and target durations (or onsets, offsets) can also give some indication of the performance of the timing model. This metric is a real number in the range $[-1, 1]$, where higher values indicate a high degree of correlation. This metric can be calculated as,

$$r = \frac{\sum_{n=1}^{N}(d_n - m_d)(\hat{d}_n - m_{\hat{d}})}{\sqrt{\sum_{n=1}^{N}(d_n - m_d)^2 \sum_{n=1}^{N}(\hat{d}_n - m_{\hat{d}})^2}}, \tag{3.23}$$

where $\hat{d}$ and d are the N predicted and target durations respectively.

3.3.4 Non-intrusive speech quality assessment

There has been a clear trend in recent work on speech quality assessment to try to leverage powerful data-driven neural networks to obtain evaluation metrics that are highly correlated to human perception, robust to a wide range of inputs, and require no human interaction. Depending on the case, such metrics can blur the line between objective and subjective evaluation, e.g., they may be computed by a deterministic calculation, but involve millions of weights trained on human ratings. In general, the goal here is to have the best of both worlds.

Many of these methods are part of a class of methods called non-intrusive speech quality assessment. In this case, unlike in all the previous metrics, no reference signal is required. This makes these approaches more widely applicable and avoids issues related to time mis-alignment and such.

Here we will briefly discuss some of these methods, although we do not use any in the main body of our work. The main reason for this is that most of these methods are still very new and many were not available when we did our research. Additionally, as far as we are aware all current methods were developed for speech rather than singing, and adaptation typically will require a non-trivial amount of work, in particular in terms of preparing a suitable dataset.

Classifier-based metrics

One group of approaches is based on the idea that if we wish to evaluate the output of our model according to some aspects which can be classified, we may leverage a data-driven classifier to aid in the evaluation. This approach was first developed as the Inception score (IS) (Salimans et al., 2016), a method for evaluating generative models of images. In this case, a pre-trained image classifier (Szegedy et al., 2016) trained on a dataset with 1000 classes is used to try to evaluate two aspects of a generative model. The first aspect is the quality of the image; the idea is that if the generative model produces samples that the classifier can classify as a single class with a high probability, this should mean that the generated image at the very least contains certain attributes that make its content recognizable. Thus, we require the conditional distribution of classes given an image to have low entropy. The second aspect is the diversity of images sampled from the generative model. That is, it is not enough for the model to be able to just output the same image, albeit realistic. Thus, we require the marginal distribution over all produced images (or over the random variable used to sample the images) to have high entropy. The idea behind this approach is quite similar to that of generative adversarial networks (GANs) (Goodfellow et al., 2014), although in this case the classifier (discriminator) is generally trained jointly with the generator and without supervision.

One shortcoming of this method is that the ability to classify an image does not necessarily mean this image is perceptually considered realistic or high quality, although it does imply at least a threshold quality. For instance, as long as some basic properties such as rough shape and colors match, the classifier may still be able to confidently classify a generated image. To mitigate these issues, the Fréchet Inception distance (FID) was proposed (Heusel et al., 2017). This metric adds an additional term that compares statistics (moments) of hidden features of the classifier computed on both real images and generated images. The term is computed using the Fréchet or Wasserstein-2 distance. This has been shown to provide a better correlation to human perception compared to vanilla IS.

The approach of FID for images has been adapted to speech by using a powerful speech recognizer, DeepSpeech (Amodei et al., 2016), as a classifier, and therefore is called the Fréchet DeepSpeech distance (FDSD) (Bińkowski et al., 2020). The same authors also propose several variations of this metric, such as the Kernel DeepSpeech Distance (KDSD), and conditional versions cFDSD, and cKDSD. Note that some papers use slightly different versions of this algorithm (e.g., Gritsenko et al., 2020), sometimes noting that the original FDSD was unreliable (e.g., Donahue et al., 2021, Appendix I). Unfortunately, there is no freely available equivalent of DeepSpeech for singing voice, and developing one is non-trivial. That said, we have not evaluated the FDSD as-is on singing voice.

Another property that can be classified and leveraged for evaluation is speaker identity. For instance, a speaker classifier could be used to ensure a multi-speaker TTS model properly reproduces the target speaker identity (e.g., Arik et al., 2018). This approach is also fairly common in the task of voice conversion. This method could be applied more easily to singing voice, although speaker identity tends to be a less desirable attribute to evaluate compared to quality.

Mean opinion score prediction

Another class of approaches uses a neural network to map acoustic features to human ratings (mean opinion scores). While several works explore this approach for TTS, voice conversion and other speech tasks (e.g., Lo et al., 2019; Serrà et al., 2021), this type of metric is currently not widely used in publications. For singing synthesis, the main hurdle to applying this approach is the lack of large-scale training data of human ratings, especially if we wish to consider aspects such as musical expression.

Traditional speech quality assessment

Finally, there is a group of more traditional speech quality assessment methods that share much of the same goals, namely to have a metric that is easily computable and highly correlated to human rating. Typically, these methods are based on signal processing and heuristics rather than a data-driven neural network, and often also are intrusive, that is, they require a reference signal. The main issue with these metrics is that they are designed by hand with a specific task or kind of signal in mind. For instance, for evaluating speech quality of telephone networks, speech codecs, and such, metrics like PESQ (ITU-T recommendation P.862, 2001) or the later POLQA (ITU-T recommendation P.863, 2011) can be used. To evaluate the intelligibility of speech in the context of noise reduction or speech separation, metrics like STOI (Taal et al., 2010) can be used. Unfortunately, designing and implementing similar metrics for other tasks and/or signals, such as the quality and naturalness of expression in singing voice, is not trivial, and arguably more difficult than for modern neural network approaches, where the principal difficulty is data collection.

Part I

Modeling timbre

Autoregressive modeling of timbre | 4

OUR INITIAL WORK on neural singing synthesis focuses on autoregressive models, in particular, adapting the influential WaveNet proposed by van den Oord et al. (2016a) from speech to singing voice. Additionally, we adapt the model from predicting waveform directly, to predicting intermediate acoustic features (in particular parametric vocoder features), which has a number of practical advantages (see §4.2).

While these days deep learning is by far the dominant approach in speech and singing synthesis, when we first published our work (Blaauw and Bonada, 2017a) the landscape of singing synthesis research was radically different. Concatenative synthesis (e.g., Bonada et al., 2016) was state of the art in terms of sound quality, especially for languages where single diphone inventory synthesis was effective, such as Japanese. It was also the most commercially successful (e.g., Kenmochi and Ohshita, 2007). However, it was plagued by difficult recording sessions due to having many constraints rather than allowing natural singing, a lot of manual labeling, segmentation and selection of data, an inability to simultaneously model timbre and pitch (and other expression, and, in general, a lack of flexibility. The obtained naturalness and ineligibility were also notably poorer for languages such as English which lend themselves less to the single inventory diphone synthesis paradigm. Hidden Markov model (HMM) synthesis (e.g., Oura et al., 2010) was also a highly researched topic, which offered much greater flexibility and lower effort voice creation. However, despite many advances, the resulting sound quality never quite reached that of concatenative synthesis. Singing synthesis based on deep learning was just starting to be explored (e.g., Nishimura et al., 2016). This approach offered similar benefits to the HMM approach, arguably with a much-simplified training pipeline and requiring fewer heuristic decisions. However, at this point very simple models were being used and the results were comparable to those of HMM-based synthesis. See §2.3 and §2.4.1 for more details on this historical background.

In this context, our initial goal was to show that a powerful deep learning model and architecture could match or outperform concatenative synthesis in terms of quality, while offering a very high degree of flexibility. To this end, in order to allow comparing systems, we first trained an autoregressive model on pseudo singing (see §3.2.2), which is a requirement for many concatenative synthesis systems. Additionally, we only predict timbre and use performance-driven synthesis (see §3.1.5), to allow better comparison of different models. This initial work is described in §4.2 and was originally published as (Blaauw and Bonada, 2017a).

While not the main focus of this work, in a second step, we extended our initial work to form a complete synthesizer. This is done by adding components that predict pitch and timing. Additionally, the models are now trained on natural singing, which is an important step towards modeling truly natural singing. This second work is described in §4.3 and was originally published as Blaauw and Bonada (2017b).

4.1 Proposed system

4.1.1 Autoregressive models

The core idea behind autoregressive models for audio is to factorize the joint probability over the to be modeled time series, $\mathbf{x} = [x_1, x_2, \dots, x_T]$, in Equations (3.1) and (3.2) as a product of conditional probabilities,

$$p_\theta(\mathbf{x} \mid \mathbf{c}) = p_\theta(x_1, x_2, \dots, x_T \mid \mathbf{c}) := \prod_{t=1}^{T} p_\theta(x_t \mid \mathbf{x}_{<t}, \mathbf{c}). \tag{4.1}$$

Here, $\mathbf{x}_{<t} = [x_1, x_2, \dots, x_{t-1}]$ are all past timesteps before timestep t, following the natural causal ordering of time. Depending on the architecture used in these models, $\mathbf{x}_{<t}$ is often a window of past timesteps in practice, e.g., the receptive field of a stack of causal convolutions. The reason this model is so effective is that while modeling joint probabilities over many timesteps is difficult, modeling a conditional probability over a single timestep given past timesteps is relatively easy. Note that, for simplicity's sake, we will assume the control signal, $\mathbf{c}$, on which the model is conditioned, is time-aligned to the acoustic features, $\mathbf{x}$, thus also consisting of T timesteps.

Once the model is trained using Equation (4.1), we can synthesize new sequences given (unseen) control inputs $\mathbf{c}$ by sequentially sampling the conditional model,

$$\hat{x}_t \sim p_\theta(x_t \mid \hat{\mathbf{x}}_{<t}, \mathbf{c}) \quad \text{for } t = 1, 2, \dots, T, \tag{4.2}$$

where $\hat{\mathbf{x}}_{<t}$ is the sequence of all previously sampled timesteps.

4.1.2 Network architecture

Besides simplifying the problem by the autoregressive factorization of Equation (4.1), the network architecture used to model $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{<t}, \mathbf{c})$ also plays an important role in

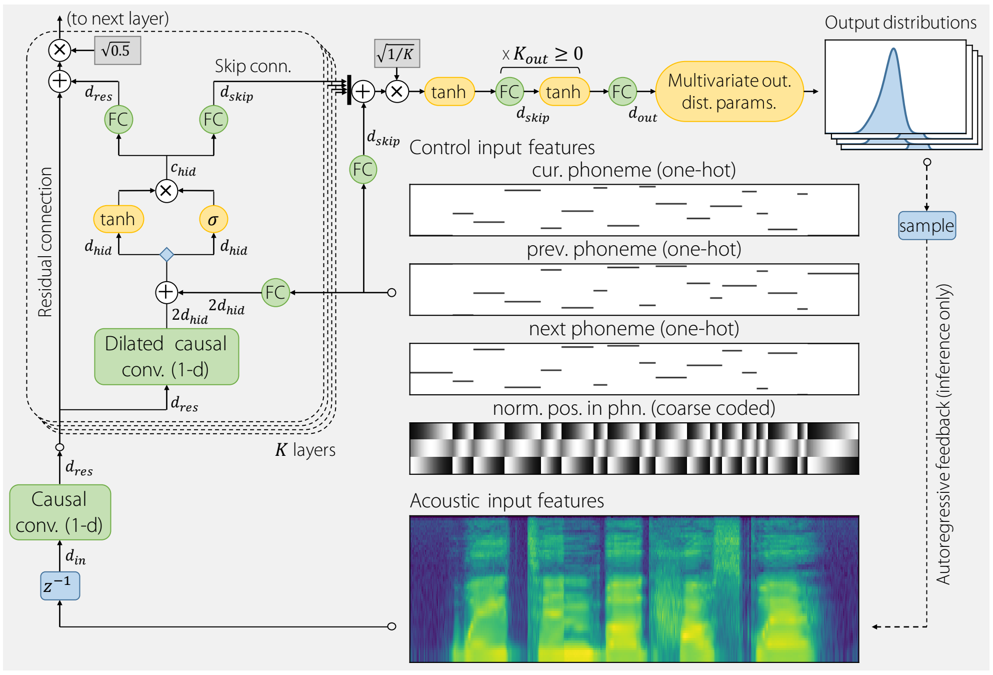

Figure 4.1: Overview of our autoregressive network architecture based on WaveNet. In this case, the network depicted predicts harmonic spectral envelope features (bottom plot), given control inputs (top plots, excluding F0 features for simplicity). Note that the "FC" modules are fully connected layers, implemented as 1×1 convolutions. The "z^{-1}" block is a single timestep delay. The network is depicted with a multi-step output stack, depending on K_{out}, however, in our experiments this is set to zero. The labels d_{in}, d_{res}, d_{hid}, d_{skip}, and d_{out} are hyperparameters that define the dimensionality of the signal at different parts of the networks.

the success of these models. The original WaveNet paper (van den Oord et al., 2016a) proposed a carefully designed architecture derived from previous work on image modeling (van den Oord et al., 2016b; van den Oord et al., 2016c). This model has now proven to stand the test of time (at least in terms of the rapidly changing field of deep learning research) and is still one of the most commonly used network architectures for speech processing (e.g., Ping et al., 2018; Yamamoto et al., 2020).

Our autoregressive network architecture based on WaveNet is depicted in Figure 4.1. One key aspect of this architecture is that it is fully convolutional rather than recurrent like many of its predecessors. This has the advantage of being parallelizable, thus speeding up training, and allowing more direct paths between distant past timesteps and the currently predicted timestep. To overcome the limited receptive field of convolutional neural networks (CNNs) compared to recurrent neural networks (RNNs), dilated convolutions are used (Yu and Koltun, 2016). This could be thought of as a kind of downsampling of the convolution's input. The dilation factor is doubled for each layer, allowing exponentially growing the model's receptive field, while linearly increasing the number of required parameters. After a number of layers (often called a *cycle*), the dilation factor is reset to one to increase the total non-linearity of the model without excessively growing its receptive field. The dilated convolutions generally use a small kernel size such as 2×1 and are ensured to be causal (by adjusting the padding parameters). To ensure an expressive power comparable to gated recurrent units such as long short-term memory (LSTM) or gated recurrent unit (GRU), a simple gating mechanism is used.

In order to facilitate training deeper networks (e.g., 30 layers), residual connections (parametrized by a 1×1 convolution) are used (He et al., 2016). Skip connections provide a direct connection between each layer's output and the network's final output, allowing to more efficiently integrate information at different time scales. Here skip connections are also parametrized by a 1×1 convolution and summed (equivalent to concatenating outputs followed by a single 1×1 convolution). The final output stack typically consists of a few 1×1 convolutions, all but the last with some activation, such as rectified linear unit (ReLU) or $\tanh(\cdot)$. The final output distribution can vary, as discussed in "Output distributions" of §4.1.3.

To condition the model on some external control inputs (in our case, e.g., notes and lyrics). Control inputs are projected for each layer individually by a 1×1 convolution and added to the output of the layer's dilated convolution, prior to the gated non-linearity. That is, by default, only control inputs corresponding to the to be predicted timestep affect the prediction directly, while control inputs corresponding to previous timesteps affect the prediction via lower hidden layers, and control inputs corresponding to future timesteps do not affect the prediction at all. In the model used in experiments of this section, we do the same thing at the output stack, similar to Reed et al. (2016), as we informally found this to provide some minor improvements.

Alternative architectures

While in this work we do not consider alternative network architectures in depth, or compare their results, many other network architectures could be used. In fact, the only real requirement of the network architecture used in autoregressive models is causality. Here we will only give a brief overview of some of the common differences, without exhaustively discussing all of the different network architectures that have been proposed for autoregressive text-to-speech (TTS) models. One common difference is that other architectures may be structured differently, in particular when they are so-called sequence-to-sequence (Seq2Seq) architectures similar to that discussed in Chapter 8. These models generally follow an encoder-decoder architecture, often with multiple encoders (i.e., one for encoding the autoregressive audio path, one for encoding control inputs), an attention mechanism, decoder, and occasionally a post-processing network (typically non-causal). Another common difference is that some architectures are recurrent rather than convolutional (e.g., Wang et al., 2017; Shen et al., 2018). In most cases, this will be the result of using an attention mechanism that is inherently recurrent. Highway networks (Srivastava et al., 2015) are also common building blocks. These are basically convolutions (e.g., dilated, causal) with an alternative gating mechanism and way of allowing deep networks (e.g., Tachibana et al., 2018; Wang et al., 2017).

4.1.3 Modified model and architecture

Modeling multivariate time series

While now a mainstay of TTS models, using powerful autoregressive models for predicting intermediate acoustic features rather than waveform directly was still a novel idea when we first published our model. The main difference between waveform data and intermediate acoustic features is that the first is a univariate time series, while the latter is a multivariate time series. One way to define an autoregressive model for 2-d time-frequency data would be to (arbitrarily) define some causal ordering of variables within a timestep, e.g., from low to high frequencies, and model the probability conditioned on all previous timesteps and all previous frequency bins. This approach is commonly taken for image modeling (van den Oord et al., 2016b; van den Oord et al., 2016c). The reason we opted not to take this approach in our work is twofold; it can make inference several orders of magnitude slower than the model we describe below (see §4.1.4), and because we consider the translation invariance that 2-d convolutions provide is actually an undesirable property for the frequency dimension of time-frequency data, unlike with images. Additionally, this approach requires a much more complicated network architecture based on masked 2-d convolutions.

Instead, we choose to model all variables within a timestep at once, assuming they are conditionally independent. In other words, we define the probability of any frequency within a timestep to only depend on all frequencies in previous timesteps, but not on other frequencies within that timestep. Thus, assuming a K-dimensional feature vector per timestep,

$$p_\theta(\mathbf{x} \mid \mathbf{c}) := \prod_{t=1}^{T} \prod_{k=1}^{K} p_\theta(x_{t,k} \mid \mathbf{x}_{<t}, \mathbf{c}). \tag{4.3}$$

Note that here $\mathbf{x}_{<t}$ is a matrix rather than a vector. In practice, we tend to implement this equation using a neural network that predicts K values or sets of parameters for

the output distribution at once. Thus this allows effective sequential sampling of one timestep at a time during inference,

$$\hat{\mathbf{x}}_t \sim p_\theta(\mathbf{x}_t \mid \hat{\mathbf{x}}_{<t}, \mathbf{c}) \quad \text{for } t = 1, 2, \dots, T, \tag{4.4}$$

where $\mathbf{x}_t$ is a K-dimensional vector rather than a scalar.

An alternative view of this approach is modeling each timestep as a K-dimensional isotropic distribution (i.e., with diagonal covariance). In this case, we can use an alternative notation for Equation (4.3) that is almost identical to Equation (4.1),

$$p_\theta(\mathbf{x} \mid \mathbf{c}) := \prod_{t=1}^{T} p_\theta(\mathbf{x}_t \mid \mathbf{x}_{<t}, \mathbf{c}), \tag{4.5}$$

where $\mathbf{x}_t$ is a K-dimensional vector rather than a scalar (similar to Equation (4.4)).

We argue that in general, this approximation is reasonable, as 2-dimensional intermediate acoustic features tend to be somewhat slowly varying over time. Thus, while frequency bins within a timestep can be highly correlated, most of this correlation can probably be inferred from considering all frequencies of a few of the preceding timesteps (at e.g., 5 msintervals. Note that this is very different from older approaches such as HMMs where each frequency dimension is effectively modeled independently.

Some autoregressive models, in particular for image modeling, such as Salimans et al. (2017), try to relax this independence assumption by predicting all three RGB channels of a pixel at once, but additionally predicting linear dependencies between channel means. We do not take this approach as the number of frequency channels in the intermediate acoustic features may equal or exceed one hundred, thus requiring a very large amount of additional outputs.

In terms of network architecture, this approach requires almost no changes compared to the original WaveNet architecture. We can still use 1-d convolutions throughout the architecture, only having to adjust the number of input and output channels to match the number of frequency dimensions. In other words, we just have to consider the time-frequency data as multi-channel 1-d data, rather than single-channel 2-d data.

Output distributions

The original WaveNet model uses a categorical (softmax) output distribution over discretized (8 bit, μ-law) waveform data. Advantages of this approach include allowing multi-modal distributions, finite support (truncated) distributions (i.e., when values are clipped), and generally arbitrary distributions due to being non-parametric. Sometimes framing the optimization as a classification rather than a regression is said to be

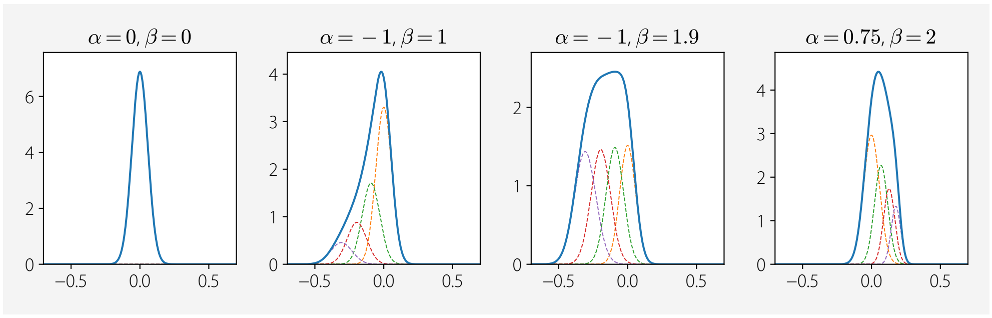

Figure 4.2: Example distributions of the constrained Gaussian mixture (CGM). All subplots use location $\xi = 0$ and scale $\omega = 6 \times 10^{-2}$, but varying skewness α and shape β. The plots show the resulting mixture distributions (solid) and the four underlying Gaussian components (dashed).

beneficial, but this is difficult to prove as both approaches cannot be compared directly. On the other hand, categorical outputs lack any kind of order or distance, which is likely to be detrimental when training data is sparse. When modeling multivariate time series, this approach quickly becomes impractical as the number of feature channels K (or quantization resolution) grows, due to requiring excessive output channels.

One alternative approach is to use a mixture of discretized distributions, such as an mixture of logistic distributions (MoL) (Salimans et al., 2017) or (less commonly) a mixture of Gaussians (MoG). Such distributions can be multi-modal and have finite support, like categorical distributions, while only requiring a small number of parameters per mixture component (e.g., three for MoL and MoG). Additionally, these kinds of output distributions allow discretization with an arbitrary quantization resolution, thus reducing issues with quantization noise that the original WaveNet could exhibit. However, compared to simpler distributions, optimization of mixture distributions tends to be more problematic; requiring more training data, and possibly leading to pathological distributions in some cases. Note that a MoL output distribution is particularly common in neural vocoders (e.g., Shen et al., 2018) and models that predict waveform directly (e.g., van den Oord et al., 2018).

In the model proposed and evaluated in this section, we use what we call a constrained Gaussian mixture (CGM) distribution, which is a special subset of a mixture of Gaussians (MoG) distribution. This distribution is a mixture of four continuous Gaussian components, constrained in such a way that there are only four free parameters (location, scale, skewness and a shape parameter). Figure 4.2 shows some of the typical distributions that the constraints imposed by this parameter mapping allow. We found such constraints to be useful to avoid certain pathological distributions, and in our case explicitly not allowing multi-modal distributions was helpful to improve results. We also found this approach speeds up convergence compared to using categorical output. This decision was also partially motivated by observing predicted output distributions to be generally close to Gaussian or skewed Gaussian in a very early 2-d version of our model with softmax output (Blaauw and Bonada, 2016). See Appendix A.1 for technical details on our CGM distribution.

One advantage of the CGM output distribution is that it has a *temperature* control, much in the same vein as the temperature softmax used in similar models (e.g., Reed et al., 2016). This temperature control allows reducing the variance when sampling the distribution during sampling, thus resulting in less noisy output, which can improve quality in some cases. In the case of modeling intermediate acoustic features, we can further adjust this temperature control independently for each frequency bin, e.g., making the prediction of lower frequencies more deterministic, while maintaining some stochasticity in higher frequencies.

Finally, using simple (mean) L_1 or (mean squared) L_2 losses (equivalent to fixed-scale Gaussian or Laplace distributions respectively) also tend to be a good option in our experience. These distributions require predicting only a single parameter (mean), while at most resulting small reduction in quality compared when a heuristic vocoder is used. Whenever a more powerful neural vocoder is used, this reduction generally becomes negligible (e.g., Shen et al., 2018) (see also §3.1.3).

Note that in some of our experiments we modify the activation functions of the network's output stack in accordance with the output distribution used. For instance, the original WaveNet uses ReLU activations in its output stack, which is a common choice for networks with softmax output. As depicted in Figure 4.1, the model we used in our experiments uses tanh activations in its output stack, which arguably is more coherent with the rest of the network (which uses gated tanh activations), and is possibly better suited for a continuous output distribution such as the CGM distribution. That said, in later experiments, we have also used ReLU activations in the output stack, and informally noted no real perceptual impact on the resulting synthesis quality.

Regularization to mitigate exposure bias

If we look closely, we can see an inconsistency between the model's training objective, Equation (3.2) using Equation (4.1) or Equation (4.3), and inference by sequential sampling, Equation (4.2) or Equation (4.4). Namely, the first uses *ground truth* past timesteps, $\mathbf{x}_{<t}$, while the latter uses past timesteps sampled from the model itself, $\hat{\mathbf{x}}_{<t}$. Even with a well-trained model, there will be some differences between the two.

The reason we do not use sequential sampling during training is twofold; first, training would become prohibitively slow, and second, the model may have trouble converging properly. Especially at the start of training, the differences between $\hat{\mathbf{x}}_{<t}$ and $\mathbf{x}_{<t}$ will be very big, possibly causing the model to ignore past timesteps altogether.

The principal issue with training on ground truth past timesteps, is that contain a lot of useful information for predicting the next timestep, thus it is likely that a powerful deep learning model can become overfitted to these ground truth past timesteps. This problem is usually referred to as *exposure bias* (Ranzato et al., 2016), as in the model becoming *biased* to the ground truth data it is exposed to during training. Viewed differently, the model becomes less robust to small prediction errors when these predictions are used as past timesteps during inference. As this sampling is sequential, such errors tend to accumulate over time. In the case of singing synthesis, these issues are most noticeable during long sustained vowels, where control inputs will be relatively constant. Here, typically unnatural resonances, similar to formants, will start to form and gradually get worse.

Mitigating exposure bias can be done by regularizing the training process. In our model we do this by adding Gaussian noise to the input of the model,

$$p_\theta(\mathbf{x} \mid \mathbf{c}) := \prod_{t=1}^{T} \prod_{k=1}^{K} p_\theta(x_{t,k} \mid \mathbf{x}_{<t} + \lambda_{\text{reg}}\epsilon, \mathbf{c}) \qquad \epsilon \sim \mathcal{N}(0, I), \tag{4.6}$$

where $\lambda_{\text{reg}} \geq 0$ is the input noise level.

This additional hyperparameter, λ_{reg}, can be optimized on a validation set, like any other hyperparameter. As a rule of thumb, we found that values of 0.2–0.4 give optimal results when intermediate acoustic features $\mathbf{x}$ are normalized in a range $[-1, 1]$. However, the exact value of λ_{reg} within this range does not significantly affect the synthesis quality in our experiments. Very small values will result in a model which is effectively unregularized, with all issues associated with exposure bias. Very large values will result in a model which effectively ignores past timesteps, thus the model will revert to a non-autoregressive model, similar to a frame-wise mapping of control features to intermediate acoustic features.

Note that the original WaveNet does not include such regularization. We speculate that this is due to the fundamental differences between waveform data and intermediate acoustic features. Compared to waveform data, intermediate acoustic features tend to vary slowly and smoothly over time, be less stochastic, and in general exhibit a higher correlation between nearby timesteps. The complexity of waveform data may mitigate overfitting to past timesteps, even in very powerful models such as WaveNet.

A common alternative to adding input noise is using dropout (Srivastava et al., 2014) on the input of the network (e.g., Salimans et al., 2017; Shen et al., 2018; Ping et al., 2018). This can be seen as multiplicative Bernoulli noise rather than additive Gaussian noise.

Typically, there will be dropout at the network input (i.e., the autoregressive feedback path) with a relatively low keep probability such as 0.5. Often there will also be dropout at the input of each convolutional layer, with a higher keep probability such as 0.9, but we found that this does contribute significantly to the regularizing effect. We choose to use additive input noise rather than dropout, because in our experiments dropout results in a slightly noisier output compared to additive input noise.

4.1.4 Training and inference speed

As the model objective uses ground truth past timesteps, sometimes referred to as *teacher forcing*, and the network architecture is fully convolutional, training is highly parallelizable. Thus, training is very fast on modern graphics processing unit (GPU) hardware. Typical training times are in the order of hours rather than days.

One of the main drawbacks of the original WaveNet model was the very slow inference speeds. In fact, most subsequent research focuses more on improving inference speed while maintaining sound quality, rather than improving the sound quality itself. As our approach predicts intermediate acoustic features rather than a waveform, most of this problem is mitigated by the much lower frame rate of intermediate acoustic features compared to the audio sample rate (e.g., 200 Hz rather than 32 kHz). Additionally, often modeling acoustic features requires smaller networks than modeling waveform directly (see §4.2.1). In fact, on GPU hardware, a naive implementation of Equation (4.4), tends to be fast enough for developing models. The sequential nature of autoregressive inference actually makes it relatively well suited for deployment on central processing units (CPUs). By caching calculations between timesteps, we were able to implement a fast generation algorithm. While this algorithm was developed independently, it is essentially identical to those proposed in other works (Ramachandran et al., 2017; Arik et al., 2017a). Using this algorithm, our model can achieve generation speeds of 10–15 $\times$ real-time on CPU. Combined with low memory and disk footprints, these relatively fast generation speeds make the system competitive with most existing systems in terms of deployability.

4.2 Modeling timbre from pseudo singing

As mentioned above, the main goal of this initial work was to show that a powerful deep learning model and architecture could match or outperform the then state of the art in terms of sound quality, concatenative synthesis, and in terms of flexibility, HMM synthesis. To allow this comparison, we train an autoregressive model on pseudo singing (see §3.2.2), which is a requirement for many concatenative synthesis systems.

Additionally, we only predict timbre and use performance-driven synthesis, meaning pitch and phonetic timings are assumed to be given as inputs to the system (see §3.1.5). The reason for this is to allow easier and better comparison of different models, which otherwise are likely to use different ways of predicting pitch and timing from a score, as this information cannot be obtained from pseudo singing and thus must come from some external source.

The main contribution of this initial work is the adaptation of the WaveNet architecture for singing voice. Unlike WaveNet, which model waveform, we opted for a more traditional approach of modeling vocoder features. The main reason for this is that it decouples the influence of pitch and timbre, thus allowing to easily and precisely synthesize any melody with any given lyric, even if not seen in the training data. Additionally, this approach makes training the system notably easier, especially when the available datasets are more modest in size compared to those used in speech synthesis.

While using a vocoder introduces some artifacts, we argue that the dominant factor of degradation in many current systems is the generative model rather than the vocoder itself, in particular due to excessive smoothing. A more powerful model, such as an autoregressive model based on WaveNet, should be able to close the gap between current results and the upper bound provided by the vocoder, i.e., round-trip vocoder analysis-synthesis without modification. Conveniently, in singing, many of the artifacts a vocoder introduces are often partially masked by background music, mixing and effects.

4.2.1 Timbre model

As this initial synthesizer only predicts timbre in a performance-driven manner, the synthesizer consists of only a single primary component, the *timbre model*. The only other notable component is the vocoder, which does not contain any trainable weights. An overview of this synthesizer is depicted in Figure 4.3.

Intermediate acoustic features

We use an acoustic frontend based on the WORLD vocoder (Morise et al., 2016) (D4C edition Morise, 2016) with a 32 kHz sample rate and 5 ms hop time. The dimensionality of the harmonic component is reduced to 60 log mel-frequency spectral coefficients (MFSCs) by truncated frequency warping in the cepstral domain (Tokuda et al., 1994) with an all-pass filter with warping coefficient $\alpha = 0.45$. The dimensionality of the aperiodic component is reduced to four coefficients by exploiting WORLD's inherently band-wise aperiodic analysis. All acoustic features are min/max normalized before

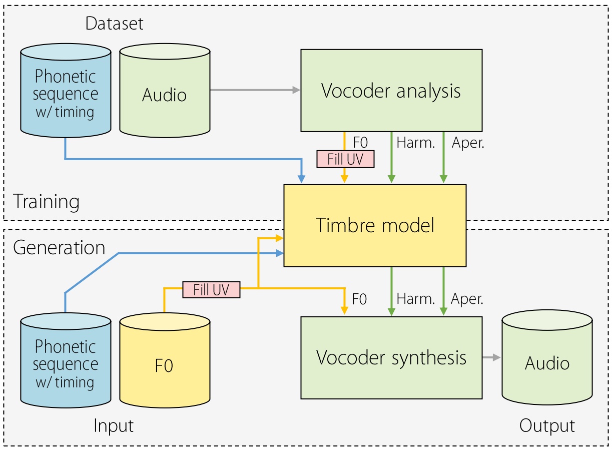

Figure 4.3: Diagram depicting an overview of the timbre-only synthesizer with its different components. Here, the "Fill UV" block fills unvoiced segments by interpolation.

feeding them to the neural network. See "Parametric vocoder features" in §3.1.3 for more details on this procedure.

Control input features

The control inputs of this model are the (timed) phonetic sequence and F0. As mentioned in §4.1.2, conditioning is mostly local to the currently predicted timestep, and notably, there is no conditioning on future information. Thus, we condition our model on triphone information, so at each timestep, we at least know the corresponding previous, current and next phoneme. We experimented with longer phonetic contexts, such as pentaphones, but did not find any notable advantages in doing so. Individual phoneme identities are encoded using simple one-hot encoding, and stacked to form triphones. This vector is repeated along the duration of each phoneme (at the same frame rate as the output acoustic features). Additionally, the position of each timestep within the corresponding phoneme is concatenated to triphone control features. We use a position $[0, 1]$ along the phoneme, which is then encoded as a 3-dimensional vector per timestep using a Gaussian coarse coding (see §3.1.4). Finally, as pitch has a very notable influence on timbre, we condition the model on F0. We encode log F0 as a 4-dimensional feature by triangular responses along the singer's pitch range (see also §3.1.4).

We consider the above conditioning signal the minimal set of features required to obtain a reasonable-quality synthesis. However, many other features could be added. In particular, we experimented with adding phoneme duration information and the absolute position of the timestep (rather than the relative position already included above). As we did not find these changes to have a significant influence on the output result, we decided to opt for the more canonical set of conditioning features described above.

We also do not condition our model on any kind of loudness or dynamics features, which typically could be features we can derive from the acoustic signal, similar to F0 (e.g., Bous and Roebel, 2019). In this case, the principal reason for this is that in this initial work we use pseudo singing datasets, which were designed to have a single constant loudness or dynamics value.

Hyperparameter differences compared to WaveNet

One notable change with respect to the original WaveNet architecture is that the receptive field in terms of timesteps can be much smaller. The reason for this is that each timestep of intermediate acoustic features corresponds to a much larger amount of time, due to the time each spectral analysis window spans, as well as the interval between timesteps. For instance, to determine a signal's pitch at a certain point in time, generally a single frame of intermediate acoustic features will suffice. However, in the case of modeling waveform directly, at the very least one period consisting of many timesteps is required. Assuming[1] a WaveNet architecture with a 1×1 initial convolution, 30 layers, 10 layer dilation cycle and 2×1 causal convolutions, the receptive field of the network would be 3070 timesteps, or approximately 192 ms at a 16 kHz sample rate. In our case, we found that a 10×1 initial convolution, 5 layers, 3 layer dilation cycle and 2×1 causal convolutions are sufficient to model intermediate acoustic features at a rate of 200 Hz. This would thus correspond to a receptive field of 20 timesteps, or 100 ms. Similarly, we assume that predicting intermediate acoustic features is generally easier than predicting waveform, even though the dimensionality per timestep is greater. Thus, we somewhat reduce the number of channels our model uses compared to the original WaveNet. See the "Timbre model" column of Table 4.2 of "Model hyperparameters" in §4.3.4 for more details on the hyperparameters used in our experiments.

Multi-stream architecture

Most parametric vocoders separate the speech signal into several components. In our case, we use three feature streams; F0 (typically including voiced/unvoiced decision), a harmonic spectral envelope, an aperiodicity envelope (which also includes information

[1]While the original WaveNet publication does not explicitly list the hyperparameters used in the experiments, we can make some assumptions based on subsequent publications and presentations. For instance, Heiga Zen, "Generative Model-Based Text-to-Speech Synthesis", Feb. 3rd, 2017, MIT Center for Brains, Minds and Machines (https://www.youtube.com/watch?v=nsrSrYtKkT8).

regarding voicing). Assuming that predicting F0 is not part of the timbre model's tasks, this still leaves the other two streams to be predicted. These components are largely independent, but their coherence is important, e.g., synthesizing a harmonic component corresponding to a voiced frame as unvoiced (highly aperiodic) will generally cause artifacts, and vice versa.

As said, there is some overlap between F0 with voiced/unvoiced decision and aperiodicity, where high values also indicate an unvoiced signal. To avoid any ambiguity due to a potential mismatch between feature streams, we use a continuous F0 stream. Generally, we fill unvoiced regions by simple linear interpolation, which also facilitates modeling this kind of data, compared to combining continuous and discrete properties in a single stream. To estimate voicing, we simply check whether the mean band aperiodicity exceeds some threshold. This threshold can be tuned as a hyperparameter, as it may depend on the voice to be modeled and the output distribution used, but typically will be somewhere in the range 0.75–0.95.

Rather than jointly modeling all data streams with a single model, in our experiments we model each component using an independent network. This approach gives us more fine-grained control over each stream's architecture and corresponding hyperparameters. This approach also avoids the possibility of streams with lower perceptual importance interfering with streams of higher perceptual importance. For instance, the harmonic component is by far the most important, therefore we would not want any other jointly modeled stream potentially reducing model capacity dedicated to this component.

To encourage predictions to be coherent, we concatenate the predictions of one network to the input of another, as depicted in Figure 4.4. In this case, we decided to let the aperiodic component depend on the harmonic component. The dependence is achieved by simply concatenating the output of the first network to the input of the second model, bypassing the unit delay applied to the autoregressive input. Both the networks are similar, but have slightly different hyperparameters (see Table 4.2 of "Model hyperparameters" in §4.3.4 for details). We found it beneficial for the aperiodicity stream to use an output distribution that allows limited support, such as MoL, as some of the band aperiodicities tend to have a lot of timesteps either zero or one, with few timesteps with in-between values.

While not used in the experiments of this section, the most obvious alternative approach is to simply concatenate all feature streams and model them with a single model. In this case, the output can optionally be split in order to allow applying different output distributions, if needed. While in our initial experiments, we found the cascaded model approach more suitable, later experiments with this single model approach resulted in only a marginal difference in output sound quality. Thus in many cases, the simplicity

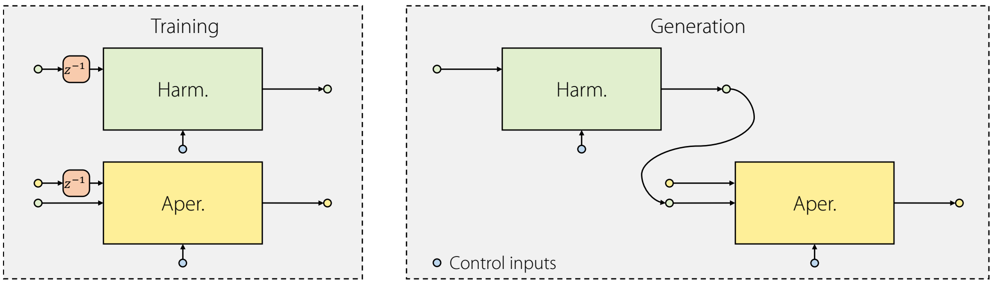

Figure 4.4: Diagram depicting the cascaded multi-stream architecture for training and generation phases. The "z^{-1}" blocks represent unit delays. The upward inputs represent control inputs, which in our case are identical for all streams. Autoregressive (feedback) connections in the generation phase are not shown.

in terms of code and training steps that the single model approach offers may outweigh any marginal quality benefits the cascaded model approach may offer.

Handling long notes

In most datasets, not all note durations will be exhaustively covered. In particular, the case of synthesizing notes significantly longer than the notes in the dataset can be problematic. This issue manifests itself mainly as a repetition in time of some of the transitions predicted by the timbre model, causing a kind of stutter. To reduce such artifacts, we compute the control feature corresponding to the frame position within the phoneme (see §3.1.4) with a non-linear mapping depending on the length of the phoneme. The idea behind this is that the edges of a phoneme, where the transitions are likely to be, will maintain their original rate, while the more stable center parts will be expanded more.

When the phoneme duration (in seconds), d, exceeds some threshold duration, $\bar{d}$, it is considered a long note. In this case, we apply a non-linear mapping to the normalized position-in-phoneme, $p \in [0, 1]$, to result in a mapped normalized position-in-phoneme, $p_{\text{mapped}} \in [0, 1]$. This mapping is shown in Figure 4.5. The idea is that the edges of the mapping are kept approximately linear, while the center portion is "expanded" more. We obtain this mapping by a piecewise cubic Hermite interpolating polynomial through four points, defined as: $(0, 0)$, $(\eta_l \bar{d}/d, \eta_l)$, $(1 - (1 - \eta_r)\bar{d}/d, \eta_r)$, $(1, 1)$. Here, $0 < \eta_l < \eta_r < 1$ are two additional free parameters that control how much of the edges are kept approximately linear. In our experiments we use $\bar{d} = 1.0$, $\eta_l = 0.25$, and $\eta_r = 0.75$.

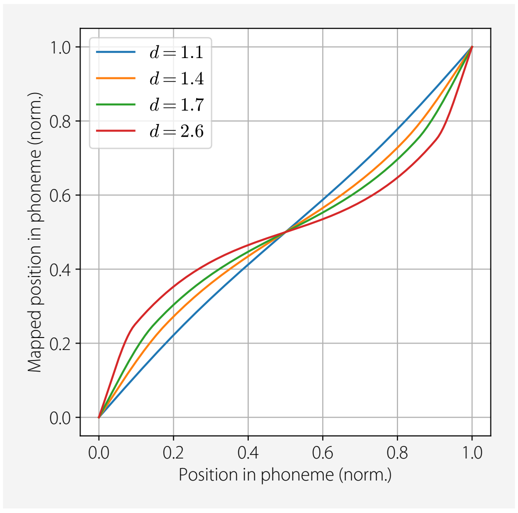

Figure 4.5: Examples of non-linear position-in-phoneme mapping for long notes. Here shown for $\overline{d} = 1$, $\eta_l = 0.25$, $\eta_r = 0.75$, and various values for $d > \overline{d}$.

4.2.2 Experiments

Datasets

We use three proprietary datasets from training systems on pseudo singing; an English male voice (EN-ZM-P), an English female voice (EN-ZF-P), and a Spanish female voice (ES-VF-P). The studio quality recordings consist of short sentences which were sung at a single pitch and an approximately constant cadence. The sentences were selected to favor high diphone coverage. The Spanish dataset contains 123 sentences, while the English datasets contain 524 sentences (approximately 16 and 35 min respectively, including silences). A randomly selected 10% of sentences are used for testing.

Note that datasets EN-ZM-P and EN-ZF-P were segmented completely automatically, without any manual correction other than correcting a few gross discrepancies between the written lyrics and what was sung. On the other hand, dataset ES-VF-P was first segmented automatically and then carefully corrected by hand, while also correcting any transcription errors that may exist.

