



**THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ**  
Spécialité **Traitements du signal**

*ED130 - École doctorale Informatique, Télécommunications et Électronique (Paris)  
Sciences et Technologie de la Musique et du Son (UMR 9912 STMS)  
Institut de Recherche et de Coordination Acoustique Musique (IRCAM)  
Équipe Analyse/Synthèse des Sons.*

---

# A neural voice transformation framework for modification of pitch and intensity

---

*Présenté par*

**Frederik Bous**

Dirigée par Axel Roebel

Soutenue le 21 septembre 2023 devant le jury composé de :

|                          |             |
|--------------------------|-------------|
| Thierry Dutoit           | Rapporteur  |
| Yannis Stylianou         | Rapporteur  |
| Christophe d'Alessandro* | Examinateur |
| Jordi Bonada             | Examinateur |
| Natalie Henrich          | Examinateur |

*\*Président du jury*

UMR 9912 STMS  
IRCAM, Sorbonne Université, CNRS, Ministère de la Culture

Équipe Analyse/Synthèse des Sons

1, place Igor Stravinsky  
75004 Paris, FRANCE

PhD thesis presented by Frederik Bous  
Thesis supervision by Axel Roebel  
Version 1.1  
Screen edition  
Compiled on 2023-12-18

This thesis is accompanied by additional media, which can be downloaded from <http://thesis.fnab.xyz>.



Copyright © 2023 Frederik Bous  
 [orcid.org/0000-0002-7477-7600](https://orcid.org/0000-0002-7477-7600)



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. The full text of the licence can be found under <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

# Abstract

Human voice has been a great source of fascination and an object of research for over 100 years. During that time numerous technologies have sprouted around the voice, such as the vocoder, which provides a parametric representation of the voice, commonly used for voice transformation. From this tradition, the limitations of purely signal processing based approaches are evident: To create meaningful transformations the codependencies between different voice properties have to be understood well and modelled precisely. Modelling these correlations with heuristics obtained by empiric studies is not sufficient to create natural results. It is necessary to extract information about the voice systematically and use this information during the transformation process automatically. Recent advances in computer hardware permit this systematic analysis of data by means of machine learning. This thesis thus uses machine learning to create a neural voice transformation framework.

The proposed neural voice transformation framework works in two stages: First a neural vocoder allows mapping between a raw audio and a mel-spectrogram representation of voice signals. Secondly, an auto-encoder with information bottleneck allows disentangling various voice properties from the remaining information. The auto-encoder allows changing one voice property while automatically adjusting the remaining voice properties. In the first part of this thesis, we discuss different approaches to neural vocoding and reason why the mel-spectrogram is better suited for neural voice transformations than conventional parametric vocoder spaces.

In the second part we discuss the information bottleneck auto-encoder. The auto-encoder creates a latent code that is independent of its conditional input. Using the latent code the synthesizer can perform the transformation by combining the original latent code with a modified parameter curve. We transform the voice using two control parameters: the fundamental frequency and the voice level. Transformation of the fundamental frequency is an objective with a long history. Using the fundamental frequency allows us to compare our approach to existing techniques and study how the auto-encoder models the dependency on other properties in a well known environment. For the voice level, we face the problem that annotations hardly exist. Therefore, first we provide a new estimation technique for voice level in large voice databases, and subsequently use the voice level annotations to train a bottleneck auto-encoder that allows changing the voice level.

# Abstract

La voix humaine est une grande source de fascination et un objet de recherche depuis plus de 100 ans. Pendant ce temps, de nombreuses technologies ont germées autour de la voix, comme le vocodeur, qui fournit une représentation paramétrique de la voix, couramment utilisée pour la transformation de la voix. Dans cette tradition, les limites des approches basées uniquement sur le traitement du signal sont évidentes : Pour créer des transformations cohérentes, les dépendances entre les différentes propriétés vocales doivent être bien comprises et modélisées avec précision. Modéliser ces corrélations avec des heuristiques obtenues par des études empiriques ne suffit pas à créer des résultats naturels. Il est nécessaire d'extraire systématiquement des informations sur la voix et d'utiliser automatiquement ces informations lors du processus de transformation. Les progrès récents de la puissance de calcul permettent cette analyse systématique des données au moyen de l'apprentissage automatique. Cette thèse utilise donc l'apprentissage automatique pour créer un système neuronal de transformation de la voix.

Le système neuronal de transformation de la voix, présenté ici, fonctionne en deux étapes : Tout d'abord, un vocodeur neuronal permet d'établir une correspondance entre la forme d'onde et une représentation mel-spectrogramme des signaux vocaux. Ensuite, un auto-encodeur avec un goulot d'étranglement permet de démêler différentes propriétés de la voix du reste de l'information. L'auto-encodeur permet de modifier une propriété de la voix tout en ajustant automatiquement d'autres caractéristiques de façon à en conserver le réalisme. Dans la première partie de cette thèse, nous comparons différentes approches du vocodage neuronal et nous expliquons pourquoi la représentation mel-spectrogramme est plus adapté pour la transformation neuronale de la voix plutôt que les espaces paramétriques du vocodeur conventionnels.

Dans la deuxième partie, nous présentons l'auto-encodeur avec goulot d'étranglement de l'information. L'auto-encodeur crée un code latent indépendant du conditionnement en entrée. En utilisant ce code latent, le synthétiseur peut effectuer la transformation en combinant le code latent original avec une courbe de paramètres modifiée. Nous transformons la voix en utilisant deux paramètres de contrôle : la fréquence fondamentale et le niveau sonore vocal. La transformation de la fréquence fondamentale est un problème qui a longtemps été abordé : Notre approche est comparable aux techniques existantes puisqu'elles utilisent la fréquence fondamentale comme paramètre. Cela nous permet également d'étudier comment l'auto-encodeur modélise les dépendances entre la fréquence fondamentale et d'autres propriétés de la voix dans un environnement connu. Quant au niveau sonore vocal, nous sommes confrontés au problème de la rareté des annotations. Par conséquent, nous proposons d'abord une nouvelle technique d'estimation du niveau sonore vocal dans

---

de grandes bases de données de voix ; puis nous utilisons ces annotations pour entraîner un auto-encodeur avec goulot d'étranglement permettant de modifier le niveau sonore vocal.