Note that these datasets are small compared to the datasets typically used to train TTS systems. However, we argue that for pseudo singing, as only timbre is captured in a very constrained setting, substantially larger datasets would likely yield diminishing returns.

Compared systems

In our experiments, we compare our proposed system to two other systems which represent the two main paradigms for singing synthesis at the time of initial publication. One system is state of the art concatenative unit selection-based system. The other system is representative of the approach using HMMs.

NPSS Our system implementing the autoregressive timbre model described above. We call this model the neural parametric singing synthesizer (NPSS). Model hyperparameters are listed in the "Timbre model" column of Table 4.2 in "Model hyperparameters" of §4.3.4.

IS16 A concatenative unit selection-based system (Bonada et al., 2016), which was the highest-rated system in the Interspeech 2016 Singing Synthesis Challenge. This is an earlier work by our research group. While described in detail in the paper, currently no implementation of this model is publicly available.

HTS An HMM-based system, similar to the system described in Oura et al. (2010), but consisting of a timbre model only, and trained on pseudo singing. The standard demo recipe from the HTS toolkit (version 2.3) (Zen et al., 2007) was followed, except for a somewhat simplified context dependency (just the two previous and two following phonemes).

Evaluation metrics

We evaluate our model with the following metrics: mel-cepstral distortion (MCD) for harmonic features, modulation spectrum log-spectral distortion (MS-LSD) for harmonic features over time, band aperiodicity distortion (BAPD) for aperiodicity features and false positive rate (FPR) and false negative rate (FNR) for voiced/unvoiced (V/UV) decision. These metrics are described in depth in §3.3.3.

For the listening tests, all stimuli were downsampled to 32 kHz, which is the lowest common denominator between the different systems.

For the systems trained on pseudo singing, we conducted an A/B preference test. The 18 participants were asked for their preference between two different stimuli, or indicate no preference. The stimuli consisted of two short excerpts (<10 s) of one song per voice/language. Versions with and without background music were presented. We perform pair-wise comparisons between our system and two other systems, resulting in a total of 24 stimuli.

Table 4.1: Quantitative results for the timbre models trained on pseudo singing, separated by voice/language. The IS16 system is excluded from the quantitative metrics because removing utterances from the dataset to use for testing would mean missing diphones would have to be replaced. The listed metrics are mel-cepstral distortion (MCD) and modulation spectrum log-spectral distortion (MS-LSD) for harmonic features, band aperiodicity distortion (BAPD) for aperiodic features, and false positive rate (FPR) and false negative rate (FNR) for voiced/unvoiced (V/UV) features.

		Harmonic		Aperiodic	V/UV	
Voice (language)	System	MCD (dB)	MS-LSD (<25 Hz/Full, dB)	BAPD (dB)	FPR (%)	FNR (%)
M1 (Eng.)	HTS	**4.95**	11.09/22.44	2.72	16.10	**2.46**
	NPSS	5.14	**7.79/8.18**	**2.44**	**11.22**	2.65
F1 (Eng.)	HTS	**4.75**	10.25/22.09	4.07	**15.60**	1.01
	NPSS	4.95	**5.68/9.04**	**3.83**	15.79	**0.56**
F2 (Spa.)	HTS	**4.88**	11.07/22.28	3.62	1.85	**2.21**
	NPSS	5.27	**8.02/6.59**	**3.38**	**1.40**	3.20

4.2.3 Results

Table 4.1 shows the results of the quantitative tests. Note that we did not include the IS16 system in these tests because removing utterances from the dataset to use for testing would mean missing diphones would have to be replaced. For the prediction of harmonic features, using frame-wise metrics, such as MCD, our system (NPSS) is slightly behind HTS. However, using sequence-wise metrics, such as MS-LSD, NPSS shows an improvement over HTS. We argue that this observation is due to systems that have a tendency to predict averages perform well on metrics that are themselves essentially averages (such as MCD), especially pseudo singing, which is much more constant than natural singing. For other features, such as BAPD on aperiodicity features, our system slightly outperforms HTS, even though this is a frame-wise metric. For metrics related to the voiced/unvoiced decision, results vary depending on the dataset.

The results of the preference test as shown in Figure 4.6 show a strong preference for NPSS over the HTS system, and a moderate preference over the IS16 system. It should be noted that the IS16 system was specifically designed with this kind of data in mind, unlike the other systems.

Some sound examples corresponding to this chapter are available online[2].

[2] https://mtg.github.io/singing-synthesis-demos/

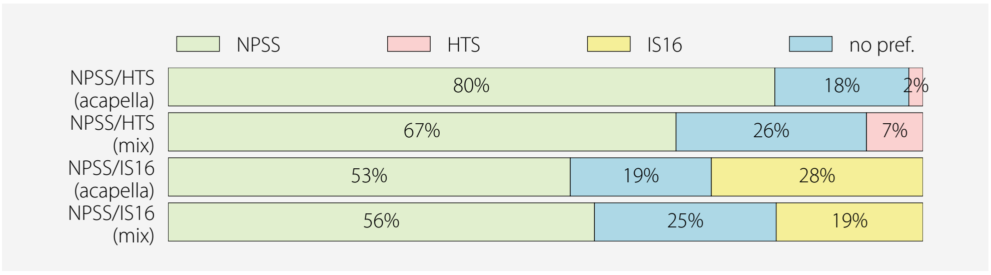

Figure 4.6: Results of the preference test for systems trained on pseudo singing. The Sinsy-HMM and Sinsy-DNN systems were excluded from this comparison, as the only available models are trained on natural singing.

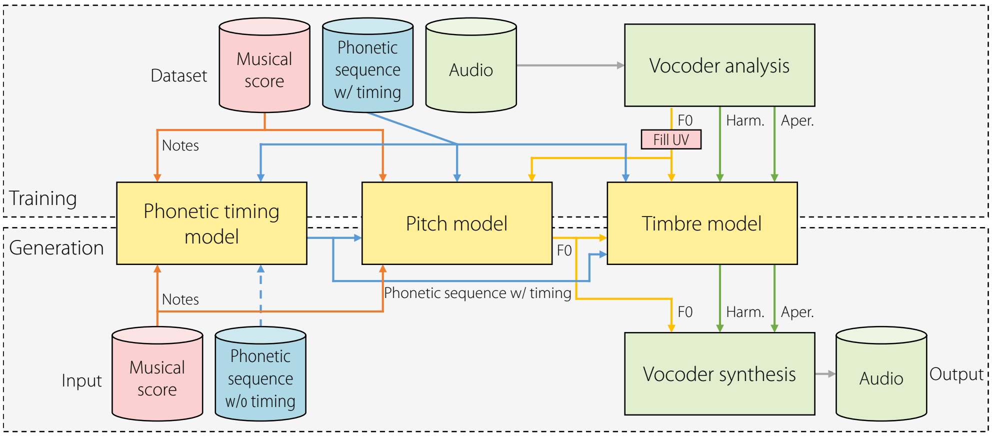

Figure 4.7: Diagram depicting an overview of the complete synthesizer with its different components. Here, the “Fill UV” block fills unvoiced segments by interpolation.

4.3 Modeling timbre, pitch and timing from natural singing

In this section, we introduce a complete singing synthesizer, able to synthesize audio from a score with lyrics. This is done by combining our existing timbre model introduced in §4.2 with necessary components that predict timing and pitch. Additionally, in our experiments, we train this system on natural singing rather than the much more constrained pseudo singing. We argue that this type of data, assuming we have a sufficient amount, should ultimately lead to more natural sounding synthetic singing, as our target output is also natural singing.

4.3.1 Complete singing synthesizer overview

A diagram of our complete singing synthesizer is depicted in Figure 4.7. The system is comprised of three main components; the phonetic timing model, the pitch model and the timbre model. While all of these components are neural networks, they are not trained jointly, in an end-to-end manner. Rather, each module is optimized independently, using so-called *teacher forcing* during training; e.g., the timbre model has an F0 input coming from the pitch model during inference, but during training the ground truth F0 is used. It should also be noted that the pipeline is not fully comprised of neural components; in the following experiments, we use a vocoder based on signal processing and heuristics, rather than a data-driven neural network. Additionally, while not depicted in Figure 4.7, inference will typically take a score with orthographic lyrics as input. In this case, an additional grapheme-to-phoneme module is needed, which in our experiments is based on either a dictionary, rules, or both (see “Grapheme-to-phoneme conversion” in §3.1.4).

The training data consists of time-aligned audio, musical scores and (timed) phonetic sequences. The vocoder is used to analyze the audio, producing F0, harmonic and aperiodicity features. The phonetic timing model takes the (quantized) notes and phonetic sequence with exact times, to predict note timing deviations and phoneme durations within each note. The pitch model takes notes as input to predict the frame-wise F0. Note that we ensure that this target F0 is continuous in order to facilitate modeling, by linearly interpolating unvoiced regions. The pitch model also takes the (timed) phonetic sequence as input, in order to properly model the influence of phonetic on pitch (microprosody, see below). Finally, the timbre model takes the (timed) phonetic sequence and frame-wise F0 sequence, to produce the harmonic and aperiodicity features.

During inference, the input to the system is a musical score with lyrics (typically orthographic). The lyrics are converted to a sequence of phonemes (without timing). The timing model then produces the timed phonetic sequence from the given score and phonetic sequence without timing. The pitch model generates a continuous, frame-wise F0 sequence from the given score and timed phonetic sequence. The timbre model then produces the harmonic and aperiodicity features from the F0 and timed phonetic sequence. Finally, the output waveform is produced by the vocoder synthesis, taking the predicted F0, harmonic and aperiodicity features.

4.3.2 Pitch model

Generating expressive F0 contours for singing voice is quite challenging. Not only is this because of its importance to the overall results, but also because in singing voice there are many factors that simultaneously affect F0. There are a number of musical factors, including melody, various types of attacks, releases and transitions, phrasing, vibratos, and so on. Additionally, phonetics can also cause inflections in F0, so-called microprosody (Taylor, 2009, Chap. 9.1.4, "Micro-prosody", p. 229). Some approaches try to decompose these factors to various degrees, for instance by separating vibratos (Oura et al., 2010) or using source material without consonants (Umbert et al., 2013; Bonada et al., 2016). In our approach, however, we model the F0 contour as-is, without any decomposition. As such, F0 is predicted from both musical and phonetic control inputs, using the same WaveNet-based architecture as used for the timbre model, with different hyperparameters (see Table 4.2 of "Model hyperparameters" in §4.3.4 for details).

Besides the phoneme-level linguistic control input features, similar to those used by the timbre model, the pitch model also takes a number of note-level musical features. The most important musical features are note pitch and duration, as one-hot and 4-state coarse coded vectors respectively. Additionally, we include the normalized position of the current frame within the current note as a sequence of 3-state coarse coded vectors, roughly corresponding to the probability of being in the beginning, middle, or end of the note.

Note that the model presented here is one of the most straightforward approaches to F0 modeling given the surrounding context of modeling timbre with an autoregressive model based on WaveNet (i.e., simply re-using the same model). However, we have also presented some other F0 models in the past such as the model presented in Bonada and Blaauw (2020). This model combines a data-driven, deep learning approach with a parametric model based on heuristics. Some of the advantages of this model are in particular that it does not require any post-processing heuristics to ensure F0 is always in tune (such as the one in "Tuning post-processing"). Additionally, the inductive biases that this model provides via the underlying parametric model, allow the model to be trained from very small datasets, such as a single song. This last advantage of course is closely related to the topics discussed in Part II.

Data augmentation

One issue with modeling pitch, is that obtaining a dataset that sufficiently covers all notes in a singer's register can be challenging. Assuming that pitch gestures are largely independent of absolute pitch, we apply data augmentation by pitch shifting the training data, similar to Mase et al. (2010). While training, we first draw a pitch shift in semitones, Δp, from a discrete uniform random distribution, for each sample in the minibatch,

$$\Delta p \sim \mathcal{U}\left(\Delta p_{\text{min}}, \Delta p_{\text{max}}\right) \tag{4.7}$$

$$\Delta p_{\text{min}} = p_{\text{min}}^{\text{singer}} - p_{\text{max}}^{\text{sample}} \tag{4.8}$$

$$\Delta p_{\text{max}} = p_{\text{max}}^{\text{singer}} - p_{\text{min}}^{\text{sample}}, \tag{4.9}$$

where Δp_{min} and Δp_{max} define the maximum range of pitch shift applied to each sample. This range is computed from the singer's register $[p_{\text{min}}^{\text{singer}}, p_{\text{max}}^{\text{singer}}]$, and the range of pitches the minibatch sample contains, $[p_{\text{min}}^{\text{sample}}, p_{\text{max}}^{\text{sample}}]$, all in semitones. This ensures that all notes of the melody within a sample can occur at any note within the singer's register. Finally, this pitch shift is applied to both the notes pitch used as a control input, p_{notes}, and the target output F0, $F0_{\text{tar}}$,

$$\hat{p}_{\text{notes}} = p_{\text{notes}} + \Delta p \tag{4.10}$$

$$\hat{F0}_{\text{tar}} = F0_{\text{tar}}\, 2^{\frac{1}{12}\Delta p}. \tag{4.11}$$

Tuning post-processing

For pitch in singing voice, one particular concern is ensuring that the predicted F0 contour is in tune. The model described above does not enforce this constraint, and in fact we observed predicted pitch to sometimes be slightly out of tune. If we define "out of tune" as simply deviating a certain amount from the note pitch, it is quite normal for F0 to be out of tune for some notes in expressive singing, without perceptually sounding out of tune. One reason why our model sometimes sounds slightly out of tune may be that such notes are reproduced in different musical contexts where they do sound out of tune. We speculate that one way to combat this is may be to use a more extensive dataset.

We improve the tuning of our system by applying moderate post-processing of predicted F0. For each note (or segment within a long note), the perceived pitch is estimated using F0 and its derivative. The smoothed difference between this pitch and the score note pitch is used to correct the final pitch used to generate the waveform. Appendix A.2 discusses the algorithm in detail.

4.3.3 Timing model

The timing model is used to predict the duration of each phoneme in the sequence to synthesize. Unlike with TTS systems where phoneme durations are generally predicted in a freerunning manner, in singing synthesis, the phoneme durations are heavily constrained by the musical score. In our proposed system we enforce this constraint using a multi-step prediction. First, the note timing model predicts the deviations of note (and rest) onsets with respect to nominal onsets in the musical score. At the same time, phoneme durations are predicted by the phoneme duration model. Finally, a simple fitting heuristic is used to ensure the predicted phoneme durations fit within the available note duration, after adjusting timing. This approach is somewhat similar to the approach taken by Nakamura et al. (2014).

Note timing model

Most singers will not follow the timing of a musical score exactly. Slightly advancing or delaying notes is part of normal expressive singing, and is the result of the given musical and linguistic context and the style of the singer. Additionally, there may be a small truly random component, simply because most singers cannot sing with exact timing.

Note onset deviations are computed from a musical score and phonetic segmentation of the corresponding utterance by the singer. We define a note onset deviation as the difference between the onset of the first syllabic nucleus in a note and that note's nominal onset as written in the musical score. These deviations are also computed for rest notes, or equivalently, note offsets before a rest.

We use a neural network to predict these deviations from note-level musical and linguistic input features. These input features are designed by hand, in part because using note-level data means we have relatively few samples compared to phoneme or frame-level data. We assume that these features contain most or all contextual information relevant to computing note time deviations, therefore we can use a simple feed-forward neural network, without the need for a recurrent or convolutional architecture. To avoid making any assumptions about the (conditional) probability distribution of the note onset deviations, we use a non-parametric approach by using a softmax output and discretizing the deviations to multiples of the hop time. Details of the input features and network architecture are available in Table 4.4 (§4.3.4).

Phoneme duration model

Phoneme durations are obtained in a similar way. They are first computed from the given phonetic segmentation, and then discretized on a log scale, similar to Arik et al. (2017b). A neural network is used to predict the phoneme durations from phoneme-level musical and linguistic input features. Unlike the note timing model, in this case we do require some local context information, so we use a convolutional architecture. Here we assume the range of context information affecting the duration of a phoneme to be limited by the musical score and the linguistic constraints on the number of possible onset and coda consonants. Therefore, the limited receptive field of a convolutional neural net should not be a significant disadvantage over a recurrent neural net's unbound receptive field. See Table 4.4 of "Model hyperparameters" in §4.3.4 for details.

Fitting heuristic

The fitting heuristic is used to conform the total of predicted phoneme durations to the available note duration predicted by the note timing model. The basic strategy is to expand or shrink the (principal) vowel, ensuring it is always at least some given percentage of the note duration, by also shrinking consonants if needed.

First, the sequence of phonemes to fit in the note duration is obtained by "shifting" onset consonants to the preceding note. The sequences will thus always start with a vowel (or silence for rests), followed by zero or more consonants formed by the note's coda consonants and the next note's onset consonants. In cases where a note contains multiple syllables, the secondary vowels are handled as if they were consonants. Then, the sequence of N predicted durations $d_0, d_1, \dots, d_{N-1}$ is fit into the available note duration d_a,

$$r = \min\left(1, \frac{d_a(1 - r_0)}{\sum_{i=1}^{N-1} d_i}\right), \tag{4.12}$$

where r_0 is the minimum fraction of the note's duration to be occupied by the primary syllabic nucleus.

$$\hat{d}_i = \begin{cases} d_a - r \sum_{j=1}^{N-1} d_j & \text{for } i = 0 \\ r d_i & \text{for } i = 1, 2, \dots, N-1. \end{cases} \tag{4.13}$$

4.3.4 Experiments

Datasets

For systems trained on natural singing, we use a public dataset published by the Nagoya Institute of Technology (Nitech), identified as NIT-SONG070-F001[3]. This dataset consists of studio-quality recordings of a female singer singing Japanese children's songs. The original dataset consists of 70 songs, but the public version consists of a 31

[3]Available from: http://hts.sp.nitech.ac.jp/archives/2.3/HTS-demo_NIT-SONG070-F001.tar.bz2

song subset (approximately 31 min, including silences). Out of these 31 songs, we use 28 for training and 3 for testing (utterances 015, 029 and 040).

As we had no involvement in the preparation of NIT-SONG070-F001, and information about it is limited, we cannot be sure how the labeling was performed. However, in our observations while working with this dataset, we consider note and phonetic labels to be generally correct and fairly precise.

Note that this dataset is small compared to the datasets typically used to train TTS systems. However, comparable dataset sizes (e.g., 40 h) would likely exceed the repertoire of most singers. In our experience, small datasets, like the one above already yield acceptable results, while slightly bigger datasets, e.g., 1–4 h, seems to be the sweet spot where results can be very good and the recording session is still manageable.

Compared systems

In our experiments, we compare our proposed system to three other systems which represent the three main paradigms for singing synthesis at the time of initial publication. One system is state of the art concatenative unit selection-based system. The second is a publicly accessible system based on HMMs. The third is a publicly accessible system based on early attempts at using deep neural networks (DNNs) for singing synthesis.

NPSS Our complete NPSS system, combining tjhe timbre model described in §4.2.1 with the pitch model described in §4.3.2 and the timing model described in §4.3.3. Model hyperparameters are listed in Table 4.2 of "Model hyperparameters" in §4.3.4.

IS16 A concatenative unit selection-based system (Bonada et al., 2016), which was the highest-rated system in the Interspeech 2016 Singing Synthesis Challenge. This is an earlier work by our research group. While described in detail in the paper, currently no implementation of this model is publicly available.

Sinsy-HMM A publicly accessible implementation of the Sinsy HMM-based synthesizer[4]. This system is described in Oura et al. (2010) and Oura et al. (2012), although the implementation may differ to some degree from any single publication, according to one of the authors in private correspondence. While the system was trained on the same NIT-SONG070-F001 dataset, it should be noted that the full 70 song dataset was used, including the 3 songs we use for testing.

Sinsy-DNN A publicly accessible implementation of the Sinsy feed-forward DNN-based synthesizer[4] (Nishimura et al., 2016). The same caveats as with Sinsy-HMM apply here. Additionally, the DNN voice is marked as "beta", and

[4]http://www.sinsy.jp/

thus should be considered still experimental. The prediction of timing and vibrato parameters in this system seems to be identical to Sinsy-HMM at the time of writing. Thus, only timbre and "baseline" F0 are predicted by the DNN system.

Model hyperparameters

Table 4.2 lists the hyperparameters for the timbre model and pitch model, which both use the same modified WaveNet architecture. Table 4.4 list the hyperparameters for timing models, which use a simpler architecture. All models are trained using the Adam optimizer (Kingma and Ba, 2015) with standard parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$. The learning rate schedule in an inverse time decay schedule, given as,

$$\mathrm{LR}_t = \mathrm{LR}_{\mathrm{init}} \frac{1}{1 + \mathrm{LR}_{\mathrm{decay}} \cdot t / \mathrm{LR}_{\mathrm{step}}}, \tag{4.14}$$

where t is the training step (in updates), and $\mathrm{LR}_{\mathrm{init}}$, $\mathrm{LR}_{\mathrm{decay}}$, and $\mathrm{LR}_{\mathrm{step}}$ are initial learning rate, decay and decay interval (in updates) respectively, as listed in the table. Training a complete system takes around 10 h on a single NVIDIA Titan X Pascal GPU. While we found these settings to work well experimentally, they have not been exhaustively optimized.

Table 4.2: Hyperparameters for our proposed autoregressive (multi-stream) timbre and pitch models. Symbols within parentheses refer to those in Figure 4.1 and the corresponding equations in the text. Coarse coding of input control features is described in Table 4.3.

	Timbre model		Pitch model
Hyperparameter	**Harmonic**	**Aperiodic**	**F0**
Feature dimensionality (d_{out})	60	4	1
Additional inputs (dim.)	-	harmonic (60)	-
Control inputs	prev. phn. identity (one-hot) cur. phn. identity (one-hot) next phn. identity (one-hot) pos.-in-phn. (coarse) F0 (coarse)		prev. phn. class (one-hot) cur. phn. class (one-hot) next phn. class (one-hot) pos.-in-phn. (coarse) prev. note pitch (one-hot) cur. note pitch (one-hot) next note pitch (one-hot) prev. note dur. (coarse) cur. note dur. (coarse) next note dur. (coarse) pos.-in-note (coarse)
Frame rate (Hz)	200	200	200
Input noise level (λ_{reg})	0.2	0.2	0.2
Initial causal convolution	10×1	10×1	20×1
Dilated convolutions	2×1	2×1	2×1
Num. layers (K)	5	5	13
Dilation cycle (num. layers)	3	3	7
Dilation factors	1, 2, 4, 1, 2	1, 2, 4, 1, 2	1, 2, 4, 8, 16, 32, 64, 1, 2, 4, 8, 16, 32
Receptive field (ms, frames)	100 (20)	100 (20)	1050 (210)
Residual channels (d_{res})	130	20	100
Hidden channels (d_{hid})	130	20	100
Skip channels (d_{skip})	240	16	100

Continued on next page.

Table 4.2: *(Cont.)*

	Timbre model		Pitch model
Hyperparameter	**Harmonic**	**Aperiodic**	**F0**
Output stage ($K_{\text{out}} = 0$)	tanh $\rightarrow$ 1×1 $\rightarrow$ 60× CGM	tanh $\rightarrow$ 1×1 $\rightarrow$ 4× CGM	tanh $\rightarrow$ 1×1 $\rightarrow$ 1× MoL
Generation temperature (τ)	piecewise linear (0,0.05; 3,0.05; 8,0.5; 60,0.5)	0.01	0.01
Batch size	32	32	64
Seq. len. (ms, valid timesteps)	210 (42)	210 (42)	105 (21)
Learning rate (schedule) (LR_{init}, LR_{decay}, LR_{step})	5×10^{-4}, 1×10^{-5}, 1	5×10^{-4}, 1×10^{-5}, 1	1×10^{-3}, -
Num. epochs (updates)	1650 (82,500)	1650 (82,500)	235 (11,750)

Table 4.3: Encoding of control features for our proposed autoregressive model. See §3.1.4 for details. The exact dimensionality and triangular encoding center points for F0 and note duration will depend on the range of pitches and note durations in the dataset.

Feature	Encoding kind	Dimensionality
Position in phoneme (norm.)	Gaussian	3
Position in note (norm.)	Gaussian	3
Log F0	Triangular	3–4
Log abs. note durations	Triangular	3–4

Evaluation metrics

We evaluate our timbre model with the following metrics: MCD for harmonic features, MS-LSD for harmonic features over time, BAPD for aperiodicity features and FPR and FNR for V/UV decision. These metrics are described in depth in §3.3.3.

For the listening tests, all stimuli were downsampled to 32 kHz, which is the lowest common denominator between the different systems. We conducted a multiple stimuli with hidden reference and anchor (MUSHRA) (ITU-R Recommendation BS.1534-3, 2015) style listening test. The 40 participants, of which 8 indicated native or good knowledge of Japanese, were asked to rate different versions of the same audio excerpt compared to a reference. The test consisted of 2 short excerpts (<10 s) for each of the 3

Table 4.4: Hyperparameters for timing networks.

Hyperparameter	**Note timing**	**Phoneme duration**
Input features	note duration (one-hot) prev. note duration (one-hot) 1st phoneme class (one-hot) note position in bar (normalized) note is rest num. coda consonants prev. note prev. note is rest	phoneme identity (one-hot) phoneme class (one-hot) phoneme is vowel phoneme kind (onset/nucleus/coda/inner) note duration (one-hot) prev. note duration (one-hot) next note duration (one-hot)
Target value range (5 ms frames)	[-15,14], [-30,29] for rests	[5,538]
Target discretization	30 or 60 bins, linear	50 bins, log scale
Architecture	input $\rightarrow$ dropout(0.81) 1×1 $\rightarrow$ 256× ReLU $\rightarrow$ dropout(0.9) 1×1 $\rightarrow$ 64× ReLU $\rightarrow$ dropout(0.9) 1×1 $\rightarrow$ 32× ReLU $\rightarrow$ dropout(0.81) 1×1 $\rightarrow$ 30-way softmax	input $\rightarrow$ dropout(0.8) 3×1 $\rightarrow$ 256× gated tanh $\rightarrow$ dropout(0.8) 3×1 (dilation = 2) $\rightarrow$ 64× gated tanh $\rightarrow$ dropout(0.8) 1×1 $\rightarrow$ 32× gated tanh $\rightarrow$ dropout(0.64) 1×1 $\rightarrow$ 50-way softmax
Batch size	32	16
Seq. len.	full (padded)	full (padded)
Learning rate	2×10^{-4}	2×10^{-4}
Number of epochs	140	210

Table 4.5: Quantitative results for the timbre models trained on natural singing. Note that for the IS16 system the modulation spectrum log-spectral distortion (MS-LSD) and voiced/unvoiced metrics are omitted as it does not use predicted harmonic features (MS-LSD is computed from predicted features, not analyzed features) or V/UV decision. The HTS system is only considered when comparing systems trained on pseudo singing, but should be roughly equivalent to Sinsy-HMM.

System	Harmonic		Aperiodic	V/UV	
	MCD (dB)	MS-LSD (<25 Hz/Full, dB)	BAPD (dB)	FPR (%)	FNR (%)
IS16	6.94	-	3.84	-	
Sinsy-HMM	7.01	8.09/18.50	4.09	15.90	0.68
Sinsy-DNN	**5.41**	13.76/29.87	5.02	**13.75**	**0.63**
NPSS	5.54	**7.60/11.65**	**3.44**	16.32	0.64

validation set songs, in 7 versions (reference, hidden reference, anchor and 4 systems), for a total of 42 stimuli. The scale used as 0–100, divided into 5 segments corresponding to a 5-scale mean opinion score (MOS) test. The anchor consisted of a distorted version of the NPSS synthesis, applying the following transformations: 2-d Gaussian smoothing (with scale $\sigma = 10$) of harmonic, aperiodic and F0 parameters, linearly expanding the spectral envelope by 5.2%, random pitch offset (± 100 cents every 250 ms, interpolated by cubic spline), and randomly “flipping” 2% of the voiced/unvoiced decisions. We excluded 59 of the total 240 tests performed, as these had a hidden reference rated below 80 (ideally the rating should be 100). We speculate that these cases could be due to the relative difficulty of the listening test for untrained listeners.

4.3.5 Results

For systems trained on natural singing, Tables 4.5–4.7 list quantitative metrics related to timbre, timing and pitch models respectively. Examples of different modulation spectra for timbre and pitch are shown in Figures 4.8–4.10. For systems trained on pseudo singing, Table 4.1 lists quantitative metrics related to timbre models.

These metrics show that, for some of the frame-wise metrics, such as harmonic MCD, our system is slightly behind. For some other metrics, such as the timing errors or aperiodic BAPD, our system is slightly ahead. For systems trained on pseudo singing the differences tend to be a little bigger, we argue that this is due that predicting averages for this kind of data results in good results for these kinds of metrics. However, in all metrics based on the modulation spectrum, which considers variations in time, NPSS shows an improvement over the other systems.

Table 4.6: Quantitative results for the timing models trained on natural singing. The table lists mean absolute error (MAE) and root mean squared error (RMSE), both in 5 ms frames, and Pearson correlation coefficient r. Note that the Sinsy-DNN system uses the same HMM-based duration model as the Sinsy-HMM system, so it is excluded from the comparison. The IS16 system used durations predicted by the NPSS system. The HTS system is only considered when comparing systems trained on pseudo singing, but should be roughly equivalent to Sinsy-HMM.

System	Note onset deviations			Note offset deviations			Consonant durations		
	MAE	RMSE	r	MAE	RMSE	r	MAE	RMSE	r
Sinsy-HMM	7.107	9.027	0.379	13.800	**17.755**	0.699	4.022	5.262	0.589
NPSS	**6.128**	**8.383**	**0.419**	**12.100**	18.645	**0.713**	**3.719**	**4.979**	**0.632**

System	MS-LSD (<25 Hz, dB)	RSME (cents)	r
Sinsy-HMM	5.052	**81.795**	**0.977**
Sinsy-DNN	2.858	83.706	0.976
NPSS	**2.008**	105.980	0.963

Table 4.7: Quantitative results of pitch models trained on natural singing. Table shows log F0 modulation spectrum log-spectral distortion (MS-LSD) in dB. The F0 root mean squared error (RMSE) in cents and Pearson correlation coefficient r are also given for reference. The IS16 and HTS systems are excluded from this comparison because they are not suitable for modeling F0 from natural singing.

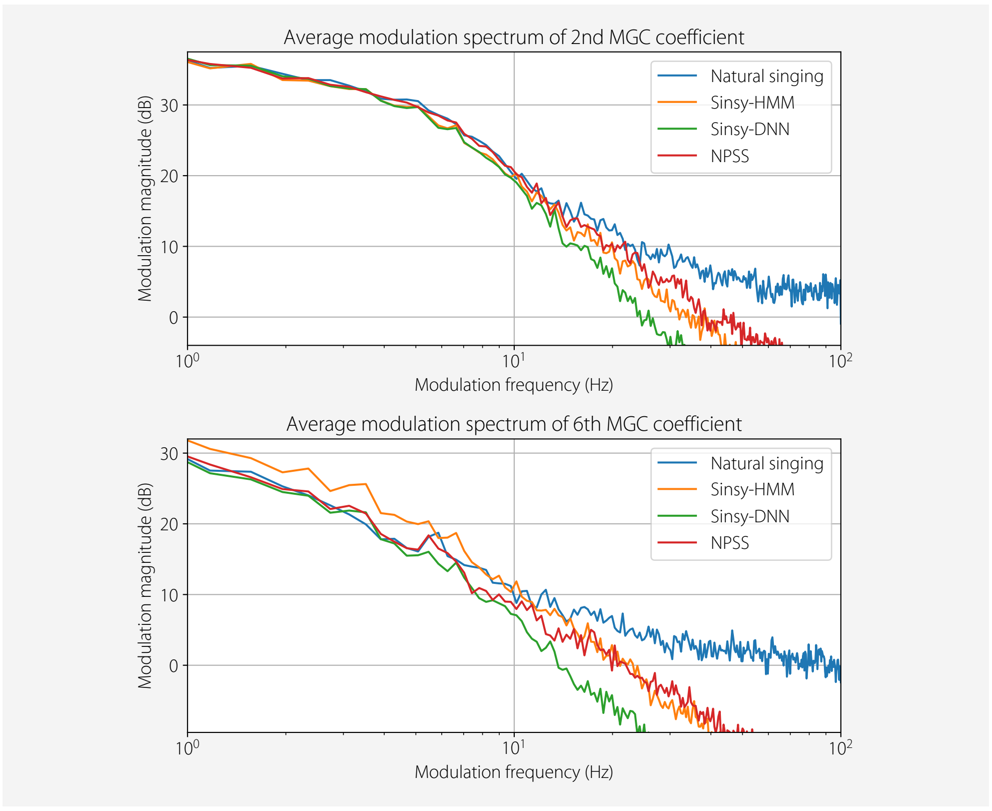

Figure 4.8: Comparing the average modulation spectrum (MS) of synthesized mel-generalized coefficient (MGC) features.

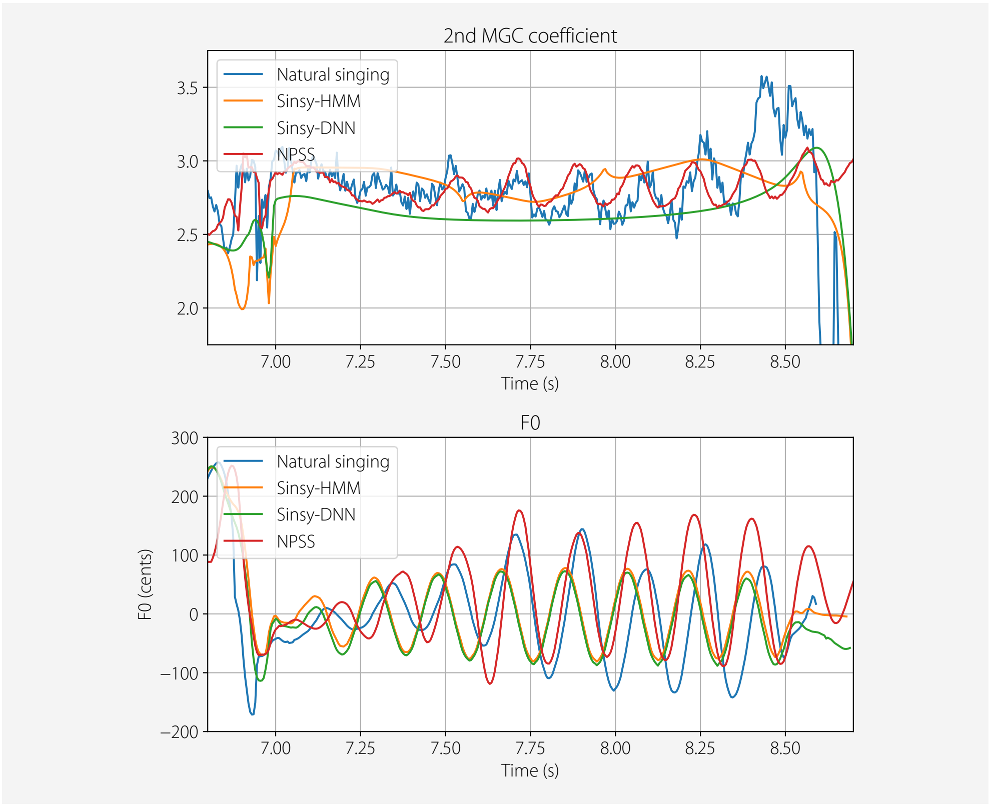

Figure 4.9: An example of synchronization between F0 and synthesized mel-generalized coefficient (MGC) features. In the plotted excerpt, the relation between pitch and timbre during vibratos can be observed.

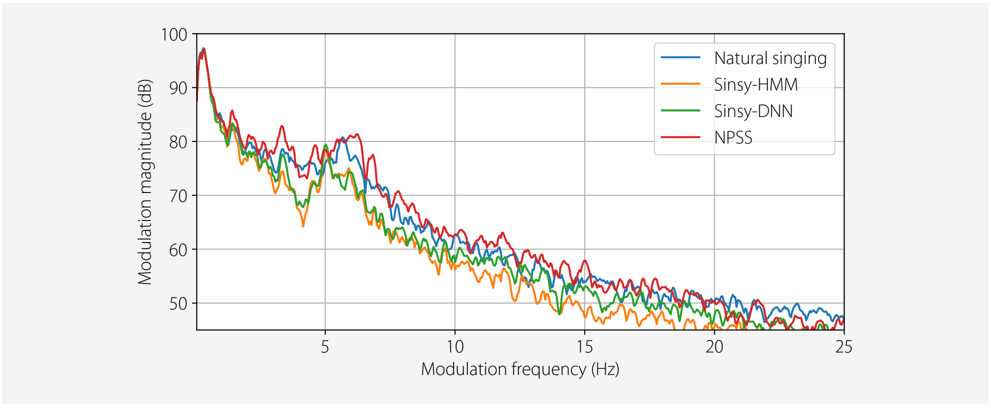

Figure 4.10: Comparing the average modulation spectrum (MS) of log F0 contours predicted by various systems and natural singing.

Table 4.8: Mean opinion score (MOS) for systems trained on natural singing, displayed on a 1–5 scale with their respective 95% confidence intervals. The HTS system is only considered when comparing systems trained on pseudo singing, but should be roughly equivalent to Sinsy-HMM.

System	Mean opinion score
Hidden reference	4.76 ± 0.04
IS16	2.36 ± 0.11
Sinsy-HMM	2.98 ± 0.10
Sinsy-DNN	2.77 ± 0.10
NPSS	$\mathbf{3.43 \pm 0.11}$

When we compare an example of generated harmonic parameters during a vibrato in Figure 4.9, we notice the features predicted by NPSS having more detail than Sinsy-HMM and Sinsy-DNN. In particular, the frame-wise conditioning of harmonic features on F0 in NPSS, causes the harmonic features to modulate along the vibrato, similar to what happens in the reference recording. In the modulation spectrum analysis of Figure 4.8, we can see that overall NPSS tends to follow the modulation spectrum of the reference recording a little closer than Sinsy-HMM and Sinsy-DNN in lower modulation frequencies. Compared to especially Sinsy-DNN, NPSS has less rolloff in higher modulation frequencies, indicating less oversmoothing over time. However, all systems have less high-frequency modulation spectrum content than the reference recording, indicating none of the systems are able to reproduce all the details of the original signal.

The analysis of the modulation spectrum of the log F0 predicted by different systems is shown in Figure 4.10. We can see that overall NPSS matches the modulation spectrum of the reference recording similarly or slightly better than Sinsy-HMM, but notably better than Sinsy-DNN. When we focus our attention to the range of modulation frequencies corresponding to vibratos in this voice, 5–7 Hz, we see that Sinsy-HMM and Sinsy-DNN have a sharp peak at 5 Hz, whereas for NPSS this whole range has increased energy, similar to the reference. This may indicate that NPSS produces a wider range of vibrato rates, similar to a real singer. In Sinsy-HMM and Sinsy-DNN vibrato parameters (rate and depth) are modeled separately from the base F0, which may explain their tendency to produce very controlled, regular vibratos.

In the listening tests, listed in Table 4.8, NPSS is clearly ahead of competing systems. In the MOS test for systems trained on natural singing, NPSS is around a third between the second-best system (Sinsy-HMM) and the (hidden) reference. The correlation between the qualitative results and the quantitative metrics based on the modulation spectrum indicate that this may be a metric with higher perceptual relevance than the frame-wise metrics such as MCD.

In our experience, the NPSS, Sinsy-HMM and Sinsy-DNN systems all produce quite coherent timbres. The concatenative system, IS16, in contrast, tends to produce more discontinuous timbres, which becomes especially apparent when using a dataset of natural singing. There are also other situations when such artifacts at concatenation boundaries become more noticeable, e.g., in fast singing or when phonetic segmentation is not perfect. We found NPSS to generally produce less static features over time, and less coloring of timbre. Compared to the Sinsy-HMM and Sinsy-DNN systems, the autoregressive generation of NPSS seems to help in reproducing rapidly varying consonants, although these can occasionally sound better still in the concatenative system. In terms of expression, Sinsy-HMM produces very coherent behavior, which, while perhaps a little less human, tends to generally sound quite pleasant. NPSS on the other hand, seems to be more varied, but this also means that results are sometimes better than other times. One notable quality of NPSS is that the frame-wise conditioning of timbre on pitch means that vibratos produce natural, synchronized modulations in both pitch and timbre (see, e.g., Figure 4.9), unlike in the other systems which condition on note pitch.

4.4 Conclusions

In this chapter, we presented a neural network capable of modeling singing voice timbre, given pitch and a timed phonetic sequence as input. Additionally, we extended this model to a complete singing voice synthesizer by adding models that model timing and pitch. These separate, but interconnected models are able to generate synthetic singing voice given a musical score with lyrics.

We first performed experiments with the timbre model trained on pseudo singing, using performance-driven synthesis, in order to allow easier comparison to the then state of the art in terms of modeling flexibility and sound quality; HMM and concatenative synthesis respectively. In listening tests, our autoregressive model with powerful network architecture based on WaveNet has shown to be a notable improvement over statistical parametric (HMM) systems, as well as simpler DNN and CNN neural networks, that were common when our model was first published. In particular, it offers improved reproduction of consonants and a more natural variation of predicted features over time. Compared to concatenative approaches, our model shows a more moderate improvement in sound quality, but allows for much greater flexibility and is more robust to small mis-alignments between phonetic and acoustic features in the training data.

In our second set of experiments, we trained a complete singing synthesizer, including timing and pitch prediction, exploiting the fact that our system allows using datasets consisting of natural singing (something that is problematic for concatenative synthesizers for instance). In listening tests, our system was rated to reduce the gap between the second-best system and the reference recording by about a third. While correlating this with quantitative metrics is challenging, metrics that take into account variations over time, such as the modulation spectrum, do seem to corroborate this.

The moderate dataset size requirements, relatively fast training times, and a relatively small CPU, memory and disk footprint allows for many practical applications of our system.

Non-autoregressive modeling of timbre using self-attention | 5

The autoregressive model introduced in Chapter 4 has many advantages compared to a naive non-autoregressive model, but this kind of model indeed also has some disadvantages. Firstly, due to its sequential nature, inference cannot take advantage of highly parallel hardware such as graphics processing units (GPUs). While this is a major disadvantage of autoregressive models in general, in our specific case this is less of a concern in practice as we use a relatively low frame rate and a smaller network architecture (see §4.1.4). Secondly, an arguably bigger concern is the discrepancy between training objective and inference algorithm, resulting in the so-called exposure bias issue (see "Regularization to mitigate exposure bias" in §4.1.3). A typical symptom of this issue is that inference can become unstable due to the feedback processes, e.g., resulting in unnatural resonances that increase over time in long sustained vowels. While there are ways to mitigate these issues, these may not be effective in all cases, or may themselves influence the final synthesis quality. Thus, to provide an alternative to autoregressive models, our next work investigates non-autoregressive models.

In our initial experiments, we found that a naive non-causal, non-autoregressive version of the convolutional neural network (CNN)-based architecture in §4.1.2 results in a step back in quality, compared to its original autoregressive formulation. In particular, we empirically found that the resulting timbre tends to lack coherence over time. To combat this issue, we turn to a model inspired by the Transformer network (Vaswani et al., 2017), which notably includes self-attention as an alternative (or complement) to convolutional or recurrent building blocks.