# Acknowledgement

Though our own vanity sometimes lets us forget this, but we are nothing without the help of others. This thesis would not have been possible if it was not due to the help of my advisors, colleagues, friends, and family. Thus, commanded by deepest gratitude, it is necessary to mention a few names to acknowledge their contribution and attest my indebtedness.

My sincerest thank goes to my thesis advisor **Axel Roebel**. The amount of advice and support I received from Axel, professional and personal, is innumerable and invaluable. The amount of attention I got from Axel during my thesis was extraordinary, which is not a matter of course for thesis supervisors. I am honoured to have been a student of Axel Roebel.

During most of my thesis I had the pleasure of working together with the artist **Judith Deschamps**. After Axel, Judith was the second most influential person on my thesis. Our collaboration was a great motivator of my work and allowed me to explore the abilities and applications of my research in an interesting project. I am happy to have had Judith at my side as a collaborator and as a friend.

It was a great pleasure to do my PhD at IRCAM thanks to all the people that make this place magical. In particular, I am grateful to **Nicolas Obin** for being persistent, where I had already given up and to **Hugues Vinet** for his support allowing me to finish my PhD without worries. I'm grateful to **Luc Ardaillon** for his preliminary work I could build on, and I'm glad I could work together with Luc to extend some of his work. Thank you to **Darick Lean**, who helped me organize all the databases, thank you to everyone, who helped me by proofreading my manuscripts, **Emily Gruber**, **Théodor Lemerle**, **Lenny Renault**, **Léane Salais**, and **Daniel Wolf**, and thank you to **Yann Teytaut** and **Clément Le Moine Veillon** for sharing their LaTeX source. A big thank you to **Pierre Guillot** for helping me get my software CIRCE ready for production which was a hell of a hassle and would not have been possible without Pierre.

My work has been greatly influenced by the artistic work of the composers **Aïda Shirazi**, **Sachie Kobayashi**, and **Omer Barash**, whom I am thankful for using my research in their compositions, helping me develop the tools for a wider audience. Likewise, collaborating with **António Breitenfeld Sá-Dantas** on the Farinelli project was a great pleasure and a source of many ideas. Furthermore, I am grateful to all the singers who gave their voice to the Farinelli project, allowing us to create a new, weird, and interesting voice.

Finally, I would like to acknowledge my friends and family who have been a great support to me during these years. Thank you to **Sascha Knüttel**, **Kay Wohlfarth**, and **Christoph von Erffa** for standing by me when things got rough, and to my family that I can rely on and that is always there for me when I need them.

# Contents

|  |           |
|--|-----------|
| <b>1. Introduction</b>   | <b>1</b>  |
| 1.1. Objectives . . . . .  | 1         |
| 1.2. Context . . . . .   | 2         |
| 1.2.1. Artistic residency <i>Farinelli</i> . . . . .                 | 2         |
| 1.2.2. ANR Project <i>ARS</i> . . . . .                              | 3         |
| 1.2.3. IRCAM Singing Synthesizer . . . . .                           | 3         |
| 1.3. Main contributions . . . . .                                    | 3         |
| 1.4. Outline . . . . .   | 5         |
| <b>2. History and fundamentals</b>                                   | <b>6</b>  |
| 2.1. A short history of voice synthesis and transformation . . . . . | 6         |
| 2.2. Human perception of sound . . . . .                             | 9         |
| 2.2.1. Perception of pitch . . . . .                                 | 11        |
| 2.2.2. Pitch scales . . . . .  | 12        |
| 2.2.3. Loudness . . . . .  | 14        |
| 2.3. Fundamentals of the voice . . . . .                             | 15        |
| 2.3.1. The theory of speech production . . . . .                     | 16        |
| 2.3.2. Glottal flow models . . . . .                                 | 18        |
| 2.4. Neural networks . . . . .                                       | 21        |
| 2.4.1. Neural arithmetic on voice signals . . . . .                  | 24        |
| 2.4.2. Common network architecture types for voice . . . . .         | 27        |
| 2.5. Datasets . . . . .  | 30        |
| 2.5.1. Speech datasets . . . . .                                     | 31        |
| 2.5.2. Singing datasets . . . . .                                    | 32        |
| 2.5.3. Combining datasets . . . . .                                  | 34        |
| 2.5.4. Dataset analysis . . . . .                                    | 36        |
| 2.6. Perceptive tests . . . . .                                      | 39        |
| 2.6.1. Conception . . . . .  | 39        |
| 2.6.2. Procedure . . . . .   | 41        |
| 2.6.3. Evaluation . . . . .  | 42        |
| 2.7. Conclusion . . . . .  | 46        |
| <b>3. State of the art</b>   | <b>47</b> |
| 3.1. Analysis of voice . . . . .                                     | 47        |
| 3.1.1. Fundamental frequency . . . . .                               | 48        |
| 3.1.2. Glottal pulse signal . . . . .                                | 48        |
| 3.1.3. Voice timbre . . . . .  | 50        |
| 3.1.4. Noise component . . . . .                                     | 50        |