This chapter is derived from work originally published as Blaauw and Bonada (2020). In this publication, we did not only try to obtain a non-autoregressive model comparable to our original autoregressive model, but also tried to exploit the sequence-to-sequence (Seq2Seq) aspect of the Transformer network[1] to reduce the effort required to create new voices. In particular, the proposed system avoids having to annotate all phoneme onsets in the training data, with just vowel onsets sufficing. This topic is discussed in depth in Chapter 8.

[1]The Transformer network was originally proposed in the context of neural machine translation (NMT), which is a prevalent sequence-to-sequence (Seq2Seq) problem.

5.1 Proposed system

5.1.1 Non-autoregressive models

Following the multivariate formulation of our autoregressive model as given in Equation (4.3) of §4.1.3, we can define a non-autoregressive equivalent,

$$p_\theta(\mathbf{x} \mid \mathbf{c}) := \prod_{t=1}^{T} \prod_{k=1}^{K} p_\theta(x_{t,k} \mid \mathbf{c}), \tag{5.1}$$

where $\mathbf{x}$ is a sequence of K-dimensional intermediate acoustic features, $\mathbf{c}$ is the time-aligned sequence of control features, $x_{t,k}$ is a scalar feature at timestep t and frequency bin k, and $p_\theta(\cdot \mid \cdot)$ is a function implemented by a neural network parametrized by θ. Note that, while not shown here, the model, $p_\theta(x_{t,k} \mid \mathbf{c})$, is generally also conditioned on output timestep, t. Often this happens implicitly due to the network architecture used, such as the step in recurrent neural network (RNN) networks or the offset in 1-d CNNs networks.

5.1.2 Attention and self-attention

We will first briefly describe the self-attention mechanism used in this work. To do so, we must first have some understanding of attention mechanisms in general. Attention mechanisms most frequently crop up in the context of Seq2Seq models. These Seq2Seq models convert an input sequence into an output sequence, where the two do not need to be aligned, i.e., have identical lengths, order and such. A typical task for such a model is neural machine translation (NMT). In text-to-speech (TTS), a typical use case would be converting an input sequence of orthographic characters or phonemes to an output waveform or intermediate acoustic features. Seq2Seq models will typically consist of an encoder that encodes the input sequence, a decoder that generates the output sequence. Traditionally, the encoder would summarize the input sequence into a fixed-length hidden vector, which is then taken by the decoder to produce the target output (Sutskever et al., 2014; Cho et al., 2014). Later, attention mechanisms were introduced in between encoder and decoder, to allow each output timestep to attend to different input timesteps in a more flexible way. In the context of TTS, we can think of this as learning the time-alignment between phonemes and the acoustic features, possibly by constraining the alignment to be monotonic. The advantages of such a Seq2Seq TTS system include allowing to train the system on <audio, text> pairs, without time-alignment, and in general a more end-to-end system (see Chapter 8 for more details).

There are many kinds of attention mechanisms (e.g., Bahdanau et al., 2014; Luong et al., 2015; Chorowski et al., 2015), however self-attention tends to be based on the scaled dot-product attention mechanism as proposed by Vaswani et al. (2017). One key aspect of this mechanism is that it is suitable for parallelized (feed-forward) computation, i.e., it does not rely on recurrent connections and such. In this mechanism, the input sequence is represented as a sequence of *key-value pairs*. These keys and values are typically simply linear projections of the input "tokens" (or embeddings thereof). Similarly, the output sequence is represented as a sequence of *queries*, again, typically linear projection of the output "tokens" (or embeddings thereof). Note that here a "token" does not necessarily need to be a categorical value, but can also be a vector of real values, such as a frame of acoustic features. The (scaled) dot-product is used as a *scoring function* to relate queries to keys. The "scaled" in scaled dot-product, simply refers to a scaling factor used to avoid small gradients that can hinder training. The output sequence is then obtained by a weighted average of the scored values, via softmax, typically followed by a linear projection. Often, in particular in natural language processing (NLP) applications, a *multi-head* variant of this mechanism is used. This means simply that the mechanism is run multiple times in parallel, using different projections. The intuition behind this is that each projection may focus on a different representation subspace, and each head learns independent attention for that subspace.

More formally, the attention operation can be defined as,

$$\text{Attention}(Q, K, V) := \text{softmax}\left(\frac{QK^\top}{\sqrt{d_\text{att}}}\right)V, \tag{5.2}$$

where Q, K, and V correspond to the (projected) query, key and value matrices respectively. For simplicity, we will assume the single-head variant of the mechanism, with d_att being the dimensionality (or number of channels) of the projected queries, keys, and values.

Self-attention (Vaswani et al., 2017) is identical to the above attention mechanism, except that rather than deriving Q from the output sequence, and K and V from the input sequence, all three matrices are derived from the same (input) sequence. Conceptually, we can think about this as rather than finding relations between input and output sequences, we find relations within the input sequence itself.

5.1.3 Self-attention as a building block

One key use for self-attention is as a building block that can replace or complement CNN or RNN blocks to learn dependencies over time. In fact, the encoder and decoder of the canonical version of the Transformer network (Vaswani et al., 2017), which is a very powerful model that excels at working with time series, only consists of point–wise fully connected layers (essentially 1×1 convolutions, i.e., operating on a single timestep) and self-attention.

We can think of RNN, CNN and self-attention as building blocks that combine information from an input sequence to produce an output sequence (of the same length). Self-attention has a number of advantages when compared to CNN or RNN operations. Firstly, self-attention, like CNNs, requires a constant number of sequential operations, regardless of the sequence length. RNNs on the other hand require a number of sequential operations that grows linearly with the sequence length. Secondly, self-attention, as well as RNNs, can access any timestep in the input sequence, while CNNs have a fixed receptive field. Thirdly, a single self-attention operation can connect any output to any input. Such a short or direct path is important in learning long-term dependencies between inputs (Hochreiter et al., 2001). CNN on the other hand will usually require a longer path, due to consisting of a number of stacked convolutional layers, typically each with a relatively small kernel size, and possibly using dilation. Similarly, in RNNs this path is directly given by the distance between an output and an input, and thus in the worst case is the sequence length. Finally, self-attention, like RNNs, tends to be a single layer, which can be stacked to form a deep architecture. In particular, when using a CNN with dilated convolutions, all layers contribute to a single big receptive field, but there often are no repeated blocks where any output can interact with any input. The principal downside of self-attention, compared to these other building blocks, is that computational complexity and memory requirements tends to be quadratic with respect to the sequence length.

Note that while models with recurrent building blocks may appear to be similar to autoregressive models and may have similar end results, in this work we consider them distinct. We consider that a model definition can be autoregressive or non-autoregressive. These models can then be implemented using network architectures using recurrent, like RNNs, or feed-forward components, like CNNs or self-attention, or both. Here, the only restriction is that autoregressive models require to be built using only causal operations. A feed-forward architecture allows an autoregressive model to have non-sequential training and sequential inference, whereas a non-autoregressive model will have non-sequential training and non-sequential inference. A recurrent architecture will make all training and inference sequential in nature. Therefore, many autoregressive models will use architectures with feed-forward building blocks.

5.1.4 Network architecture

Our proposed network architecture is depicted in Figure 5.1. Unlike the architecture described in §4.1.2, this network is explicitly structured as an encoder-decoder. This type of architecture is popular in TTS, in particular for Seq2Seq models, where linguistic input and acoustic output are not aligned. The encoder takes the linguistic input sequence (i.e., phonemes or orthographic characters), and generates a sequence of hidden features of the same length. The decoder can then take these hidden features to generate the acoustic output. As these sequences will be of different lengths, typically an attention mechanism connects the two, learning the (often monotonic) alignment between outputs and inputs. While this Seq2Seq approach is discussed in depth in Chapter 8, in this chapter we assume that the timing of phonemes is part of the training data (and input for inference). Thus, in this case, we do not need to use an attention mechanism, but instead, we use a much simpler *alignment module*. This alignment module takes the sequence of hidden features derived from phonetic input sequence by the encoder, together with a sequence of phoneme durations, and then repeats the hidden feature timesteps to match the acoustic target sequence's length.

Our model architecture is inspired by the architecture of FastSpeech proposed by Ren et al. (2019), which is based on the influential Transformer network (Vaswani et al., 2017). There are also similarities with the architecture proposed for ParaNet (Peng et al., 2019), which is a continuation of their earlier work Deep Voice 3 (Ping et al., 2018), which itself is based on a somewhat simplified version of the WaveNet architecture (van den Oord et al., 2016a). Note that after our publication, other singing synthesizers have been proposed that are also inspired by the FastSpeech architecture (e.g., Lu et al., 2020; Chen et al., 2020).

Encoder

Our encoder is based on the encoder proposed in (Ping et al., 2018). First, we look up a learned embedding for each phoneme in the input sequence. Then, a series of convolutional blocks with gated linear units (GLUs)[2] (Dauphin et al., 2017) allows encoding information about the phonetic context of each phoneme, e.g., corresponding to triphones or pentaphones, depending on the receptive field of this stack of convolutional blocks. Finally, a residual shortcut connection from the monophone embeddings is added to the local context output of the convolutional blocks.

This approach is quite different from the system proposed in Chapter 4, where the control signal was provided at the same rate as the acoustic features, and the phonetic context had to be "hardcoded" by including previous and next phoneme identities in the control signal. While the end result does not tend to be radically different, the approach of an explicit encoder operating on a phoneme-level control signal is more

[2]Note that the gated linear unit (GLU) activation is a popular choice for natural language processing (NLP) models, in particular those based on the Transformer network. Compared to e.g., gated tanh units (GTUs), it still provides gating and non-linear activation, but with less chance of vanishing gradients.

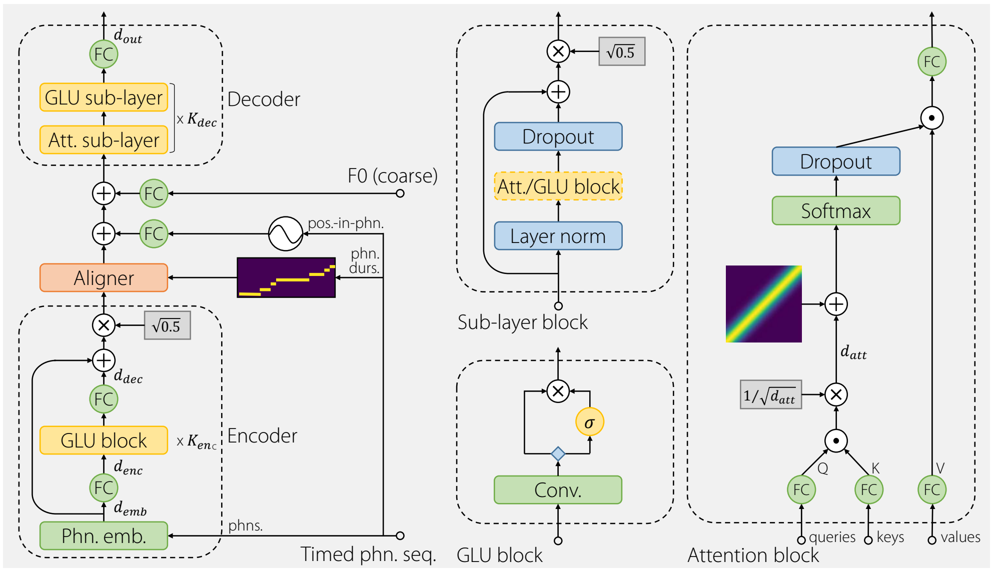

Figure 5.1: A diagram of the complete model architecture. On the left is the full system, composed of encoder, aligner and decoder, which themselves are composed of different higher-level blocks. On the right, these higher-level blocks (sub-layer, gated linear unit (GLU) and attention) are shown in detail. The "FC" modules are fully connected layers, implemented as 1×1 convolutions.

elegant and flexible. For instance, in theory including duration information would allow a learned phonetic context dependency based on the speed of singing (i.e., fast singing is likely to include more significant coarticulation). However, in the experiments in this work, we opted for a simpler encoder that only combines monophone and triphone information, without considering durations.

Decoder

Our decoder is based on a feed-forward variant of the Transformer network (Vaswani et al., 2017), similar to (Ren et al., 2019), which we will refer to as the feed-forward Transformer (FFT). Each layer consists of a self-attention sub-layer block and a convolutional sub-layer block. Both sub-layer blocks have layer normalization (Ba et al., 2016), dropout (Srivastava et al., 2014) and a residual shortcut connection (He et al., 2016).

As described in §5.1.2, the self-attention mechanism we use is based on the scaled dot-product. Additionally, similar to (Sperber et al., 2018), we bias the scores with a Gaussian along the diagonal to favor a more localized self-attention. This is done by adding a term to scaled dot-product of Equation (5.2), prior to the softmax operation,

$$\text{Attention}(Q, K, V) := \text{softmax}\left(\frac{QK^{\top}}{\sqrt{d_{\text{all}}}} + M\right)V, \tag{5.3}$$

where Q, K, and V correspond to the (projected) query, key and value matrices respectively, and d_{att} is their dimensionality. The diagonal Gaussian bias, $M \in \mathbb{R}^{T \times T}$, is computed as,

$$M_{j,k} = \frac{-(j-k)^2}{2\sigma_{\text{bias}}^2}, \tag{5.4}$$

where σ_{bias} is a learned scale parameter, and T is the sequence length. While the use of multi-head attention is typical for NLP applications, we did not find this improved results in our case.

For the convolutional blocks, we use a non-causal convolution without dilation, followed by a GLU activation, much like a very stripped-down version of the WaveNet block (see e.g., Figure 4.1). We found that for our case this approach outperforms the two-layer convolutional network with central rectified linear unit (ReLU) activation and higher inner dimensionality that is typically used in Transformer architectures.

When using intermediate acoustic features with multiple streams, such as WORLD vocoder features, we simply concatenate the different features in this model. Compared to the approach described in "Multi-stream architecture" of §4.2.1, we found that this simplifies the model considerably, while only resulting in a marginal difference in output sound quality.

F0 and position conditioning

As with our autoregressive timbre model (§4.2.1), continuous log F0 is encoded as a low-dimensional vector between zero and one by evaluating several triangular basis functions whose centers are placed at frequencies appropriate for the training data's pitch range. This triangular encoding is described in depth in §3.1.4.

Transformers typically use an additive trigonometric positional encoding to give the self-attention blocks a sense of the position of their inputs, and provide a linear inductive bias along early on in training. However, in our case, we found that a simple K-dimensional cyclical encoding (see §3.1.4, Equation (3.9)) of the normalized frame position within each phoneme gave slightly better results.

Reduction factor

As mentioned, one of the principal disadvantages of self-attention is that computation complexity and memory requirements are quadratic with respect to the sequence length. This can cause issues whenever sequences can be very long, such as when modeling intermediate acoustic features, which often are sampled at a rate of 200 Hz. It should be noted here, that this kind of model tends to use minibatches that consist of full phrases, while purely convolutional architectures (e.g., the architecture in §4.2.1) can use minibatches of arbitrary length slices without loss of generality.

One way to mitigate the issue of quadratic complexity is to reduce the number of timesteps of the acoustic features by using an integer reduction factor $r \geq 1$, which means r frames are predicted per output timestep (Wang et al., 2017; Ping et al., 2018). That is, if we have some acoustic features, $x \in \mathbb{R}^{T \times d_{\text{feat}}}$, we can reshape this matrix to become, $\mathbb{R}^{(T/r) \times (d_{\text{feat}} r)}$, where T is the number of timesteps, and d_{feat} is the feature dimensionality.

Additionally, we found that using a slightly lower rate for the intermediate acoustic features (e.g., 10 ms rather than 5 ms), still yields acceptable results. This is the case in particular when the system is used in conjunction with a more powerful neural vocoder, capable of "recovering" more waveform detail from lower resolution acoustic features.

Note that a lot of the work on Transformer-based models after the publication of our initial work on this model has been focused on making the model more scalable to very long sequences (e.g., Kitaev et al., 2020; Wang et al., 2020a). While we have not experimented which such newer approaches, our intuition is that perhaps they would not have a very big effect on the final result in our use case. In particular, we are interested in global coherence, but at the same time, most effects are fairly localized. That said, a more scalable network would greatly benefit practical deployment of this model.

5.1.5 Training and inference speed

Training times for this kind of model are quite comparable, albeit slightly higher, than our previously proposed autoregressive model (see §4.1.4), around 10–15 h. Inference speeds are notably higher however, around 770 × real-time. All of these numbers are for an NVIDIA GTX 1080Ti GPU. We did not benchmark performance on central processing unit (CPU), as this model is much less suited for hardware with relatively low parallelism.

5.2 Relation to prior work

Not considering Seq2Seq singing synthesizers, which are discussed in §8.2, a number of other non-autoregressive and related singing synthesizers have been proposed. For instance, Nakamura et al. (2019) propose a non-autoregressive CNN-based singing synthesizer. Here, downsampling and upsampling operations are used for capturing long-term dependencies in the singing voice (somewhat similar to the U-Net architecture (Ronneberger et al., 2015)). Another group of approaches uses generative adversarial network (GAN)-based approaches (e.g., Hono et al., 2019; Chandna et al., 2019). Here, typically a CNN-based generator is combined with a discriminator to provide an adversarial loss. A key point of such adversarial losses is that they do not operate on a single timestep, as is common with most explicit losses, but rather take a full sequence or window of timesteps, thus likely improving coherence over time. One downside of this approach is that GAN-based networks generally take longer to train and may not converge properly depending on many interacting factors.

One indirectly related, but nevertheless interesting work also uses (multi-head) self-attention for singing synthesis. The model proposed by Yi et al. (2019) includes a timbre model that is autoregressive, but the past timesteps are processed by a pre-net module that includes multi-head self-attention. They argue that this pre-net module helps to unify the distribution spaces of past timesteps of acoustic features on the one hand, and control input features on the other.

After the publication of our work, one notable new approach for generative models in general are based on diffusion probabilistic models (and denoising score matching), which has recently also been applied to singing synthesis (Liu et al., 2021a). A key aspect of this model is that at inference it gradually denoises an initial random sequence toward the target output, such as intermediate acoustic features. Thus, while this denoising process is feed-forward, it is executed in an iterative manner. However, the number of steps required is still much less than the number of steps required for autoregressive inference, and work is being done to make inference even more efficient. At the same time, this approach promises better temporal coherence, less oversmoothing, and overall better synthesis quality, compared to naive non-autoregressive approaches.

5.3 Experiments

5.3.1 Datasets

For the experiments in this work, we train a model on a proprietary dataset of 41 pop songs performed by a professional English male singer. From this dataset 35 songs were used for training (1 h 26 total), 4 for validation and 2 for testing.

The labeling of the dataset was done by first correcting the orthographic lyrics with respect to what was actually sung. Then a dictionary based transcription was used to obtain the initial phonetic transcription. The dataset was then segmented by first training a hidden semi-Markov model (HSMM) from scratch using the deterministic annealing expectation maximization (DAEM) algorithm on the audio data and initial phonetic transcription. Using this trained model to perform a forced alignment between transcription and audio results in initial phonetic segmentation. This segmentation was then corrected by hand, mainly to avoid gross segmentation errors. In a few cases the phonetic transcription was also corrected while correcting the phonetic segmentation.

We use the same intermediate acoustic features as in Chapter 4, that is WORLD vocoder features (see “Intermediate acoustic features” of §4.2.1). The only difference is that we use a lower frame rate (100 Hz rather than 200 Hz), for reasons mentioned in “Reduction factor” above.

5.3.2 Compared systems

In our experiments, we compare our proposed system to two baseline systems. The first is a version of our proposed system without self-attention layers. The second is an autoregressive model, similar to that proposed in §4.2, although also incorporating some aspects from our proposed system to be a fairer comparison.

FFT-NPSS-D This is our proposed non-autoregressive model. The identifier is derived from it being a version of our neural parametric singing synthesizer (NPSS) based on the FFT, and using phonetic durations as input (hence the "D"). Model hyperparameters are listed in Table 5.1 of §5.3.3.

FFT-NPSS-D-NoSA This system is identical to the FFT-NPSS-D system, except that it has no self-attention layers (hence the "NoSA" suffix). The idea is to have a baseline non-autoregressive system without self-attention, essentially a CNN (the FFT in the name is a bit of a misnomer, but used to indicate the architecture is equivalent except for lacking self-attention). This ablation study allows us to evaluate how much (or little) the self-attention mechanism contributes to the resulting sound quality. Model hyperparameters are listed in Table 5.1 of §5.3.3.

AR-NPSS This is an autoregressive baseline model, similar to that proposed in §4.2 (NPSS). However, the model and its hyperparameters have been slightly modified to be more similar to FFT-NPSS-D. Some notable changes include using more residual channels (same number as FFT-NPSS-D) and using a simple (mean) L_1 loss rather than the constrained Gaussian mixture (CGM) output distribution. To keep the total number of model parameters comparable to NPSS after the increase in residual channels, we no longer use parametrized residual skip connections[3]. Similarly, while FFT-NPSS-D uses a smaller kernel size for the initial causal convolution compared to NPSS, 1×1 rather than 10×1, by also using a bigger dilation cycle for the same total number of layers, the resulting receptive field is still comparable (actually bigger).

5.3.3 Model hyperparameters

The hyperparameters that we use in our experiments for our proposed systems (FFT-NPSS-D and FFT-NPSS-D-NoSA) are listed in Table 5.1. We use 64-dimensional acoustic features similar to (Blaauw and Bonada, 2017b), extracted with a 10 ms hop time. A reduction factor, $r = 2$, is used to further reduce the complexity of the attention layers. We use 256-dimensional phoneme embeddings, and an encoder with a single 3×1 GLU block with 64 channels. F0 is coarse coded to a 4-dimensional vector, as is the position within the note, albeit with a cyclical encoding. The decoder consists of six layers with (single-head) self-attention and 3×1 GLU blocks, all with 256 channels. The final output projection is to $64r$ channels. Dropout probability is set to 0.1 throughout the model. The learned standard deviation of the Gaussian

[3]Not using parametrized residual skip connections simply means removing the left "FC" (1×1) module after the gated activation in Figure 4.1, and ensuring $d_{\text{hid}} = d_{\text{res}}$. Especially when d_{res} is big, this notably reduces the number of model parameters, and is reported to result in no perceptual change in quality (Arik et al., 2017b).

bias of the self-attention blocks is initialized to 30. Initialization of convolutional layers follows (Gehring et al., 2017). We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1 \times 10^{-9}$, and a batch size of 32. We follow the learning rate schedule from (Vaswani et al., 2017), with a 4000-step linear warm-up, a base learning rate of 1×10^{-3}, and an inverse square root decay. We train for a total of 50 k updates, and use Polyak averaging, also known as exponential moving average (EMA) updates, with a decay of 0.995 for validation and testing (Polyak and Juditsky, 1992). The objective that we optimize is a simple (mean) L_1 loss between output and target features.

The hyperparameters for our autoregressive baseline (AR-NPSS) are listed in Table 5.2. Some of the principal differences between this model and NPSS from §4.2 are listed in §5.3.2 above.

5.3.4 Listening test

We ran a mean opinion score (MOS) listening test that corresponds to the experiments in this chapter, as well as those of Chapter 8, as these two chapters are based on what was a single publication originally. The listening test had 20 participants, which each rated a random subset of 11 out of 22 phases. For each phrase, the participant had to rate six stimuli for overall quality and naturalness, in accordance to a presented reference. The scale used as 0–-100, divided into 5 segments corresponding to a 5-scale MOS test. Three of the six stimuli to be rated correspond to systems described above, FFT-NPSS-D, FFT-NPSS-D-NoSA and AR-NPSS. Two other stimuli correspond to two additional systems (FFT-NPSS and FFT-NPSS-NoSA) that are relevant to the experiments of Chapter 8, but not to the experiments of this chapter. The final stimulus corresponds to a hidden reference. The visible and hidden references consist of a WORLD re-synthesis of the target recording. All systems are presented and rated together to encourage a comparison between them. In this listening test we did not use an anchor as a kind of low scoring (hidden) reference, mainly because preparing such stimuli in a way that are representative of bad quality TTS or singing synthesis, is not an easy task.

5.4 Results

The results of our listening test are shown in Table 5.3. We can see that our proposed non-autoregressive model with self-attention, FFT-NPSS-D, is rated best, with the autoregressive baseline, AR-NPSS, second by a significant margin. Rated worst of the compared system is the non-autoregressive model without self-attention, FFT-NPSS-D-NoSA, indicating that self-attention is a crucial component for the

Hyperparameter	
Feature dimensionality	60 + 4 (concat.)
Frame rate (Hz)	100
Reduction factor (r)	2
Phoneme embedding	
Dimensionality (d_{emb})	256
Initialization	$\mathcal{N}(0, 0.01^2)$
Encoder	
Num. layers (K_{enc})	1
Kernel size	3×1
Num. channels (d_{enc})	64
F0 encoding	4-dim. triangular
Pos.-in-phn. encoding	4-dim. cyclical
Decoder	
Num. layers (K_{dec})	6
Kernel size	3×1 (dilation = 1)
Num. channels (d_{dec})	256
Output linear projection (d_{out})	$64r$
Dropout probability	0.1
Attention	
Num. heads, d_{att}	1, 256
Init. learned scale of diag. bias (σ_{bias})	30.0
Batch size	32
Seq. len.	full
Learning rate schedule	Noam; 1×10^{-3}; 4000 step warm-up
Num. epochs (updates)	1000 (50,000)
EMA decay	0.995
Loss	L1
Receptive field (FFT-NPSS-D)	full
Receptive field (FFT-NPSS-D-NoSA)	13 frames (260 ms)

Table 5.1: Hyperparameters for the proposed non-autoregressive system (FFT-NPSS-D and FFT-NPSS-D-NoSA). Symbols within parentheses refer to those in Figure 5.1 and the corresponding equations in the text.

Hyperparameter	
Feature dimensionality	60 + 4 (concat.)
Control inputs	prev. phn. identity (one-hot) cur. phn. identity (one-hot) next phn. identity (one-hot) pos.-in-phn. (coarse) F0 (coarse)
Frame rate (Hz)	200
Reduction factor (r)	1
Input noise level (λ_{reg})	0.2
Initial causal convolution	1×1
Dilated convolutions	2×1
Num. layers (K)	5
Dilation cycle (num. layers)	5
Dilation factors	1, 2, 4, 8, 16
Receptive field (ms, frames)	160 (32)
Residual channels (d_{res})	256
Parametrized residual	no
Hidden channels (d_{hid})	256
Skip channels (d_{skip})	256
Output stage ($K_{out} = 1$)	ReLU $\rightarrow$ 1×1 (256) $\rightarrow$ ReLU $\rightarrow$ 1×1 (64) $\rightarrow$ 64× L1
Generation temperature (τ)	-
Batch size	32
Seq. len.	full
Learning rate (schedule)	3×10^{-4} (fixed)
Num. epochs (updates)	1000 (85,000)

Table 5.2: Hyperparameters for the autoregressive baseline system (AR-NPSS). Symbols within parentheses refer to those in Figure 4.1 and the corresponding equations in the text of Chapter 4. Coarse coding of input control features is described in Table 4.3 of the same chapter.

System	**Mean opinion score**
Hidden reference	4.56 ± 0.07
AR-NPSS	2.63 ± 0.10
FFT-NPSS-D (proposed)	$\mathbf{2.92 \pm 0.10}$
FFT-NPSS-D-NoSA (w/o self-attention)	2.53 ± 0.11

Table 5.3: Mean opinion score (MOS) for evaluating the proposed non-autoregressive timbre model. Ratings on a 1–5 scale with their respective 95% confidence intervals.

success of the model. When listening to the sound examples, we observe the non-autoregressive model without self-attention exhibits notable variations in timbre over time, often in a way that is not natural. For instance, there may be sudden differences in timbre along a note, or a sudden increase in energy just before a release. In our experience, we could not obtain the same result of self-attention by simply increasing the receptive field of the purely CNN variant (e.g., by using dilated convolutions or using more layers). One explanation for this may be that each of the multiple self-attention operations has a large receptive field, while a CNN typically consists of many smaller convolutions which together result in a potentially large receptive field, combined with simply being different operations altogether. In Figure 5.2 we try to visualize these differences by plotting waveforms and mel-spectrograms produced our CNN-based decoder without and with self-attention. While somewhat difficult to appreciate without listening (although sound examples are available in the supplemental material for this work[4]), just looking at the waveforms' envelope gives some indication of these issues.

It should be noted that these MOS scores can be considered relatively low. Both generally, in terms of state of the art in TTS, and compared to MOS scores from previous experiments. In particular, our previous autoregressive model, NPSS, scored 3.43 in §4.3.5, while the similar baseline system AR-NPSS only scored 2.63 here. It is difficult to say why this happens exactly, but we argue there can be a few causes. Firstly, it is quite possible that people's expectations have grown in the time that expired between the two listening tests (2017 to 2020). In particular, high-quality neural TTS has become much more commonplace, than it was. The fact that we use a heuristic vocoder rather than predicting mel-spectrogram features and using a neural vocoder, probably made this aspect worse (even though this arguably makes the comparison "fairer" because the vocoder component is more predictable). Secondly, results are obviously dependent on the datasets used, and perhaps more importantly the language used. The test in §4.3.5 used a Japanese voice, with only 8 out of the 40 participants being native or having a good understanding of the language. Here on the other hand, an English voice is used, and all participants are assumed to have a good grasp of this language. Arguably, Japanese is easier to model than English, and in our experience tends to be perceived as higher quality, at least by people not so familiar with the language. Lastly, there are some notable differences between the NPSS and AR-NPSS models. These may have some influence on the resulting sound quality, and overall the latter model was perhaps a little less aggressively optimized for sound quality, in order to have an autoregressive baseline that is very comparable to our proposed non-autoregressive model.

While the results of this listening test seem to indicate that non-autoregressive models with self-attention outperform autoregressive model, we should perhaps be careful to

[4]https://mtg.github.io/singing-synthesis-demos/supplemental/

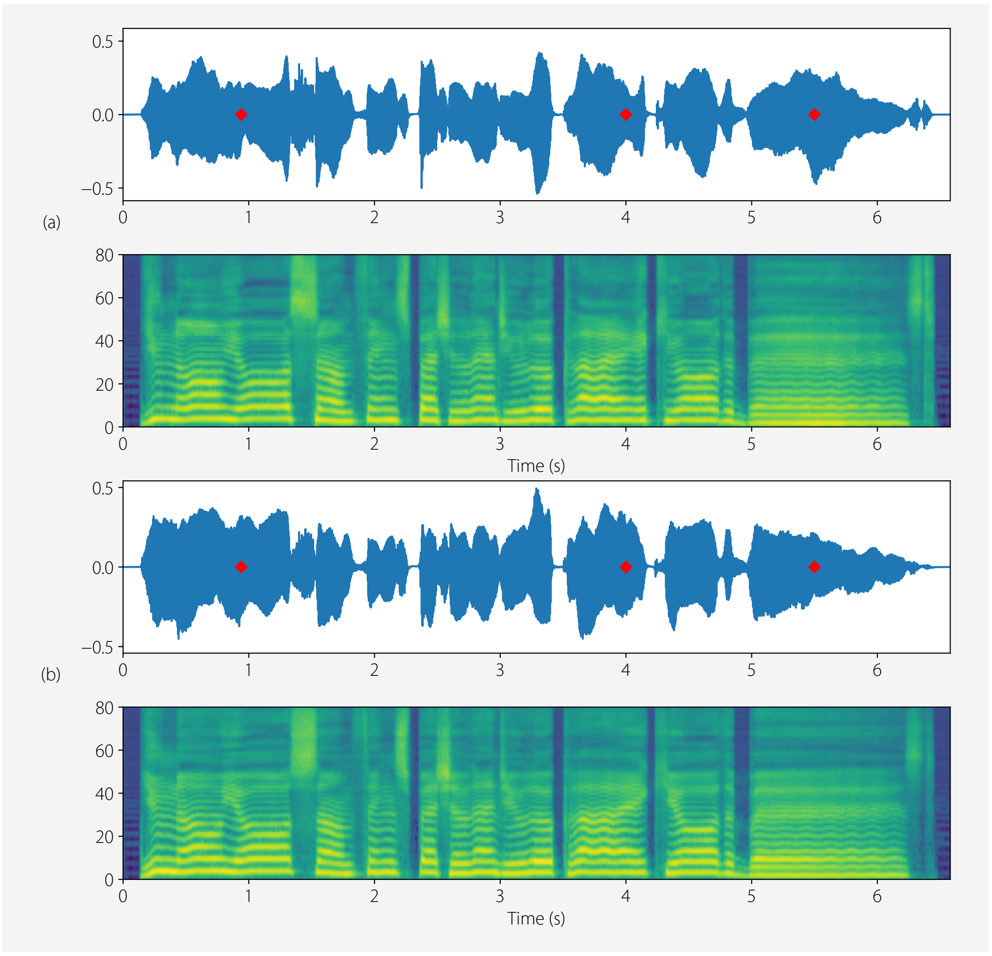

Figure 5.2: (a) Shows an example of the result of the non-autoregressive model without self-attention. (b) Shows the same example as a result of the non-autoregressive model with self-attention. While from this visual representation of waveform and mel-spectrogram alone it may be hard to appreciate the inconsistencies in timbre over time in the case self-attention is not used, the red diamond-shaped markers indicate some places where there are sudden shifts in levels, brightness, and so on. Notice in particular, in (b) a more consistent energy in the fundamental, and cleaner formant transitions compared to (a). These examples are available as sound files in the supplemental material of this work: https://mtg.github.io/singing-synthesis-demos/supplemental/

draw any generalized conclusions from these somewhat limited experiments. Notably, the experiments here are quite limited in terms of the number of voices used, number of participants, and even variations of autoregressive and non-autoregressive models compared. In our opinion, a more reasonable conclusion would be that non-autoregressive models, at least when using self-attention, can at least be competitive with autoregressive models. In our experience, the differences between these two approaches are diminished even further when the models predict mel-spectrogram features and a neural vocoder is used to produce the waveform. In particular, when a neural vocoder is used (especially an autoregressive neural vocoder), many of the details that may be missing from the predicted mel-spectrogram can be recovered in the produced waveform.

5.5 Conclusions

In this chapter, we have proposed a non-autoregressive model for generating singing voice timbre. Compared to autoregressive models this formulation has a number of advantages. Firstly, the inference no longer needs sequential operations, thus can be faster on highly parallel hardware such as GPUs. Secondly, it avoids issues related to exposure bias, such as unnaturally evolving resonances in long sustained vowels.

The network architecture we propose is based on a feed-forward version of the Transformer network. A key aspect of this network architecture is that it uses self-attention, alongside convolutions, as its primary building blocks. Self-attention is a fundamentally different operation compared to convolutional or recurrent operations, with the principal difference being that it can attend to any timestep in a sequence in a single operation. In our experiments, we found that this ability seems to help increase the coherence of the generated timbre over time, whereas a convolutional architecture without self-attention can produce unnatural variations in timbre over time. These observations are corroborated by the results of our listening tests.

Compared to a baseline autoregressive model, our proposed non-autoregressive model is rated better in listening tests, indicating that non-autoregressive models with carefully designed network architectures can at least be competitive with autoregressive models. Issues related to exposure bias, such as unnaturally evolving resonances in long sustained vowels are absent. Additionally, monitoring training progress is simplified. Inference speed is also significantly faster on GPU compared to the baseline.

Non-autoregressive source-filter neural vocoder | 6

One important disadvantage of autoregressive models is that inference requires sequentially computing each timestep, thus not being able to take advantage of modern highly parallel hardware, such as graphics processing units (GPUs). This is one of the reasons why we proposed a non-autoregressive timbre model in Chapter 5. However, for timbre models, lack of parallelization tends to be less problematic, as the predicted intermediate acoustic features tend to have a relatively low frame rate, and the networks themselves tend to be relatively small. Where autoregressive models tend to be truly problematic is when predicting waveform signals directly. Here, the sample rate tends to be high (e.g., 32 kHz) and as the signal itself is more complex, the networks used tend to have higher complexity (e.g., more layers, more channels). Thus, when combining a timbre model with a neural vocoder, using a non-autoregressive neural vocoder is often an attractive choice.

Another desirable property of neural vocoders is stability and interpretability. Although possible to mitigate, autoregressive models are inherently less stable than non-autoregressive models due to the recurrent feedback signal. In neural vocoders, this may cause artifacts such as clicks or "explosions". Similarly, if a model is interpretable, any issues with sound quality can be investigated and more easily corrected, compared to a more "black box" model. Unfortunately, interpretability often comes at the cost of introducing constraints in the model, which may ultimately limit the accuracy of the model.

The work in this chapter is as of yet unpublished.

6.1 Non-autoregressive neural vocoders

Research on non-autoregressive neural vocoders, also called parallel neural vocoders, is currently a very active topic, with many competing types of models being proposed. The most prominent of these are listed in §2.4.2 under "Probability density distillation", "Normalizing flows", "Diffusion probabilistic models" and "Generative adversarial networks".

All of these different types of models tend to obtain fairly good-quality results and inference speeds, but also have some weaknesses. Probability density distillation model (e.g., van den Oord et al., 2018; Ping et al., 2019) tends to be cumbersome to train. Models based on normalizing flows (e.g., Prenger et al., 2019; Kim et al., 2019) tend to require huge networks in order to obtain good results, which limits practical applicability. Diffusion probabilistic models (e.g., Chen et al., 2021b; Kong et al., 2021) still require sequential inference, although the number of iterations is independent of the sequence length. Arguably, the approach that has the most desirable properties is based on generative adversarial networks (GANs) (e.g., Kumar et al., 2019; Yamamoto et al., 2020; Bińkowski et al., 2020). This method allows single-pass training, places almost no constraints on the network architectures used, and has single-step inference. That said, some general drawbacks of adversarial models are that training can be unstable, there are many interacting "moving parts", and that training generally requires many steps to converge.

6.1.1 Adversarial neural vocoders for singing

Neural vocoders based on GANs are generally close to state of the art in terms of sound quality for speech. That is, the results obtained are generally very close to those of autoregressive models in listening tests. However, applying these models to singing voice in our experience results in a number of artifacts that may not exist or go unnoticed when modeling speech.

Some common artifacts include

Discontinuities One notable issue is that these models can produce discontinuities, in particular in long, sustained vowels. Of course, the duration of vowels is one of the key differences between speech and singing.

Phasiness The synthesized sound can exhibit what is generally referred to a "phasiness" (e.g., Laroche and Dolson, 1997). This is especially noticeable in the lower end of pitches, lower than most text-to-speech (TTS) voices.

Quasi-periodic pulses and "metallic" artifacts Unvoiced fricatives, affricates and aspirations may show quasi-periodic pulses. These pulses can be perceived as a pitched, "buzzy" sound. In other cases, the same sounds can have a certain "metallic" quality that sounds unnatural (i.e., something akin to inharmonic partials within unvoiced noise).

6.2 Proposed system

Finding exact causes of the above problems can be hard as neural networks tend to be somewhat of a “black box”. We can certainly think of reasons why these things may be happening, and try to address these issues. However, in our experience, it seems that certain symptoms may be caused by a number of different underlying issues, and whether these occur or not can depend on stochastic optimization processes involving millions of weights, competing networks (a generator and one or more discriminators), multiple losses, and so on. Thus, a common scenario with very flexible models is that we address a certain symptom somehow, the symptom may go away entirely or occur only very rarely, but some additional change in something seemingly unrelated may cause the symptoms to re-appear.

Thus, in order to avoid the above issues in a more systematic way, we propose a much more constrained, and in certain ways less powerful, model based on the source-filter model of speech production (see §2.5.1). Our model is very similar to conventional parametric vocoders, such as STRAIGHT (Kawahara et al., 1999; Kawahara, 2006) or WORLD (Morise et al., 2016), with certain signal processing blocks replaced by neural networks.

Besides viewing this approach as a way to avoid common artifacts of non-autoregressive neural vocoders, we can also view it as a way to improve upon weaknesses of traditional vocoders. In our experience, for instance WORLD is generally able to obtain high-quality results, however it has the following weaknesses 1) poor frequency resolution of the aperiodic component of the voice (especially below 3 kHz), 2) being very sensitive to voiced/unvoiced errors of the F0 estimator used, and 3) the inability to recover from oversmoothed predicted parameters when used in the context of TTS or singing synthesis. By replacing certain components by neural networks, we should be able to improve all of these aspects.

An overview of our model is depicted in Figure 6.1. The input to the model is a mel-spectrogram and corresponding F0 sequence. While it should generally be possible to infer F0 from the mel-spectrogram alone, we skip this step, as we assume that ultimately our vocoder will be used together with a pitch model within a singing synthesizer. From the input F0 a periodic excitation is generated. This periodic excitation is combined with an aperiodic component to obtain a voice source signal. This signal is then filtered by a vocal tract filter. The aperiodic envelope and the vocal tract filter are predicted from the mel-spectrogram using a neural network.

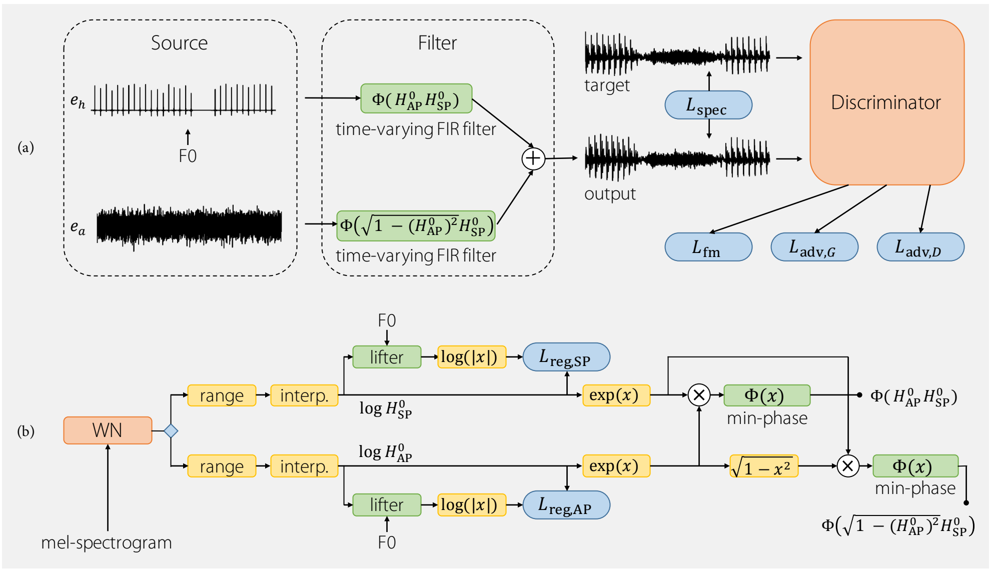

Figure 6.1: A diagram of the proposed non-autoregressive source-filter neural vocoder. (a) Shows the generator, consisting of a source and a filter module, and the discriminator. (b) Shows the filter prediction module. Here, the “WN” block denotes a non-causal WaveNet. Note that L_{reg} in Equation (6.11) corresponds to the sum of $L_{\mathrm{reg,SP}}$ and $L_{\mathrm{reg,AP}}$ in this diagram.

6.2.1 Periodic and aperiodic excitation

We argue that one of the reasons for discontinuities in the output waveform is that in certain cases the network has no reasonable way to predict the phase of the harmonics. The input to most GAN-based neural vocoders for TTS is a mel-spectrogram, or a mel-spectrogram and a noise sequence[1]. If we imagine a simplified voice signal as just a fundamental sinusoid with constant amplitude, but varying frequency, determining the phase of this sinusoid at the given output timestep would require the initial phase of the sinusoid, as well as the entire trajectory of frequencies leading up to that timestep. In speech or singing, such trajectories would start at every unvoiced to voiced boundary, e.g., whenever there is a silence or unvoiced phoneme. Assuming that the generator network is able to infer harmonic frequencies from the input spectrogram, and that the

[1]Typically, the noise input will have the same dimensionality as the output. This can be interpreted as the generator network converting a sample from a simple, e.g., Gaussian, distribution to a sample from the complex data distribution. In a neural vocoder, this will typically be the data distribution conditioned on the input mel-spectrogram. Thus a different interpretation can be that the noise input helps to generate stochastic components of the waveform from the less stochastic input mel-spectrogram. That said, models that omit this noise input altogether tend to also be able to generate seemingly aperiodic voice components.

initial phases are fixed to some values, the network should be able to produce coherent phases.

For GAN-based neural vocoders applied to TTS discontinuities are generally not considered an important issue. Thus, we can assume that given a powerful enough network, and enough data, the network should be able to infer phases, even for unseen input mel-spectrograms. As the same does not hold true for singing voice (e.g., Chen et al., 2021a), we argue that the reason for this may be that the receptive field of the typical generator networks is insufficient to capture the entire F0 trajectory from the last unvoiced-voiced boundary in singing voice. This is due to a fundamental difference between speech and singing voice, where voiced sections can be much longer, notably due to long, sustained vowels. It is also not unlikely that these issues also occasionally arise in TTS, but much less frequently, and possibly masked by the rapid succession of phonemes in speech.

A common solution to this issue is to provide a periodic signal derived from F0 as an input to the generator network (in addition, or instead of the noise input). This approach was initially proposed for speech (Wang et al., 2019), albeit with the aim of providing a fast feed-forward alternative for autoregressive neural vocoders. Later, neural vocoders for singing also incorporated this approach (Liu et al., 2021a, in footnote) (Chen et al., 2021a; Hono et al., 2021b; Roebel and Bous, 2021). The exact excitation signal used varies a little, e.g., only the fundamental (Hono et al., 2021b), the eight first harmonics (e.g., Wang et al., 2019; Chen et al., 2021a), or cyclic decaying noise (Wang and Yamagishi, 2020).

In our model, we first upsample the input F0 sequence to the output sample rate by repeating values, as a simple way to avoid interpolating F0 between voiced and unvoiced frames. Then, given this upsampled input F0 sequence, $\mathbf{f}_0 = [f_{0,1}, f_{0,2}, \dots, f_{0,T}]$, we generate a periodic excitation signal, $\mathbf{e}_p = [e_{p,1}, e_{p,2}, \dots, e_{p,T}]$, by summing all harmonics up to Nyquist for voiced timesteps, and silence for unvoiced timesteps,

$$e_{p,t} = \begin{cases} g_p \sum_{k=1}^{K} \cos\left(2\pi k \sum_{i=i_{<t}}^{t} \frac{f_{0,i}}{f_s} + \phi\right) \left(k f_{0,t} < \frac{f_s}{2}\right) & \text{for } f_{0,t} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Here, $K = 250$ is the maximum number of harmonics, $i_{<t}$ is the timestep corresponding to the first unvoiced-voiced boundary to the left of t (or $i_{<t} = 1$, if there is none), $\phi = \pi$ is the initial phase (selected somewhat arbitrarily), and g_p is a scaling factor.

We also generate an aperiodic excitation signal, $\mathbf{e}_a = [e_{a,1}, e_{a,2}, \dots, e_{a,T}]$, by simply sampling a Gaussian distribution,

$$e_{a,t} \sim g_a \, \mathcal{N}(0, 1), \tag{6.2}$$

where g_a is a scaling factor.

6.2.2 Minimum-phase vocal tract filter and aperiodicity

In our experience, concatenating $\mathbf{e}_a$ and $\mathbf{e}_p$, and feeding this excitation signal to a generator network significantly reduces discontinuities, but does not fully solve the issue. We argue, that while the losses used to train the generator should penalize discontinuous output, this is still not enough to ensure that there never are sudden changes in phase, especially for unseen mel-spectrogram inputs. On a fundamental level, the issue is mainly that generating artifact-free waveform is still a difficult task for feed-forward models.

To further aid the network, we decide to apply a vocal tract filter to the excitation signal, following the source-filter model of speech production (see §2.5.1). While less powerful than directly generating waveform, it should be considerably easier for a network to produce a reasonable, smooth vocal tract filter. Once we predict a zero-phase filter (i.e., magnitude response), H^0, we can obtain a corresponding minimum-phase filter, H, by ensuring its cepstrum is causal (Smith, 2011, Chap. 4.9, "Minimum-Phase and Causal Cepstra", p. 147), i.e.,

$$C^0 = \mathcal{F}^{-1}\left(\log H^0\right), \tag{6.3}$$

$$C(n) = \begin{cases} 0 & \text{for } n < 0 \\ C^0(n) & \text{for } n = 0 \\ 2C^0(n) & \text{for } n > 0, \end{cases} \tag{6.4}$$

$$H = \exp \mathcal{F}\left(C\right), \tag{6.5}$$

where C^0 is the cepstrum of the zero-phase filter, and C is the cepstrum of the corresponding minimum-phase filter, obtained by "flipping" the anti-causal part onto the causal part. Thus, if we assume our network predicts a vocal tract filter with a magnitude response that is smooth over frequency and smoothly varies over time, the corresponding minimum-phase response derived from this magnitude response will also be smooth and smoothly varying over time, thus making discontinuities due to rapid variations in phase response unlikely.

Combining periodic and aperiodic components

Following traditional vocoders like WORLD (Morise et al., 2016), we model our target signal as a combination of periodic and aperiodic components. The ratio between (the power spectra of) periodic and aperiodic components for any given frequency is determined by an aperiodicity filter. Thus, we can describe our model in the frequency domain as,

$$X = E_p \Phi(\sqrt{1 - (H^0_{\text{AP}})^2} H^0_{\text{SP}}) + E_a \Phi(H^0_{\text{AP}} H^0_{\text{SP}}), \tag{6.6}$$

where X is the output spectrum, E_p the periodic excitation spectrum, E_a the aperiodic excitation spectrum, H^0_{SP} and H^0_{AP} the zero-phase vocal tract and aperiodicity filters, and $\Phi(\cdot)$ the process of turning a zero-phase filter into a minimum-phase filter following Equations (6.3)–(6.5).

Note that the relative scaling factors used in the excitation signal g_p and g_a, while not affecting the output sound quality significantly, have some effect on the aperiodicity filter predicted. In our model, we empirically set these to $g_p = 0.177$ and $g_a = 1$ so the predicted aperiodicity is closer to that of the WORLD vocoder.

Network and filters

In order to predict H^0_{SP} and H^0_{AP} we use a network with a non-causal WaveNet architecture (van den Oord et al., 2016a), which takes as input the given mel-spectrogram. In this case, the network has a short receptive field, but a moderately high capacity in terms of the number of layers and the dimensionality of the hidden layers (all hyperparameters are listed in Table 6.1). While predicting these filters from given a mel-spectrogram seems like a relatively easy task, we have to keep in mind that this is the only neural network component in our entire model. Thus, any improvement over traditional vocoders comes mainly from this network. Additionally, this network operates at the frame rate of the input mel-spectrogram, not the output waveform. Therefore, we opted for a moderately powerful network for this task.

The outputs of this network are the normalized log magnitude responses for the vocal tract filter and the aperiodicity filter. These normalized responses are then scaled to an appropriate range, e.g., −140 to 20 dB for the vocal tract filter, and −60 to 0 dB for the aperiodicity filter. To reduce the number of values that have to be predicted and to ensure some degree of smoothness, we predict these filters with a smaller dimensionality (e.g., 257) and then linearly interpolate them to the target dimensionality (e.g., 1025). An example of the vocal tract and aperiodicity filters predicted by our neural network is shown in Figure 6.2.

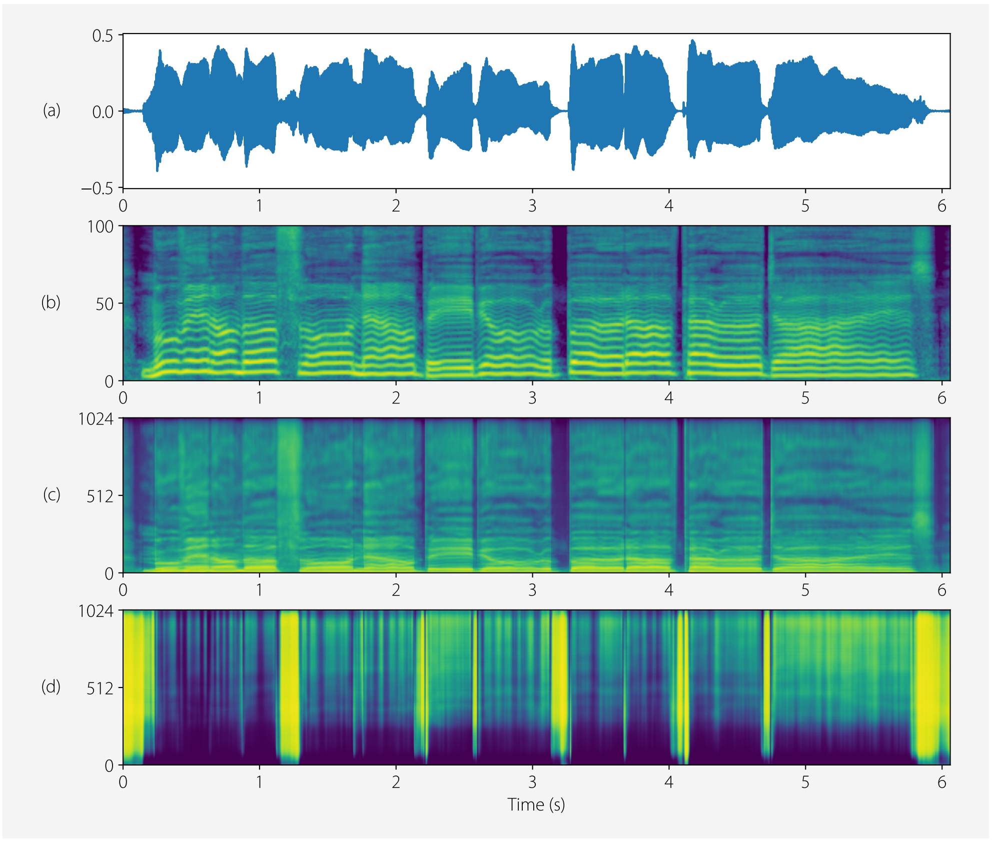

Figure 6.2: Example of the filters predicted by our source-filter neural vocoder. Shown here are; (a) the waveform, (b) log mel-spectrogram, (c) vocal tract filter log H^0_{SP}, and (d) (squared) aperiodicity filter $(H^0_{\text{AP}})^2$. For reference, the audio here matches that of the example of WORLD features obtained by signal processing algorithms, shown in Figure 3.5.

It should be noted that we did experiment with some additional neural components within our model. For instance, transforming the excitation signal, transforming the output, or predicting residual signals. In all these cases, either the improvement from such components turned out to be rather lackluster, or performance was even degraded[2].

To apply the time-varying finite impulse response (FIR) filters, we use the overlap-save algorithm, implemented in the fast Fourier transform (FFT) domain (Wefers and Vorländer, 2014). The cross-fading functions we used are $\sin^2$ and $\cos^2$ (equivalent to a Hann window and a shifted Hann window).

Additional regularization by cepstral liftering

We apply some additional regularization to ensure the predicted filters are smooth by applying two different ceptral domain lifters. The first, following Morise (2015), is a F0-adaptive sinc lifter,

$$l_{\text{sinc},t}(n) = \begin{cases} 0 & \text{for } n < 0 \\ 1 & \text{for } n = 0 \\ \dfrac{\sin(\pi f_{0,t} n)}{\pi f_{0,t} n} & \text{for } n > 0. \end{cases} \tag{6.7}$$

The second lifter is a general low-time lifter with a cosine taper to reduce high quefrency content,

$$l_{\text{taper},n_c}(n) = \begin{cases} 0 & \text{for } n < 0 \\ \cos(\frac{1}{2}\pi n / n_c) & \text{for } n \geq 0 \text{ and } n < n_c \\ 0 & \text{for } n \geq n_c, \end{cases} \tag{6.8}$$

where n_c is the cut-off quefrency expressed as an integer index.

Using these two lifters, we generate smoothed versions of the vocal tract and aperiodicity filters, H^*_{SP} and H^*_{AP}. Whenever F0 is voiced, the l_{sinc} lifter is applied to both H_{SP} and H_{AP}. This has a smoothing effect, equivalent to convolving the filter's (log) magnitude response with a rectangular window, and additionally has zeros at the quefrencies corresponding to the harmonics, thus avoiding the harmonic structure of the signal being present in the vocal tract filter. At the same time, high quefrency content is cut

[2]For instance, transforming the excitation signal would often cause a notable phasiness at low pitches. This may be because the transformation would result in an excitation signal that was no longer zero-phase, most likely with a phase response that varied over time.

beyond 10 ms and 1.6 ms for H_{SP} and H_{AP} respectively using l_{taper}. For unvoiced frames, we only apply l_{taper} to cut content beyond 1.6 ms for both H_{SP} and H_{AP}. That is,

$$l_{\mathrm{SP},t}(n) = \begin{cases} l_{\mathrm{sinc},t}(n) l_{\mathrm{taper},n_{\mathrm{high}}}(n) & \text{for } f_{0,t} > 0 \\ l_{\mathrm{taper},n_{\mathrm{low}}}(n) & \text{otherwise,} \end{cases} \tag{6.9}$$

$$l_{\mathrm{AP},t}(n) = \begin{cases} l_{\mathrm{sinc},t}(n) l_{\mathrm{taper},n_{\mathrm{low}}}(n) & \text{for } f_{0,t} > 0 \\ l_{\mathrm{taper},n_{\mathrm{low}}}(n) & \text{otherwise,} \end{cases} \tag{6.10}$$

where $n_{\mathrm{high}} = \lfloor 10 f_s / 1000 \rfloor$ and $n_{\mathrm{low}} = \lfloor 1.6 f_s / 1000 \rfloor$ are the cut-off quefrencies used.

Rather than applying these lifters directly to the vocal tract and aperiodicity filters used to generate the output waveform, we use them indirectly as part of a regularization loss. If we apply the lifters directly, the predicted filters, H^0_{SP} and H^0_{AP}, tend to be noisy, which we argue may result in poorer generalization compared to predicting smooth filters directly. Additionally, the lifters would place certain "hard" constraints on the lifters, i.e., forcing certain quefrencies to zero, which may not be beneficial. In particular, the lifters were designed empirically, and it is hard to tell if these designs lead to results that are close to optimal in all cases. Instead, we compute a regularization loss that tries to ensure the difference between the predicted filters, $\log |H^0_{\mathrm{SP}}|$ and $\log |H^0_{\mathrm{AP}}|$, and their smoothed (liftered) counterparts, $\log |H^*_{\mathrm{SP}}|$ and $\log |H^*_{\mathrm{AP}}|$, is small,

$$L_{\mathrm{reg}} = \frac{1}{N} \Big[\, \| \log |H^0_{\mathrm{SP}}| - \log |H^*_{\mathrm{SP}}| \, \|_2^2 + \| \log |H^0_{\mathrm{AP}}| - \log |H^*_{\mathrm{AP}}| \, \|_2^2 \, \Big], \tag{6.11}$$

where N is the filter length. Here, we assume this error will go towards zero the smoother the filters get. While this implies that the error reaches zero once the filter is maximally smooth (i.e., just a constant), we observe that during training the regularization loss tends to stay relatively constant at a non-zero value after a certain amount of updates. We argue that this happens because predicting overly smooth filters is naturally penalized by the other competing losses used to train the network.

Avoiding upsampling operators

One advantage of our approach is that it does not require any learned upsampling operators, this contrary to most GAN-based neural vocoders, that typically bridge the significant difference in sample rates between the input mel-spectrogram and the output waveform through such operators. While very powerful and necessary for many applications, these operators can easily be a source of artifacts, depending on many different factors, and may not be sufficiently penalized by the loss functions used. In the case of GAN-based neural vocoders, we argue that upsampling operators may be the culprit behind certain types of artifacts, especially those affecting unvoiced fricatives (see Pons et al., 2021, for a general discussion of upsampling artifacts in audio synthesis). While there are many ways to improve the performance of upsamplers, such as careful selection of the kind of upsampler used and its hyperparameters (e.g., Odena et al., 2016; Shi et al., 2016a; Shi et al., 2016b), special initialization of the learned kernel (Aitken et al., 2017), ensuring smoothness of the learned kernel (Sugawara et al., 2019; Kinoshita and Kiya, 2020), and so on, in our experience, it is very difficult to ensure no artifacts will occur.

6.2.3 Spectral losses

As is common for GAN-based neural vocoders, besides the adversarial loss we use several auxiliary losses to help the model converge and generally obtain better results. The most important of these are multi-resolution short-time Fourier transform (STFT) losses (e.g., Arik et al., 2019; Yamamoto et al., 2020). Because of the strong constraints of our model, we can even obtain fairly good results training using these losses only (in which case training is very fast).

The auxiliary multi-resolution STFT loss between target signal $\mathbf{x}$ and predicted signal $\hat{\mathbf{x}}$ is defined as,

$$L_{\text{spec}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{|M|} \sum_{m \in M} \left[\lambda_{\text{mag}} L^m_{\text{mag}}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda_{\text{sc}} L^m_{\text{sc}}(\mathbf{x}, \hat{\mathbf{x}}) \right], \tag{6.12}$$

where M is a set of spectral analysis parameters (window size, hop size, FFT size), and $L^m_{\text{mag}}(\mathbf{x}, \hat{\mathbf{x}})$ and $L^m_{\text{sc}}(\mathbf{x}, \hat{\mathbf{x}})$ are *log STFT magnitude* and *spectral convergence* losses respectively, combined using weights λ_{mag} and λ_{sc}. The log magnitude loss is defined as,

$$L^m_{\text{mag}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{T_m N_m} \| \log |\text{STFT}_m(\mathbf{x})| - \log |\text{STFT}_m(\hat{\mathbf{x}})| \, \|_1, \tag{6.13}$$

where $\| \cdot \|_1$ denotes the L_1 norm, and $|\text{STFT}_m(\cdot)|$ computes a $T_m \times N_m$ magnitude spectrogram, as a result of using spectral analysis parameters m. The spectral convergence loss is defined as,

$$L^m_{\text{sc}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{\| \, |\text{STFT}_m(\mathbf{x})| - |\text{STFT}_m(\hat{\mathbf{x}})| \, \|_F}{\| \, |\text{STFT}_m(\mathbf{x})| \, \|_F}, \tag{6.14}$$

where $\| \cdot \|_F$ denotes the Frobenius norm. Note that the spectral convergence loss is equivalent to a normalized root mean squared error (RMSE) between predicted and target magnitude spectrograms, which, in our experience, helps both to converge faster and obtain slightly better end results. While in our case we average these losses over the different samples in the minibatch during training, depending on the hyperparameters used (e.g., short sequence lengths), it may be beneficial to compute the Frobenius norm over all tensor dimensions (i.e., the L_2 norm over the flattened tensor), including the batch dimension, so that the normalizing factor is more constant and training more stable. Similarly, we should clamp magnitude values under some small threshold. We empirically set $\lambda_{\text{sc}} = 0.5$ and $\lambda_{\text{mag}} = 1$, as while the spectral convergence loss is generally beneficial, in our experience, it does have a tendency to increase the likelihood that a given model produces artifacts, in particular in unvoiced fricatives.

6.2.4 Adversarial training

We use mostly the same discriminators and adversarial losses from HiFi-GAN (Kong et al., 2020). This model uses two types of discriminators, a multi-scale discriminator, and a multi-period discriminator, each consisting of several sub-discriminators. The sub-discriminator of the multi-scale discriminator follows MelGAN (Kumar et al., 2019), a stack of 1-d strided convolutions, with relatively large kernel sizes, and an increasing number of channels, with grouped connectivity to keep the number of parameters reasonable. The output of this discriminator is scalar value indicating whether the input is "real or fake" for a sequence of overlapping windows on the input waveform (Isola et al., 2017). The original HiFi-GAN multi-scale discriminator uses three sub-discriminators at the original sample rate, halve the sample rate and one forth the sample rate, respectively. The sub-discriminator of the multi-period discriminator first reshapes the input waveform to a 2-d signal, assuming some integer period P. That is, the time dimension will be divided by P, the first row consisting of every P-th sample, and subsequent rows the following offsets into the period. Then, a 2-d convolution with $k \times 1$ is applied to this signal, thus sharing weights between the different offsets into the period, and steps over (decimated) time. Otherwise, the network design is similar to that of the multi-scale sub-discriminator, except using smaller kernel sizes and not using grouped connectivity. The original HiFi-GAN multi-period discriminator uses periods $P \in [2, 3, 5, 7, 11]$.

The discriminators we use in our model only differ by removing some of the sub-discriminators. We argue that because in our model things like periodicity are already ensured by the excitation signal, the adversarial training will mainly be beneficial to more time-localized detail. Thus, we use a multi-scale discriminator that operates on the original sample rate and halve sample rate, and a multi-period discriminator that uses periods $P \in [2, 3, 5]$. In this case, the biggest receptive field among the sub-discriminators will be around 250 ms, which we argue should be sufficient. While the discriminators do not affect inference speed, they do have a significant effect on training time and batch size.

The adversarial losses are based on the LS-GAN formulation (Mao et al., 2017),

$$L_{\text{adv},G} = \sum_k \frac{1}{T_k} \sum \left(D_k(G(\mathbf{c})) - 1\right)^2, \tag{6.15}$$

for the generator, and,

$$L_{\text{adv},D} = \sum_k \frac{1}{T_k} \sum \left[\left(D_k(\mathbf{x}) - 1\right)^2 + \left(D_k(G(\mathbf{c}))\right)^2\right], \tag{6.16}$$

for the discriminators, where D_k is the k-th sub-discriminator, G is the generator, $\mathbf{x}$ is the target waveform, $\mathbf{c}$ are the conditioning features (mel-spectrogram and F0), and T_k is the number of output timesteps resulting from the k-th sub-discriminator.

Also following HiFi-GAN, we use a feature matching loss, which tries to make sure the hidden features of the discriminators are similar between corresponding real and fake inputs,

$$L_{\text{fm}} = \sum_k \sum_i \frac{1}{N_{k,i}} \parallel D_k^i(\mathbf{x}) - D_k^i(G(\mathbf{c})) \parallel_1, \tag{6.17}$$

where $N_{k,i}$ is the dimensionality of the output feature map of the i-th hidden layer within the k-th sub-discriminator.

The final losses can thus be computed as,

$$L_G = L_{\text{adv},G} + \lambda_{\text{spec}} L_{\text{spec}} + \lambda_{\text{fm}} L_{\text{fm}} + \lambda_{\text{reg}} L_{\text{reg}}, \tag{6.18}$$

for the generator, and,

$$L_D = L_{\text{adv},D} \tag{6.19}$$

for the discriminator. Considering the recommendations proposed in HiFi-GAN, and adjusting for inclusion of the spectral convergence loss and smoothness regularization loss, we empirically set the weights to $\lambda_{\text{spec}} = 36$, $\lambda_{\text{fm}} = 2$, and $\lambda_{\text{reg}} = 36$.

6.3 Relation to prior work

While developed independently, our model shares many similarities with the Multi-Band Excited WaveNet (MBExWN) model (Roebel and Bous, 2021). The main differences being that this model transforms the periodic source with an efficient multi-band variant of a non-causal WaveNet conditioned on the mel-spectrogram, while we do not. This transformation also provides the aperiodic component of the signal, thus not requiring the prediction of an aperiodicity filter. The vocal tract filter is predicted directly in the cepstral domain, with a reduced number of coefficients (equivalent to a rectangular lifter). Additionally, this model predicts F0 from the mel-spectrogram during inference (but does require F0 for training), as well as several other minor differences.

The idea of using source-filter models in neural vocoders is not new. Some proposed models use excitations generated by neural networks, e.g., in LPCNet (Valin and Skoglund, 2019), iLPCNet (Hwang et al., 2020a), LP-WaveNet (Hwang et al., 2020b) or GELP (Juvela et al., 2019), others use periodic excitations similar to our model, e.g., in NSF (Wang et al., 2019; Wang et al., 2020b; Wang and Yamagishi, 2019; Wang and Yamagishi, 2020) or uSFGAN (Yoneyama et al., 2021), and for singing, e.g., in DiffSinger (Liu et al., 2021a, in footnote), SingGAN (Chen et al., 2021a), or PeriodNet (Hono et al., 2021b). Most of these models generate an aperiodic component through a neural network that transforms the source excitation, whereas our model predicts an aperiodicity filter, similar to traditional parametric vocoders such as WORLD (Morise et al., 2016). As our aperiodicity filter is predicted by a neural network it is arguably more powerful than signal processing approaches, that often lack adequate frequency resolution. Our approach arguably also is more powerful than predicting a single voicing frequency (e.g., Wang and Yamagishi, 2019), or band-wise aperiodicity (e.g., Hwang et al., 2021). On the other hand, while one could argue using a neural network transformation is ultimately the most powerful approach, we found that this can also be a source of artifacts that can be difficult to mitigate.

Most, although not all, of these models also include adversarial training. In most GAN-based neural vocoders, the generator will produce an audio rate signal from an input mel-spectrogram at a much lower frame rate. Approaches tend to either first upsample the mel-spectrogram and then use this to condition a network operating at the audio rate (e.g., Yamamoto et al., 2020), or take a more multi-scale approach, where the mel-spectrogram is gradually upsampled and transformed through different layers of the generator network (Kumar et al., 2019; Bińkowski et al., 2020; Kong et al., 2020, e.g.,). Multi-band approaches can be used to improve the efficiency of this process (e.g., Okamoto et al., 2018; Yu et al., 2019; Yang et al., 2021; Tian et al., 2020b; Cui et al., 2020). Our model is unique in that it does not require upsampling operators which can introduce artifacts (see “Avoiding upsampling operators” in §6.2.2).

6.4 Experiments

6.4.1 Datasets

We are mainly interested in a so-called "universal" vocoder (e.g., Lorenzo-Trueba et al., 2019; Jang et al., 2020; Jiao et al., 2021; Huang et al., 2021; Roebel and Bous, 2021). In this case, the model is trained on a multi-singer dataset and should be able to generalize to unseen voices, possibly including unseen languages and singing styles. From a practical point of view, this is a significant advantage over a singer-dependent model, as we do not have to train a dedicated vocoder for each voice we want to use. Additionally, this allows using a larger amount of total data, and also provides a solution for cases where data is scarce, such as for voice cloning (see Chapter 7).

As neural vocoders can be trained from only audio, without any annotations, we train our model on a relatively large, multi-singer dataset sourced from a variety of other, proprietary datasets. The common denominator among these datasets is that recording conditions are similar (studio recordings), and singers are professional level. In total the dataset includes 75 singers, spanning 39 h 33, divided in 31,148 short phrases. Languages include English, Japanese, Spanish, Catalan, and German. Singing styles include pop and choir singing. Recording styles include both natural singing and pseudo singing (see §3.2.2). Thus, this dataset is fairly varied. However, the data is not balanced among the different categories, e.g., some singers will only contribute a single song, while others over two hours.

The loudness of all songs in the dataset are normalized using an ITU-R BS.1770 meter (ITU-R Recommendation BS.1770-4, 2015), prior to segmentation into phrases. In order to be more robust to varying input levels, we also apply some random gain data augmentation during training (see Table 6.1). We estimate F0 prior to training using the SAC algorithm (Villavicencio et al., 2015). Mel-spectrograms are computed during training, using the hyperparameters listed in Table 6.1. It should be noted that we use fairly high resolution (in time and frequency) mel-spectrograms. In our experience, such a high resolution can be especially beneficial to capture non-modal aspects of singing voice, such as sub-harmonics. While our model is mainly focused on producing model singing voice, we feel that for comparing different systems, these kinds of features are most appropriate.

While left as future work, we expect a singer-dependent version of our model can be successfully trained on a relatively small amount of data, due to the high degree of domain knowledge included in the model itself.

6.4.2 Compared systems

In our experiments, we compare our proposed system to several baseline systems, including a traditional vocoder, an autoregressive neural vocoder, and another non-autoregressive GAN-based neural vocoder. Additionally, we perform a small ablation study to investigate the importance of adversarial training.

NeuralWORLD This system is our proposed non-autoregressive source-filter neural vocoder, as described above.

NeuralWORLD-NoAdv This system is a variation of our proposed model that does not use adversarial training. That is, no discriminator is used and $L_{\mathrm{adv},G}$, and L_{fm} are not used to train the generator. This is a kind of ablation study to see whether adversarial training has a notable effect on sound quality. While using adversarial training or not does not affect inference speed, in our model the discriminator is by far the most demanding component in terms of computational complexity, and thus discarding it significantly reduces training time (down to a few hours).

Excited-PWG This system is a baseline system consisting of a slightly modified version of Parallel WaveGAN[3] (Yamamoto et al., 2020), where the input is the excitation signal of our proposed model (concatenating the pulse train and noise channels), rather than simply noise. Furthermore, we try to use all applicable hyperparameters from Table 6.1, such as multi-resolution STFT loss parameters. Notable exceptions are that we use $\lambda_{\mathrm{spec}} = 1/4$ and a minibatch consisting of 8 1.0 s samples, following the original Parallel WaveGAN. We upsample the input mel-spectrogram using upsample ratios [5, 4, 4, 2] to accommodate the modified audio sample rate and mel-spectrogram hop time. This system serves as a baseline GAN-based non-autoregressive neural vocoder. In particular, we would like to know whether our system improves over this system by mitigating certain artifacts (see §6.1.1). On the other hand, this system may also improve over our proposed system, due to allowing for a more powerful neural transformation of the excitation signal, rather than just a simple time-varying filter. This model is trained to 500 k steps, with the first 100 k steps training just the generator, without adversarial loss.

Note that while we only compare a single GAN-based baseline system, we empirically found the same artifacts we discuss consistently across different generator and discriminator architectures (e.g, Kumar et al., 2019; Bińkowski et al., 2020).

AR-WNV This system is a baseline autoregressive neural vocoder. We use an internal implementation of a model derived from WaveNet (van den Oord et al., 2016a),

[3]Code adapted from: https://github.com/kan-bayashi/ParallelWaveGAN

which has been heavily tuned for good performance on a wide range of singing voices. This model is trained to around 540 k steps.

WORLD This system is a baseline traditional parametric vocoder, based on signal processing and heuristics. We use the version of WORLD (Morise et al., 2016) that uses the D4C algorithm for aperiodicity estimation Morise, 2016, combined with our own F0 estimator SAC (Villavicencio et al., 2015), which we also use in all the other systems. This system serves as a kind of lowest baseline, upon which we expect all neural vocoders to improve. Note that this is the vocoder used in many of the other chapters, in part due to its predictable behavior making it suitable for comparing different systems.

6.4.3 Model hyperparameters

While several model hyperparameters are already listed in the text, a more complete listing is given in Table 6.1.

6.4.4 Listening test

We ran a mean opinion score (MOS) listening test to compare the different systems. The listening test had 12 participants, all active in the field of music and sound technology, including several specializing in speech and singing. Each of the participants was presented 14 excerpts. The participant were asked to rate stimuli corresponding to each of the compared systems above, which were presented simultaneously, together with a reference and hidden reference. Ratings were done on a 0–100 scale, asking participants to rate for overall sound quality with respect to the reference. The (hidden) references are recordings of professional singers resampled to the vocoder sample rate. The input mel-spectrograms, F0 and WORLD features were extracted from these reference recordings. The references included around 7 different singers, some seen during training and some unseen, different languages, pitch ranges, genders, and singing styles. Comparing systems on synthetic input features is left as future work, partially because obtaining matching ground truth reference audio, synthetic mel-spectrogram and synthetic WORLD features is somewhat problematic.

6.5 Results

The results of the listening test are summarized in Table 6.2. The best-performing system is the autoregressive neural vocoder AR-WNV, which was to be expected as

Table 6.1: Hyperparameters for the proposed neural vocoder.

Hyperparameter		**Hyperparameter**	
Sample rate (f_s; Hz)	32,000	Source scaling factors	
Input features		g_p	0.177
Hop time (ms)	5	g_a	1.0
Window time (ms)	45	Multi-res. STFT losses	
Window	Hann	Hop times (ms)	[2.5, 5, 40]
FFT length (samples)	1440	Window times (ms)	[5, 20, 120]
Num. mel	100	FFT lengths (samples)	[256, 1024, 4096]
Freq. range (Hz)	10–15,200	Window	Hann
Gain data aug.	[-3, 0] dB	Clamp (dB)	[-120, ∞]
Mel norm. range	–140 to 0 dB	Loss weights	
F0 estimator	SAC	λ_{mag}	1.0
	(Villavicencio et al., 2015)	λ_{sc}	0.5
Filter prediction		λ_{spec}	36.0
Filter size (N)	1025 bins	λ_{fm}	2.0
H^0_{SP} pred. size	257 bins	λ_{reg}	36.0
H^0_{AP} pred. size	257 bins	Optimization	
H^0_{SP} pred. range	–140 to 20 dB	Batch size	16
H^0_{AP} pred. range	–60 to 0 dB	Seq. len. (s)	0.5
Kernel size	3×1	Optimizer	Adam
Num. layers	8	Params.	$\beta_1 = 0.9$
Dilation	1		$\beta_2 = 0.999$
Num. ch. residual	256		$\epsilon = 1 \times 10^{-8}$
Num. ch. skip	256	Learning rate	1×10^{-4}
Activation	Gated tanh	Schedule	Const. for 100 k steps,
Re-parametrization	Weight norm.		then exp. decay
Output	ReLU $\rightarrow$ 1×1(256) $\rightarrow$		(×0.5 per 100 k steps)
	ReLU $\rightarrow$ 1×1(257+257)	Clip grad. norm	0.5
	$\rightarrow$ sigmoid	Num. steps	500,000

System	Mean Opinion Score
Hidden reference	4.71 ± 0.03
NeuralWORLD (proposed)	4.13 ± 0.11
NeuralWORLD-NoAdv	3.21 ± 0.15
Excited-PWG	3.21 ± 0.09
AR-WNV	**4.36 ± 0.08**
WORLD	3.27 ± 0.15

Table 6.2: Mean opinion score (MOS) ratings to evaluate our proposed non-autoregressive source-filter neural vocoder. MOS ratings on a 1–5 scale with their respective 95% confidence intervals.

it is a very strong baseline, rated fairly close to the hidden reference score. Our proposed system, NeuralWORLD, is scored an (arguably) close second. The remaining three systems are rated notably worse. The version of our system without adversarial training, NeuralWORLD-NoAdv, the traditional vocoder, WORLD, and the baseline GAN-based neural vocoder with fewer constraints, Excited-PWG, are all rated similarly. We discuss some of the common artifacts we observe these systems produce in §6.5.1. Some sound examples corresponding to this chapter are available online[4].

6.5.1 Discussion of artifacts

Observing the results of the different systems, we can discuss some of the properties of the different artifacts they produce. Empirically, the traditional vocoder baseline WORLD, and the version of our proposed system without adversarial training, NeuralWORLD-NoAdv, produce perceptually similar artifacts. Although both systems are based on a very similar source-filter model, this may just be a coincidence. This artifact is best described as "buzziness", and is characterized as unvoiced or mixed-voicing regions (e.g., voiced fricatives), being "overly harmonics". In the waveform often periodic pulses can be observed where there should be none, e.g., see Figure 6.3. Considering that our proposed system with adversarial training, NeuralWORLD, does not produce such artifacts, we can conclude that training the model on just the multi-resolution STFT loss is not enough to clearly distinguish whether the produced spectra are sufficiently aperiodic in certain frequency bands. Related to this is also the model's ability to "recover" from voiced/unvoiced prediction errors by the F0 estimator.

We find that the Excited-PWG system, in addition to the occasional "buzziness" described above, can also produce a kind of "phasiness". This artifact can perhaps best be described as a kind of "robotic" effect due to an unnatural alignment of phases, and in the worst cases sounding like a kind of chorus effect. The degree of these artifacts seems to vary a bit from excerpt to excerpt, but is especially noticeable at low pitches.

[4]https://mtg.github.io/singing-synthesis-demos/nar-vocoder/

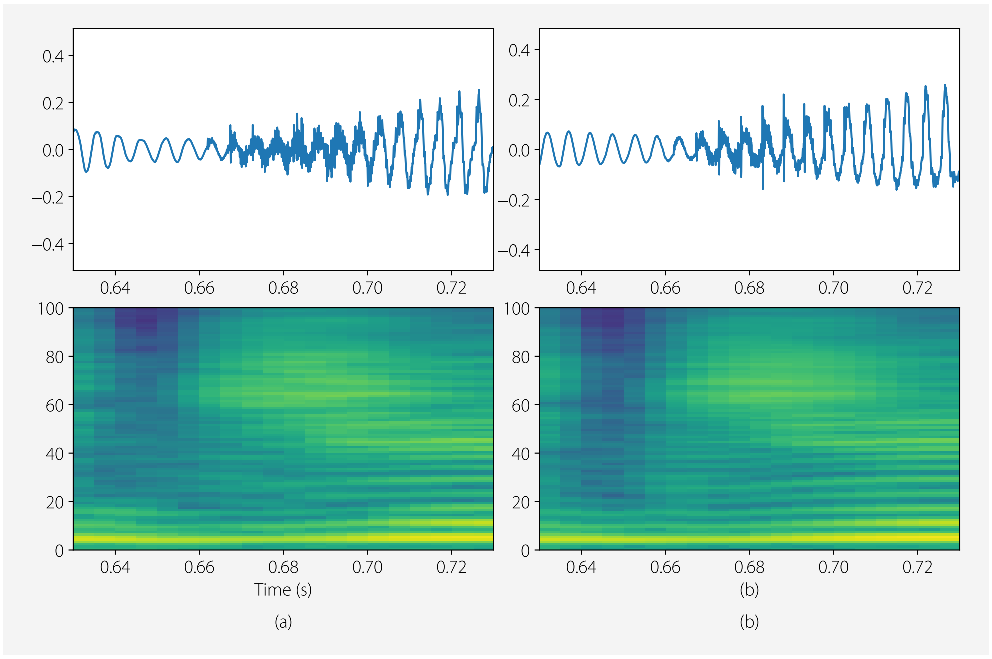

Figure 6.3: Example of an artifact when training our neural vocoder without adversarial loss. (a) Shows the output of our proposed model. (b) Shows the output of our proposed model without adversarial training. Notice here the voiced affricate [dʒ] has excessive harmonics in the mel-spectrogram (bottom), and an excessively pulse-like waveform (top).

Although WORLD uses a similar minimum-phase model to our own system, we found that occasionally it can also produce notable phasiness. We observe that in this case the phasiness tends to have a time-varying component, sometimes sounding like a kind of flanging effect. These artifacts are most noticeable in longer unvoiced segments, such as aspirations, unvoiced fricatives, unvoiced affricates, and so on.

Finally, our proposed system, NeuralWORLD, and the autoregressive baseline, AR-WNV, are both generally of high quality. Where AR-WNV notably improves over our non-autoregressive approach, is in non-modal voicing such as vocal fry or growls. This is not surprising as our source-filter model is designed with modal speech or singing in mind. A more subtle difference is that especially in very low pitch ranges, NeuralWORLD can lack a certain "presence", compared to AR-WNV or a reference recording.

6.5.2 Inference speed

Inference is quite efficient, at around 213.6 × real-time on a single NVIDIA GeForce RTX2080 Ti GPU, and 9.7 × real-time on single core of an i9-9960X central processing unit (CPU). These numbers are computed using a single sample batch, and averaged over a representative selection of phrases, as the speed varies somewhat depending on the sequence length. While these are competitive numbers, there are some approaches notably faster, e.g., HiFi-GAN's v3 configuration (Kong et al., 2020) runs at over 1000 × on a GPU (albeit a V100 GPU and at 22 kHz). That said, we have not applied any optimization to our code, beyond the most basic. In particular, the band-limited impulse train generation could be implemented using a much more optimized algorithm (e.g., Stilson and Smith, 1996). It is also quite likely that the filter estimation network could be further optimized in terms of computational complexity and memory requirements. For certain applications where speed is of the utmost importance, we could even optimize the FIR filtering.

6.6 Conclusions

We have proposed a non-autoregressive neural vocoder for singing, based on a quite constrained source-filter model. Unlike most competing models where either the source excitation is (partially) predicted by a neural network, or the filter itself is a non-linear neural network, our model predicts time-varying linear filters from the input mel-spectrogram, and generates a source excitation signal from the input F0.

Compared to competing autoregressive neural vocoders, our proposed model is rated fairly closely, albeit slightly lower. However, inference is much faster (and with much room for improvement still). Additionally, any potential instability due to relying on autoregressive feedback is avoided. We feel that for many applications our approach can be a practical solution. Especially for cases where significantly faster inference, and a stable and easily interpretable model are desirable.

Compared to competing non-autoregressive neural vocoders, our much more constrained model provides a straightforward way to practically guarantee certain types of artifacts will not be present. In particular, we avoid discontinuities and "phasiness" by using a continuous excitation signal, combined with a smoothly varying minimum-phase filter response.

In an ablation study, we found adversarial training to be especially useful to mitigate issues of "buzziness". From this we may conclude that these kind of artifacts are not sufficiently penalized by a multi-resolution STFT loss alone.

One downside of our approach is that the underlying source-filter model only considers modal speech or singing. Extending our model to cover other voice qualities, such as growls, vocal fry, rough voices, etc., in a general and flexible way is interesting future work.

Part II

Data-efficient and reduced effort voice creation

Data-efficient voice creation via voice cloning | 7

Many applications of singing synthesis require efficiently creating new voices. For example, in many cases we want to reproduce a specific target singer's voice, not just any voice. Or, in some cases like music production, where there may not be a specific target voice, the user will want to be able to select a voice from a wide array of possibilities. A singer may want to model their own voice, in order to let other musicians use it in their productions, or augment their own performances. In other cases, such as reproducing a large choir, a wide range of timbres is required. Thus, in singing synthesis there arguably is a greater need to model many voices compared to text-to-speech (TTS), where for many applications the exact voice is somewhat irrelevant.

With this demand for modeling many voices, data efficiency of the voice modeling process becomes of increasing importance. In this work, we discuss both improving data efficiency for creating new voices, and reducing the effort required, notably by reducing the number of annotations needed in the training data. That said, one important pathway to reducing the voice creation effort is improving data efficiency, as recording and annotating a small amount of data obviously entails far less work than recording and annotating a large amount of data. Similarly, even in the case of almost zero-effort voice creation, data efficiency can still be of importance. For instance, in the case of modeling a deceased singer's voice, the amount of available data may be quite limited. In other cases, such as creating user voices or large choirs, recording very large amounts of data may be impractical. For most applications, the ideal system would simultaneously be data efficient, as well as allow low-effort voice creation.

Voice cloning, also known as voice fitting or speaker adaptation, is a technique that leverages data from many speakers[1] combined with a small amount of data from the target speaker, e.g., 2 min, to allow creating a voice model that outperforms a model trained on just the adaptation target data from scratch. These kinds of ideas have been around for a long time in TTS and singing synthesis, in particular for models based on hidden Markov models (HMMs) (e.g., Shirota et al., 2014). With the advent of TTS and singing synthesis models based on deep learning (e.g., van den Oord et al., 2016a; Shen

[1]Throughout this document, the term "speaker" is used regardless of whether the subject is speaking or singing.

et al., 2018), these techniques have become notably easier to implement and arguably have better results.