|           |  |           |
|-----------|--|-----------|
| 3.2.      | Vocoders . . . . .   | 51        |
| 3.2.1.    | Pitch synchronous overlap and add . . . . .                    | 52        |
| 3.2.2.    | Non-parametric vocoders . . . . .                              | 52        |
| 3.2.3.    | Source-filter parametric vocoder . . . . .                     | 53        |
| 3.2.4.    | Neural vocoder . . . . .                                       | 53        |
| 3.2.5.    | Neural source-filter vocoders . . . . .                        | 53        |
| 3.2.6.    | Neural vocoder using the mel-spectrogram . . . . .             | 54        |
| 3.3.      | Synthesis . . . . .  | 55        |
| 3.3.1.    | Concatenative synthesis . . . . .                              | 55        |
| 3.3.2.    | Statistical parametric speech (singing) synthesis . . . . .    | 56        |
| 3.3.3.    | Neural parametric synthesis . . . . .                          | 56        |
| 3.3.4.    | Neural end-to-end synthesis . . . . .                          | 56        |
| 3.4.      | Transformation . . . . .                                       | 57        |
| 3.4.1.    | Modelling dependencies . . . . .                               | 57        |
| 3.4.2.    | Disentanglement of parameters . . . . .                        | 58        |
| 3.4.3.    | Higher level voice conversion . . . . .                        | 60        |
| 3.5.      | Conclusion . . . . .   | 61        |
| <b>4.</b> | <b>Glottal closure instance extraction</b>                     | <b>63</b> |
| 4.1.      | Introduction . . . . .   | 63        |
| 4.1.1.    | Glottal closure instant . . . . .                              | 64        |
| 4.1.2.    | Overview . . . . .   | 64        |
| 4.2.      | Approach . . . . .   | 65        |
| 4.2.1.    | The synthetic database . . . . .                               | 65        |
| 4.2.2.    | The analysis-synthesis framework . . . . .                     | 65        |
| 4.2.3.    | Loss functions . . . . .                                       | 66        |
| 4.2.4.    | Training procedure . . . . .                                   | 69        |
| 4.3.      | Evaluation . . . . .   | 69        |
| 4.3.1.    | Test setup . . . . .   | 70        |
| 4.3.2.    | Evaluation metrics . . . . .                                   | 70        |
| 4.4.      | Results . . . . .  | 71        |
| 4.5.      | Conclusion . . . . .   | 72        |
| <b>5.</b> | <b>Neural vocoder</b>  | <b>73</b> |
| 5.1.      | Desired properties of a vocoder . . . . .                      | 73        |
| 5.1.1.    | Mathematical model . . . . .                                   | 74        |
| 5.1.2.    | Desired properties of the parametric space . . . . .           | 75        |
| 5.1.3.    | Practical requirements on the vocoder implementation . . . . . | 77        |
| 5.2.      | Parametric spaces for voice . . . . .                          | 77        |
| 5.2.1.    | Time-domain signal and end-to-end processing . . . . .         | 78        |
| 5.2.2.    | Classical vocoder parameters . . . . .                         | 78        |
| 5.2.3.    | Mel-spectrogram . . . . .                                      | 80        |
| 5.2.4.    | Conclusion . . . . .   | 83        |
| 5.3.      | Neural vocoding using the mel-spectrogram . . . . .            | 83        |

|           |  |            |
|-----------|--|------------|
| 5.4.      | Multi-Band Excited WaveNet . . . . .                   | 85         |
| 5.4.1.    | Network architecture . . . . .                         | 85         |
| 5.4.2.    | Training . . . . .                                     | 87         |
| 5.4.3.    | Discussion . . . . .                                   | 88         |
| 5.5.      | Conclusion . . . . .                                   | 90         |
| <b>6.</b> | <b>Bottleneck auto-encoder</b>                         | <b>91</b>  |
| 6.1.      | Introduction . . . . .                                 | 92         |
| 6.1.1.    | Applications . . . . .                                 | 92         |
| 6.1.2.    | Challenges of pitch transformation . . . . .           | 93         |
| 6.1.3.    | Mathematical description of disentanglement . . . . .  | 93         |
| 6.1.4.    | Goals of this chapter . . . . .                        | 95         |
| 6.1.5.    | Outline . . . . .                                      | 96         |
| 6.2.      | Methodology . . . . .                                  | 96         |
| 6.2.1.    | Proposed model . . . . .                               | 96         |
| 6.2.2.    | Datasets . . . . .                                     | 98         |
| 6.2.3.    | Nomenclature . . . . .                                 | 99         |
| 6.2.4.    | Experimental setup . . . . .                           | 99         |
| 6.3.      | Experimental results . . . . .                         | 101        |
| 6.3.1.    | $f_0$ accuracy . . . . .                               | 101        |
| 6.3.2.    | Synthesis quality . . . . .                            | 104        |
| 6.3.3.    | Investigation of the latent code . . . . .             | 107        |
| 6.4.      | Conclusion . . . . .                                   | 111        |
| 6.5.      | Future work . . . . .                                  | 111        |
| <b>7.</b> | <b>Transformation of perceived voice level</b>         | <b>113</b> |
| 7.1.      | Introduction . . . . .                                 | 113        |
| 7.1.1.    | Terminology . . . . .                                  | 113        |
| 7.1.2.    | Problem formulation . . . . .                          | 115        |
| 7.1.3.    | Outline . . . . .                                      | 115        |
| 7.2.      | Extracting voice level from audio recordings . . . . . | 116        |
| 7.2.1.    | Objective . . . . .                                    | 116        |
| 7.2.2.    | Learnt recording factor . . . . .                      | 117        |
| 7.2.3.    | Adaptive recording factor . . . . .                    | 118        |
| 7.3.      | Proposed voice level transformations . . . . .         | 120        |
| 7.4.      | Experiments . . . . .                                  | 121        |
| 7.4.1.    | Data . . . . .   | 121        |
| 7.4.2.    | Target curves . . . . .                                | 121        |
| 7.4.3.    | Architecture . . . . .                                 | 122        |
| 7.4.4.    | Auto-encoders . . . . .                                | 122        |
| 7.4.5.    | Evaluation methods . . . . .                           | 122        |
| 7.5.      | Results . . . . .                                      | 123        |
| 7.5.1.    | Classification of dynamic . . . . .                    | 123        |
| 7.5.2.    | Accuracies . . . . .                                   | 125        |
| 7.5.3.    | Perceptive test . . . . .                              | 125        |