In this chapter, we apply voice cloning techniques to an autoregressive singing synthesizer, based on the work from Chapter 4. Here, we focus only on modeling timbre of the voice, not other expressive features, such as F0 or timing. Instead, we consider pitch and phonetic timings control inputs given by some external source, such as a recording (e.g., Janer et al., 2006), or another model (e.g., Umbert et al., 2015; Hua, 2018). While applying voicing cloning techniques to such expressive features is important future work, we argue voice cloning for just timbre is still useful for many practical applications.

While in this chapter we base our model on the autoregressive model from Chapter 4, we have also applied these techniques to other models, such as the non-autoregressive model from Chapter 5, as well as others. Although we do not report the results of these (informal) experiments, we feel fairly confident that these techniques can be applied to virtually any model with good results.

The content of this chapter was originally published as Blaauw et al. (2019). This work investigates adapting modern voice cloning techniques proposed for TTS to singing synthesis. More importantly, using a number of different listening tests, we evaluate several practical aspects of singing voice cloning, such as what kind of data to use, across datasets in English, Japanese and Spanish/Catalan.

7.1 Proposed method

As a reminder, the autoregressive timbre proposed in Chapter 4, was defined as (copying Equation (4.5)),

$$p_\theta(\mathbf{x} \mid \mathbf{c}) := \prod_{t=1}^{T} p_\theta(\mathbf{x}_t \mid \mathbf{x}_{<t}, \mathbf{c}), \tag{7.1}$$

where $\mathbf{x}$ is a sequence of multivariate intermediate acoustic features, $\mathbf{x}_t$ is a single timestep of that sequence, $\mathbf{x}_{<t}$ are acoustic features corresponding to all timesteps preceding timestep t, and $\mathbf{c}$ is the time-aligned sequence of control features. In this case, the model uses a network architecture based on WaveNet (van den Oord et al., 2016a) which is parametrized by θ.

7.1.1 Multi-speaker model

In order to leverage data from many speakers, we first extend this base model to a multi-speaker variant. Such a model is a single network able to model all voices in the training data simultaneously, by conditioning it on an additional low-dimensional vector representing the different speaker identities. We follow Arik et al. (2017b) and use a learned *speaker embedding* which depends on the acoustic properties of each speaker. We thus extend the model in Equation (7.1) to become,

$$p_{\theta,\mathbf{s}}(\mathbf{x} \mid \mathbf{c}, i) := \prod_{t=1}^{T} p_{\theta,\mathbf{s}}(\mathbf{x}_t \mid \mathbf{x}_{<t}, \mathbf{c}, i). \tag{7.2}$$

The main difference here is that the model is conditioned on an additional speaker index, $1 \leq i \leq N$, where N is the total number of speakers in the dataset. Additionally, the model now includes an additional table of speaker embeddings, $\mathbf{s} \in \mathbb{R}^{N \times M}$, where M is the dimensionality of the speaker embeddings. This table is initialized from a uniform random distribution in the range $[-0.1, 0.1]$, and then learned jointly with the other model parameters, θ. During training and inference, the speaker index, i, is used to select a specific speaker embedding vector, $\mathbf{s}_i$, selected from the table. Conditioning of the model on the speaker embedding is done by simply concatenating the speaker embedding, $\mathbf{s}_i$, to the control vector for the current timestep, $\mathbf{c}_t$, to form an augmented control vector,

$$\hat{\mathbf{c}}_t = [\mathbf{c}_t; \mathbf{s}_i]. \tag{7.3}$$

This augmented control vector is then processed by the network architecture as normal (see §4.1.2).

The underlying idea here is that if we train such a model on sufficiently many speakers, the speaker embeddings should be learned to represent key characteristics of each speaker's voice. Consequently, it should be possible to generalize to new speakers by simply finding the corresponding speaker embeddings, but keeping all the other weights in the model fixed. As the dimensionality of the speaker embeddings tends to be tiny (e.g., $M = 16$ or $M = 32$) compared to the total number of other model parameters, we argue that it is plausible to learn these effectively from significantly less data.

7.1.2 Speaker adaptation

In order to adapt a multi-speaker model to a new unseen speaker, a randomly initialized vector is added to the speaker embedding table, s, and the training of the model is continued on data of the new speaker, using the corresponding speaker index and generative loss as usual. In this case, the multi-speaker weights, θ, are kept fixed and only the new speaker embedding, $\mathbf{s}_{i=N+1}$, is updated. However, as previously reported in e.g., Arik et al. (2018) and Chen et al. (2019), the generalization capabilities of a multi-speaker model tend to be limited in practice, and better results in terms of synthesis quality and speaker similarity are obtained using so-called *fine-tuning* of all the model weights.

When fine-tuning the whole model, both the speaker embedding, $\mathbf{s}_i$, and the multi-speaker model weights, θ, are updated using the generative loss computed over data of the new speaker. As the goal of voice cloning is to use very few material of the target voice, this data tends to have a very small size. Thus, unlike learning only the new speaker embedding, fine-tuning the whole model tends to overfit. A simple solution is to use early stopping and only fine-tune the whole model for a small number of updates. In contrast, learning a new speaker embedding tends to converge after many more updates. Intuitively, it therefore makes sense to first optimize the speaker embedding for many updates and then do a fine-tuning of the multi-speaker model weights for few updates. However, we empirically found that this approach does not clearly benefit the results compared to jointly learning all parameters for few updates, and thus we tend to use this latter approach as it is much faster.

7.1.3 Singing voice specifics

Compared to speech, expressive singing tends to span a wider range of pitches and timbres. As our proposed system uses a vocoder (Morise et al., 2016) which separates pitch and timbre to a great extent, generalizing to a wide range of pitches can be done efficiently. However, the high degree of variability in terms of timbre should be considered, especially since datasets in singing synthesis tend to have less data for the single-speaker case or fewer speakers for the multi-speaker case, compared to datasets in TTS.

In order to keep the intra-speaker timbre variability manageable, we propose to use what we call *pseudo singing* for the multi-speaker model. Pseudo singing are phrases that are sung with approximately constant pitch, cadence and dynamics, forcing a more clear and coherent pronunciation; see §3.2.2 for more details. Using this kind of data thus provides a convenient way of ensuring a more homogeneous base voice. Additionally, as we ideally want a large number of speakers for the multi-speaker model, another important advantage of this kind of data is that phonetic transcription and automatic segmentation tend to be more straightforward. The recordings used for the target voice to clone can be either natural or pseudo singing.

7.1.4 Combination with a neural vocoder

The use of a traditional heuristics-based vocoder has some benefits in the case of singing synthesis, but at the same time places an upper bound on the obtainable sound quality. Especially when the used generative model introduces some form of smoothing, the synthesized waveform can often sound overly “buzzy”. One way to mitigate these problems is to use a neural vocoder, e.g., a WaveNet model that predicts waveform from vocoder features (Tamamori et al., 2017; Sotelo et al., 2017); see §3.1.3 for more details. While the amount of data required to train such a neural vocoder may seem contrary to the goals of voice cloning, we have obtained promising results by combining data from different speakers in order to learn what is sometimes called a *universal* mapping of intermediate acoustic features to waveform (Jia et al., 2018; Lorenzo-Trueba et al., 2019). As training data in this case only consists of audio, without requiring text, speaker identities or other annotations, the burden of collecting such a dataset is notably reduced.

7.2 Relation to prior work

There have been several recent works on voice cloning. Following Arik et al. (2018), the two main approaches can be classified as *speaker adaptation* and *speaker encoding*. Speaker adaptation, which is used in this work, optimizes a speaker embedding for a new speaker within a multi-speaker model using gradient descent. Speaker encoding on the other hand uses a secondary network to predict a new speaker embedding from acoustic features. The latter approach has the advantage that voice cloning only requires a single forward pass rather than a costly iterative optimization, and that no transcription of the acoustic adaptation target material is needed. However, both approaches have been shown to benefit significantly from a fine-tuning of all model weights (Arik et al., 2018; Chen et al., 2019), effectively negating these benefits from the latter method. Furthermore, while the speaker encoding approach can be highly data efficient, the secondary network needs to be trained with a lot of speakers to be effective (e.g., Jia et al. (2018) mention that training the encoder on 18 k speakers improves over 1.2 k speakers). The number of speakers in singing datasets tends to be notably smaller.

Arik et al. (2018) compare speaker adaptation and encoding approaches for the Deep Voice 3 model, which is a convolutional sequence-to-sequence (Seq2Seq) model that predicts mel-spectrogram features combined with a Griffin-Lim vocoder (Griffin and Lim, 1984). Jia et al. (2018) propose an encoding approach in the context of the Tacotron 2 synthesizer, using speaker vectors obtained using a network trained for speaker verification using a discriminative loss rather than a generative loss. In this case, the waveform is generated from predicted mel-spectrogram features using a multi-speaker WaveNet vocoder. Taigman et al. (2018) and Nachmani et al. (2018) discuss speaker adaptation and encoding respectively, in the context of the novel shifting buffer VoiceLoop model, with the latter using a jointly optimized generator and encoder network using a combination of generative, contrastive and cycle constancy losses. Here, like in our work, the model predicts vocoder features. Similarly, Chen et al. (2019) use speaker adaptation and speaker encoding using *d-vectors* (as in (Jia et al., 2018)), but adapt these techniques to the WaveNet model, notably using waveform directly rather than mel-spectrogram or vocoder features. There are several small architectural differences between these works, such as how conditioning on speaker embedding is implemented.

One notably different approach is to use unsupervised pre-training. For instance, Chung et al. (2018) propose this method for a model based on the first Tacotron. Large, independent text and audio corpora are used to pre-train word embeddings and an unconditioned autoregressive (next-step frame prediction). Finally, the entire Seq2Seq network is fine-tuned on the target data consisting of <audio, text> pairs. Data efficiency is improved with this approach, e.g., good results are obtained using 1 shard (24 min) of data, but this is still notably more than some of the voice cloning approaches above. However, at the same time, the adaptation target data does not need to be pre-aligned, and no annotations are required for the pre-training audio data.

The approach proposed in this chapter has similarities to many of the above works. However, we apply these techniques to the model proposed in Chapter 4, and evaluate the model in a context relevant to singing rather than speech. Related prior works in singing synthesis have been limited to speaker adaptive training of HMMs (Shirota et al., 2014) to increase data efficiency.

After we published our work on voice cloning, several other proposed singing synthesizers have also included multi-speaker models, fine-tuning and similar techniques (e.g. Lee et al., 2020; Wu and Luan, 2020).

7.3 Experiments

7.3.1 Datasets

The datasets used in this work, consisting of audio with aligned phonetic transcription and speaker identities, are listed in Table 7.1. The pseudo singing datasets generally are optimized for phoneme or diphone coverage. The natural singing datasets mostly consist of songs with a somewhat similar style from the repertoire of the singer. The phonetic transcription and segmentation was done automatically for the pseudo singing datasets, but some manual correction was required in the case of natural singing. Except for NUS-48E (Duan et al., 2013), all datasets are proprietary.

Table 7.1: The different datasets used in the experiments and their properties.

Tag	Speakers	Kind	Language	Gender	Style	Size[1]	Pitches	Avg. dur.[2]
JP-MULTI-P	35	Pseudo	Japanese	8M/27F	Mixed	80	3	0:14:48
EN-MULTI-P	13	Pseudo	English, mixed native	8M/5F	Mixed	524	12×1,1×2	0:32:28
NUS-48E (Duan et al., 2013)	12	Natural	English, non-native	6M/6F	Pop	4	-	0:10:35
ES-MULTI-A,B-P	2×8[3]	Pseudo	Spanish, Catalan	2×4M/4F	Choir	219	3	0:51:47
JP-TAR-K-P	1	Pseudo	Japanese	Female	Pop	9	3	0:01:49
EN-TAR-PS-N	1	Natural	English	Male	Pop	2	-	0:03:31
EN-TAR-AM-N	1	Natural	English	Female	Pop	6 (excerpts)	-	0:02:14
EN-TAR-AM-P	1	Pseudo	English	Female	Pop	10	3	0:02:23
ES-TAR-*-P	2×4×1[3]	Pseudo	Spanish, Catalan	2×2M/2F	Choir	10	3	0:02:22
JP-FULL-K-P	1	Pseudo	Japanese	Female	Pop	80	3	0:14:35
EN-FULL-PS-N	1	Natural	English	Male	Pop	39	-	1:29:01

[1] Size in number of utterances per speaker per pitch for pseudo singing datasets, or number of songs per speaker for natural singing datasets.

[2] Average duration per speaker (combining all pitches) for multi-speaker datasets, or total duration of single-speaker datasets.

[3] In the experiments, from total of 12 speakers, we create two 8 speaker multi-speaker models with a different set of 4 held-out speakers. Each of these models is then adapted to its 4 held-out speakers, for a total of 8 targets.

7.3.2 Model hyperparameters

The hyperparameters used for the experiments generally follow "Model hyperparameters" of §4.3.4, but were simplified slightly and the number of channels was increased to increase capacity for the multi-speaker model. In particular, we use an initial 1×1 convolution, followed by 5 causal 2×1 convolutional layers with gated tanh non-linearity, dilation factors $\{1, 2, 4, 8, 16\}$, and non-parameterized residual connections. These layers use 384 or 256 channels, and 256 or 128 skip channels for multi-speaker and single-speaker models respectively. The output stack is comprised of a fully connected rectified linear unit (ReLU) layer, a fully connected output layer and a simple (mean) L_1 loss, rather than a mixture density negative log-likelihood loss. We used a 16-dimensional speaker embedding for all experiments. The multi-speaker models are trained for around 800 k iterations, with an initial learning rate of 3×10^{-4}. Adaptation fine-tuning is done for 4 k additional iterations, applying Polyak averaging (Polyak and Juditsky, 1992). The baseline single-speaker models are trained for 30–90 k iterations with a learning rate of 5×10^{-4}.

7.3.3 Listening tests

We conducted a series of listening tests to evaluate the proposed system perceptually, in lieu of reliable quantitative metrics. The listening tests consisted of simple AB preference tests, where participants were asked to select the preferred stimulus ("A", "B" or "no preference"), considering a reference recording of the target singer. In all tests, this reference recording is also used to control pitch and phonetic timings. In total there were 19 participants, each rating 20 pairs of acapella stimuli, divided over 5 tests. The different tests and their results are described below.

7.4 Results

The first test tries to measure the effectiveness of voice cloning, in the case of a Japanese pseudo singing multi-speaker model, JP-MULTI-P (see Table 7.1), and a pseudo singing adaptation target, JP-TAR-K-P. The test compares the adapted model to a model trained on the full dataset, JP-FULL-K-P, and the adapted model to a model trained from random initialization on the same small adaptation dataset. In Figure 7.1, we can see that the adapted model actually outperforms the model trained on the full dataset. One explanation could be that the relatively large amount of data used to train the underlying multi-speaker model improves overall sound quality, while both models capture the target speaker identity similarly well. As expected, the adapted model is

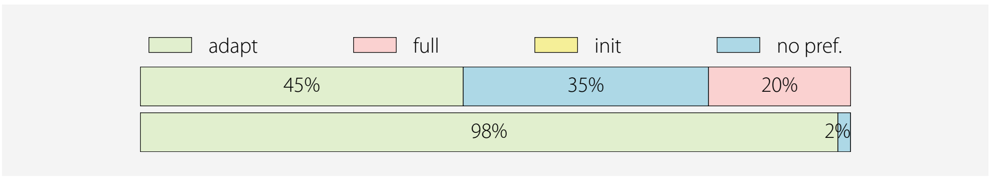

Figure 7.1: Results of the test comparing adapted voice ("adapt", 1 min 49) to a voice trained on the full dataset ("full", 14 min 48) and a voice trained on adaptation data from random initialization ("init", 0% preference) respectively. In this case with Japanese pseudo singing multi-speaker and target data.

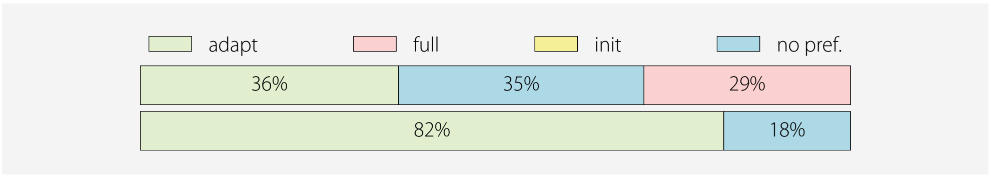

Figure 7.2: Results of the test comparing adapted voice ("adapt", 3 min 31) to a voice trained on the full dataset ("full", 1 h 29) and a voice trained on adaptation data from random initialization ("init", 0% preference) respectively. In this case with English pseudo singing multi-speaker data and natural singing target data.

consistently preferred over the model trained on a small amount of data from random initialization.

The second test is similar to the first test, but for the case of an English pseudo singing multi-speaker model, EN-MULTI-P, a natural singing adaptation target, EN-TAR-PS-N, and a full dataset of natural singing, EN-FULL-PS-N. In Figure 7.2, we see that the adapted model and the full dataset model perform similarly. This shows that in this case cloning is as effective as training on a larger dataset. Again, the adapted model is consistently preferred over the model trained on a small amount of data from random initialization.

The third test considers the use case of choir singing. In particular, using cloning to create larger choirs, e.g., 8 full voices and 24 cloned voices adapted from the former

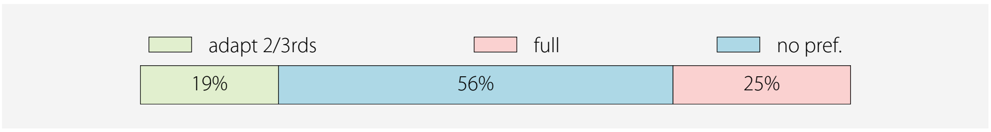

Figure 7.3: Results of the test comparing a synthetic choir consisting of 8 adapted voices and 4 full dataset voices ("adapt 2/3rds", 8 × 2 min 22, 4 × 51 min 47) to a choir trained on full datasets ("full"). In this case with Spanish/Catalan pseudo singing multi-speaker and target data.

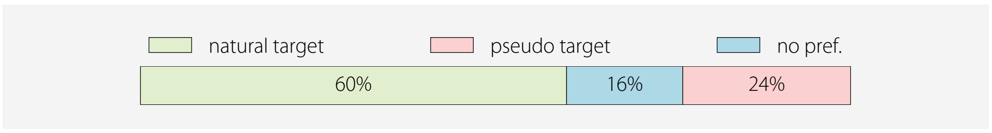

Figure 7.4: Results of a test comparing adaption of a pseudo singing multi-speaker model to natural singing (2 min 14) and pseudo singing (2 min 23) respectively. In this case with English data.

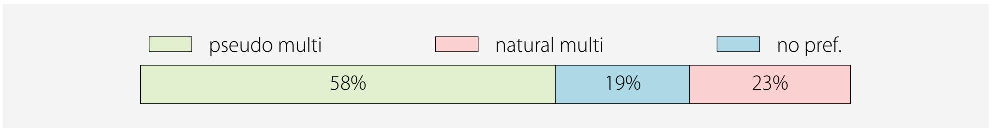

Figure 7.5: Results of a test comparing adaption of a multi-speaker model from pseudo singing (13×32 min 28) or from natural singing (12×10 min 35) to natural singing. In this case with English data.

set of full voices. As the available dataset consisted of just 12 voices, 3 per soprano, alto, tenor and bass part, we performed a scaled-down experiment, generating a choir of 4 full voices and 2×4 cloned voices, adapted from two different 8 voice multi-speaker models. First, a multi-speaker model is trained on 8 voices, ES-MULTI-A,B-P, holding out one voice per part. Then, adapted models are created for each of the held-out voices, ES-TAR-*-P, and the whole process is repeated for a different set of held-out voices. Finally, we generate the choir synthesis by mixing 4 full voices and 8 cloned voices. We compare this result with a result of a single multi-speaker model trained on the full dataset of all 12 voices. In Figure 7.3, we can see that both models perform on par, showing that in this case using cloned voices does not significantly degrade the result.

The fourth test compares using a natural singing adaptation target, EN-TAR-AM-N, to using a pseudo singing adaptation target, EN-TAR-AM-P, in the case of English. Both use a multi-speaker model trained on pseudo singing, EN-MULTI-P. In Figure 7.4, we see that a natural singing target is preferred. One explanation for this is that pseudo singing tends to be a little overpronounced compared to natural singing, thus producing a timbre a little farther from the reference stimulus.

The fifth test compares using a multi-speaker model trained on pseudo singing, EN-MULTI-P, to a multi-speaker model trained on natural singing, NUS-48E, in the case of English. Both are then adapted to natural singing. In Figure 7.5, we see that a multi-speaker model trained on pseudo singing is preferred. One possible explanation for this could be that the more coherent pseudo singing provides a more homogeneous base voice. However, as the two multi-speaker datasets have different sizes and consist of different speakers, it is difficult to draw any definitive conclusions from this single result.

A number of sound examples are available online[2].

7.5 Conclusions

We feel that voice cloning provides a simple yet effective and widely applicable tool for improving data efficiency in deep learning-based singing synthesis. From small amounts of data, the target speaker identity can be convincingly reproduced, while maintaining a sound quality comparable to non-cloned voices. We have shown that these techniques taken from TTS research are also applicable to singing synthesis, with few modifications. One convenient approach is to combine a multi-speaker model trained on pseudo singing, with a natural singing adaptation target. This provides a very coherent multi-speaker base voice, that is relatively easy to annotate (i.e., automatically), while at the same time resulting in a final target voice to sound natural and expressive.

While the proposed system, focused on cloning timbre only, this approach still has many practical applications. Ultimately, want to clone both timbrical and expressive aspects of the target voice. In this case, going away from a traditional TTS pipeline and towards an end-to-end system would be a promising direction.

[2]https://mtg.github.io/singing-synthesis-demos/voice-cloning/

8 Creating voices using annotated vowel onsets and self-attention

In recent years, modern text-to-speech (TTS) systems have largely moved to sequence-to-sequence (Seq2Seq) models, where the alignment between the phonetic or orthographic input sequence and the acoustic output sequence is learned during training and inferred during synthesis (e.g., Wang et al., 2017; Shen et al., 2018; Ping et al., 2018). One advantage of this approach is that it leads to a more end-to-end system. This means that we can avoid the need for pre-aligned training data, avoid the need for a separate phonetic duration model, and possibly allow integration of grapheme-to-phoneme conversion in the system. For singing synthesis, not requiring pre-aligned training data is particularly attractive, as many existing tools to do this automatically (e.g., forced alignment with a hidden Markov model (HMM) model) do not yield sufficiently accurate results on expressive singing, often requiring manual correction. This manual correction forms the bottleneck in process of creating new voices, in terms of effort (time and expertise) required compared to the other steps.

In addition to reducing the effort needed to create new voices, a Seq2Seq approach may also improve the resulting sound quality of the synthesizer. In particular, when correcting phonetic segmentation there often is some degree of ambiguity, and disagreement between different annotators. As such, there often is some inconsistencies in the annotations of a dataset of several hours. We argue that a learned alignment may ultimately be better as the model can decide what alignment is most favorable for obtaining the desired target output, likely showing greater coherence.

A common approach for Seq2Seq models in TTS is to use a content-based attention mechanism (e.g., Gehring et al., 2017), sometimes additionally using location-based information (e.g., Graves, 2013). As these mechanisms require access to acoustic information at inference, they are normally used in combination with an autoregressive decoder. Recently, some systems have been proposed that use a feed-forward decoder and an alternative attention mechanism that does not rely on access to acoustic information (Peng et al., 2019; Ren et al., 2019). These notably provide faster, parallelizable inference, and are reported to produce more robust alignments with fewer mispronounced, repeated or skipped phonemes.

As discussed in Chapter 5, this non-autoregressive, feed-forward approach is also interesting in the case of singing synthesis, as it avoids the exposure bias problem (see

"Regularization to mitigate exposure bias" in §4.1.3), caused by the discrepancy between teacher forced training and fully autoregressive inference. This problem can be especially noticeable in long sustained vowels where prediction errors tend to accumulate over time. Additionally, in our experience for singing synthesis, learning clean alignments reliably with autoregressive, content-based attention mechanisms can be quite challenging. As a result, reaching similar-quality results compared to non-Seq2Seq systems can also be difficult.

Note that the issue of inconsistencies in manually corrected phonetic segmentations used in non-Seq2Seq models mentioned above is exacerbated when combined with a non-autoregressive model. As such models learn a mapping from time-aligned control features to acoustic features, without any additional inputs, it is unlikely that they are able to compensate for errors in this time-alignment. Autoregressive models on the other hand do take additional inputs in the form of past timesteps, which may allow for a greater degree of compensation.

This chapter is derived from work originally published as Blaauw and Bonada (2020). This original publication also contained the topics discussed in Chapter 5. Whereas this chapter deals with reducing the effort required for creating new voices by only using note onset annotations, the almost identical model from Chapter 5 uses full phonetic onset annotations and focuses on using self-attention to model time dependencies in a non-autoregressive model.

8.1 Proposed system

In singing synthesis, the alignment between the input phonetic sequence and the output acoustic sequence is strongly constrained by the given musical score. This is a notable difference from TTS, which is generally only weakly constrained by the (average) speech rate. Exploiting this fact, we propose to first generate an approximate initial alignment using the given note onset timings and a phoneme duration model. Once the input sequence is roughly aligned to the target output timesteps, we assume that the network is able to gradually refine the alignment through a series of transformations, notably using self-attention, until reaching something close to the target. Note that this approach is quite different from the approach using content-based attention, as here the initial alignment does not use any acoustic features at all.

An important point here is that we assume that the accuracy of the phoneme duration model is not critical to the end results. We assume that the decoder is powerful enough to be able to recover from errors in the initial alignment, to a certain degree. At the same time, the initial alignment can never hugely deviate from the true alignment, as it is heavily constrained by the note timings. To see if this assumption is correct,

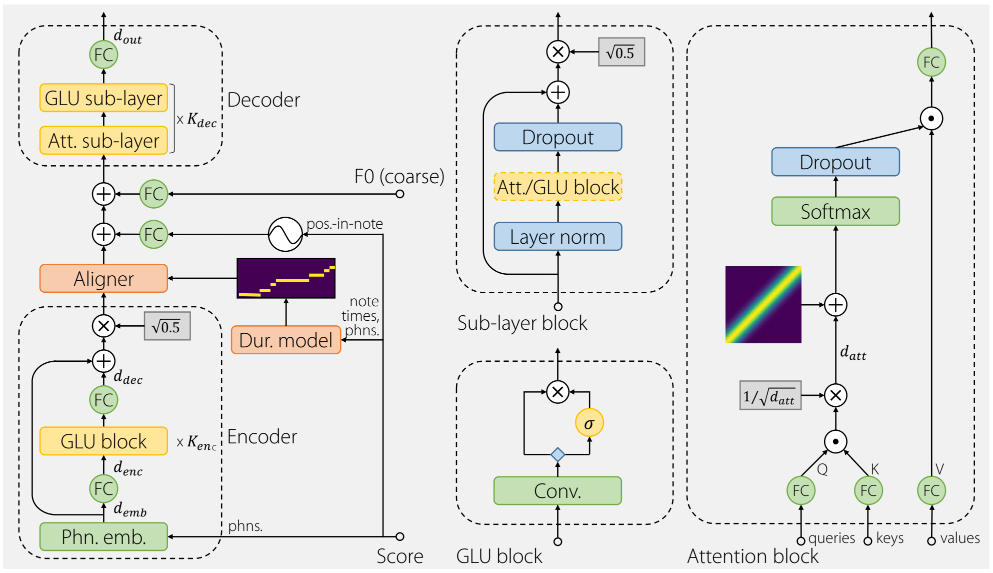

Figure 8.1: A diagram of the complete model architecture. On the left is the full system, composed of encoder, aligner and decoder, which themselves are composed of different higher-level blocks. On the right, these higher-level blocks (sub-layer, gated linear unit (GLU) and attention) are shown in detail.

we purposely use a very simplistic, approximate phoneme duration model, based on average phoneme durations computed on a different dataset whose segmentation was corrected by hand. While language dependent, in this case the approximate phoneme duration model is not singer dependent and the values could simply be copied from a table, without the need for any data with phonetic timings.

8.1.1 Model architecture

The input to our system is a musical score, consisting of a sequence of notes. Each note consists of an onset, duration, pitch, and a sequence of phonemes, typically corresponding to a syllable. In this work, we define the note onset as the vowel onset, and note end as the onset of the following vowel or silence. Additionally, we provide an external F0 to our system, in order to capture the effect of pitch on timbre. The output of our system is a sequence of harmonic and aperiodic vocoder features, which in this case are simply concatenated.

The main components of our proposed system, as depicted in Figure 8.1, are the encoder, the aligner and the decoder. The encoder takes the input phonetic sequence and computes a sequence of hidden states corresponding to each phoneme and their local context. The aligner provides a hard alignment by repeating these states according to the predicted approximate phoneme durations, obtaining a sequence of the same length as the output acoustic sequence. Next, some additional conditioning signals derived from F0 and position are added. The decoder, based on the Transformer model (Vaswani et al., 2017), finally transforms the sequence of encoder hidden states to the target output sequence, through a series of self-attention and convolutional layers.

Given this encoder-decoder structure, it becomes clear that one of the tasks of the decoder is to learn the differences between the initial approximate phonetic timing and the true phonetic timing. That is, the decoder has to convert somewhat misaligned control features (or some transformation thereof) into the output acoustic features, taking differences in timing into account. This is mainly achieved by operations provided by the self-attention mechanism of the feed-forward Transformer (FFT) layers. The joint action of the approximate duration model and decoder self-attention, thus replaces the need for a traditional (accurate) duration model that may be used in a typical TTS or singing synthesis pipeline.

8.1.2 Approximate phoneme duration model

As noted, we purposely choose to use a very simplistic approximate phoneme duration model in this work. It consists of a simple lookup table, populated with average phoneme durations computed from a dataset of a different singer with manually corrected phonetic segmentation. A simple heuristic is then used to ensure that the sum of predicted phoneme durations matches the target note duration.

As we assume the note onset to correspond to the vowel onset, we first shift all onset consonants of each note to the preceding note (or silence). Then, we look up the sequence of average phoneme durations for each note, $[\overline{d}_1, \overline{d}_2, \dots, \overline{d}_N]$, where N is the corresponding number of phonemes. Thus, $\overline{d}_1$ will correspond to the average duration of the vowel (unused here), and $\overline{d}_2, \dots, \overline{d}_N$ will correspond to the coda consonants of the current note and the onset consonants of the following note. In order to match the target note duration, d_n, we use the predicted consonant durations and fill the remaining duration with the vowel. However, we also ensure at least half of the note's duration is occupied by the vowel by fixing $r_v = 0.5$. The scaling factor for all consonants in the note, r_c, then becomes,

$$r_c = \begin{cases} 1 & \text{for } N = 1, \\ \min\left(1, \dfrac{d_n - \lfloor r_v d_n \rceil}{\sum_{i=2}^{N} \overline{d}_i}\right) & \text{otherwise.} \end{cases} \tag{8.1}$$

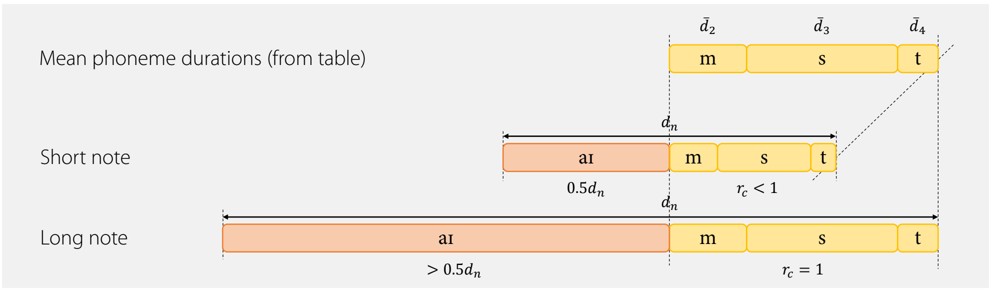

Figure 8.2: An example of the phoneme duration fitting heuristic. Here, the phonetic sequence [aɪmst] is fitted into a short (top) and long (bottom) note. For the short note, the vowel occupies half the note duration and the consonants are shrunk to fit. For the long note, the vowel occupies more than half the note duration and the consonants are kept at their average durations.

And, the final fitted phoneme durations, $[d_1, d_2, \ldots, d_N]$, then becomes,

$$d_i = \begin{cases} d_n - \sum_{j=2}^{N} \max\left(1, \lfloor r_c \bar{d}_j \rceil\right) & \text{for } i = 1, \\ \max\left(1, \lfloor r_c \bar{d}_i \rceil\right) & \text{for } i = 2, 3, \ldots, N. \end{cases} \tag{8.2}$$

Note that all durations here are in integer number of frames, and that $\lfloor \cdot \rceil$ rounds to the nearest integer. There are corrections for rounding errors and avoiding zero frame durations. In the case of notes where the nucleus is a syllabic consonant, the syllabic consonant is handled as a vowel. An example of this phoneme duration fitting heuristic is illustrated in Figure 8.2.

One obvious shortcoming of this simplistic model is that it is based on globally averaged phoneme durations. This do not consider context such as note duration or the surrounding phonetic sequence. Furthermore, the distribution of phonetic durations can easily have multiple modes. That said, a more powerful, data-driven duration model, such as the one proposed in §4.3.3, needs a dataset with annotated phoneme timings, defeating much of the purpose of our proposed approach to reducing the dataset annotation effort.

8.2 Relation to prior work

Our work is most closely related to the recently proposed FastSpeech model for TTS (Ren et al., 2019). This model is also based on the feed-forward Transformer (FFT) and an initial alignment obtained from a duration model. However, in this case, the duration model is trained with the help of a teacher model based on an autoregressive Transformer (Li et al., 2019), which is also used for generating the target mel-spectrogram features. We wanted to avoid the need to train an autoregressive teacher model, as we found this generally challenging for the case of singing voice. Additionally, we apply some modifications to the architecture, such as the use of gated linear unit (GLU) convolutional blocks, alternative positional encoding and a Gaussian bias for the self-attention layers.

The ParaNet model (Peng et al., 2019) proposes a different approach to feed-forward TTS. Here, standard content-based encoder-decoder attention is used, but the model is trained with the help of attention distillation with an autoregressive teacher model based on (Ping et al., 2018). Besides the reasons mentioned above, we found that the hard alignment used in our approach makes it easier to obtain a quality similar to non-Seq2Seq models, compared to the soft alignment of encoder-decoder attention.

At the time of publication of our model, in singing synthesis, the only Seq2Seq system we are aware of was (Lee et al., 2019). This model is based on the DCTTS model (Tachibana et al., 2018), using content-based encoder-decoder attention, with autoregressive decoder. Similar to our approach, there is an initial alignment of the input states to the output timesteps. However, relying on the fact that the Korean syllable structure has at most one onset and one coda consonant, the first frame and the last frame of the note are assigned to each consonant respectively, and the remaining frames are assigned to the vowel. After which, learning the attention alignment can be facilitated by using diagonally guided attention (Tachibana et al., 2018). It is not yet clear if this approach can be extended to languages with more complex syllable structures, such as English, which we focus on in our work.

After we published our model, several other singing synthesizers have been proposed. While some of these have been Seq2Seq models, no radically different approaches have been introduced so far. Some use a more traditional Seq2Seq approach derived from TTS (e.g., Angelini et al., 2020). Others, like our model, use an aligner module that repeats encoder timesteps to match decoder timesteps. In this case, phoneme durations are either given by a ground truth (e.g., Wu and Luan, 2020; Shi et al., 2020), or are obtained from a model that predicts phoneme durations (e.g., Lu et al., 2020; Chen et al., 2020).

8.3 Experiments

The experiments performed to evaluate our proposed Seq2Seq model from this chapter where combined with those to evaluate the non-autoregressive timbre model using self-attention from Chapter 5. Or rather, the experiments Chapter 5 are a subset of the experiments described here, which contain two additional systems that include the approximate phoneme duration module rather than using ground truth phoneme durations. As such, the dataset used is the same, the design of the listening test is the same, and the hyperparameters used are the same. Details of these aspects of the experiments can be found in §5.3.

8.3.1 Compared systems

In our experiments, we aim to compare our proposed system that includes the approximate phoneme duration model to several other systems. Firstly, we want to see how well it performs against the same model that instead of the duration model, uses ground truth phoneme durations. This would correspond to a kind of upper bound of how well the system would perform if the duration model was perfect, and at the same time serves as a way to evaluate the importance of the accuracy of the duration model; i.e., if the difference between ground truth phoneme durations and approximate phoneme durations from a very simple model is small, this means that the importance of the accuracy of the duration model is low, probably because the self-attention layers can effectively compensate for any errors. Secondly, we compare to identical models with duration model and with ground truth durations, but without self-attention layers. This ablation study will hopefully evaluate the effectiveness of self-attention, in the case of accurate phoneme timings and in the case of inaccurate phoneme timings. Thirdly, we compare to an autoregressive baseline, which uses ground truth phonetic durations, just to get a sense of performance compared to state of the art alternative systems.

FFT-NPSS This is our proposed model, using phoneme timings predicted by the approximate phoneme duration model and given note onsets. All hyperparameters are identical to those described in Table 5.1 of §5.3.3 for the network architecture, and §8.1.2 for the heuristic approximate duration model.

FFT-NPSS-D This is the version of our proposed model which uses ground truth phoneme timings. This could be seen as the upper bound quality achievable with an ideal phoneme duration model. This system is identical to that described in §5.3.

FFT-NPSS-NoSA This is a version of our proposed model, with an approximate phoneme duration model, but without self-attention layers. This system is included to perform an ablation study about the importance of self-attention in correcting for mis-aligned control features.

FFT-NPSS-D-NoSA This is a version of our proposed model, with ground truth phoneme durations, but without self-attention layers. This system is included to

Table 8.1: Mean opinion score (MOS) for evaluating the proposed non-autoregressive timbre model using only note onsets, rather than onsets for all phonemes. Ratings on a 1–5 scale with their respective 95% confidence intervals.

System	Mean opinion score
Hidden reference	4.56 ± 0.07
AR-NPSS	2.63 ± 0.10
FFT-NPSS (proposed)	2.85 ± 0.11
FFT-NPSS-NoSA (w/o self-attention)	2.50 ± 0.10
FFT-NPSS-D (ground truth dur.)	$\mathbf{2.92 \pm 0.10}$
FFT-NPSS-D-NoSA (ground truth dur., w/o self-attention)	2.53 ± 0.11

have a reference of how well the system performs with ideal phoneme timings, but no self-attention layers. This system is identical to that described in §5.3.

AR-NPSS An autoregressive baseline system, representative of state of the art singing synthesis at the time. This system is identical to that described in §5.3.

8.3.2 Model hyperparameters

The model hyperparameters of the network architectures for FFT-NPSS and FFT-NPSS-NoSA are identical to those of FFT-NPSS-D and FFT-NPSS-D-NoSA, described in §5.3. These are listed in Table 5.1 of §5.3.3. A few additional hyperparameters related to the approximate phoneme duration model are described in §8.1.2. The system AR-NPSS is unchanged with respect to that described in Chapter 5, where its hyperparameters are listed in Table 5.2.

8.4 Results

The results of our listening test are shown in Table 8.1. As expected, the model using ground truth phoneme durations, FFT-NPSS-D, outperforms our proposed method with a simplistic approximate phoneme duration model, FFT-NPSS. However, the difference is arguably small, indicating that the accuracy of the phoneme duration model is not very important. Viewed in a different way, by improving the phonetic duration model, our proposed system, FFT-NPSS, may reach scores very close to that of the reference using ground truth durations, FFT-NPSS-D.

When comparing the proposed system, FFT-NPSS, to its equivalent without self-attention, FFT-NPSS-NoSA, and likewise the system using ground truth durations, FFT-NPSS-D, to its equivalent without self-attention, FFT-NPSS-D-NoSA, we can see that removing the self-attention layers from the model leads to a significant drop in performance in both cases. However, the rating is quite similar for models without self-attention, regardless of whether ground truth phoneme durations or the approximate phoneme duration model is used. This seems to indicate that the performance gap when removing self-attention is dominated by the lack of timbrical coherence over time, rather than artifacts introduced by mis-aligned control features. That said, we did observe that using self-attention does to some degree allow to correct errors in the initial time-alignment provided by the approximate phoneme duration model. For instance, the duration of phrase-final consonants tends to be systematically underpredicted by the average phoneme durations. However, when using self-attention, these phonemes have a duration much closer to the reference recording.

Our model also outperforms the baseline autoregressive model, AR-NPSS. Here we should mention the same caveats apply as those mentioned in §5.4. That is, our model can at least be competitive with autoregressive models, but we can probably not generalize the conclusion that our model is always better than all autoregressive models.

8.5 Conclusions

We presented a singing synthesizer based on the Transformer model, with a practical Seq2Seq mechanism allowing feed-forward operation. This approach allows training models from just audio, phonetic transcriptions, and note (or vowel) onsets, rather than the precise onset of each phoneme, which can be cumbersome to prepare for singing data. Compared to a baseline autoregressive model, the proposed model allows for faster inference, avoids issues related to exposure bias, and rates somewhat better in listening tests. The use of self-attention resulted to be a key factor in obtaining good-quality results, especially in terms of producing coherent timbre. As our model relies on an initial alignment provided by a phoneme duration model, we compared using a very simplistic duration model to using ground truth durations. In listening tests, using ground truth durations was rated highest, but the difference was relatively small, indicating that the accuracy of the phoneme duration model is not crucial for obtaining good results.

9 Creating voices without any annotations using semi-supervised timbre modeling

In the previous chapters we have presented some techniques to reduce the effort required to create new voices to use in singing synthesis. In Chapter 7, voice cloning techniques are used to significantly reduce the amounts of training data needed (e.g., to 3 min), while producing results of comparable quality to models trained on full datasets (e.g., 1 h). However, annotating all phonemes and their onsets in even such a small set of recordings often still tends to be a non-trivial amount of work[1]. Additionally, as singing synthesis systems improve, it is likely that the data requirements for cloning will grow. For instance, when cloning not just timbre, but also pitch and timing, or cloning a wider range of timbres and singing styles, etc.

Alternatively, in Chapter 8 we propose a system that can be trained from audio, phonetic transcription and vowel onsets only. This implies a notable reduction in the voice creation effort, compared to annotating the onsets of all phonemes. However, the annotation effort is still non-zero, and generally has to be done by hand for good accuracy. Additionally, while the underlying idea of using a sequence-to-sequence (Seq2Seq) model is fairly universal, the implementation of this idea is fairly specific to the model and architecture used. For instance, the non-autoregressive approach from Chapter 8 cannot be directly applied to the autoregressive model from Chapter 4.

What the above approaches have in common is that they are still *supervised* approaches, that is, we require audio and some form of annotations to train the model. Ideally, we would have a system that is *unsupervised*, where we can train voices from audio only. This would greatly reduce the effort required to create new voices, allowing us to easily model a wide array of vocal timbres, which can be useful for many creative applications, and in cases such as choir synthesis. While data efficiency is still an important aspect, i.e., for certain applications we may not have a lot of data, a big advantage of unsupervised training is that also opens a pathway to more easily creating bigger datasets, as recording data is easy compared to annotating them.

While less straightforward than supervised training, unsupervised modeling of the voice is definitely within the realm of possibilities (e.g., van den Oord et al., 2017).

[1]Typically, the target data used for voice cloning will exclude any silences, avoid repeated lyrics, and use moderately high tempos. As such, the amount of phonemes that need to be annotated still tends to be considerable. When using expressive natural singing voice cloning target data, these annotations generally need to be done or corrected by hand in order to have sufficient accuracy.

However, in our case we not only want to train our model on audio data without any annotations such as timed phonemes, we also want to be able to control the synthesizer using precisely those kinds of annotations at inference. Satisfying these two, seemingly orthogonal goals simultaneously in a completely unsupervised manner seems very hard, if not downright impossible. Therefore, we choose to use a *semi-supervised* approach, which combines both supervised training on annotated data, required for allowing controllable inference, and also unsupervised training on data without annotations, to learn new timbres with very low effort. Before going into details of the method, the key point is that the supervised training is independent of the unsupervised training, and can be made speaker-independent when using training data of a wide range of speakers. As a result, while the system as a whole is semi-supervised, once the supervised training is completed, the system is able to learn new voices in a fully unsupervised way.

The content of this chapter was originally published as Bonada and Blaauw (2021).

9.1 Proposed system

Our proposed model follows an encoder-decoder structure, similar to that used in e.g., §5.1.4 and many others. In a conventional encoder-decoder model, the encoder takes linguistic input and produces some hidden features (or embeddings), which are then taken by the decoder to produce the acoustic output. The key difference of our model is that we have a secondary encoder that takes acoustic input. Thus, the model now has two pathways; linguistic-to-acoustic (similar to conventional text-to-speech (TTS)) and acoustic-to-acoustic (similar to an autoencoder), with a single, shared decoder. The idea behind this approach is that we can now use supervised training (i.e., on <acoustic, linguistic> pairs) to ensure that both encoders produce similar hidden features for matching inputs. At the same time, these hidden features should still be useful for generating acoustic features by the decoder, while at the same time being speaker independent. In other words, we expect the acoustic encoder to produce hidden features similar to those produced by the linguistic encoder (which are inherently speaker independent). Once the encoders are trained in a supervised and speaker independent manner, e.g., using an annotated multi-speaker dataset, we can train a new decoder for an unseen target speaker in a fully unsupervised way (using audio data only) using the pre-trained acoustic encoder. Similarly, to control this newly trained decoder for the target speaker from linguistic inputs, we can use the pre-trained linguistic encoder at inference.

The three main stages of operation of our model are summarized in Figure 9.1. In a first step (Figure 9.1a), the system is trained in a supervised manner, using an annotated multi-speaker dataset, i.e., consisting of <acoustic, linguistic> pairs. Both encoders are trained to produce similar embeddings, which should still produce good acoustic output by the decoder (given a speaker embedding). Once this initial model is converged, the decoder is discarded and the weights of the encoders are frozen (i.e., no longer updated in subsequent steps). In a second step (Figure 9.1b), a new decoder for the target speaker is trained in a fully unsupervised way, using the pre-trained acoustic encoder. That is, the decoder is trained to reconstruct audio examples of the target speaker, from the embeddings produced by the pre-trained acoustic encoder. Finally, at inference (Figure 9.1c), the pre-trained linguistic encoder is combined with the newly trained decoder for the target speaker. This allows controlling the synthesis using linguistic features, similar to conventional TTS or singing synthesis.

Both encoders and the decoder are based on the same building block; a WaveNet architecture (van den Oord et al., 2016a) consisting of a set of dilated 1-d convolutional layers featuring gated units, residual shortcut connections and skips, and with the skip sum feeding an output stack of two convolutional layers. Thus, the decoder alone is very similar to our initial autoregressive model proposed in Chapter 4.

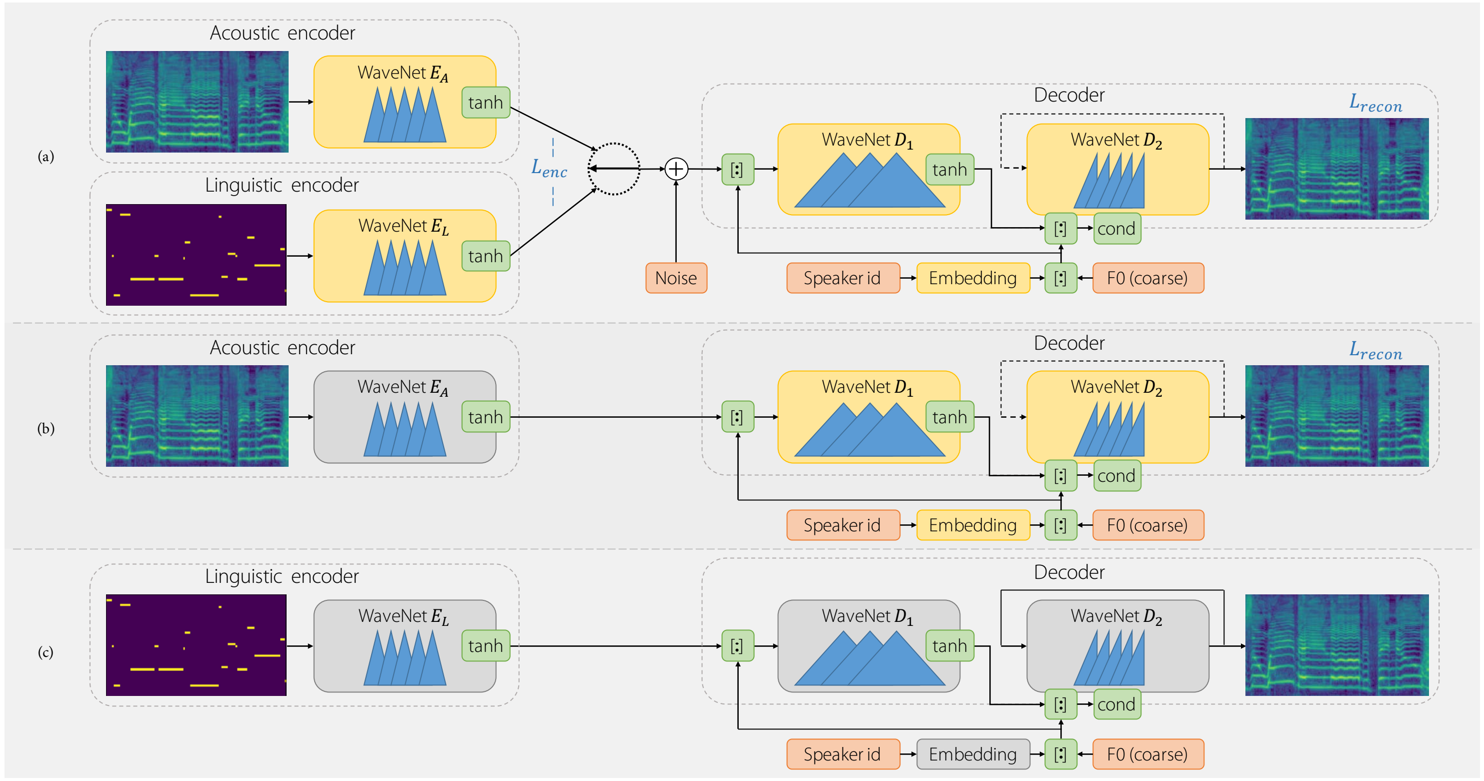

Figure 9.1: A diagram of the model architecture in three different phases: (a) Training the encoder-decoder from annotated audio (supervised). (b) Training the decoder from audio (unsupervised). (c) Inference from linguistic features. Gray-colored modules indicate their weights are kept fixed. The shape of the triangles in the WaveNet blocks represents the size of the receptive field and whether it is causal. A dashed autoregressive connection in the $D_2(\cdot, \cdot)$ WaveNet block indicates teacher forced training with additive noise to avoid overfitting, while a solid connection indicates true autoregressive inference.

9.1.1 Encoder

Our system combines an acoustic and a linguistic encoder that are trained to produce similar embeddings when computed from an annotated singing voice. Both encoders are non-causal and share the same structure and hyperparameters. The acoustic encoder, $E_A(\cdot)$, takes as input a mel-spectrogram, while the linguistic encoder, $E_L(\cdot)$, takes as input a timed phonetic sequence, as a frame-wise sequence of one-hot phoneme encodings. The encoders have a rather short receptive field of few hundred milliseconds. The reason is that our aim is not to capture large-scale variations, but to focus on a rather short scale, and get embeddings closer to what might be a phonetic transcription. Modeling longer phonetic sequences or clusters is a task left for the decoder.

Bottleneck and stochastic switch

In order to favor the encoders producing similar embeddings from matching acoustic and linguistic data, inspired by (Wan et al., 2017), we randomly switch between acoustic and linguistic encoders during training. Additionally, we add noise to the encoder output (after the non-linearity) as a bottleneck, with the idea that the encoder should not encode mel-spectrogram details, also to encourage more stable embeddings along phonemes. A similar bottleneck is used in (Salakhutdinov and Hinton, 2009; Kaiser and Bengio, 2018), where additive noise is added to the embeddings, however in this case before the non-linearity for saturating it and producing more binary-like embeddings, which is not our goal.

9.1.2 Decoder

The decoder is divided into a long-scope and a short-scope network. The first network, $D_1(\cdot)$, focuses on capturing the encoded timbre variations at a large scale of a few seconds, while the second network, $D_2(\cdot,\cdot)$, focuses on producing a detailed timbre output. Vowels can last for several seconds in singing, thus for capturing the phonetic context, a large receptive field, like that of $D_1(\cdot)$, is required. $D_1(\cdot)$ is a non-causal, non-autoregressive convolutional network and receives the encoder output concatenated with F0 and speaker embedding. $D_2(\cdot,\cdot)$ is a causal, autoregressive convolutional network with a short receptive field (less than 200 ms) that produces the final mel-spectrogram. For the latter network, autoregressive and control inputs are combined as in §4.1.2. In our case, the input is the previous mel-spectrogram, and the control input is the output of $D_1(\cdot)$ concatenated with F0 and speaker embeddings.

9.1.3 Training loss

The training loss, L, is a weighted sum of two components,

$$L = \lambda_{\text{recon}} L_{\text{recon}} + \lambda_{\text{enc}} L_{\text{enc}}. \tag{9.1}$$

Here, L_{enc} is the (mean squared) L_2 distance between acoustic and linguistic embeddings, produced by acoustic and linguistic encoders, $E_A(\cdot)$ and $E_L(\cdot)$, from aligned acoustic and linguistic input features, $\mathbf{x}$ and $\mathbf{y}$ respectively,

$$L_{\text{enc}} = \frac{1}{TN_{\text{emb}}} \| E_A(\mathbf{x}) - E_L(\mathbf{y}) \|_2^2, \tag{9.2}$$

where T is the number of timesteps and N_{emb} is the dimensionality of the embeddings. We do this in order to constraint the system to produce similar embeddings from either type of input feature. The reconstruction loss, L_{recon}, is the (mean squared) L_2 distance between the output of the decoder, $\hat{\mathbf{x}}$, and the ground truth acoustic features, $\mathbf{x}$,

$$L_{\text{recon}} = \frac{1}{TN_{\text{feat}}} \| \mathbf{x} - \hat{\mathbf{x}} \|_2^2, \tag{9.3}$$

where T is the number of timesteps and N_{feat} is the dimensionality of the acoustic features. To compute the decoder output, $\hat{\mathbf{x}}$, we first compute its input embedding as a random switch between the acoustic and linguistic embeddings,

$$\mathbf{e} = kE_A(\mathbf{x}) + (1-k)E_L(\mathbf{y}), \tag{9.4}$$

where $k \sim$ Bern (i.e., randomly switch each sample in the minibatch). Then random noise, $\epsilon_1 \sim \mathcal{N}(0, \sigma_1^2 I)$, is added to the selected embedding, we concatenate with control features, $\mathbf{c}$, derived from speaker embedding and F0, and feed the result to the first (non-causal) decoder network, $D_1(\cdot)$. The output of this first decoder network is then concatenated with $\mathbf{c}$ and used as a control input on which the second, autoregressive decoder network, $D_2(\cdot, \cdot)$, is conditioned. During the training of $D_2(\cdot, \cdot)$ we use teacher forcing, where past timesteps are ground truth acoustic features with added noise, $\epsilon_2 \sim \mathcal{N}(0, \sigma_2^2 I)$, to reduce overfitting. Thus, the resulting computation becomes,

$$\tilde{\mathbf{e}} = \begin{bmatrix} \mathbf{e} + \epsilon_1 \\ \mathbf{c} \end{bmatrix} \tag{9.5}$$

$$\tilde{\mathbf{c}} = \begin{bmatrix} D_1(\tilde{\mathbf{e}}) \\ \mathbf{c} \end{bmatrix} \tag{9.6}$$

$$\hat{\mathbf{x}}_t = D_2(\mathbf{x}_{<t} + \epsilon_2, \tilde{\mathbf{c}}). \tag{9.7}$$

The constants, λ_{recon}, λ_{enc}, σ_1, and σ_2, are defined in §9.3.3.

9.1.4 Data augmentation for improved invariance

When training our system, we randomly transpose the pitch of the acoustic input without informing the acoustic encoder of the actual transposition factor. This transposition is performed by combining resampling of the input audio signal with time-scaling of the corresponding mel-spectrogram (repeating or dropping frames). This transformation modifies the pitch of the signal but also linearly scales the timbre in frequency. Since both acoustic and linguistic embeddings are constrained to be similar, and the linguistic encoder does not depend on the pitch transposition factor, then transposing the acoustic input helps to produce a more speaker and pitch independent embedding.

9.2 Relation to prior work

While semi-supervised singing synthesis has not been widely studied, the task of non-parallel voice conversion is closely related. Most of these approaches try to extract the linguistic content from a given audio signal, independently of factors such as speaker identity, pitch, loudness, and so on. The majority of approaches are based on the autoencoder, with several reoccurring themes that ensure only linguistic content is encoded. One such theme is to use information-restrictive bottleneck, e.g., by temporal downsampling (Qian et al., 2019), carefully selected dimensionality (Qian et al., 2019), variational regularization (Luo et al., 2020), or vector quantization (van den Oord et al., 2017). This is often combined with a decoder conditioned on non-linguistic factors, such as speaker embedding (Qian et al., 2019), or F0 (Qian et al., 2020a; Polyak et al., 2020). Data augmentation can be used to make the encoder more invariant to aspects we do not wish to encode, e.g., using pitch shifting or time stretching (Qian et al., 2020a; Qian et al., 2020b). The negative gradient of an auxiliary classifier can be used to reduce undesirable information in the bottleneck, e.g., a speaker classifier to reduce information related to speaker identity (Nachmani and Wolf, 2019). Cycle-consistency is another common theme, which relies on the fact that conversion to another speaker identity and back should result in a similar output (Kaneko and Kameoka, 2018; Kaneko et al., 2019a; Kameoka et al., 2018; Kaneko et al., 2019b). This technique is often combined with adversarial training, which is also applied in a number of other approaches (Polyak et al., 2020). Backtranslation techniques can be used to generate parallel training data (Nachmani and Wolf, 2019; Polyak et al., 2020). Finally, using a phonetic recognizer trained on audio with transcription can aid in extracting content features, albeit in a more supervised manner (Polyak et al., 2020; Sun et al., 2016).

In singing synthesis, several works aim to go towards a reduction in the burden of dataset annotation. As discussed in Chapter 8, Seq2Seq models generally avoid the need for detailed phonetic segmentation, but do require a fairly well aligned musical score with lyrics (e.g., Lee et al., 2019; Angelini et al., 2020; Gu et al., 2020; Wu and Luan, 2020; Lu et al., 2020; Chen et al., 2020). Similarly, voice cloning techniques, like those discussed in Chapter 7, require only a small amount of training data with phonetic segmentation for the target voice (e.g., 3 min versus an hour or more). However, this limited data regime may in some cases ultimately affect sound quality, and still requires some annotation effort. Finally, some work has been done to generate voices from data mined from the web (audio and lyrics) in a completely automatic manner by aligning lyrics to audio (Ren et al., 2020).

9.3 Experiments

9.3.1 Datasets

For the experiments in this chapter, we use two proprietary datasets. For training the encoders we use a dataset of 7 native English singers (5 h 47), which we label EN-MULTI-Y7-N, with approximately 10 songs per singer (one used for validation, the rest for training). The audio files were phonetically segmented with manual corrections. For training the decoders we use a dataset of 41 pop songs performed by a professional English male singer, labeled EN-FULL-PS-V2-N. From this dataset 38 songs were used for training (2 h 7 total), 3 for validation (10 min).

9.3.2 Compared systems

We compare our proposed semi-supervised model to a similar supervised model, being the only difference between both systems is that the supervised model does not have an acoustic encoder. Thus, it learns to predict acoustic features from the input linguistic features using an annotated dataset of the target singer. For the semi-supervised case, we first train the encoder-decoder with EN-MULTI-Y7-N (as in Figure 9.1a), and next retrain the decoder with EN-FULL-PS-V2-N (as in Figure 9.1b). For the supervised model, we train the encoder-decoder directly with EN-FULL-PS-V2-N, as in Figure 9.1a but without the acoustic encoder.

Additionally, we also evaluate the case of using only a small dataset of training data for the target voice (a 3 min subset of EN-FULL-PS-V2-N), labeled EN-TAR-PS-V2-N. We compare our proposed semi-supervised cloning approach to a supervised approach. For the semi-supervised case, we first train the encoder-decoder with EN-MULTI-Y7-N, and next we fine-tune the model with EN-TAR-PS-V2-N for a few thousand updates without using dataset annotations (as in Figure 9.1b). For the supervised cloning case, we first train the supervised encoder-decoder with EN-MULTI-Y7-N, and then fine-tune the model with annotated EN-TAR-PS-V2-N for a few thousand updates.

9.3.3 Model hyperparameters

Our proposed system uses 100-dimensional mel-spectrogram acoustic features, extracted with a 45 ms window and a 5 ms hop time, and computed between 10–15,200 Hz. Linguistic features are computed with 1-hot encodings using 43 phonetic symbols. The encoder networks $E_A(\cdot)$ and $E_L(\cdot)$ have 9 non-causal 1-d convolutional layers (3×1), with dilation factors $[1, 2, 4, 1, 2, 4, 1, 2, 4]$, and 70 residual channels. A leaky rectified linear unit (ReLU) activation follows the skip sum, and then the output stack has 2 convolutional layers with 120 channels, and leaky ReLU and tanh activations respectively. The first block of the decoder $D_1(\cdot)$ contains 10 non-causal 1-d convolutional layers (3×1), dilation factors $[1, 2, 4, 1, 2, 4, 1, 2, 4, 1]$, and 70 residual channels. The output stack has the same configuration as for the encoders. Finally, the second block of the decoder $D_2(\cdot, \cdot)$ has 8 causal 1-d convolutional layers (2×1), 200 residual channels, and dilation factors $[1, 2, 4, 8, 16, 1, 2, 4]$. The skip sum is followed by a leaky ReLU activation. The first convolution in the output stack has 200 channels and leaky ReLU activation, and the second one directly predicts the output mel-spectrogram. All leaky ReLU activations use $\alpha = 0.2$. Speaker embeddings are computed as one-hot encodings followed by a 1×1 convolution with 16 channels.

We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and a batch size of 12. We follow the learning rate schedule from (Vaswani et al., 2017), with a 700-step warm-up, a base learning rate of 5×10^{-4}, a decay rate of 0.15 every 10,000 steps. The objective that we optimize is a (mean squared) L_2 loss between output and target features. When training encoders, we use an additional (mean squared) L_2 loss between acoustic and linguistic encoded features, with a weighting of $\lambda_{enc} = 0.2$ and $\lambda_{recon} = 1$. In addition, we add normal noise with $\sigma_1 = 0.3$ to the embeddings, and with $\sigma_2 = 0.2$ to the $D_2(\cdot, \cdot)$ input. Each sample in the minibatch produces a valid output length of 1.5 s (i.e., excluding the receptive field number of outputs computed using padding).

9.3.4 Listening test

We ran a mean opinion score (MOS) listening test with 12 participants, which each rated a random subset of 12 out of 24 phrases. Per test 6 stimuli were presented; the 4 systems mentioned previously, and visible and hidden references consisting of a re-synthesis

Table 9.1: Mean opinion score (MOS) ratings on a 1–5 scale with their respective 95% confidence intervals.

System	Mean opinion score
Hidden reference	4.80 ± 0.05
Supervised	$\mathbf{3.42 \pm 0.12}$
Semi-supervised	3.37 ± 0.11
Supervised cloning	2.66 ± 0.12
Semi-supervised cloning	2.80 ± 0.11

of the target recording. All systems were presented and rated together to encourage a comparison between them. THe final waveform was generated with a mel-spectrogram driven neural vocoder (Tamamori et al., 2017) trained with EN-FULL-PS-V2-N. Sound examples are available online[2].

9.4 Results

The results of our listening tests are shown in Table 9.1. We can see that the semi-supervised and the supervised systems perform similarly, without a very significance difference. Both systems outperform the cloning approaches, probably due to the small amount of target data available for those. Finally, the semi-supervised cloning system is rated slightly better than the supervised one.

9.5 Conclusions

In this chapter, we have proposed a semi-supervised method for learning a new voice timbre model from a singing dataset without any annotations. Our system produces a synthetic acoustic rendition given F0 and a timed phonetic sequence as input. According to our evaluation results, our proposed system performs similarly when compared to an equivalent supervised system using manually corrected annotations. This means that we can effectively reduce the effort to learn a new voice, by removing the dataset annotation task, and without significantly degrading the synthesis quality. This method could be very useful in the context of choir singing, allowing to model many singers without the dataset annotation burden. Also, we showed that the proposed method performs acceptably in low-resource scenarios, where we only have access to a small amount of acapella audio material.

At inference, we can produce acoustic features from linguistic or acoustic inputs. In some informal experiments, we observed that our approach can be effectively used as a

[2]https://mtg.github.io/singing-synthesis-demos/semisupervised/

voice conversion system when controlled by acoustic inputs, performing promisingly in cross-lingual scenarios.

Conclusions 10

10.1 Summary

One of the aims of our research was to answer whether deep learning approaches for modeling timbre in singing synthesis could equal or exceed more traditional approaches. In particular, using concatenative synthesis and hidden Markov model synthesis as baselines, which were the prevailing methods when starting this work. We first focus on a modern deep learning approach based on autoregressive modeling and a powerful dilated convolutional architecture. Through qualitative and quantitative evaluation, we show that this approach can outperform the then state of the art in terms of sound quality, concatenative synthesis, in particular for more "phonetically rich" languages such as English. Combining a neural timbre model with a neural vocoder to produce the final waveform further increases the obtained results. At the same time, this approach offers all of the flexibility of the competing machine learning approach, hidden Markov model synthesis. Notably, we can train a model on natural singing rather than the specialized recordings required for concatenative synthesis.

Closely related to the above aim, we also try to answer whether singing synthesis based on deep learning can be fast and stable, which are important qualities in many applications. Notably, autoregressive models, while powerful, tend to have slow inference due to the inherently sequential process which cannot be parallelized. Additionally, the recurrent feedback connection of autoregressive models makes them inherently less stable compared to feed-forward approaches. This is compounded by the fact that there is a discrepancy between training, where past timesteps come from the ground truth training data, and inference, where past timesteps are themselves predictions made by the model. This can result in so-called exposure bias issues where a neural network overfits to the training condition and cannot generalize well to the inference condition. Non-autoregressive models solve both of these problems, but generally tend to underperform compared to autoregressive timbre models and neural vocoders. In our experiments, we combine a timbre model based on a feed-forward convolutional neural network with self-attention, a way to integrate information across all timesteps in a sequence. Through listening tests, we show that this approach improves the performance of non-autoregressive models to be similar to their autoregressive counterparts. In particular, this approach improves the coherence of timbre over time, which is one of the weak points of more vanilla feed-forward approaches. For waveform generation, where parallelization can be especially beneficial, we propose a model based on a fairly constrained source-filter model. In this model, neural networks are only used to predict a minimum-phase vocal tract filter, and an aperiodicity filter, which determines the mixture of periodic and aperiodic source signals for any given frequency. Through listening tests, we show that this approach can outperform traditional vocoders based on signal processing alone, as well as avoid many of the common artifacts of less constrained non-autoregressive neural vocoders when applied to singing voice. Compared to autoregressive neural vocoders, performance is close, albeit slightly lower, in particular for very low pitches and non-model voice qualities. Combining these non-autoregressive models, we end up with a system that can perform inference several orders of magnitude faster than autoregressive models, as well as avoid being affected by any issues related to exposure bias.

The second major aim of this work is to answer whether the additional flexibility that deep learning approaches provide could be utilized to reduce the effort required to create new voices, and possibly also reducing the amount of data required. The effort required to create new voices in singing synthesis tends to form a bottleneck for many applications, as well as research on the topic itself. Particularly problematic are the annotations that are required such as phonetic segmentation, score transcription, and so on. These not only can take a significant amount of time, they generally also require expert knowledge. In our experiments, we have shown that voice cloning techniques from text-to-speech are also effective in singing synthesis. At the cost of a small reduction in fidelity, timbres can be modeled using datasets an order of magnitude smaller in size, by leveraging not only data of the target voice, but also data from other singers. Alternatively, or complementary, utilizing self-attention, we obtained a timbre model that does not require precise phonetic segmentation to be trained. Instead, this approach only requires vowel onsets and a rough initial segmentation in phonemes via a generic phoneme duration model. Again, according to the results of our listening tests, while this approach results in a small reduction in fidelity, this may be an acceptable compromise in order to reduce the voice creation effort. Taking this line of research one step further, we introduced a semi-supervised training approach. Unlike the previous approach, this does require an initial multi-speaker dataset with full phonetic segmentation. However, once this data is used to train some speaker-independent components, a new voice can be trained in a completely unsupervised manner. That is, no annotations are required, just audio. While also resulting in a very small degradation in quality, we feel that the benefits of this approach significantly outweigh the potential drawbacks. Semi-supervised training truly allows for creating new voices practically instantly. This not only allows rapidly creating voices for artistic endeavors or productization, it also allows singing synthesis research to be done at a much larger scale, e.g., evaluating systems on a wider range of voices, singing styles and languages.

10.2 Limitations of our research and future work

We feel that the main limitation of our research is its scope. We chose to limit our research to modeling timbre, however, this is only one of several aspects of singing synthesis. While we briefly discuss the additional components required to build a complete singing synthesizer, such as pitch and timing models, this is still an area that could be explored much more in-depth. Although many of the recent works on singing synthesis do include such components to some degree, we feel some more large-scale evaluation and comparison of different approaches is still somewhat lacking. For instance, synthesizers are often evaluated on a single dataset, thus typically only giving results for a single language and singing style.

This brings us to another limitation of our research, and singing synthesis research in general; evaluation is very difficult. One of the reasons we limit our scope to modeling timbre, is that evaluating for instance pitch or timing is notably more difficult. As a rule of thumb, objective metrics are not very reliable for comparing singing synthesis systems. Even subjective evaluation, which is the current "gold standard" method of evaluation, can be somewhat problematic, as for instance the meaning of "poor" can vary greatly between participants. We hope that ongoing work on objective evaluation based on deep learning will alleviate these issues in the future.

Besides more elementary pitch and timing components, expressive singing of course encompasses a much wider range of aspects that have not been investigated in much detail. For instance, it is not clear to what degree current models can model more long-term expressive aspects such as dynamics, phrasing, or for instance performing a chorus and verse in contrasting manners. Singing voice models currently typically only model a single timbre or singing style, while many singers may be able to perform several, often combining these in a single song. In our case, we only focus on pop, pop/rock, and classical or choir singing styles. Clearly, this is only a small fraction of the total range of singing styles, some of which are significantly different, such as rap. Another topic that lacks research is the synthesis of non-modal voice qualities. These are fairly common in singing, but tend to be problematic for several reasons. In datasets, such non-modal resources tend to be very sparse, which makes data-driven modeling more difficult. Their waveform and spectral properties can be very different compared to modal singing, e.g., including time-varying modulations, and non-trivial modification of formants. Finally, when and how to apply such non-modal resources given an input score is not obvious, and may for instance depend on the semantic content of the lyrics.

In our view of a singing synthesizer, we consider it a model that outputs a waveform given an input score with lyrics. This view is an analogy of the process a real singer may perform; interpret a score and render the acoustic performance. However, this view is not entirely accurate. A real singer is not a black box whose only input is the musical score. In reality, a performance may be influenced by a wide range of things; obviously the music, possibly other musicians in a live setting, cultural and social context, the setting of the recording session, explicit instructions from a producer, and so on. Similarly, in singing synthesis, we often do not just want a black box that outputs a rendition of a score in a given singing style, we typically want additional user control. Exactly what this user control should be, and how it should affect the model is also important future work in our opinion.

10.3 Final thoughts

Deep learning in general and text-to-speech using deep learning are fields of research that are progressing at a breakneck pace. During the course of this thesis singing synthesis research has started moving at similar speeds, which is a stark contrast to the decades prior. We can notice the field is growing, with a notable uptick in publications on singing synthesis, and more and more institutions and companies starting lines of research on the topic. In our opinion, this is one of the greatest contributions that deep learning has made to the field. We could argue that deep learning has in a way lowered the barrier of entry to singing synthesis research. That is, it no longer relies on complicated signal processing algorithms that take years to develop, or require painstaking amounts of detailed annotations of singing data. At the same time, public source code and datasets are becoming more and more common. All of this makes has led to an increase in expected baseline quality in singing synthesis.

In a way, this work sits in a kind of transitional phase of the field. When we started our work, whether deep learning approaches could outperform, e.g., concatenative synthesis for singing was still very much up for debate. The results from WaveNet had just shown what was possible in text-to-speech. While closely related, singing synthesis is not exactly the same as text-to-speech. Thus, for instance having to cover a much wider pitch range, while simultaneously having to work with much smaller dataset sizes, made the answer to this question not immediately obvious. While all the models we describe in this work can still be considered "current", questions like these are perhaps not. In fact, deep learning has become so dominant, that previous approaches are no longer considered or used as baselines.

As the technology matures, we expect "elementary" singing synthesis to become kind of a solved problem on the one hand, and much more accessible on the other (i.e., not requiring a large amount of specialized knowledge). We feel that our work has contributed towards such a future. However, at the same time, we are still a long way off from truly natural, truly expressive singing synthesis. Thus, we expect the field to move from more "elementary" topics to more specialized aspects of the singing voice. For this, we consider one promising approach to be (re-)introducing more domain knowledge in the models. Especially considering many practical issues, at some point purely data-driven approaches may reach their limit. For instance, the amounts of data required may be impractical, or the models required as so computationally complex that certain applications would be no longer possible. We expect that in many such cases, machine learning and domain knowledge can be complementary to each other; perhaps we can think of using domain knowledge as a way to regularize powerful deep learning models, allowing us to maintain most or all of the flexibility, while guiding the model towards the results we want to obtain.

10.4 List of contributions

- We have proposed a neural singing synthesizer that predicts vocoder features from linguistic and F0 features using an autoregressive model and a powerful network architecture based on gated dilated convolutions (WaveNet). As this was one of the first models based on modern neural networks, we performed a detailed comparison to existing approaches. Our model was rated notably higher in listening tests than the then state of the art in terms of quality, concatenative synthesis, while providing similar or better flexibility than the prevalent machine learning approach, hidden Markov model synthesis. Details of this model and the experiments are given in §4.2.
- We have extended the above model for synthesizing timbre to a complete synthesizer that can synthesize sung vocals from a score with lyrics, by including pitch and timing models. Using listening tests we show that our autoregressive model can outperform previous approaches based on hidden Markov model synthesis and simpler feed-forward deep neural networks. The proposed synthesizer and the comparison to baseline systems can be found in §4.3.
- We have proposed an alternative, non-autoregressive model, based on the feed-forward Transformer, notably combining a convolutional neural network with self-attention. Compared to the autoregressive model, this approach provides comparable quality, but with faster inference and more stable results as it does not rely on a recurrent feedback connection. We also show that the use of self-attention is beneficial compared to a vanilla convolutional neural network, notably for improving the coherence of timbre over time. Details of this model and the experiments are given in Chapter 5.
- As non-autoregressive modeling is especially beneficial to waveform generation, we have also proposed a non-autoregressive neural vocoder. Combining this

model with the above non-autoregressive timbre model, we can generate the final waveform output of a singing synthesizer in a completely non-autoregressive manner. Compared to competitive approaches, this much more constrained model avoids many of the potential artifacts that can occur with a less constrained neural vocoder. The sound quality obtained using this approach is comparable to a strong autoregressive baseline, while inference is several orders of magnitude faster. Thanks to its fast inference and stable nature, this neural vocoder is an attractive option for many practical applications. This model is described and evaluated in Chapter 6.

- ❧ In order to reduce the amount of data required to create a new voice, we have investigated applying voice cloning techniques initially introduced for text-to-speech to singing synthesis. In particular, focusing on representing the singer identity by an embedding, leveraging big multi-speaker datasets, and fine-tuning pre-trained models. We have investigated how different kinds of singing recordings affect the results of voice cloning. We show that voice cloning can be used to create voices with significantly smaller datasets, while only resulting in a small degradation in performance. Voice cloning for singing synthesis is discussed in Chapter 7.
- ❧ Besides reducing the amount of data required to create a new voice, we also try to reduce the effort needed to prepare the training data. For our non-autoregressive model, we proposed a sequence-to-sequence technique that allows training the model note timings rather than phonetic timings. This approach is based on an approximate phonetic duration model, and self-attention provided by the feed-forward Transformer architecture. This approach is described and evaluated in Chapter 8.
- ❧ Reducing the voice creation effort even further, we propose a semi-supervised training approach. This method consists of a speaker-independent autoencoder trained in a supervised manner, after which a decoder for a new target voice can be trained in a fully unsupervised manner. That is, once the initial supervised model is trained, we can use it to create a timbre model for a new target voice from just audio recordings, without any form of annotations. The results obtained using this approach are comparable to fully supervised models. The semi-supervised training approach is described and evaluated in Chapter 9.

Technical details | A

A.1 Details constrained Gaussian mixture

As discussed in "Output distributions" of §4.1.3, some of our models use a specialized output mixture density that we call constrained Gaussian mixture (CGM). This distribution is a mixture of $K = 4$ Gaussians,

$$p(x) = \sum_{k=0}^{K-1} w_k \, \mathcal{N}(x; \, \mu_k, \sigma_k^2), \tag{A.1}$$

with additional constraints applied on the mixture parameters to only allow a certain subset of distributions that the underlying mixture of Gaussians (MoG) allows.

The 12 mixture parameters w_k, μ_k, σ_k for $k = 0, 1, \dots, K-1$ are computed from four free parameters: location ξ, scale ω, skewness α and shape β (see Figure 4.2 for some example distributions). Assuming the network predicts four outputs with linear activations, a_0, a_1, a_2, a_3, we apply some non-linearities to obtain the free parameters in suitable ranges,

$$\xi = 2\,\mathrm{sigm}(a_0) - 1 \qquad \xi \in [-1, 1] \tag{A.2}$$

$$\omega = \frac{2}{255} e^{4\,\mathrm{sigm}(a_1)} \qquad \omega \in [\frac{2}{255}, \frac{2e^4}{255}] \tag{A.3}$$

$$\alpha = 2\,\mathrm{sigm}(a_2) - 1 \qquad \alpha \in [-1, 1] \tag{A.4}$$

$$\beta = 2\,\mathrm{sigm}(a_3) \qquad \beta \in [0, 2]. \tag{A.5}$$

Then, we map predicted location ξ, scale ω, skewness α and shape β to Gaussian mixture parameters μ_k, σ_k, w_k for $k = 0, 1, \dots, K-1$,

$$\sigma_k = \omega e^{(|\alpha|\gamma_s - 1)k} \tag{A.6}$$

$$\mu_k = \xi + \sum_{i=0}^{k-1} \sigma_k \gamma_u \alpha \tag{A.7}$$

$$w_k = \frac{\alpha^{2k} \beta^k \gamma_w^k}{\sum_{i=0}^{K-1} \alpha^{2i} \beta^i \gamma_w^i}, \tag{A.8}$$

where γ_u, γ_s and γ_w are constants tuned by hand,

$$\gamma_u = 1.6 \tag{A.9}$$

$$\gamma_s = 1.1 \tag{A.10}$$

$$\gamma_w = \frac{1}{1.75}. \tag{A.11}$$

A temperature control is achieved by first shifting component means towards their global weighted average,

$$\bar{\mu} = \sum_{k=0}^{K-1} \mu_k w_k \tag{A.12}$$

$$\hat{\mu}_k = \mu_k + (\bar{\mu} - \mu_k)(1 - \tau), \tag{A.13}$$

where $0 < \tau \leq 1$ is the temperature. Then, the component variances are scaled by the temperature,

$$\hat{\sigma}_k = \sigma_k \sqrt{\tau}. \tag{A.14}$$

A.2 Details tuning post-processing

In our complete autoregressive singing synthesizer (see §4.3), we mentioned that the pitch model includes a tuning correction post-processing step (see "Tuning post-processing" in §4.3.2). The principal idea behind this tuning correction post-processing is simple; apply the difference between the perceived pitch of a note, given its predicted F0 contour, and the pitch of the corresponding note in the score. However, robustly estimating the perceived pitch of a note from the corresponding F0 contour is non-trivial. In singing voice, there are many factors that affect F0, but may not influence the perceived note pitch. These factors include vibratos, scoops, releases, transitions, microprosody due to consonants and so on. Therefore, simple estimators, such as directly taking the mean of the frame-wise F0 over the note duration, will typically yield poor results.

To obtain a more robust estimate of the perceived note pitch, $\bar{\mathbb{c}}$, we compute a weighted average of the predicted frame-wise pitch (log F0 in semitones), $\mathbb{c} = 1200 \log_2(\mathbf{f}_0/440)$, over the note's duration,

$$\bar{\mathbb{c}} = \frac{\sum_i \mathbb{c}_i w_i}{\sum_i w_i}, \tag{A.15}$$

where $\mathbb{c}_i$ and w_i correspond to the i-th frame within a given note of the predicted pitch vector and weighting vector respectively. The weighting vector $\mathbf{w}$ is composed of a number of different factors that correspond to different heuristics designed to make the estimate more robust,

$$\mathbf{w} = \mathbf{w}_e \mathbf{w}_d \mathbf{w}_p \mathbf{w}_t. \tag{A.16}$$

The first of these factors, $\mathbf{w}_e$, is a weighting to reduce the influence of the edges of the note, where most of the transition effects will typically be located. We compute $\mathbf{w}_e$ as a Tukey window with $\alpha = 0.5$. That is, we apply a cosine-taper weighting along the first and last 25% of the note duration.

The second factor, $\mathbf{w}_d$, is a weighting depending on the derivative of the pitch contour. The idea is that the portion of the note where pitch is mostly flat will contribute more to the perceived pitch than portions where pitch fluctuates due to transitions or microprosody. We first estimate the derivative by convolving the signal with a 3rd order 1st derivative Savitzky-Golay finite impulse response (FIR) filter, $\mathbf{s}_d$, with a length of 11 frames (55 ms),

$$d\mathbb{c} := \mathbb{c} * \mathbf{s}_d, \tag{A.17}$$

where $*$ denotes the convolution operator. Then, we compute the weighting factor, $\mathbf{w}_d$, as follows,

$$w_{d,i} = \frac{1}{\min(1 + 27|d\mathbb{c}_i|, 15)}, \tag{A.18}$$

where the constants were obtained empirically.

The third factor, $\mathbf{w}_p$, is a weighting depending on the phoneme corresponding to each frame p_i,

$$w_{p,i} = \begin{cases} 2 & \text{for } p_i \in \{\textit{vowel, syllabic consonant}\} \\ 0 & \text{for } p_i \in \{\textit{silence, pause, breath}\} \\ 1 & \text{otherwise.} \end{cases} \tag{A.19}$$

The idea is that frames corresponding to vowels typically contribute more to the perceived pitch than consonants, which often contain microprosody effects.

The last factor, $\mathbf{w}_t$, is a weighting depending on the distance from the target pitch, based on the assumption that detuning in the perceived pitch will typically be caused by relatively small deviations. Other factors, such as scoops or microprosody, may cause relatively big deviations, but these tend not to contribute to the perceived detuning. We use a pitch deviation of ± 1 semitone as a threshold,

$$w_{t,i} = \begin{cases} 1 & \text{for } |\mathbb{c}_{\text{tar}} - \mathbb{c}_i| \leq 1 \\ 1/|\mathbb{c}_{\text{tar}} - \mathbb{c}_i| & \text{otherwise.} \end{cases} \tag{A.20}$$

Finally, the required amount of (frame-wise) pitch correction, $\Delta\mathbb{c}$, is computed for each frame in a note as follows,

$$\Delta\mathbb{c}_i = \mathbb{c}_{\text{tar}} - \overline{\mathbb{c}}, \tag{A.21}$$

where $\mathbb{c}_{\text{tar}}$ is the note's target pitch, as is written in the score. For rests, we do not apply any correction, $\Delta\mathbb{c}_i = 0$. These frame-wise correction vectors are then concatenated for all notes and rests in the sequence. As the resulting vector may be discontinuous, we smooth it by zero-phase filtering with a Gaussian window with a length of 30 frames (150 ms).

As the above method computes a note-wise correction, it is based on the assumption that the detuning will be approximately constant along a note. However, this is not always the case, especially for longer notes. There can for instance be a pitch trend along a note's duration, which may sound like the singer is slowly trying to reach the correct pitch. To reduce this kind of detuning, we divide longer notes in smaller sub-note segments, and compute the per-segment correction as described above. However, prior to the final smoothing step, instead of a constant correction per segment, we obtain the frame-wise correction by linearly interpolating each segment's correction at its center.

Publications by the author

Journal articles

Blaauw, M., & Bonada, J. (2017). A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs. *Applied Sciences*, *7*(12) Special Issue on Sound and Music Computing. doi:10.3390/app7121313

Peer-reviewed conference articles

Bonada, J., & **Blaauw, M.** (2021). Semi-Supervised Learning for Singing Synthesis Timbre. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 7083–7087). Toronto, ON, Canada.

Blaauw, M., & Bonada, J. (2020). Sequence-to-Sequence Singing Synthesis Using the Feed-Forward Transformer. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7229–7233). Barcelona, Spain.

Bonada, J., & **Blaauw, M.** (2020). Hybrid Neural-Parametric F0 Model for Singing Synthesis. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7244–7248). Barcelona, Spain.

Chandna, P., **Blaauw, M.**, Bonada, J., & Gómez, E. (2020). Content Based Singing Voice Extraction from a Musical Mixture. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 781–785). Barcelona, Spain.

Chandna, P., **Blaauw, M.**, Bonada, J., & Gómez, E. (2019). WGANSing: A Multi-Voice Singing Voice Synthesizer Based on the Wasserstein-GAN. In *Proceedings of the 27th European Signal Processing Conference (EUSIPCO)*, September 2–6, 2019, A Coruña, Spain.

Blaauw, M., & Bonada, J. (2019). Data Efficient Voice Cloning for Neural Singing Synthesis. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 6840–6844). Brighton, UK.

Chandna, P., **Blaauw, M.**, Bonada, J., & Gómez, E. (2019). A Vocoder Based Method for Singing Voice Extraction. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 990–994). Brighton, UK.

Blaauw, M., & Bonada, J. (2017). A Neural Parametric Singing Synthesizer. In *Proceedings of the 18th Annual Conference of the International Speech Communication Association (Interspeech)*, August 20–24, 2017 (pp. 1230–1234). Stockholm, Sweden.

Blaauw, M., & Bonada, J. (2016). Modeling and Transforming Speech Using Variational Autoencoders. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, September 8–12, 2016 (pp. 1770–1774). San Francisco, CA, USA.