|   |            |
|---|------------|
| 7.6. Conclusions . . . . .  | 127        |
| <b>8. Applications</b>  | <b>129</b> |
| 8.1. Farinelli singing voice . . . . .  | 129        |
| 8.1.1. The castrato voice of Farinelli . . . . .                                  | 130        |
| 8.1.2. Related work . . . . .   | 130        |
| 8.1.3. Musical material . . . . .   | 131        |
| 8.1.4. Creating a hybrid voice . . . . .  | 132        |
| 8.1.5. Improvisations over <i>Quell’usignolo</i> . . . . .                        | 134        |
| 8.1.6. Observations and difficulties . . . . .                                    | 136        |
| 8.1.7. Conclusion . . . . .   | 137        |
| 8.2. Musical compositions with CIRCE . . . . .                                    | 138        |
| 8.2.1. Aïda Shirazi: <i>Né entre corps</i> (2022) . . . . .                       | 138        |
| 8.2.2. Sachie Kobayashi:<br><i>Day 0 - trans-instrumentalism</i> (2022) . . . . . | 139        |
| 8.2.3. Omer Barash: <i>G.N.Z.</i> (2023) . . . . .                                | 141        |
| <b>9. Conclusion</b>  | <b>143</b> |
| 9.1. Recapitulation . . . . .   | 143        |
| 9.1.1. Neural vocoders . . . . .  | 143        |
| 9.1.2. Bottleneck auto-encoder . . . . .  | 144        |
| 9.2. The neural voice transformation framework . . . . .                          | 144        |
| 9.3. Applications . . . . .   | 145        |
| 9.3.1. Audio post-production . . . . .  | 146        |
| 9.3.2. Creation of new, hybrid-voices . . . . .                                   | 146        |
| 9.3.3. Obfuscation and anonymization . . . . .                                    | 146        |
| 9.3.4. Singing voice synthesis . . . . .  | 146        |
| 9.3.5. Speech enhancement and humanization . . . . .                              | 147        |
| 9.4. Limitations and Future work . . . . .  | 147        |
| 9.4.1. Issues with synthesis quality . . . . .                                    | 148        |
| 9.4.2. Controllable voice properties . . . . .                                    | 152        |
| 9.4.3. Understanding the latent space of the auto-encoder . . . . .               | 155        |
| <b>List of figures</b>  | <b>157</b> |
| <b>List of tables</b>   | <b>158</b> |
| <b>A. List of model configurations and their layers</b>                           | <b>159</b> |
| A.1. Models from Chapter 4: Pulse detection . . . . .                             | 160        |
| A.2. Models from Chapter 6: Bottleneck auto encoder . . . . .                     | 161        |
| A.3. Models from Chapter 7: Voice level transformation . . . . .                  | 162        |
| <b>Glossary</b>   | <b>163</b> |
| <b>Bibliography</b>   | <b>170</b> |

# 1. Introduction

This thesis is the result of the last three years of my research at **IRCAM**. It extends the research from my master thesis (**Bous, 2019**) where we were investigating means to synthesize spectral envelopes from the fundamental frequency and phonetic transcription. The limitations of previous methods of modelling the voice were apparent already then: The classical parameters used to describe voice, spectral envelope and fundamental frequency, are highly dependent on each other. Changing the fundamental frequency without changing the spectral envelope creates unnatural results and previous attempts to adjust the spectral envelope to changes of the fundamental frequency were unsatisfactory. The voice has been an object of research for a long time and relationships between different aspects of voice, like the vocal tract filter and the fundamental frequency are well known. Nevertheless, this knowledge is rather approximative than precise: While we know from the research on voice production mechanism that different aspects of the voice interact, the theories remain approximative and do not provide mathematical models that are precise enough for natural voice synthesis. To obtain mathematical descriptions other methods are required.

## 1.1. Objectives

Machine learning provides a tool to systematically and automatically obtain mathematical models from data. Technological advance in computational power has caught up with the requirements for large-scale deep learning, which makes it a suitable tool to formalize the relationships of different aspects in the voice. This leads to the central research objective of this thesis:

**We want to use deep learning to transform interesting voice properties in voice recordings with high audio quality and natural sounding voice.**

The objective of this work is to develop a neural voice transformation framework that allows modelling the relationships between different voice properties. The neural voice transformation framework should be able to learn these relationships from available voice recordings. When changing one voice property, the **neural voice transformation framework** should automatically adapt the remaining voice properties such that the speech remains natural. Notably, we want to address the following problems:

**Transformable properties** In the past, Voice transformation has largely been oriented around transformations of pitch. Nowadays, we observe that abstract

properties like voice identity are popular in the voice conversion community. In this thesis, the goal is to extend the canon of signal related transformable properties. While in this research, we are mainly occupied with pitch, we will do this by developing a transformation framework that can be applied to other properties as well, and we will use the voice level as another such property.

**Transformation range** In traditional pitch transformation methods the range of possible transformations is limited. If the transposition is too large the result sounds unnatural. We want to extend the amount of transposition that keeps producing natural voice. This requires automatic adjustments in either voice identity, vocal technique or both. Similar reasoning applies to other voice properties.

While pioneer work has already established solid ground for voice synthesis using deep learning, the area of transformation has still large unexplored territories. Only a handful of topics of voice transformation are covered by neural signal processing, notably speaker identity transformation and some speaking style tokens. The present work addresses transformation of fundamental signal parameters, i.e. the pitch and intensity. We will be concerned with human voice in general but give special attention to singing voice. The treatment of speech is addressed to use synergies between speech and singing voice. Since training data for singing voice is scarce, we hope to use additional information about human voice from speech to improve the quality of our models and would obtain models that work on speech and singing as well.

## 1.2. Context

During the last three years the research around this thesis was part of three larger projects: the artistic residency by Judith Deschamps, the ANR project *ARS*, and the IRCAM Singing Synthesizer. While this thesis was originally motivated by the shortcomings of existing methods in voice processing, the surrounding projects have given this thesis a purpose and thus have influenced the direction of this research.

### 1.2.1. Artistic residency *Farinelli*

During the last two years I had the pleasure of working together closely with Judith Deschamps in the context of her artistic research residency at **IRCAM**. The subject of her artistic residency was to recreate the voice of 18th century castrato singer Carlo Maria Michelangelo Nicola Broschi, known as Farinelli. The research behind this thesis is closely tied to the Farinelli project as a lot of the research questions were focused around developing the technologies that would allow us to recreate the voice of Farinelli. The recreated voice was a central device in the documentary film *La Mue, un conte vidéographique* (Deschamps, 2022b) and was part of an installation at Casino Luxembourg in early 2023. The project will be discussed in more detail in Chapter 8.