Bonada, J., Umbert, M., & **Blaauw, M.** (2016). Expressive Singing Synthesis Based on Unit Selection for the Singing Synthesis Challenge 2016. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, September 8–12, 2016 (pp. 1230–1234). won challenge. San Francisco, CA, USA.

Umbert, M., Bonada, J., & **Blaauw, M.** (2013). Systematic Database Creation for Expressive Singing Voice Synthesis. In *Proceedings of the 8th ISCA Speech Synthesis Workshop (SSW8)*, August 31–September 2, 2013 (pp. 213–216). Barcelona, Spain.

Umbert, M., Bonada, J., & **Blaauw, M.** (2013). Generating Singing Voice Expression Contours Based on Unit Selection. In *Proceedings of the 4th Stockholm Music Acoustics Conference (SMAC)*, July 30–August 3, 2013 (pp. 315–320). Stockholm, Sweden.

Bonada, J., & **Blaauw, M.** (2013). Generation of Growl-Type Voice Qualities by Spectral Morphing. In *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 26–31, 2013 (pp. 213–216). Vancouver, BC, Canada.

Bonada, J., **Blaauw, M.**, & Loscos, A. (2006). Improvements to a Sample-Concatenation Based Singing Voice Synthesizer. In *Proceedings of the 121st AES Convention*, October 5–8, 2006, San Francisco, CA, USA.

Bonada, J., **Blaauw, M.**, Loscos, A., & Kenmochi, H. (2006). Unisong: A Choir Singing Synthesizer. In *Proceedings of the 121st AES Convention*, October 5–8, 2006, San Francisco, CA, USA.

Janer, J., Bonada, J., & **Blaauw, M.** (2006). Performance-Driven Control for Sample-Based Singing Voice Synthesis. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx)*, September 18–20, 2006, Montreal, QC, Canada.

Non peer-reviewed articles

Gómez, E., **Blaauw, M.**, Bonada, J., Chandna, P., & Cuesta, H. (2018). Deep Learning for Singing Processing: Achievements, Challenges and Impact on Singers and Listeners. arXiv: 1807.03046 [cs.SD]

Posters, talks and miscellaneous

Weinberg, G., Savery, R., Yang, N., Zahray, L., **Blaauw, M.**, & Bonada, J. (2020). Shimon: Now a Singing, Songwriting Robot. Retrieved from https://www.news.gatech.edu/2020/02/25/shimon-now-singing-songwriting-robot

Blaauw, M., & Bonada, J. (2019). Recent Work on Neural Singing Synthesis. Poster presented at the BCN Music Technology Forum. Barcelona, Spain.

Blaauw, M., & Bonada, J. (2018). Current Work on Neural Singing Synthesis. Deep Learning Barcelona Symposium. Barcelona, Spain.

Blaauw, M. (2018). Singing Machines: From Early Mechanical Attempts to Methods Based on Machine Learning. Talk at l'Ull Cec, Ex-Designer Project Bar. Barcelona, Spain.

Gómez, E., **Blaauw, M.**, Bonada, J., Chandna, P., & Cuesta, H. (2018). Deep Learning for Singing Processing: Achievements, Challenges and Impact on Singers and Listeners. Keynote speech, 2018 Joint Workshop on Machine Learning for Music. The Federated Artificial Intelligence Meeting (FAIM), a joint workshop program of ICML, IJCAI/ECAI, and AAMAS. Stockholm, Sweden.

Bonada, J., & **Blaauw, M.** (2018). A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs. Invited speaker at the joint 118th Special Interest Group on Music and Computer (SIG-MUS) and 120th Special Interest Group on Spoken Language Processing (SIG-SLP) workshop. Tsukuba, Ibaraki Prefecture, Japan.

Blaauw, M., & Bonada, J. (2016). A Singing Synthesizer Based on PixelCNN. Poster presented at the María de Maeztu Seminar on Music Knowledge Extraction Using Machine Learning (collocated with NIPS). Barcelona, Spain.

Blaauw, M., & Ramos, S. (2015). Singing Voice Conversion Using Layer-Wise Generative Trained DNN. Poster presented at the RTTH Summer School on Speech Technology: A Deep Learning Perspective. Barcelona, Spain.

Patents

Bonada, J., **Blaauw, M.**, & Daido, R. (2020). *Sound Signal Synthesis Method, Generative Model Training Method, Sound Signal Synthesis System, and Program*. WO/2020/171034. Yamaha Corp. Extended to US20210383816, JPWO2020171034.

Bonada, J., **Blaauw, M.**, & Daido, R. (2020). *Sound Signal Synthesis Method, Generative Model Training Method, Sound Signal Synthesis System, and Program*. WO/2020/171033. Yamaha Corp. Extended to US20210375248, JPWO2020171033.

Daido, R., **Blaauw, M.**, & Bonada, J. (2020). *Information Processing Method and Information Processing System*. WO/2020/095950. Yamaha Corp. Extended to US20210256960, EP3879524, JP2020184092, JP2020076843, CN112970058.

Bonada, J., **Blaauw, M.**, Keijiro, S., Daido, R., Wilson, M., & Yuji, H. (2018). *Voice Synthesis Method*. WO/2018/084305. Yamaha Corp. Extended to US20190251950, EP3537432, JPWO2018084305, CN109952609.

Blaauw, M., Bonada, J., Daido, R., & Yuji, H. (2018). *Sound Processing Method, Sound Processing Device, and Program*. WO/2019/181767. Yamaha Corp. Extended to US20210005176, EP3770906, JP2019168542, CN111837184.

Yuji, H., Daido, R., Keijiro, S., Bonada, J., & **Blaauw, M.** (2018). *Voice Synthesizing Device and Voice Synthesizing Method*. WO/2018/003849. Yamaha Corp. Extended to US20190130893, EP3480810, JP2018004870, CN109416911.

Saino, K., Bonada, J., & **Blaauw, M.** (2016). *Voice Synthesis Method, Voice Synthesis Device, Medium for Storing Voice Synthesis Program*. JP2016-161919. Yamaha Corp. Extended to US2016260425, EP3065130, CN105957515.

Bonada, J., **Blaauw, M.**, & Saino, K. (2016). *Voice Processing Method and Apparatus, and Recording Medium Therefor*. JP2016-122157. Yamaha Corp. Extended to US2016189725.

Bonada, J., **Blaauw, M.**, & Hisaminato, Y. (2012). *Voice Synthesis Apparatus: Voice Quality Modification by Spectral Morphing*. JP2012-139455. Yamaha Corp. Extended to US20140006018.

Bonada, J., **Blaauw, M.**, & Tachibana, M. (2012). *Voice Synthesis Apparatus: Phase Model*. JP2012-129798. Yamaha Corp.

Bonada, J., **Blaauw, M.**, & Tachibana, M. (2012). *Voice Synthesis Apparatus: Unit Interpolation*. JP2012-110359. Yamaha Corp. Extended to EP2530671.

Articles on other topics

Bonada, J., Lachlan, R., & **Blaauw, M.** (2016). Bird Song Synthesis Based on Hidden Markov Models. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, September 8–12, 2016 (pp. 2582–2586). San Francisco, CA, USA.

Pérez, A., Bonada, J., Maestre, E., Guaus, E., & **Blaauw, M.** (2012). Performance Control Driven Violin Timbre Model Based on Neural Networks. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, *20*(3), 1007–1021. doi:10.1109/TASL.2011.2170970

Maestre, E., **Blaauw, M.**, Bonada, J., Guaus, E., & Pérez, A. (2010). Statistical Modeling of Bowing Control Applied to Violin Sound Synthesis. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, *18*(4), 855–871. doi:10.1109/TASL.2010.2040783

Guaus, E., Bonada, J., Maestre, E., Pérez, A., & **Blaauw, M.** (2009). Calibration Method to Measure Accurate Bow Force for Real Violin Performances. In *Proceedings of the 2009 International Computer Music Conference (ICMC)*, August 16–21, 2009 (pp. 251–254). Montreal, QC, Canada.

Pérez, A., Bonada, J., Maestre, E., Guaus, E., & **Blaauw, M.** (2008). Score Level Timbre Transformations of Violin Sounds. In *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx)*, September 1–4, 2008, Espoo, Finland.

Pérez, A., Bonada, J., Maestre, E., Guaus, E., & **Blaauw, M.** (2008). Measuring Violin Sound Radiation for Sound Equalization. In *Proceedings of Acoustics'08*, June 29–July 4, 2008, Paris, France.

Guaus, E., Bonada, J., Pérez, A., Maestre, E., & **Blaauw, M.** (2007). Measuring the Bow Pressing Force in a Real Violin Performance. In *Proceedings of the International Symposium on Musical Acoustics (ISMA)*, September 9–12, 2007, Barcelona, Spain.

Pérez, A., Bonada, J., Maestre, E., Guaus, E., & **Blaauw, M.** (2007). Combining Performance Actions with Spectral Models for Violin Sound Transformation. In *Proceedings of the 19th International Congress on Acoustics (ICA)*, September 2–7, 2007 (pp. 1904–1909). Madrid, Spain.

Maestre, E., Bonada, J., **Blaauw, M.**, Pérez, A., & Guaus, E. (2007). Acquisition of Violin Instrumental Gestures Using a Commercial EMF Device. In *Proceedings of the 2007 International Computer Music Conference (ICMC)*, August 27–31, 2007 (pp. 386–393). Copenhagen, Denmark.

Glossary

ACR The Absolute Category Rating is a 1–5 rating scale commonly used in mean opinion score (MOS) listening tests. Values used in this case are: 5 excellent, 4 good, 3 fair, 2 poor, and 1 bad. ↑ 66, 67

aperiodicity Aperiodicity, aperiodicity envelope or band aperiodicity is generally a vector between of values between zero and one, indicating whether a frequency or frequency band is fully voiced (zero), fully aperiodic (unvoiced, noisy; one), or somewhere in between. It can be thought of as a kind of frequency-wise continuous voiced/unvoiced feature. ↑ 93, 94, 97, 98, 100, 108

ASR Automatic Speech Recognition systems automatically transcribe speech from audio. This is sometimes also referred to as "speech-to-text" (especially in more modern deep learning contexts). ↑ 55, 62

BAPD Band Aperiodicity Distortion, also known as band aperiodicity distance, is a distance in dB between two band aperiodicity spectra. ↑ 71, 72, 97, 98, 108, 110

CGM A Constrained Gaussian Mixture (or constrained mixture of Gaussian) is a special case of MoG where the number of free parameters and possible distributions are reduced by a set of heuristics. These heuristics and the term itself are fairly specific to this work and currently not widely used in other works. ↑ 87, 88, 127, 199

CMOS A Comparison Mean Opinion Score is the result of a kind of qualitative evaluation where participants are typically asked to rate stimuli on a -3–+3 scale (much worse, worse, slightly worse, about the same, slightly better better, much better), compared to a reference. *See also:* MOS. ↑ 68

CNN A Convolutional Neural Network is kind of neural network that implements a convolution operation. Traditionally used for image applications due to their translation invariance when used with 2-d spatial data, they can also be an alternative to RNNs for time series. In this case, the main difference is that CNNs integrate a fixed window of past and/or future information, while RNNs in theory integrate any number of past and/or future timesteps. ↑ 26, 28, 49, 51, 83, 115, 117–120, 125, 127, 131

CPU General processing hardware not suited for very parallelized computations. ↑ 28, 90, 116, 125, 155

CTC Connectionist Temporal Classification is a kind of loss between a continuous (unsegmented) time series, e.g., audio features, and a target sequence, e.g., text or phonemes. By summing over the probability of all possible alignments of input to target, it produces a loss value which is differentiable with respect to each input node. A CTC loss is typically used in conjunction with RNNs. ↑ 23

DAEM A variant of the expectation maximization (EM) algorithm which can be used to optimize the parameters of an HMM or HSMM (Ueda and Nakano, 1998). By using deterministic annealing, this approach tends to converge to better maxima, especially from a so-called flat start. As this approach is iterative, one downside is that it does tend to take a longer time compared to standard expectation maximization. ↑ 63, 126

DBN A Deep Belief Network is an early form of deep neural network, where each layer is sequentially trained in an unsupervised manner as an RBM. ↑ 19, 20

DCT The Discrete Cosine Transform expresses a discrete time-domain signal as a sum of cosine functions oscillating at evenly spaced frequencies, with different amplitudes. It is similar to the DFT, with a notable difference that the DCT results in a real sequence, whereas the DFT results in a complex sequence. ↑ 46

DFT The Discrete Fourier Transform expresses a discrete time-domain signal as a discrete frequency domain spectrum, the sum of sinusoids oscillating at evenly spaced frequencies, with different amplitudes and phases. A complex spectrum can also be represented as magnitude and phase spectrum. The inverse of this transform recovers the original signal without loss. ↑ 46, 73

DNN A Deep Neural Network is a neural network with many layers, at least more than one hidden layer. Typically, this refers to a feed-forward neural network. ↑ 20, 21, 105, 106, 115

DTW An efficient algorithm for aligning two time series using Dynamic Programming. ↑ 69

dynamics Dynamics in music and singing is related to the loudness of the signal, and more importantly, the intended intensity of the performer. In particular, dynamics are related to the symbolic dynamic markings in a musical score (discrete labels such as piano, forte, etc.). In this work, we may also use continuous values to indicate dynamics, typically in a range $[0, 1]$. In this work, we use the terms loudness, intensity and dynamics mostly interchangeably. *See also:* . ↑ 93, 162

EM An iterative method to find (local) maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. In this work, a special case of this algorithm, Baum-Welch, is used to optimize the parameters of an HMM. ↑ 63

EMA An exponential moving average, is a first-order infinite impulse response filter, for smoothing time series, by taking into account previous timesteps. It is usually defined in the form of $\overline{x}_t = \alpha x_t + (1-\alpha)\overline{x}_{t-1}$, where the decay $0 \leq \alpha \leq 1$ (higher values discount previous timesteps faster), and $\overline{x}_1 = x_1$. ↑ 128

FDSD A classifier-based learned metric for speech, that also considers the distance between statistics of higher-level features of real and generated speech. The speech equivalent of the Fréchet Inception distance (FID) metric for images. ↑ 76

FFT

1) The Fast Fourier Transform is an efficient algorithm to compute the DFT. ↑ 41, 143, 145

2) Feed-Forward Transformer, a non-autoregressive (feed-forward) version of the Transformer model. ↑ 123, 127, 174, 175

FID An improved version of the classifier-based Inception score (IS) metric for images, which additionally considers the distance between statistics of higher-level features of real and generated images. ↑ 76

FIR A Finite Impulse Response filter is a digital filter with a finite impulse response. This is in contrast to Infinite Impulse Response (IIR) filters whose impulse response continues indefinitely. FIR filters are formulated as a weighted sum of different inputs, while IIR filters are formulated as a weighted sum of inputs as well as previous outputs (hence a recurrence). ↑ 143, 155, 201

FNR The False Negative Rate is an error metric for binary features, such as the voiced/unvoiced decision. Computed as $\text{FNR} = \frac{\text{FN}}{\text{FN}+\text{TP}}$, where FN and TP are false negatives and true positives respectively. ↑ 73, 97, 98, 108

FPR The False Positive Rate is an error metric for binary features, such as the voiced/unvoiced decision. Computed as $\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}}$, where FP and TN are false positives and true negatives respectively. ↑ 73, 97, 98, 108

GAN A Generative Adversarial Network (Goodfellow et al., 2014) is a kind of model that consists of a generator and a discriminator. The generator is trained to produce realistic samples of the data distribution. The discriminator is trained to distinguish between real samples taken from the dataset distribution and fake samples produced by the generator. Thus, the generator can be improved using the error signal from the discriminator, providing a kind of implicitly learned loss function. And, at the same time, the discriminator itself is updated as well. ↑ 26–28, 75, 125, 136, 138, 139, 144, 145, 148, 150, 153

GLA The Griffin-Lim Algorithm (Griffin and Lim, 1984) is a method of reconstructing a corresponding phase spectrogram given a magnitude spectrogram. This algorithm is iterative and the resulting waveform often has notable artifacts. ↑ 24, 42

GLU An activation function that combines a linear activation with a gate. Computed as $\text{GLU}(a, b) = a \odot \text{sigm}(b)$, where $\odot$ denotes element-wise multiplication. This activation function is popular in NLP applications, possibly due to the reduced chance of vanishing gradients (compared to e.g., gated tanh units allowing deeper networks. *See also:* GTU. ↑ 121–123, 127, 173, 176

GMM Gaussian Mixture Model. *See:* MoG. ↑ 16, 19, 20

GPU Specialized hardware especially suited for parallelized computations such as training a deep learning model. ↑ 19, 25, 38, 90, 106, 117, 125, 133, 135, 155

GRU A Gated Recurrent Unit network is a kind of RNN composed of memory cells with gating operations. It was designed to be a reduced computational complexity alternative to the LSTM recurrent unit. *See also:* LSTM. ↑ 22, 84

GTU An activation function that combines a tanh activation with a gate. Computed as $\text{GTU}(a, b) = \tanh(a) \odot \text{sigm}(b)$, where $\odot$ denotes element-wise multiplication. This activation function is commonly used to give CNN similar performance to gated recurrent networks such as LSTMs. *See also:* GLU. ↑ 121

GV Global Variance is a statistic used to describe speech parameters, computed as the per-utterance variance of the given parameter. As a statistic, it is used in the evaluation of speech and singing synthesis systems. Global Variance may also refer to a parameter generation technique that attempts to preserve the global variance of natural speech during synthesis.

In particular, this is a way to combat the oversmoothing problem some generative models tend to have. ↑ 16, 70

HMM A Hidden Markov Model is a probabilistic model in which the system being modeled is assumed to be a Markov process with unobserved (i.e., hidden) states. ↑ 4, 15–20, 22, 55, 62, 63, 81, 86, 90, 97, 105, 115, 159, 164, 171

HSMM A Hidden Semi-Markov Model has the same structure as a hidden Markov model, but the probability of a state change depends on the time elapsed in the state, rather than being constant. Typically, state durations may be modeled using a Gaussian distribution. *See also:* HMM. ↑ 16, 17, 63, 126

HTS The HMM Speech Synthesis System (H Triple S) is a collection of tools and scripts for building HMM (HSMM) speech and singing synthesizers. It is developed by Nagoya Institute of Technology (Nitech, Japan) and based on HTK (Hidden Markov Model Toolkit). Later versions of the toolkit also support DNN and DNN/HMM hybrid models. ↑ 56, 63

hyperparameter A hyperparameter is a numerical parameter of a machine learning model that is not learned from data, but set by hand. Typical examples include learning rate, number of hidden units, regularization weights, etc. ↑ 89, 94

IS A classifier-based data-driven evaluation metric, commonly used to evaluate generative models of images. *See also:* FID. ↑ 75, 76

loudness Loudness is a feature of a speech and music signals, related to the subjective perception of sound pressure. In particular in broadcasting some standardized definitions of loudness have been proposed, such as ITU-R BS.1770 (e.g., implemented in Steinmetz and Reiss, 2021). This algorithm includes filtering, overlapping windows (or "blocks"), energy calculation, converting to dB, and gating low energy blocks. Within this work, we use loudness and dynamics mostly interchangeably. In singing voice, the slope of the spectrum is also an important feature related to dynamics, rather than just the fullband energy. *See also:* . ↑ 40, 93, 187

LSD Log-Spectral Distortion, also known as Log-Spectral Distance, is a distance in dB between two spectra or spectrograms. In the case of two $T \times K$-dimensional spectrograms X and $\hat{X}$, the (mean) log-spectral distortion is defined as $D_{\text{LSD}} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\left[20\log_{10}\frac{X(t,k)}{\hat{X}(t,k)}\right]^2}$. ↑ 70–72

LSTM A Long Short-Term Memory network is a kind of RNN composed of memory cells with gating operations. The gates help avoid the vanishing or exploding gradient problem of the traditional RNN, where gradients become every smaller or larger with each recurrent timestep. Additionally, this type of RNN allows modeling longer time dependencies compared to the classical RNN. ↑ 20, 22, 28, 84

MAE The Mean Absolute Error is a distance metric. Computed as $D_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i|$. ↑ 74, 111

MCD Mel-Cepstral Distortion, also known as Mel-spectral Distance, is a distance in dB between two mel scale cepstra or mel scale cepstrograms. In the case of two $T \times K$-dimensional cepstrograms C and $\hat{C}$, the (mean) mel-cepstral distortion in dB is defined as $D_{\text{MCD}} = \frac{10}{\log 10}\sqrt{2}\frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\left[C(t,k)-\hat{C}(t,k)\right]^2}$. ↑ 69, 71, 72, 97, 98, 108, 110, 114

MDN A Mixture Density Network is a neural network whose output is interpreted as a mixture density, often an MoG. ↑ 20, 21

MFCC Mel-Frequency Cepstral Coefficients are a speech feature that consists of a vector of values that represent the transform (e.g., DCT) of the log magnitude spectrum at frequencies on a mel scale. It is typically used as a perceptually motivated low-dimensional representation of timbre. ↑ 46, 63

MFSC Mel-Frequency Spectral Coefficients are a speech feature that consists of a vector of values that represent the log magnitude spectrum at frequencies on a mel scale. It is typically used as a perceptually motivated low-dimensional representation of timbre. ↑ 46, 47, 91

MGC Mel-Generalized Coefficients are a generalized cepstral representation of timbre. It has two hyperparameters; a frequency warping factor, typically to something close to a mel scale, and a pole/zero weighting factor, which controls the emphasis on peaks or valleys in the spectral envelope. ↑ 16, 46, 112, 113

MoG A Mixture of Gaussians is a multi-modal distribution consisting of the sum of multiple scaled Gaussian distributions. ↑ 15, 16, 87, 199

MoL A Mixture of Logistics is a multi-modal distribution consisting of the sum of multiple scaled logistic distributions. In the context of autoregressive models and speech synthesis, MoL often refers to a mixture of discretized logistic distributions (e.g., Salimans et al., 2017; Shen et al., 2018). ↑ 24, 87, 94

MOPPG Maximum Output Probability Parameter Generation is a technique where a model predicts static, delta and delta-delta distributions, and using these predictions over the whole output sequence, the most probable output trajectory is computed. In particular, this method is used to void discontinuities in the predicted output. Also known as: Maximum Likelihood Parameter Generation (MLPG). ↑ 16, 20

MOS A Mean Opinion Score is the result of a kind of qualitative evaluation where participants are typically asked to rate stimuli on a 1–5 scale. *See also:* ACR. ↑ 21, 24, 25, 28, 66–68, 76, 110, 114, 128, 130, 131, 151, 153, 178, 189, 190

MS The Modulation Spectrum is the magnitude spectrum of a speech parameter time series (e.g., F0 or mel-cepstral coefficients). It is used in the evaluation of speech and singing synthesis systems. It is also used in parameter generation algorithms, where the goal is to preserve the modulation spectrum of natural speech at synthesis. ↑ 70, 72, 73, 112, 113

MS-LSD The Modulation Spectrum Log-Spectral Distortion is the log-spectral distortion between two modulation spectra. *See also:* MS, LSD. ↑ 72, 97, 98, 108, 110, 111

MUSHRA A MUltiple Stimuli with Hidden Reference and Anchor test is a type of listening test where listeners compare multiple stimuli generated from a single reference. A hidden copy of the reference is included to get an upper bound score. Similarly, one or more anchors, distorted versions of the reference, are included in order to obtain a lower bound score. Typically, a fine-grained 0–100 rating is used. *See also:* MOS. ↑ 67, 108

NLP Natural Language Processing is an area of research concerned with how to program computers to process and analyze large amounts of natural language data. ↑ 119, 121, 123

NMT Neural Machine Translation is a topic of research that deals with translating text data using neural networks. ↑ 117, 118

NPSS Neural Parametric Singing Synthesizer is the singing synthesis system proposed in this work. The name is derived from the fact that it is based on a neural network and predicts features of a parametric vocoder. ↑ 97, 105, 127

one-hot A one-hot representation or encoding is a way to express a categorical variable with N possible values as a binary vector of length N with one position set to one and the rest to zero. ↑ 92, 101, 185, 189

QRNN Quasi-Recurrent Neural Networks are an approach to neural sequence modeling that alternates convolutional layers, which apply in parallel across timesteps, and a minimalist recurrent pooling function that applies in parallel across channels. Networks with this type of recurrent unit reported has similar performance to LSTM RNN units, while potentially an order of magnitude faster. ↑ 23

RBM A Restricted Boltzmann Machine is an undirected probabilistic model with binary (Bernoulli) latent variables. This was a popular way to do unsupervised feature learning in early deep learning approaches. *See also:* DBN. ↑ 19, 20

ReLU A Rectified Linear Unit is common (de facto standard) activation function that is linear for inputs greater than zero and zero for inputs smaller than zero. In particular, it avoids the vanishing gradients that other activation functions can exhibit, thus allowing deeper networks and faster training (e.g., Goodfellow et al., 2016, Chapter 6.3.1). ↑ 84, 88, 123, 166, 189

RMSE The Root Mean Squared Error is a distance metric. Computed as $D_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2}$. ↑ 72, 74, 111, 145

RNN A Recurrent Neural Network is a kind of neural network typically used for working with time series. As there are connections from the previous timestep's hidden state to the current timestep's hidden state, the network's output can depend on past information. ↑ 20–22, 24, 27, 28, 83, 118–120

Seq2Seq A sequence-to-sequence model, is a model that can learn a mapping between two unaligned sequences. An example may be an input phonetic sequence to output acoustic frames. ↑ 6, 23, 24, 26–28, 37, 39, 84, 117, 118, 121, 125, 163, 164, 171, 172, 176, 179, 181, 188

softmax The softmax function takes an input vector of K real numbers, called logits, and normalizes them into a categorical probability distribution, $\text{softmax}(x) = \frac{e^x}{\sum_{j=1}^{K} e^{x_j}}$. It is often used in neural networks as an activation function, and can be seen as a smooth (differentiable) approximation of the arg max function. A temperature softmax scales the logits by some factor to make the distribution more or less "peaky", which is sometimes used, in particular when sampling from this distribution. ↑ 86, 88, 103, 119, 123

STFT The Short-Time Fourier Transform is a variant of the DFT, used to analyze long, non-stationary time-domain signals. The signal is first windowed into shorter, overlapping segments that are considered to be quasi-stationary. These windowed signals can then be analyzed using the DFT to obtain meaningful time series of magnitude and phase spectra, called spectrograms. Unlike the DFT, the inverse of this transform generally incurs some loss, especially if the magnitude and phase information is modified somehow. ↑ 41, 71, 72, 145, 150, 153, 156

TTS Text-to-speech is another term for speech synthesis, i.e., a system that converts written text to a speech waveform. ↑ 4, 5, 7, 11, 15, 16, 19, 21–24, 26–28, 41, 55, 59, 63–68, 72, 76, 84, 85, 96, 102, 105, 118, 121, 128, 131, 136–139, 159, 160, 162, 169, 171, 172, 174–176, 182, 183

V/UV A voiced/unvoiced decision is typically part of F0 estimation algorithms. It is a binary feature that indicates whether a frame has pitch (is voiced) or not (is unvoiced). *See also:* aperiodicity. ↑ 72, 73, 97, 98, 108, 110

vocoder A vocoder (portmanteau of the words *voice* and *coder*) consists of an analysis (or encoder) and synthesis (or decoder) step. The analysis step takes the input speech signal and converts it into a typically low-dimensional representation, e.g., F0 and spectral envelope bands. The synthesis step attempts to reconstruct the original signal from this low-dimensional representation, often with some degree of loss. ↑ 7, 10, 71, 81, 87, 88, 91, 93, 100, 123, 124, 126, 131, 133, 162–164, 173, 190

Bibliography

Aitken, A. P., Ledig, C., Theis, L., Caballero, J., Wang, Z., & Shi, W. (2017). Checkerboard Artifact Free Sub-Pixel Convolution: A Note on Sub-Pixel Convolution, Resize Convolution and Convolution Resize. arXiv: 1707.02937 [cs.CV]. ↑145

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., … Zhu, Z. (2016). Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, June 19–24, 2016 (pp. 173–182). New York City, NY, USA. ↑76

Angelini, O., Moinet, A., Yanagisawa, K., & Drugman, T. (2020). Singing Synthesis: With a Little Help from my Attention. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 1221–1225). Shanghai, China. ↑24, 176, 188

Ardaillon, L. (2017). *Synthesis and Expressive Transformation of Singing Voice* (Doctoral dissertation, Université Pierre et Marie Curie – Paris VI). ↑14

Ardaillon, L., Chabot-Canet, C., & Roebel, A. (2016). Expressive Control of Singing Voice Synthesis Using Musical Contexts and a Parametric F0 Model. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, September 8–12, 2016 (pp. 1250–1254). San Francisco, CA, USA. ↑14

Arik, S. Ö., Chen, J., Peng, K., Ping, W., & Zhou, Y. (2018). Neural Voice Cloning with a Few Samples. In *Advances in Neural Information Processing Systems 31 (NIPS)*, December 3–8, 2018 (pp. 10040–10050). Montreal, QC, Canada. ↑76, 161, 163

Arik, S. Ö., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., … Shoeybi, M. (2017a). Deep Voice: Real-Time Neural Text-to-Speech. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, August 6–11, 2017 (pp. 195–204). Sydney, Australia. ↑22, 90

Arik, S. Ö., Diamos, G., Gibiansky, A., Miller, J., Peng, K., Ping, W., … Zhou, Y. (2017b). Deep Voice 2: Multi-Speaker Neural Text-to-Speech. In *Advances in Neural Information Processing Systems 30 (NIPS)*, December 4–9, 2017 (pp. 2962–2970). Long Beach, CA, USA. ↑23, 24, 103, 127, 161

Arik, S. Ö., Jun, H., & Diamos, G. F. (2019). Fast Spectrogram Inversion Using Multi-Head Convolutional Neural Networks. *IEEE Signal Processing Letters*, *26*(1), 94–98. ↑145

Ba, L. J., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. arXiv: 1607.06450 [stat.ML]. ↑123

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv: 1409.0473 [cs.CL]. ↑119

Bengio, Y., Delalleau, O., & Simard, C. (2010). Decision Trees do not Generalize to New Variations. *Computational Intelligence*, *26*(4), 449–267. doi:10.1111/j.1467-8640.2010.00366.x. ↑19

Bińkowski, M., Donahue, J., Dieleman, S., Clark, A., Elsen, E., Casagrande, N., … Simonyan, K. (2020). High Fidelity Speech Synthesis with Adversarial Networks. In *8th International Conference on Learning Representations (ICLR)*, April 26–30, 2020, Addis Ababa, Ethiopia. ↑26, 76, 136, 148, 150

Birkholz, P. (2007). Articulatory Synthesis of Singing. In *Proceedings of the 8th Annual Conference of the International Speech Communication Association (Interspeech)*, August 27–31, 2007 (pp. 4001–4004). Antwerp, Belgium. ↑10

Blaauw, M., & Bonada, J. (2017a). A Neural Parametric Singing Synthesizer. In *Proceedings of the 18th Annual Conference of the International Speech Communication Association (Interspeech)*, August 20–24, 2017 (pp. 1230–1234). Stockholm, Sweden. ↑81, 82

Blaauw, M., & Bonada, J. (2017b). A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs. *Applied Sciences*, *7*(12) Special Issue on Sound and Music Computing. doi:10.3390/app7121313. ↑23, 82, 127

Blaauw, M., & Bonada, J. (2016). A Singing Synthesizer Based on PixelCNN. Poster presented at the María de Maeztu Seminar on Music Knowledge Extraction Using Machine Learning (collocated with NIPS). Barcelona, Spain. ↑88

Blaauw, M., & Bonada, J. (2020). Sequence-to-Sequence Singing Synthesis Using the Feed-Forward Transformer. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7229–7233). Barcelona, Spain. ↑117, 172

Blaauw, M., Bonada, J., & Daido, R. (2019). Data Efficient Voice Cloning for Neural Singing Synthesis. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 6840–6844). Brighton, United Kingdom. ↑160

Black, A. W., & Taylor, P. (1997). Automatically Clustering Similar Units for Unit Selection in Speech Synthesis. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech)*, September 22–25, 1997 (pp. 601–604). Rhodes, Greece. ↑12

Bonada, J., & Blaauw, M. (2013). Generation of Growl-Type Voice Qualities by Spectral Morphing. In *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 26–31, 2013 (pp. 213–216). Vancouver, BC, Canada. ↑14, 32

Bonada, J., & Blaauw, M. (2020). Hybrid Neural-Parametric F0 Model for Singing Synthesis. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7244–7248). Barcelona, Spain. ↑101

Bonada, J., & Blaauw, M. (2021). Semi-Supervised Learning for Singing Synthesis Timbre. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 7083–7087). Toronto, ON, Canada. ↑182

Bonada, J., Blaauw, M., & Loscos, A. (2006). Improvements to a Sample-Concatenation Based Singing Voice Synthesizer. In *Proceedings of the 121st AES Convention*, October 5–8, 2006, San Francisco, CA, USA. ↑14

Bonada, J., Celma, Ò., Loscos, A., Ortolà, J., Serra, X., Yoshioka, Y., … Kenmochi, H. (2001). Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models. In *Proceedings of the 2001 International Computer Music Conference (ICMC)*, September 17–22, 2001, Havana, Cuba. ↑14

Bonada, J., & Serra, X. (2007). Synthesis of the Singing Voice by Performance Sampling and Spectral Models. *IEEE Signal Processing Magazine*, *24*(2), 67–79. doi:10.1109/MSP.2007.323266. ↑14

Bonada, J., Umbert, M., & Blaauw, M. (2016). Expressive Singing Synthesis Based on Unit Selection for the Singing Synthesis Challenge 2016. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, September 8–12, 2016 (pp. 1230–1234). San Francisco, CA, USA. ↑14, 81, 97, 101, 105

Bous, F., & Roebel, A. (2019). Analysing Deep Learning-Spectral Envelope Prediction Methods for Singing Synthesis. In *Proceedings of the 27th European Signal Processing Conference (EUSIPCO)*, September 2–6, 2019, A Coruña, Spain. ↑23, 40, 93

Bozkurt, B., Öztürk, Ö., & Dutoit, T. (2003). Text Design for TTS Speech Corpus Building Using a Modified Greedy Selection. In *Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH)*, September 1–4, 2003, Geneva, Switzerland. ↑59

Bradbury, J., Merity, S., Xiong, C., & Socher, R. (2017). Quasi-Recurrent Neural Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, April 24–26, 2017, Toulon, France. ↑23

Carlsson-Berndtsson, G., & Sundberg, J. (1993). The MUSSE DIG Singing Synthesis. In *Proceedings of the 2nd Stockholm Music Acoustics Conference (SMAC)*, July 28–August 1, 1993 (pp. 279–281). Stockholm, Sweden. ↑8

Chandna, P., Blaauw, M., Bonada, J., & Gómez, E. (2019). WGANSing: A Multi-Voice Singing Voice Synthesizer Based on the Wasserstein-GAN. In *Proceedings of the 27th European Signal Processing Conference (EUSIPCO)*, September 2–6, 2019, A Coruña, Spain. ↑26, 125

Chen, F., Huang, R., Cui, C., Ren, Y., Liu, J., Zhao, Z., … Huai, B. (2021a). SingGAN: Generative Adversarial Network for High-Fidelity Singing Voice Generation. arXiv: 2110.07468 [eess.AS]. ↑26, 139, 148

Chen, J., Tan, X., Luan, J., Qin, T., & Liu, T.-Y. (2020). HiFiSinger: Towards High-Fidelity Neural Singing Voice Synthesis. arXiv: 2009.01776 [eess.AS]. ↑26, 28, 121, 176, 188

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., & Chan, W. (2021b). WaveGrad: Estimating Gradients for Waveform Generation. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, May 3–7, 2021, Virtual, Austria. ↑27, 136

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., Dehak, N., & Chan, W. (2021c). WaveGrad 2: Iterative Refinement for Text-to-Speech Synthesis. arXiv: 2106.09660 [eess.AS]. ↑27

Chen, Y., Assael, Y., Shillingford, B., Budden, D., Reed, S., Zen, H., ... de Freitas, N. (2019). Sample Efficient Adaptive Text-to-Speech. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, May 6–9, 2019, New Orleans, LA, USA. ↑161, 163, 164

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, October 25–29, 2014 (pp. 1724–1734). Doha, Qatar. ↑118

Choi, S., Kim, W., Park, S., Yong, S., & Nam, J. (2020). Korean Singing Voice Synthesis Based on Auto-Regressive Boundary Equilibrium GAN. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7234–7238). Barcelona, Spain. ↑26

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-Based Models for Speech Recognition. In *Advances in Neural Information Processing Systems 28 (NIPS)*, December 7–12, 2015 (pp. 577–585). Montreal, QC, Canada. ↑119

Chowning, J. (1989). Frequency Modulation Synthesis of the Singing Voice. In M. V. Mathews & J. R. Pierce (Eds.), *Current Directions in Computer Music Research* (pp. 57–64). Cambridge, MA, USA: The MIT Press. ↑10

Chung, Y.-A., Wang, Y., Hsu, W.-N., Zhang, Y., & Skerry-Ryan, R. (2018). Semi-Supervised Training for Improving Data Efficiency in End-to-End Speech Synthesis. arXiv: 1808.10128 [cs.CL]. ↑164

Cook, P. R. (1996). Singing Voice Synthesis: History, Current Work, and Future Directions. *Computer Music Journal*, *20*(3), 38–46. doi:10.2307/3680822. ↑10

Cui, Y., Wang, X., He, L., & Soong, F. K. (2020). An Efficient Subband Linear Prediction for LPCNet-Based Neural Synthesis. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 3555–3559). Shanghai, China. ↑148

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). Language Modeling with Gated Convolutional Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, August 6–11, 2017 (pp. 933–941). Sydney, Australia. ↑121

Depalle, P., García, G., & Rodet, X. (1994). A Virtual Castrato (!?) In *Proceedings of the International Computer Music Conference (ICMC)*, September 12–17, 1994 (pp. 357–360). Aarhus, Denmark. ↑9

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2017). Density Estimation Using Real NVP. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, April 24–26, 2017, Toulon, France. ↑25

Dodge, C. (1989). On Speech Sounds. In M. V. Mathews & J. R. Pierce (Eds.), *Current Directions in Computer Music Research* (pp. 9–18). Cambridge, MA, USA: The MIT Press. ↑9

Donahue, J., Dieleman, S., Binkowski, M., Elsen, E., & Simonyan, K. (2021). End-to-End Adversarial Text-to-Speech. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, May 3–7, 2021, Virtual, Austria. ↑28, 38, 39, 76

Doval, B., & d'Alessandro, C. (1997). Spectral Correlates of Glottal Waveform Models: An Analytic Study. In *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 21–24, 1997 (pp. 1295–1298). Munich, Germany. ↑30

Duan, Z., Fang, H., Li, B., Sim, K. C., & Wang, Y. (2013). The NUS Sung and Spoken Lyrics Corpus: A Quantitative Comparison of Singing and Speech. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, October 29–November 1, 2013 (pp. 1–9). Kaohsiung, Taiwan. ↑56, 164, 165

Dudley, H. (1940). The Carrier Nature of Speech. *The Bell System Technical Journal*, *19*(4). doi:10.1002/j.1538-7305.1940.tb00843.x. ↑10

Fan, Y., Qian, Y., Xie, F.-L., & Soong, F. K. (2014). TTS Synthesis with Bidirectional LSTM Based Recurrent Neural Networks. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association (Interspeech)*, September 14–18, 2014 (pp. 1964–1968). Singapore. ↑20, 21

Fant, G. (1960). *Acoustic Theory of Speech Production*. The Hague: Mouton. ↑29

Feugère, L., d'Alessandro, C., Doval, B., & Perrotin, O. (2017). Cantor Ditigalis: Chironomic Parametric Synthesis of Singing. *EURASIP Journal on Audio, Speech, and Music Processing*, *2017*(2). doi:10.1186/s13636-016-0098-5. ↑10

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, August 6–11, 2017 (pp. 1243–1252). Sydney, Australia. ↑24, 128, 171

Gfeller, B., Frank, C. H., Roblek, D., Sharifi, M., Tagliasacchi, M., & Velimirovic, M. (2020). SPICE: Self-Supervised Pitch Estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 1118–1128. doi:10.1109/TASLP.2020.2982285. ↑45

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. ↑36, 214

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Networks. arXiv: 1406.2661 [stat.ML]. ↑75, 211

Goto, M., Nakano, T., Kajita, S., Matsusaka, Y., Nakaoka, S., & Yokoi, K. (2012). VocaListener and VocaWatcher: Imitating a Human Singer by Using Signal Processing. In *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 25–30, 2012 (pp. 5393–5396). Kyoto, Kyoto Prefecture, Japan. ↑54

Graves, A. (2013). Generating Sequences with Recurrent Neural Networks. arXiv: 1308.0850 [cs.NE]. ↑171

Graves, A., Fernández, S., Gomez, F. J., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Machine Learning, Proceedings of the 23rd International Conference on Machine Learning (ICML)*, June 25–29, 2006 (pp. 369–376). Pittsburgh, PA, USA. ↑23

Griffin, D. W., & Lim, J. S. (1984). Signal Estimation from Modified Short-Time Fourier Transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *32*(2), 236–243. doi:10.1109/TASSP.1984.1164317. ↑24, 42, 163, 211

Gritsenko, A. A., Salimans, T., van den Berg, R., Snoek, J., & Kalchbrenner, N. (2020). A Spectral Energy Distance for Parallel Speech Synthesis. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, December 6–12, 2020, Virtual event. ↑26, 76

Gu, Y., Yin, X., Rao, Y., Wan, Y., Tang, B., Zhang, Y., … Ma, Z. (2020). ByteSing: A Chinese Singing Voice Synthesis System Using Duration Allocated Encoder-Decoder Acoustic Models and WaveRNN Vocoders. arXiv: 2004.11012 [eess.AS]. ↑28, 188

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 27–30, 2016 (pp. 770–778). Las Vegas, NV, USA. ↑22, 84, 123

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems 30 (NIPS)*, December 4–9, 2017 (pp. 6629–6640). Long Beach, CA, USA. ↑76

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, December 6–12, 2020, Virtual event. ↑27

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies. In J. F. Kolen & S. C. Kremer (Eds.), *A Field Guide to Dynamical Recurrent Neural Networks* (pp. 237–243). doi:10.1109/9780470544037.ch14. ↑120

Hono, Y., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2019). Singing Voice Synthesis Based on Generative Adversarial Networks. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 6955–6959). Brighton, United Kingdom. ↑26, 125

Hono, Y., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2021a). Sinsy: A Deep Neural Network-Based Singing Voice Synthesis System. arXiv: 2108.02776 [eess.AS]. ↑28

Hono, Y., Takaki, S., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2021b). Periodnet: A Non-Autoregressive Waveform Generation Model with a Structure Separating Periodic and Aperiodic Components. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 6049–6053). Toronto, ON, Canada. ↑26, 139, 148

Hua, K. (2018). Modeling Singing F0 with Neural Network Driven Transition-Sustain Models. arXiv: 1803.04030 [eess.AS]. ↑160

Huang, R., Chen, F., Ren, Y., Liu, J., Cui, C., & Zhao, Z. (2021). Multi-Singer: Fast Multi-Singer Singing Voice Vocoder with a Large-Scale Corpus. arXiv: 2112.10358 [eess.AS]. ↑26, 57, 149

Hunt, A. J., & Black, A. W. (1996). Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 7–10, 1996 (pp. 373–376). Atlanta, GA, USA. ↑12