### 1.2.2. ANR Project ARS

The *Agence nationale de la recherche* (French national research agency, ANR) provides research grants for various research topics. Just before the beginning of the thesis such a grant was acquired in the context of the project *Analyse et transformation du style de chant* (Analysis and transformation of singing style, ARS). As the name suggests, the objective of the project is to investigate singing style and find new means to analyse and transform it. The ARS project is an opportunity to work together with musicologists and use voice transformation to investigate the effect of the changed parameter on the style. Furthermore, the collaboration with the private sector has motivated the creation of a robust product with high expectations to synthesis quality. The ARS project is a collaboration between the *STMS lab*, the research institutes *Institut Jean Le Rond d'Alembert* and *Passages Arts & Littératures (XX-XXI)*, and the company *Flux Software Engineering*.

### 1.2.3. IRCAM Singing Synthesizer

The IRCAM Singing Synthesizer (ISIS) is a parametric concatenative synthesizer. To synthesize singing voice from the text and score, first the phonetic sequence is synthesized by concatenating parameter contours of the corresponding diphones using the parameters of a parametric vocoder. Rhythm and melody are imposed by time-stretch and pitch-shift using the same parametric vocoder. This thesis was in part motivated by the idea to improve IRCAM Singing Synthesizer by using the parametric space of a parametric vocoder and use the neural transformation algorithms for the time-stretch and pitch-shift operations. Furthermore, the singing voice could be customized by other voice transformations, in particular the voice level changes that were developed in this thesis. While this thesis is not directly involved in the development of the IRCAM Singing Synthesizer, it provides one very interesting application that we want to address in the future.

## 1.3. Main contributions

As scientific research is a collective effort it is difficult to disentangle individual contributions from the scientific results. Therefore, not all ideas presented here are my own ideas. Attribution is given where necessary either as citations or a mentioning of the original author of an idea. My main contributions that are presented here in the thesis are the application of the **bottleneck auto-encoder** to the fundamental frequency ([Chapter 6](#)) and the development of the voice level estimation model ([Chapter 7](#)). Furthermore, I contributed in the development of the **neural vocoder** presented in [Chapter 5](#) through fundamental groundwork, in particular the analysis-synthesis framework for glottal closure instant detection from [Chapter 4](#). I used the results of this research to write the software application **CIRCE**<sup>1</sup> which provides a graphical interface for the neural pitch and voice level transformations developed in this thesis.

---

1. <https://forum.ircam.fr/projects/detail/circe/>

### Contributions of Chapter 4: Glottal closure instant detection

The results of [Chapter 4](#) were published in Bous, Ardaillon, et al. (2020). [Chapter 4](#) builds on existing work of Luc Ardaillon and extends the glottal closure instant detection of Ardaillon and Roebel (2020). A novel analysis-synthesis framework is introduced allowing to train the glottal pulse estimator in parallel with a neural vocoder.

### Contributions of Chapter 5: Neural vocoder

A significant fraction of my research was involved in developing a neural vocoder using the mel-spectrogram as its parametric space. While I contributed to the theoretical considerations and performed my own experiments, the [Multi-Band Excited WaveNet](#) presented at the end of [Chapter 5](#) cannot be seen as an independent contribution of this thesis.

### Contributions of Chapter 6: Bottleneck auto-encoder

The results of [Chapter 6](#) were published in Bous and Roebel (2022). In that chapter we propose the first deep-learning based algorithm explicitly focusing on transformation of fundamental frequency  $f_0$  of singing and spoken voice using the explicit  $f_0$  values (previous work like (Qian, Jin, et al., 2020) use  $f_0$ -contours normalized by speaker and only focus on speech). The proposed system is able to accurately follow the desired  $f_0$  within typical ranges of human voice and even outside the range of the source voice, while surpassing the audio quality of classical vocoders. We study the proposed bottleneck auto-encoder with regard to several aspects:

1. The quality of the pitch transformation is evaluated by analysing the  $f_0$  error on the test set in [Section 6.3.1](#) and by a perceptive test which we present in [Section 6.3.2](#).
2. Since the proposed system uses an auto-encoder with information bottleneck like (Qian, Yang Zhang, Chang, X. Yang, et al., 2019), we perform a thorough study on how disentanglement depends on different hyperparameters. We will investigate the bottleneck size, the size of the neural network, dataset size and the voice type (speech vs. singing voice). The results of this analysis are given in [Section 6.3.1](#).
3. Finally, we carry out a visual analysis on the latent code of the auto-encoder where we find phonetic content to be represented as periodic discontinuous patterns by the auto-encoder.

### Contributions of Chapter 7: Voice level

The results of [Chapter 7](#) were published in Bous and Roebel (2023). In [Chapter 7](#) we use the bottleneck auto-encoder to transform the perceived voice level of singing voice recordings. None of the available singing voice recordings are annotated with

voice level. Thus, we provide a method to train a neural network to extract voice level without the need for explicitly annotated voice level. Voice level is given with regard to a fixed reference point where the signal power (or loudness) would be theoretically identical to the (perceived) voice level. The voice level estimation presented here, can be calibrated using the reference point of any recording condition present in the singing database. The contributions are the following:

1. We present an unsupervised algorithm that allows training a deep neural network to predict the voice level without the need to create annotated data.
2. We use this voice level measure to train the bottleneck auto-encoder from [Chapter 6](#) to transform the voice level in recordings of singing voice.

## 1.4. Outline

The remainder of this thesis is structured as follows: We begin this thesis with an introduction to the fundamentals related to the topics of voice transformation in [Chapter 2](#). There, we will discuss topics that are not necessarily state of the art but nevertheless important in the context of this thesis. This includes a historic overview ([Section 2.1](#)), an introduction to hearing ([Section 2.2](#)), speech production ([Section 2.3](#)), and neural networks ([Section 2.4](#)), followed by a discussion about the datasets ([Section 2.5](#)), and the evaluation methods ([Section 2.6](#)). We continue with a review of the state of the art in [Chapter 3](#) where we will discuss work closely related to pitch transformation. In particular, we will discuss voice analysis methods ([Section 3.1](#)), vocoders ([Section 3.2](#)), speech- and singing synthesis ([Section 3.3](#)), and voice transformation ([Section 3.4](#)).

The main part of this thesis consist of [Chapters 4](#) to [7](#) and includes the principal contributions of this thesis. [Chapter 4](#) describes the initial work on neural vocoding resulting in an improvement of glottal closure instant detection using a neural vocoder. In [Chapter 5](#) we will be discussing the neural vocoder for the purpose of voice transformation. In that chapter we will motivate the choices that led to the development of the neural vocoder that provides the parametric space on which we will implement the voice transformations in the subsequent chapters. [Chapter 6](#) describes the bottleneck auto-encoder, which is used for pitch- and vocal effort transformation. In this chapter we discuss the general working principle of the auto-encoder and evaluate it on pitch transformation. In [Chapter 7](#) we develop a method to estimate the voice level in singing voice from the timbre using an unsupervised estimator. The voice level measure is used to condition the bottleneck auto-encoder to allow transformation of voice level.

The last part consists of [Chapters 8](#) and [9](#) and puts the presented material into perspective. In [Chapter 8](#) we present a few applications that testify about the usefulness of the technology developed in this thesis. We will discuss the residency of Judith Deschamps and a few musical compositions. [Chapter 9](#) is a recapitulation of the thesis, puts the presented material into a larger context and discusses possible future work.

## 2. History and fundamentals

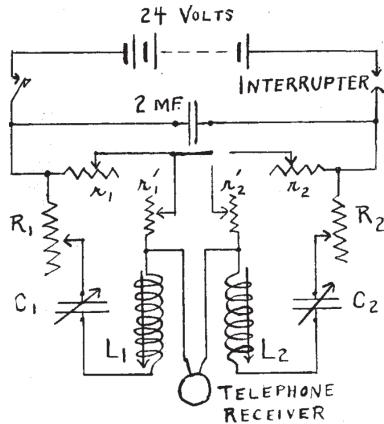
**H**uman voice has been subject of study for a long time. This thesis builds on research that has been going on for over 100 years. Throughout this long period, different technologies were utilized and had their popularity at one time or another. We see voice technology being realized through mechanics, electrical circuitry, digital signal processing and recently through neural networks. While this work heavily uses deep learning to achieve the goals we set out for us, the other technologies have not vanished from the domain of voice processing. Rather, they have become such common visitors that we grew used to their presence and do not notice them every time any more. In this chapter we introduce the main characters of this thesis. We pay tribute to the extensive work that has allowed my own work to be produced and provide a context within the research of this thesis resides.

### 2.1. A short history of voice synthesis and transformation

The first documented systematic investigation on speech production is the work by Kempelen (1791) in the late 18th century. Kempelen was an infamous inventor and engineer, at his time most known for his chess-playing machine, an apparatus that was able to play chess but which was later exposed as a hoax; the central logic unit was an actual human chess player concealed from the spectators (Linggard, 1985, pp. 4-8). While the conceptualization and construction of the chess-playing hoax took six months to finish, Kempelen's next goal was much more ambitious. For the next 20 years he would work fiercely on his speaking machine, this time for real. This invention-driven research yielded some groundbreaking insights into the speech production mechanism. Finally, Kempelen built his speaking machine using a bagpipe reed for the glottis to produce the excitation signal, a bellow for the lungs to control the strength of the sound and different resonant pipes to form different phonemes.

Kempelen's apparatus was reproduced several times during the 19th century by Farber, Willis, Wheatstone, and Scripture (Dudley, 1939). Others succeeded to imitate voice sounds by other means, like Helmholtz, using tuning forks, or Miller, using organ pipes (Stewart, 1922). One noteworthy invention was Joseph Farber's *Euphonia*, maybe the first singing synthesizer ever to be build (Blaauw, 2022, Sec. 2.1).

In the 20th century electronic devices started to replace mechanical ones, and we see voice synthesis being implemented with electronic circuits. The first such device was the speech synthesizer of Stewart (1922). This synthesizer implemented a source-filter model by filtering different source signals by analogue filters. A circuit diagram of this synthesizer is shown in Fig. 2.1. The filters were obtained by



**Figure 2.1:** Electrical circuit of the first electrical speech synthesizer. A motor-driven circuit interrupter produces a periodic excitation that is filtered by the analogue filter to produce a speech signal fed to the telephone receiver. Different vowels can be produced by adjusting the variable components to generate different resonant frequencies. [From the original publication (Stewart, 1922). Public Domain.]

experimentation and Stewart provides frequency and damping constant of the filters impulse response for six different vowels in the English language. The excitation signals are generated from a hiss for noise excitation and a motor-driven circuit interrupter for the voiced excitation. The synthesizer was able to produce singing, speech, and whispering of vowels, semi-vowels, diphthongs and fricative consonants but failed to produce plosives.

The speech synthesizer of Stewart can be seen as an electrical implementation of the mechanical speech apparatus in the sense that the mechanical components have been replaced by electrical ones. Apart from electrical filtering and pulse generation being more flexible than their mechanical counterpart, the real benefits of using electrical components became clear through the invention of the vocoder (Dudley, 1939). Stewart relied on experimental fine-tuning of the circuits in his speech synthesizer to match the filter characteristics by adjusting the variable components by hand and fixing the values when the result sounded correct. Dudley, on the other hand, automated the analysis process and obtained an automatic transcription of the synthesis parameters. This analysis-synthesis cycle was named *Vocoder* as a portmanteau of the words *voice* and *encoder*. An immediate consequence is the increased number of parameters: As the speech analysis happens automatically Dudley was able to parametrize the voice using ten frequency bands and the fundamental frequency thus yielding an 11-dimensional parametric space. Further noteworthy analogue voice synthesizers were *Voder*, a voice synthesizer based on Dudley's vocoder, *Orator Vox Electrica (OVE)*, a speech synthesizer created at the Royal Institute of Technology (KTH) in Stockholm (Fant and Martony, 1962; Liljencrants, 1968), and the *Music and Singing Synthesis Equipment (MUSSE)* singing synthesizer (also from KTH) (Larsson, 1977; Sundberg, 2006).