Hwang, M.-J., Song, E., Yamamoto, R., Soong, F. K., & Kang, H.-G. (2020a). Improving LPCNET-Based Text-to-Speech with Linear Prediction-Structured Mixture Density Network. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7219–7223). Barcelona, Spain. ↑148

Hwang, M.-J., Soong, F. K., Song, E., Wang, X., Kang, H., & Kang, H.-G. (2020b). LP-WaveNet: Linear Prediction-Based WaveNet Speech Synthesis. In *Proceedings of the 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, December 7–10, 2020 (pp. 810–814). Auckland, New Zealand. ↑148

Hwang, M.-J., Yamamoto, R., Song, E., & Kim, J.-M. (2021). High-Fidelity Parallel WaveGAN with Multi-Band Harmonic-plus-Noise Model. In *Proceedings of the 22nd Annual Conference of the International Speech Communication Association (Interspeech)*, August 30–September 3, 2021, Brno, Czechia. ↑148

Iglewicz, B., & Hoaglin, D. C. (1993). *How to Detect and Handle Outliers*. Milwaukee, WI, USA: ASQC Quality Press. ↑69, 71

Ihm, H. R., Lee, J. Y., Choi, B. J., Cheon, S. J., & Kim, N. S. (2020). Reformer-TTS: Neural Speech Synthesis with Reformer Network. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 2012–2016). Shanghai, China. ↑28

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 21–26, 2017 (pp. 5967–5976). Honolulu, HI, USA. ↑146

ITU-R Recommendation BS.1534-3. (2015). *Method for the Subjective Assessment of Intermediate Quality Levels of Coding Systems*. International Telecommunication Union. Geneva, Switzerland. ↑67, 108

ITU-R Recommendation BS.1770-4. (2015). *Algorithms to Measure Audio Programme Loudness and True-Peak Audio Level*. International Telecommunication Union. Geneva, Switzerland. ↑149

ITU-R Recommendation P.800. (1996). *Methods for Subjective Determination of Transmission Quality*. International Telecommunication Union. Geneva, Switzerland. ↑66

ITU-T recommendation P.862. (2001). *Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*. International Telecommunication Union. Geneva, Switzerland. ↑77

ITU-T recommendation P.863. (2011). *Perceptual Objective Listening Quality Prediction*. International Telecommunication Union. Geneva, Switzerland. ↑77

Janer, J., Bonada, J., & Blaauw, M. (2006). Performance-Driven Control for Sample-Based Singing Voice Synthesis. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx)*, September 18–20, 2006, Montreal, QC, Canada. ↑54, 160

Jang, W., Lim, D., & Yoon, J. (2020). Universal MelGAN: A Robust Neural Vocoder for High-Fidelity Waveform Generation in Multiple Domains. arXiv: 2011.09631 [eess.AS]. ↑149

Jeong, M., Kim, H., Cheon, S. J., Choi, B. J., & Kim, N. S. (2021). Diff-TTS: A Denoising Diffusion Model for Text-to-Speech. arXiv: 2104.01409 [eess.AS]. ↑27

Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., … Wu, Y. (2018). Transfer Learning from Speaker Verification to Multispeaker Text-to-Speech Synthesis. In *Advances in Neural Information Processing Systems 31 (NIPS)*, December 3–8, 2018 (pp. 4485–4495). Montreal, QC, Canada. ↑163, 164

Jiao, Y., Gabrys, A., Tinchev, G., Putrycz, B., Korzekwa, D., & Klimkov, V. (2021). Universal Neural Vocoding with Parallel WaveNet. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 6044–6048). Toronto, ON, Canada. ↑149

Juvela, L., Bollepalli, B., Yamagishi, J., & Alku, P. (2019). GELP: GAN-Excited Linear Prediction for Speech Synthesis from Mel-Spectrogram. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech)*, September 15–19, 2019 (pp. 694–698). Graz, Austria. ↑148

Kaegi, W., & Tempelaars, S. (1978). VOSIM – A New Sound Synthesis System. *Journal of the Audio Engineering Society*, *26*(6), 418–425. ↑10

Kaiser, L., & Bengio, S. (2018). Discrete Autoencoders for Sequence Models. arXiv: 1801.09797 [cs.LG]. ↑185

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., … Kavukcuoglu, K. (2018). Efficient Neural Audio Synthesis. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, July 10–15, 2018 (pp. 2415–2424). Stockholm, Sweden. ↑27

Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., & Kavukcuoglu, K. (2016a). Neural Machine Translation in Linear Time. arXiv: 1610.10099 [cs.LG]. ↑22

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., & Kavukcuoglu, K. (2016b). Video Pixel Networks. arXiv: 1610.00527 [cs.CV]. ↑22

Kameoka, H., Kaneko, T., Tanaka, K., & Hojo, N. (2018). StarGAN-VC: Non-Parallel Many-to-Many Voice Conversion Using Star Generative Adversarial Networks. In *Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT)*, December 18–21, 2018 (pp. 266–273). Athens, Greece. ↑187

Kaneko, T., & Kameoka, H. (2018). CycleGAN-VC: Non-Parallel Voice Conversion Using Cycle-Consistent Adversarial Networks. In *Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, September 3–7, 2018 (pp. 2100–2104). Roma, Italy. ↑187

Kaneko, T., Kameoka, H., Tanaka, K., & Hojo, N. (2019a). CycleGAN-VC2: Improved CycleGAN-Based Non-Parallel Voice Conversion. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 6820–6824). Brighton, United Kingdom. ↑187

Kaneko, T., Kameoka, H., Tanaka, K., & Hojo, N. (2019b). StarGAN-VC2: Rethinking Conditional Methods for StarGAN-Based Voice Conversion. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech)*, September 15–19, 2019 (pp. 679–683). Graz, Austria. ↑187

Kang, M., Lee, J., Kim, S., & Kim, I. (2021). Fast DCTTS: Efficient Deep Convolutional Text-to-Speech. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 7043–7047). Toronto, ON, Canada. ↑24, 69

Kang, S., Qian, X., & Meng, H. (2013). Multi-Distribution Deep Belief Network for Speech Synthesis. In *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 26–31, 2013 (pp. 8012–8016). Vancouver, BC, Canada. ↑20

Karaali, O., Corrigan, G., & Gerson, I. (1996). Speech Synthesis with Neural Networks. In *Proceedings of the World Congress on Neural Networks (WCNN)*, September 15–18, 1996 (pp. 45–50). San Diego, CA, USA. ↑19

Kawahara, H. (2006). STRAIGHT, Exploitation of the Other Aspect of VOCODER: Perceptually Isomorphic Decomposition of Speech Sounds. *Acoustical Science and Technology*, *27*(6), 349–353. doi:10.1250/ast.27.349. ↑44, 137

Kawahara, H., Masuda-Katsuse, I., & de Cheveigné, A. (1999). Restructuring Speech Representations Using a Pitch-Adaptive Time-Frequency Smoothing and an Instantaneous-Frequency-Based F0 Extraction: Possible Role of a Repetitive Structure in Sounds. *Speech Communication*, *27*(3-4), 187–207. doi:10.1016/S0167-6393(98)00085-5. ↑44, 137

Kelly, J. L., & Lochbaum, C. C. (1962), In *Proceedings of the 4th International Congress on Acoustics (ICA)*, August 21–28, 1962 (pp. 1–4). Paper G42. Copenhagen, Denmark. ↑10

Kenmochi, H., & Ohshita, H. (2007). VOCALOID - Commercial Singing Synthesizer Based on Sample Concatenation. In *Proceedings of the 8th Annual Conference of the International Speech Communication Association (Interspeech)*, August 27–31, 2007 (pp. 4009–4010). Antwerp, Belgium. ↑14, 81

Kim, J., Kim, S., Kong, J., & Yoon, S. (2020). Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, December 6–12, 2020, Virtual event. ↑26, 28

Kim, J., Choi, H., Park, J., Hahn, M., Kim, S., & Kim, J.-J. (2018). Korean Singing Voice Synthesis System Based on an LSTM Recurrent Neural Network. In *Proceedings of the 19th Annual Conference of the International Speech Communication Association (Interspeech)*, September 2–6, 2018 (pp. 1551–1555). Hyderabad, India. ↑28

Kim, S., Lee, S.-g., Song, J., Kim, J., & Yoon, S. (2019). FloWaveNet: A Generative Flow for Raw Audio. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, June 9–15, 2019 (pp. 3370–3378). Long Beach, CA, USA. ↑25, 136

Kingma, D. P., & Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, May 7–9, 2015, San Diego, CA, USA. ↑106

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems 31 (NIPS)*, December 3–8, 2018 (pp. 10236–10245). Montreal, QC, Canada. ↑25

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems 29 (NIPS)*, December 5–10, 2016 (pp. 4743–4751). Barcelona, Spain. ↑25

Kinoshita, Y., & Kiya, H. (2020). Fixed Smooth Convolutional Layer for Avoiding Checkerboard Artifacts in CNNs. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 3712–3716). Barcelona, Spain. ↑145

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The Efficient Transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, April 26–30, 2020, Addis Ababa, Ethiopia. ↑124

Kob, M. (2002). *Physical Modeling of the Singing Voice* (Master's thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen). ↑10

Koguchi, J., Takamichi, S., & Morise, M. (2020). PJS: Phoneme-Balanced Japanese Singing-Voice Corpus. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, December 7–10, 2020 (pp. 487–491). Auckland, New Zealand. ↑58

Kong, J., Kim, J., & Bae, J. (2020). HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, December 6–12, 2020, Virtual event. ↑26, 146, 148, 155

Kong, Z., Ping, W., Huang, J., Zhao, K., & Catanzaro, B. (2021). DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, May 3–7, 2021, Virtual, Austria. ↑27, 136

Kröger, B. J., & Birkholz, P. (2009). Articulatory Synthesis of Speech and Singing: State of the Art and Suggestions for Future Research. In A. Esposito, A. Hussain, M. Marinaro, & R. Martone (Eds.), *Multimodal Signals: Cognitive and Algorithmic Issues* (pp. 306–319). doi:10.1007/978-3-642-00525-1_31. ↑10

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., … Courville, A. C. (2019). MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, December 8–14, 2019 (pp. 14881–14892). Vancouver, BC, Canada. ↑26, 136, 146, 148, 150

Laroche, J., & Dolson, M. (1997). Phase-Vocoder: About This Phasiness Business. In *Proceedings of 1997 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, October 19–22, 1997, New Paltz, NY, USA. ↑136

Larsson, B. (1977). Music and Singing Synthesis Equipment (MUSSE). *Speech Transmission Laboratory Quarterly Progress and Status Report (STL-QPSR)*, *18*(1), 38–40. ↑8

Lee, G.-H., Kim, T.-W., Bae, H., Lee, M.-J., Kim, Y.-I., & Cho, H.-Y. (2021). N-Singer: A Non-Autoregressive Korean Singing Voice Synthesis System for Pronunciation Enhancement. arXiv: 2106.15205 [eess.AS]. ↑26

Lee, J., Choi, H.-S., Jeon, C.-B., Koo, J., & Lee, K. (2019). Adversarially Trained End-to-End Korean Singing Voice Synthesis System. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech)*, September 15–19, 2019 (pp. 2588–2592). Graz, Austria. ↑25, 27, 42, 176, 188

Lee, J., Choi, H.-S., Koo, J., & Lee, K. (2020). Disentangling Timbre and Singing Style with Multi-Singer Singing Synthesis System. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7224–7228). Barcelona, Spain. ↑27, 48, 164

Li, N., Liu, S., Liu, Y., Zhao, S., & Liu, M. (2019). Neural Speech Synthesis with Transformer Network. In *The 33rd AAAI Conference on Artificial Intelligence (AAAI)*, January 27–February 1, 2019 (pp. 6706–6713). Honolulu, HI, USA. ↑28, 176

Li, X., & Wang, Z. (2016). A HMM-Based Mandarin Chinese Singing Voice Synthesis System. *IEEE/CAA Journal of Automatica Sinica*, *3*(2), 192–202. ↑18

Ling, Z.-H., Deng, L., & Yu, D. (2013). Modeling Spectral Envelopes Using Restricted Boltzmann Machines and Deep Belief Networks for Statistical Parametric Speech Synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, *21*(10), 2129–2139. doi:10.1109/TASL.2013.2269291. ↑19

Liu, J., Li, C., Ren, Y., Chen, F., Liu, P., & Zhao, Z. (2021a). DiffSinger: Singing Voice Synthesis via Shallow Diffusion Mechanism. arXiv: 2105.02446 [eess.AS]. ↑27, 126, 139, 148

Liu, Z., Miao, C., Zhu, Q., Chen, M., Ma, J., Wang, S., & Xiao, J. (2021b). EfficientSing: A Chinese Singing Voice Synthesis System Using Duration-Free Acoustic Model and HiFi-GAN Vocoder. In *Proceedings of the 22nd Annual Conference of the International Speech Communication Association (Interspeech)*, August 30–September 3, 2021 (pp. 1609–1613). Brno, Czechia. ↑26

Lo, C.-C., Fu, S.-W., Huang, W.-C., Wang, X., Yamagishi, J., Tsao, Y., & Wang, H.-M. (2019). MOSNet: Deep Learning-Based Objective Assessment for Voice Conversion. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech)*, September 15–19, 2019 (pp. 1541–1545). Graz, Austria. ↑76

Lorenzo-Trueba, J., Drugman, T., Latorre, J., Merritt, T., Putrycz, B., Barra-Chicote, R., ... Aggarwal, V. (2019). Towards Achieving Robust Universal Neural Vocoding. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech)*, September 15–19, 2019 (pp. 181–185). Graz, Austria. ↑149, 163

Lu, P., Wu, J., Luan, J., Tan, X., & Zhou, L. (2020). XiaoiceSing: A High-Quality and Integrated Singing Voice Synthesis System. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 1306–1310). Shanghai, China. ↑28, 121, 176, 188

Luo, Y.-J., Hsu, C.-C., Agres, K., & Herremans, D. (2020). Singing Voice Conversion with Disentangled Representations of Singer and Vocal Technique Using Variational Autoencoders. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 3277–3281). Barcelona, Spain. ↑187

Luong, M., & Tran, V.-A. (2021). FlowVocoder: A Small Footprint Neural Vocoder Based Normalizing Flow for Speech Synthesis. arXiv: 2109.13675 [cs.SD]. ↑26

Luong, T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-Based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, September 17–21, 2015 (pp. 1412–1421). Lisbon, Portugal. ↑119

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2017). Least Squares Generative Adversarial Networks. In *IEEE International Conference on Computer Vision (ICCV)*, October 22–29, 2017 (pp. 2813–2821). Venice, Italy. ↑147

Mase, A., Oura, K., Nankaku, Y., & Tokuda, K. (2010). HMM-Based Singing Voice Synthesis System Using Pitch-Shifted Pseudo Training Data. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (Interspeech)*, September 26–30, 2010 (pp. 845–848). Makuhari, Chiba Prefecture, Japan. ↑17, 102

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., ... Bengio, Y. (2017). SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, April 24–26, 2017, Toulon, France. ↑24

Mehta, D., & Quatieri, T. F. (2005). Synthesis, Analysis, and Pitch Modification of the Breathy Vowel. In *Proceedings of the 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, October 16–19, 2005 (pp. 199–202). New Paltz, NY, USA. ↑31

Miao, C., Liang, S., Chen, M., Ma, J., Wang, S., & Xiao, J. (2020). Flow-TTS: A Non-Autoregressive Network for Text to Speech Based on Flow. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7209–7213). Barcelona, Spain. ↑26, 28

Morise, M. (2015). CheapTrick, A Spectral Envelope Estimator for High-Quality Speech Synthesis. *Speech Communication*, *67*, 1–7. doi:10.1016/j.specom.2014.09.003. ↑143

Morise, M. (2016). D4C, a Band-Aperiodicity Estimator for High-Quality Speech Synthesis. *Speech Communication*, *84*, 57–65. doi:10.1016/j.specom.2016.09.001. ↑42, 45, 91, 151

Morise, M., Miyashita, G., & Ozawa, K. (2017). Low-Dimensional Representation of Spectral Envelope Without Deterioration for Full-Band Speech Analysis/Synthesis System. In *Proceedings of the 18th Annual Conference of the International Speech Communication Association (Interspeech)*, August 20–24, 2017 (pp. 409–413). Stockholm, Sweden. ↑46

Morise, M., Yokomori, F., & Ozawa, K. (2016). WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications. *IEICE Transactions on Information and Systems*, *E99.D*, 1877–1884. doi:10.1587/transinf.2015EDP7457. ↑42, 45, 71, 91, 137, 141, 148, 151, 162

Nachmani, E., Polyak, A., Taigman, Y., & Wolf, L. (2018). Fitting New Speakers Based on a Short Untranscribed Sample. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, July 10–15, 2018 (pp. 3683–3691). Stockholm, Sweden. ↑28, 163

Nachmani, E., & Wolf, L. (2019). Unsupervised Singing Voice Conversion. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech)*, September 15–19, 2019 (pp. 2583–2587). Graz, Austria. ↑187

Nakamura, K., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2019). Singing Voice Synthesis Based on Convolutional Neural Networks. arXiv: 1904.06868 [eess.AS]. ↑28, 125

Nakamura, K., Oura, K., Nankaku, Y., & Tokuda, K. (2014). HMM-Based Singing Voice Synthesis and its Application to Japanese and English. In *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–9, 2014 (pp. 265–269). Florence, Italy. ↑17, 103

Nakamura, K., Takaki, S., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2020). Fast and High-Quality Singing Voice Synthesis System Based on Convolutional Neural Networks. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7239–7243). Barcelona, Spain. ↑28

Nishimura, M., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2016). Singing Voice Synthesis Based on Deep Neural Networks. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, September 8–12, 2016 (pp. 2478–2482). San Francisco, CA, USA. ↑20, 21, 81, 105

Odena, A., Dumoulin, V., & Olah, C. (2016). Deconvolution and Checkerboard Artifacts. *Distill*. doi:10.23915/distill.00003. ↑145

Ogawa, I., & Morise, M. (2021). Tohoku Kiritan Singing Database: A Singing Database for Statistical Parametric Singing Synthesis Using Japanese Pop Songs. *Acoustical Science and Technology*, *42*(3), 140–145. doi:10.1250/ast.42.140. ↑59

Okamoto, T., Tachibana, K., Toda, T., Shiga, Y., & Kawai, H. (2018). An Investigation of Subband WaveNet Vocoder Covering Entire Audible Frequency Range with Limited Acoustic Features. In *Proceedings of the 43rd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 15–20, 2018 (pp. 5654–5658). Calgary, AB, Canada. ↑148

Oura, K., Mase, A., Nankaku, Y., & Tokuda, K. (2012). Pitch Adaptive Training for HMM-Based Singing Voice Synthesis. In *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 25–30, 2012 (pp. 5377–5380). Kyoto, Kyoto Prefecture, Japan. ↑17, 105

Oura, K., Mase, A., Yamada, T., Muto, S., Nankaku, Y., & Tokuda, K. (2010). Recent Development of the HMM-Based Singing Voice Synthesis System - Sinsy. In *Proceedings of the 7th*

ISCA Workshop on Speech Synthesis (SSW7), September 22–24, 2010 (pp. 211–216). Kyoto, Kyoto Prefecture, Japan. ↑17, 81, 97, 101, 105

Peng, K., Ping, W., Song, Z., & Zhao, K. (2019). Parallel Neural Text-to-Speech. arXiv: 1905.08459 [cs.CL]. ↑28, 121, 171, 176

Ping, W., Peng, K., & Chen, J. (2019). ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, May 6–9, 2019, New Orleans, LA, USA. ↑25, 136

Ping, W., Peng, K., Gibiansky, A., Arik, S. Ö., Kannan, A., Narang, S., … Miller, J. (2018). Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, April 30–May 3, 2018, Vancouver, BC, Canada. ↑24, 38, 83, 89, 121, 124, 171, 176

Ping, W., Peng, K., Zhao, K., & Song, Z. (2020). WaveFlow: A Compact Flow-Based Model for Raw Audio. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, July 13–18, 2020 (pp. 7706–7716). Virtual event. ↑26

Polyak, A., Wolf, L., Adi, Y., & Taigman, Y. (2020). Unsupervised Cross-Domain Singing Voice Conversion. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 801–805). Shanghai, China. ↑187

Polyak, B. T., & Juditsky, A. B. (1992). Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, *30*(4), 838–855. doi:10.1137/0330046. ↑128, 166

Pons, J., Pascual, S., Cengarle, G., & Serrà, J. (2021). Upsampling Artifacts in Neural Audio Synthesis. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 3005–3009). Toronto, ON, Canada. ↑145

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., & Kudinov, M. A. (2021). Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, July 18–24, 2021 (pp. 8599–8608). Virtual event. ↑27

Prenger, R., Valle, R., & Catanzaro, B. (2019). WaveGlow: A Flow-Based Generative Network for Speech Synthesis. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 3617–3621). Brighton, United Kingdom. ↑25, 42, 136

Pucher, M., Villavicencio, F., & Yamagishi, J. (2016). Development of a Statistical Parametric Synthesis System for Operatic Singing in German. In *Proceedings of the 9th ISCA Speech Synthesis Workshop (SSW9)*, September 13–15, 2016 (pp. 68–73). Sunnyvale, CA, USA. ↑18

Qian, K., Jin, Z., Hasegawa-Johnson, M., & Mysore, G. J. (2020a). F0-Consistent Many-to-Many Non-Parallel Voice Conversion via Conditional Autoencoder. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 6284–6288). Barcelona, Spain. ↑187

Qian, K., Zhang, Y., Chang, S., Hasegawa-Johnson, M., & Cox, D. (2020b). Unsupervised Speech Decomposition via Triple Information Bottleneck. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, July 13–18, 2020 (pp. 7836–7846). Virtual event. ↑187

Qian, K., Zhang, Y., Chang, S., Yang, X., & Hasegawa-Johnson, M. (2019). AutoVC: Zero-Shot Voice Style Transfer with Only Autoencoder Loss. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, June 9–15, 2019 (pp. 5210–5219). Long Beach, CA, USA. ↑187

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, *77*(2), 257–286. doi:10.1109/5.18626. ↑15

Ramachandran, P., Le Paine, T., Khorrami, P., Babaeizadeh, M., Chang, S., Zhang, Y., … Huang, T. (2017). Fast Generation for Convolutional Autoregressive Models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, April 24–26, 2017, Toulon, France. ↑23, 90

Ranzato, M., Chopra, S., Auli, M., & Zaremba, W. (2016). Sequence Level Training with Recurrent Neural Networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, May 2–4, 2016, San Juan, Puerto Rico. ↑89

Reed, S., van den Oord, A., Kalchbrenner, N., Bapst, V., Botvinick, M., & de Freitas, N. (2016). *Generating Interpretable Images with Controllable Structure*. Google DeepMind. London, UK. ↑84, 88

Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., & Liu, T.-Y. (2019). FastSpeech: Fast, Robust and Controllable Text to Speech. In *Advances in Neural Information Processing Systems 32 (NIPS)*, December 8–14, 2019 (pp. 3165–3174). Vancouver, BC, Canada. ↑28, 121, 123, 171, 175

Ren, Y., Tan, X., Qin, T., Luan, J., Zhao, Z., & Liu, T.-Y. (2020). DeepSinger: Singing Voice Synthesis with Data Mined from the Web. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 23–27, 2020 (pp. 1979–1989). San Diego, CA, USA. ↑188

Ribeiro, F. P., Florêncio, D. A. F., Zhang, C., & Seltzer, M. L. (2011). CROWDMOS: An Approach for Crowdsourcing Mean Opinion Score Studies. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 22–27, 2011 (pp. 2416–2419). Prague, Czech Republic. ↑66

Ríos Mestre, A. (1999). La transcripción fonética automática del diccionario electrónico de formas simples flexivas del español: estudio fonológico en el léxico. *Estudios de lingüística del español*, *4*. ↑54

Rodet, X. (1984). Time-Domain Formant-Wave-Function Synthesis. *Computer Music Journal*, *8*(3), 9–14. doi:10.2307/3679809. ↑10

Roebel, A., & Bous, F. (2021). Towards Universal Neural Vocoding with a Multi-Band Excited WaveNet. arXiv: 2110.03329 [eess.AS]. ↑139, 147, 149

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, October 5–9, 2015 (pp. 234–241). Munich, Germany. ↑125

Saino, K., Zen, H., Nankaku, Y., Lee, A., & Tokuda, K. (2006). An HMM-Based Singing Voice Synthesis System. In *Proceedings of the 9th International Conference on Spoken Language Processing (ICSLP - Interspeech)*, September 17–21, 2006 (pp. 2274–2277). Pittsburgh, PA, USA. ↑17

Salakhutdinov, R., & Hinton, G. E. (2009). Semantic Hashing. *International Journal Approximate Reasoning*, *50*(7), 969–978. doi:10.1016/j.ijar.2008.11.006. ↑185

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems 29 (NIPS)*, December 5–10, 2016 (pp. 2226–2234). Barcelona, Spain. ↑75

Salimans, T., Karpathy, A., Chen, X., & Kingma, D. P. (2017). PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, April 24–26, 2017, Toulon, France. ↑22, 24, 86, 87, 89, 213

Sankaran, S., Nanjundan, S., & Anand, G. P. (2021). Anyone GAN Sing. arXiv: 2102.11058 [cs.SD]. ↑26

Serrà, J., Pons, J., & Pascual, S. (2021). SESQA: Semi-Supervised Learning for Speech Quality Assessment. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 381–385). Toronto, ON, Canada. ↑76

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., … Wu, Y. (2018). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. In *Proceedings of the 43rd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 15–20, 2018 (pp. 4779–4783). Calgary, AB, Canada. ↑24, 38, 85, 87–89, 159, 171, 213

Shi, J., Guo, S., Huo, N., Zhang, Y., & Jin, Q. (2020). Sequence-to-Sequence Singing Voice Synthesis with Perceptual Entropy Loss. arXiv: 2010.12024 [eess.AS]. ↑28, 176

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., … Wang, Z. (2016a). Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 27–30, 2016 (pp. 1874–1883). Las Vegas, NV, USA. ↑145

Shi, W., Caballero, J., Theis, L., Huszar, F., Aitken, A. P., Ledig, C., & Wang, Z. (2016b). Is the deconvolution layer the same as a convolutional layer? arXiv: 1609.07009 [cs.CV]. ↑145

Shirota, K., Nakamura, K., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2014). Integration of Speaker and Pitch Adaptive Training for HMM-Based Singing Voice Synthesis. In *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–9, 2014 (pp. 2559–2563). Florence, Italy. ↑17, 159, 164

Smith, J. O. (2011). *Spectral Audio Signal Processing*. W3K Publishing. ↑47, 140

Sotelo, J., Mehri, S., Kumar, K., Santos, J. F., Kastner, K., Courville, A., & Bengio, Y. (2017). Char2Wav: End-to-End Speech Synthesis. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, April 24–26, 2017, Toulon, France. ↑24, 38, 163

Sperber, M., Niehues, J., Neubig, G., Stüker, S., & Waibel, A. (2018). Self-Attentional Acoustic Models. In *Proceedings of the 19th Annual Conference of the International Speech Communication Association (Interspeech)*, September 2–6, 2018 (pp. 3723–3727). Hyderabad, India. ↑123

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958. ↑89, 123

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway Networks. arXiv: 1505.00387 [cs.LG]. ↑85

Steinmetz, C. J., & Reiss, J. D. (2021). pyloudnorm: A Simple Yet Flexible Loudness Meter in Python. In *Proceedings of the 150th AES Convention*, May 25–28, 2021, Virtual event. ↑212

Stilson, T., & Smith, J. O. (1996). Alias-Free Digital Synthesis of Classic Analog Waveforms. In *Proceedings of the 1996 International Computer Music Conference (ICMC)*, August 19–24, 1996, Hong Kong. ↑155

Sugawara, Y., Shiota, S., & Kiya, H. (2019). Checkerboard Artifacts Free Convolutional Neural Networks. *APSIPA Transactions on Signal and Information Processing*, *8*. ↑145

Sun, L., Li, K., Wang, H., Kang, S., & Meng, H. M. (2016). Phonetic Posteriorgrams for Many-to-One Voice Conversion Without Parallel Data Training. In *Proceedings of the 2016 IEEE International Conference on Multimedia and Expo (ICME)*, July 11–15, 2016 (pp. 1–6). Seattle, WA, USA. ↑187

Sundberg, J. (2001). Level and Center Frequency of the Singer's Formant. *Journal of voice*, *15*(2), 176–186. ↑30

Sundberg, J. (2006). The KTH Synthesis of Singing. *Advances in Cognitive Psychology*, *2*(2–3), 131–143. doi:10.2478/v10053-008-0051-y. ↑8

Sundberg, J. (1989). *The Science of the Singing Voice*. Northern Illinois University Press. ↑8, 30

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, December 8–13, 2014 (pp. 3104–3112). Montreal, QC, Canada. ↑118

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 27–30, 2016 (pp. 2818–2826). Las Vegas, NV, USA. ↑75

Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2010). A Short-Time Objective Intelligibility Measure for Time-Frequency Weighted Noisy Speech. In *Proceedings of the 2010*

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), March 14–19, 2010 (pp. 4214–4217). Dallas, TX, USA. ↑77

Tachibana, H., Uenoyama, K., & Aihara, S. (2018). Efficient Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention. In *Proceedings of the 43rd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 15–20, 2018 (pp. 4784–4788). Calgary, AB, Canada. ↑24, 85, 176

Taigman, Y., Wolf, L., Polyak, A., & Nachmani, E. (2018). VoiceLoop: Voice Fitting and Synthesis via a Phonological Loop. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, April 30–May 3, 2018, Vancouver, BC, Canada. ↑28, 163

Takamichi, S., Ikawa, S., Tanji, N., & Saruwatari, H. (2018). *JSUT Collection Ver. 1: Voice Corpora of Reading-Style Speech, Singing Voice and Vocal Imitation Uttered by a Single Japanese Speaker*. Graduate School of Information Science and Technology, The University of Tokyo. Retrieved from https://sites.google.com/site/shinnosuketakamichi/publication/jsut-song. ↑59

Takamichi, S., Toda, T., Black, A. W., Neubig, G., Sakti, S., & Nakamura, S. (2016). Postfilters to Modify the Modulation Spectrum for Statistical Parametric Speech Synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(4), 755–767. doi:10.1109/TASLP.2016.2522655. ↑70, 72

Tamamori, A., Hayashi, T., Kobayashi, K., Takeda, K., & Toda, T. (2017). Speaker-Dependent WaveNet Vocoder. In *Proceedings of the 18th Annual Conference of the International Speech Communication Association (Interspeech)*, August 20–24, 2017 (pp. 1118–1122). Stockholm, Sweden. ↑42, 163, 190

Taylor, P. (2009). *Text-to-Speech Synthesis*. doi:10.1017/CBO9780511816338. ↑32, 54, 101

Theis, L., van den Oord, A., & Bethge, M. (2016). A Note on the Evaluation of Generative Models. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, May 2–4, 2016, San Juan, Puerto Rico. ↑65

Tian, Q., Chen, Y., Zhang, Z., Lu, H., Chen, L., Xie, L., & Liu, S. (2020a). TFGAN: Time and Frequency Domain Based Generative Adversarial Network for High-Fidelity Speech Synthesis. arXiv: 2011.12206 [eess.AS]. ↑26

Tian, Q., Zhang, Z., Lu, H., Chen, L.-H., & Liu, S. (2020b). FeatherWave: An Efficient High-Fidelity Neural Vocoder with Multi-Band Linear Prediction. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 195–199). Shanghai, China. ↑148

Titze, I. R., & Story, B. H. (1993). The Iowa Singing Synthesis. In *Proceedings of the 2nd Stockholm Music Acoustics Conference (SMAC)*, July 28–August 1, 1993 (p. 294). Stockholm, Sweden. ↑9

Toda, T., & Tokuda, K. (2007). A Speech Parameter Generation Algorithm Considering Global Variance for HMM-Based Speech Synthesis. *IEICE Transactions on Information and Systems*, *E90-D*(5), 816–824. doi:10.1093/ietisy/e90-d.5.816. ↑16, 70

Tokuda, K., Kobayashi, T., Masuko, T., & Imai, S. (1994). Mel-Generalized Cepstral Analysis - A Unified Approach to Speech Spectral Estimation. In *Proceedings of the 3rd International Conference on Spoken Language Processing (ICSLP)*, September 18–22, 1994, Yokohama, Kanagawa Prefecture, Japan. ↑46, 91

Tokuda, K., Masuko, T., Miyazaki, N., & Kobayashi, T. (1999). Hidden Markov Models Based on Multi-Space Probability Distribution for Pitch Pattern Modeling. In *Proceedings of the 8th Annual Conference of the International Speech Communication Association (Interspeech)*, March 15–19, 1999 (pp. 229–232). Phoenix, AZ, USA. ↑17, 45

Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., & Kitamura, T. (2000). Speech Parameter Generation Algorithms for HMM-Based Speech Synthesis. In *Proceedings of the 25th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, July 5–9, 2000 (Vol. 3, pp. 1315–1318). Istanbul, Turkey. ↑16

Tuerk, C., & Robinson, T. (1993). Speech Synthesis Using Artificial Neural Networks Trained on Cepstral Coefficients. In *Proceedings of the 3rd European Conference on Speech Communication and Technology (Eurospeech)*, September 21–23, 1993 (pp. 1713–1716). Berlin, Germany. ↑19

Ueda, N., & Nakano, R. (1998). Deterministic Annealing EM Algorithm. *Neural networks*, *11*(2), 271–282. doi:10.1016/S0893-6080(97)00133-0. ↑63, 209

Umbert, M., Bonada, J., & Blaauw, M. (2013). Generating Singing Voice Expression Contours Based on Unit Selection. In *Proceedings of the 4th Stockholm Music Acoustics Conference (SMAC)*, July 30–August 3, 2013 (pp. 315–320). Stockholm, Sweden. ↑14, 101

Umbert, M., Bonada, J., Goto, M., Nakano, T., & Sundberg, J. (2015). Expression Control in Singing Voice Synthesis: Features, Approaches, Evaluation, and Challenges. *IEEE Signal Processing Magazine*, *32*(6), 55–73. doi:10.1109/MSP.2015.2424572. ↑14, 73, 160

Uneson, M. (2002). *Burcas - A Simple Concatenation-Based MIDI-to-Singing Voice Synthesis System for Swedish* (Master's thesis, Lund University). ↑14

Valin, J.-M., & Skoglund, J. (2019). LPCNET: Improving Neural Speech Synthesis Through Linear Prediction. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 5891–5895). Brighton, United Kingdom. ↑148

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016a). WaveNet: A Generative Model for Raw Audio. arXiv: 1609.03499 [cs.SD]. ↑19, 21, 52, 81, 83, 121, 141, 150, 159, 160, 183

van den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016b). Pixel Recurrent Neural Networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, June 19–24, 2016 (Vol. 48, pp. 1747–1756). New York, NY, USA. ↑22, 83, 85

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., & Kavukcuoglu, K. (2016c). Conditional Image Generation with PixelCNN Decoders. In *Advances in Neural Information Processing Systems 29 (NIPS)*, December 5–10, 2016 (pp. 4790–4798). Barcelona, Spain. ↑22, 83, 85 van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., … Hassabis, D. (2018). Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, July 10–15, 2018 (pp. 3915–3923). Stockholm, Sweden. ↑25, 87, 136

van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems 30 (NIPS)*, December 4–9, 2017 (pp. 6306–6315). Long Beach, CA, USA. ↑181, 187

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems 30 (NIPS)*, December 4–9, 2017 (pp. 6000–6010). Long Beach, CA, USA. ↑28, 51, 117, 119, 121, 123, 128, 174, 189

Villavicencio, F., Bonada, J., Yamagishi, J., & Pucher, M. (2015). *Efficient Pitch Estimation on Natural Opera-Singing by a Spectral Correlation Based Strategy*. Information Processing Society of Japan. ↑44, 149, 151, 152

Wada, Y., Nishikimi, R., Nakamura, E., Itoyama, K., & Yoshii, K. (2018). Sequential Generation of Singing F0 Contours from Musical Note Sequences Based on WaveNet. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, November 12–15, 2018 (pp. 983–989). Honolulu, HI, USA. ↑23

Wan, V., Agiomyrgiannakis, Y., Silén, H., & Vít, J. (2017). Google's Next-Generation Real-Time Unit-Selection Synthesizer Using Sequence-to-Sequence LSTM-Based Autoencoders. In *Proceedings of the 18th Annual Conference of the International Speech Communication Association (Interspeech)*, August 20–24, 2017 (pp. 1143–1147). Stockholm, Sweden. ↑185

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020a). Linformer: Self-Attention with Linear Complexity. arXiv: 2006.04768 [cs.LG]. ↑124

Wang, X., Takaki, S., & Yamagishi, J. (2020b). Neural Source-Filter Waveform Models for Statistical Parametric Speech Synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 402–415. doi:10.1109/TASLP.2019.2956145. ↑48, 148

Wang, X., Takaki, S., & Yamagishi, J. (2019). Neural Source-Filter-Based Waveform Model for Statistical Parametric Speech Synthesis. In *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 12–17, 2019 (pp. 5916–5920). Brighton, United Kingdom. ↑42, 48, 139, 148

Wang, X., & Yamagishi, J. (2019). Neural Harmonic-plus-Noise Waveform Model with Trainable Maximum Voice Frequency for Text-to-Speech Synthesis. In *Proceedings of the 10th ISCA Workshop on Speech Synthesis (SSW10)*, September 20–22, 2019 (pp. 1–6). Vienna, Austria. ↑48, 148

Wang, X., & Yamagishi, J. (2020). Using Cyclic Noise as the Source Signal for Neural Source-Filter-Based Speech Waveform Model. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 1992–1996). Shanghai, China. ↑48, 139, 148

Wang, Y., Wang, X., Zhu, P., Wu, J., Li, H., Xue, H., … Bi, M. (2022). Opencpop: A High-Quality Open Source Chinese Popular Song Corpus for Singing Voice Synthesis. arXiv: 2201.07429 [cs.SD]. ↑57

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., … Saurous, R. A. (2017). Tacotron: A Fully End-to-End Text-to-Speech Synthesis Model. In *Proceedings of the 18th Annual Conference of the International Speech Communication Association (Interspeech)*, August 20–24, 2017 (pp. 4006–4010). Stockholm, Sweden. ↑24, 85, 124, 171

Wefers, F., & Vorländer, M. (2014). Efficient Time-Varying FIR Filtering Using Crossfading Implemented in the DFT Domain. In *Proceedings of the 7th Forum Acusticum*, September 7–12, 2014 (pp. 7–12). Krakow, Poland. ↑143

Weiss, R. J., Skerry-Ryan, R. J., Battenberg, E., Mariooryad, S., & Kingma, D. P. (2021). Wave-Tacotron: Spectrogram-Free End-to-End Text-to-Speech Synthesis. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 5679–5683). Toronto, ON, Canada. ↑26, 28, 38, 39

Wu, Y.-C., Hayashi, T., Okamoto, T., Kawai, H., & Toda, T. (2021). Quasi-Periodic Parallel Wave-GAN: A Non-Autoregressive Raw Waveform Generative Model with Pitch-Dependent Dilated Convolution Neural Network. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *29*, 792–806. ↑26

Wu, J., & Luan, J. (2020). Adversarially Trained Multi-Singer Sequence-to-Sequence Singing Synthesizer. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, October 25–29, 2020 (pp. 1296–1300). Shanghai, China. ↑26, 164, 176, 188

Wu, N.-Q., & Ling, Z.-H. (2020). WaveFFJORD: FFJORD-Based Vocoder for Statistical Parametric Speech Synthesis. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 7214–7218). Barcelona, Spain. ↑26

Yamamoto, R., Song, E., Hwang, M.-J., & Kim, J.-M. (2021). Parallel Waveform Synthesis Based on Generative Adversarial Networks with Voicing-Aware Conditional Discriminators. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 6–11, 2021 (pp. 6039–6043). Toronto, ON, Canada. ↑26

Yamamoto, R., Song, E., & Kim, J.-M. (2020). Parallel WaveGAN: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 4–8, 2020 (pp. 6199–6203). Barcelona, Spain. ↑26, 42, 83, 136, 145, 148, 150

Yang, G., Yang, S., Liu, K., Fang, P., Chen, W., & Xie, L. (2021). Multi-Band MelGan: Faster Waveform Generation for High-Quality Text-to-Speech. In *IEEE Spoken Language Technology Workshop, SLT 2021*, January 19–22, 2021 (pp. 492–498). Shenzhen, China. ↑26, 148

Yi, Y.-H., Ai, Y., Ling, Z.-H., & Dai, L.-R. (2019). Singing Voice Synthesis Using Deep Autoregressive Neural Networks for Acoustic Modeling. In *Proceedings of the 20th Annual*

Conference of the International Speech Communication Association (Interspeech), September 15–19, 2019 (pp. 2593–2597). Graz, Austria. ↑23, 125

Yoneyama, R., Wu, Y.-C., & Toda, T. (2021). Unified Source-Filter GAN: Unified Source-Filter Network Based on Factorization of Quasi-Periodic Parallel WaveGAN. arXiv: 2104.04668 [cs.SD]. ↑26, 148

Yu, C., Lu, H., Hu, N., Yu, M., Weng, C., Xu, K., … Yu, D. (2019). DurIAN: Duration Informed Attention Network for Multimodal Synthesis. arXiv: 1909.01700 [cs.CL]. ↑148

Yu, F., & Koltun, V. (2016). Multi-Scale Context Aggregation by Dilated Convolutions. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, May 2–4, 2016, San Juan, Puerto Rico. ↑22, 83

Zen, H., Agiomyrgiannakis, Y., Egberts, N., Henderson, F., & Szczepaniak, P. (2016). Fast, Compact, and High Quality LSTM-RNN Based Statistical Parametric Speech Synthesizers for Mobile Devices. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, September 8–12, 2016 (pp. 1230–1234). San Francisco, CA, USA. ↑20

Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A. W., & Tokuda, K. (2007). The HMM-Based Speech Synthesis System (HTS) Version 2.0. In *Proceedings of the 6th ISCA Workshop on Speech Synthesis (SSW6)*, August 22–24, 2007 (pp. 294–299). Bonn, Germany. ↑56, 63, 97

Zen, H., & Sak, H. (2015). Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis. In *Proceedings of the 40th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 19, 3015–April 24, 2015 (pp. 4470–4474). South Brisbane, Australia. ↑20, 21

Zen, H., & Senior, A. (2014). Deep Mixture Density Networks for Acoustic Modeling in Statistical Parametric Speech Synthesis. In *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 4–9, 2014 (pp. 3844–3848). Florence, Italy. ↑20, 21

Zen, H., Senior, A., & Schuster, M. (2013). Statistical Parametric Speech Synthesis Using Deep Neural Networks. In *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 26–31, 2013 (pp. 7962–7966). Vancouver, BC, Canada. ↑20, 21

Zera, J., Gauffin, J., & Sundberg, J. (1984). Synthesis of Selected VCV-Syllables in Singing. In *Proceedings of the International Computer Music Conference (ICMC)*, October 19–23, 1984 (pp. 83–86). Paris, France. ↑9

Zhai, B., Gao, T., Xue, F., Rothchild, D., Wu, B., Gonzalez, J. E., & Keutzer, K. (2020). SqueezeWave: Extremely Lightweight Vocoders for On-Device Speech Synthesis. arXiv: 2001.05685 [cs.SD]. ↑26