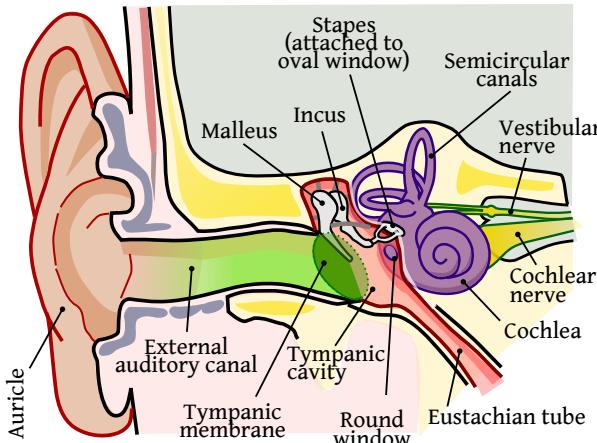
At this point the need arose to understand the acoustics of the voice to utilize the possibilities of the increasingly parameter rich synthesizers. Thus, the analysis of voice and its acoustical properties emerged as its own research topic. Among the enormous amount of publications we find a great variety of different subtopics. Vocal intensity has been studied with its relationship to various other voice properties. Ladefoged and McKinney (1963) and Isshiki (1964) provide first studies about how vocal intensity is controlled physically by the voice. Further studies have deepened

our understandings about how different voice properties depend on each other and interact (Holmberg et al., 1988; Titze and Sundberg, 1992; Sundberg et al., 1993; Henrich et al., 2005). Studies on the effect of external factors on the voice, such as distance between speakers can be found in Traunmüller and Eriksson (2000). The study of the voice production mechanism has yielded several glottal pulse models, like the Rosenberg C model (Rosenberg, 1971), the LF model (Fant, Liljencrants, et al., 1985), the KLGLOTT88 model (D. H. Klatt and L. C. Klatt, 1990) and the R++ model (Veldhuis, 1998).

With the advent of computers the vocoder could be reimplemented digitally and synthesis could be performed even more flexibly. Again, new digital synthesizers were first just an implementation of analogue synthesizers in software which came with some benefit. The big power of digital synthesis, however, became apparent through inventions like the phase vocoder (Flanagan and Golden, 1966), linear predictive coding (LPC) (Atal and Hanauer, 1971) and FM synthesis (Chowning, 1989). Computer based calculations of the parameters required by these new technologies made their application possible and had opened new possibilities for voice research. Early digital synthesizers were the work of Kelly and Lochbaum (1962), MUSSE DIG, a digital reimplementation of MUSSE in 1978 (Sundberg, 2006), and Chant (Rodet et al., 1984) developed at IRCAM. Voice transformations could be achieved thanks to pitch synchronous overlap and add (Moulines and Charpentier, 1990), harmonic plus noise model (Laroche et al., 1993; Stylianou et al., 1995) and the shape-invariant sinusoidal models including the phase vocoder (Quatieri and McAulay, 1992; Roebel, 2010).

Around the turn of the century concatenative synthesis (Moulines and Charpentier, 1990; Sagisaka et al., 1992; Hunt and Black, 1996) was a popular choice for speech synthesis. The idea of concatenative synthesis is to have a large database of linguistic units (phonemes, diphones, etc.) and concatenate the units according to the target text. The concatenation could be done on the audio signal or on the parameters of a parametric vocoder. The latter is called parametric concatenative synthesis and was particularly popular in singing synthesis (Umbert, Bonada, and Blaauw, 2013; Bonada et al., 2016; Ardaillon, 2017). An alternative was statistical parametric speech synthesis (Zen, Tokuda, et al., 2009), where a statistical model, like hidden Markov models (HMMs) (Yoshimura, 2002), is used to generate vocoder parameters from the target text. These methods could produce voice with high flexibility and reasonable quality thanks to the development of high quality parametric vocoders (Kawahara, 2006; Morise et al., 2016; Huber and Roebel, 2015).

Since the last decade, we can observe a new technology emerging within the voice research community, the technology of deep learning. Like in other domains, deep learning is in the process of revolutionizing the field. Early work using deep learning applied to voice can be found in statistical parametric speech synthesis, where neural networks were used as the statistical models (Zen, Senior, et al., 2013; Zen, Agiomyrgiannakis, et al., 2016) while the actual audio synthesis was still performed by a parametric synthesizer. In recent years, however, thanks to deep learning we have seen many breakthroughs that would have not been possible using previous approaches. Nowadays, the best speech synthesis (Donahue et al., 2021; Ren et al.,



**Figure 2.2:** Cross-section of the auditory system. Sound enters through the ear, travels through the auditory canal and excites the tympanic membrane. The vibrations are transmitted via the malleus, incus, and stapes to the cochlea where the sound is split into its frequency components. [Adaption from Wikimedia Commons *Anatomy\_of\_the\_Human\_Ear.svg*. Creative Commons Attribution 2.5 Generic Licence.]

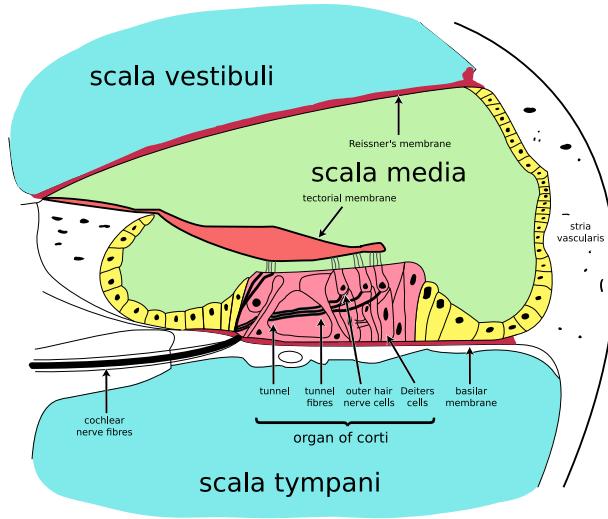
2021; J. Kim et al., 2021; C. Wang et al., 2023; K. Shen et al., 2023), singing synthesis (J. Wu and Luan, 2020; P. Lu et al., 2020) and speaker identity transformation (J.-X. Zhang et al., 2019) and other applications are achieved by neural networks. We see the same pattern unfolding as we have before during the introduction of electrical circuitry and later with digital signal processing: a new technology emerges which causes us to rethink the way we approach the problem. Great inventions are being discovered continuously with more foreseeable in the future. While we have arrived at the present in our narrative of historic overview, this is far from the end of the story of voice research.

## 2.2. Human perception of sound

In this thesis we aim to synthesize audio signals that *sound like* human voice. The verb ‘to sound like’ is a subjective action. We need to emphasize at this point that we are not trying to recreate human voice exactly but rather to create signals that are indistinguishable from real voice to human listeners. This slight difference is often neglected but gives us leeway to reduce complexity and focus our resources on relevant acoustic features. In order to produce the necessary features in the synthetic voice signals we first need to study how humans perceive voice.

Sound is perceived in the auditory system (Moore, 2012, Ch. 1.6), which is depicted in Fig. 2.2. Incoming sound is caught by the auricle, or pinna, and channelled into the auditory canal which due to its decreasing diameter amplifies the sound pressure. At the end of the auditory canal is the tympanic membrane. The variations in sound pressure are picked up by the tympanic membrane and transmitted via the hearing bones, *malleus*, *incus* and *stapes*, to the *inner ear* (depicted in purple in Fig. 2.2).

In the inner ear, the sound is perceived in the cochlea (the other part, the semi-circular canals, are responsible for our sense of balance). The cochlea is a lengthy cavity surrounded by bony structure. It is about 30 mm long and wound like a snail turning 2.75 times. The interior of the cochlea is filled by almost incompressible



**Figure 2.3:** Cross-section of the cochlea. Vibration of the **basilar membrane** is what we perceive as sound. [Adaption from Wikimedia Commons [Cochlea-crosssection.svg](#). Creative Commons Attribution-Share Alike 3.0 Unported Licence.]

fluids. At its base are two membrane-covered openings, the *oval window* and the *round window*. The oval window connects the interior of the cochlea to the stapes and allows sound to enter the cochlea. Since the cochlea is filled with incompressible fluid, the round window exists to allow the displaced volume to be absorbed by connecting the interior of the cochlea to the tympanic cavity. The interior of the cochlea is split lengthwise by two membranes, *Reissner's membrane* and the **basilar membrane**. A cross-section of the cochlea can be seen in Fig. 2.3.

The **basilar membrane** is where the vibrations of the incoming sound are detected. The mechanical properties of the **basilar membrane** change along the length of the cochlea, making it react differently to each frequency: Incoming sinusoids cause the **basilar membrane** to vibrate. The amplitude of the vibration is different along the **basilar membrane** with its maximum somewhere along the length of the cochlea. The exact position of the maximum depends on the frequency of the sinusoid; the part closer to the oval window (the base) reacts more sensitive to high frequencies, while the tip (the apex) reacts more sensitive to low frequencies (Ruggero, 1992). Each point along the length of the **basilar membrane** can be thought of as a band pass filter. Measurements have shown, that the bandwidth of these band pass filters are about 1/8 to 1/2 of an octave (Moore, 2012, Ch. 1.6). Thus, the **basilar membrane** decomposes the frequency spectrum over space. Finally, the vibrations on the **basilar membrane** are detected along its length, translated into nervous impulses and sent to the brain.

There is much more to the hearing mechanism which is beyond the scope of this thesis. A very interesting and detailed description can be found in Moore (2012). The important take-away for us is, that sound is decomposed into its frequency components before it enters the brain. Hearing is the perception of frequencies, not air pressure itself.

### 2.2.1. Perception of pitch

Pitch and frequency are related but different concepts. Frequency is a physical quantity and describes how often a periodic process repeats. Pitch is a psychoacoustical property related to the frequency of sinus tones and can be extended to more complex sounds that evoke the same perception of pitch (Moore, 2012, Ch. 6.1). That is, in the world of pure sine tones, pitch and frequency are the same scales. Further stimuli can evoke the perception of the same pitch as a sine tone, though their harmonic structure may be very different.

While the phenomenon of pitch is well known to everyone with functioning ears, how pitch is perceived is still a matter of debate (Chevigné, 2010). While several theories for the perception of pitch exist, ‘physiological and psychophysical investigations have failed to rule decisively in favor of either hypothesis [...]’ (Chevigné, 2010, Sec. 4.12). Nevertheless, most naïve models of pitch perception can be ruled out through simple counter examples. Camacho (2007, Sec. 1.2) provides illustrative counter examples, which I will summarize here:

**Pitch is not the lowest partial** Generally, the pitch remains the same when adding or removing harmonics. This is even true for the fundamental frequency. Removing all the energy from the fundamental frequency does not necessarily change the pitch.

**Pitch is not the distance between partials** Camacho (2007) shows the example of a square wave, which has only odd harmonics. Thus, the distance between the partials is  $2f_0$ , but the pitch is perceived as  $f_0$ .

**Pitch is not the fundamental frequency** Adding subharmonics does not change the perceived pitch if the energy from the subharmonics is low compared to the remaining signal. This effect can be observed in rough voice where subharmonics might be introduced through the glottal source.

**Pitch can be evoked by inharmonic signals** As shown by Patel and Balaban (2001), even inharmonic signals can evoke the perception of pitch. Stimuli consisting of very low periodicity might evoke a perceived pitch of much higher frequency. For example, Patel and Balaban (2001) show that the combination of sinusoids of 650 Hz, 950 Hz and 1250 Hz evoke a pitch of 315 Hz or 650 Hz.

**Pitch can be evoked by binaural effects** A particular curiosity is the phenomenon of Huggins pitch (Cramer and Huggins, 1958; Culling et al., 1998). This stimulus is created by an all pass applied to white noise. Then the original and the filtered signal are played simultaneously in the left ear and the right ear respectively. Though each of the channels, the filtered and the unfiltered component individually are perceived only as white noise, listening to them at the same time creates the perception of a pitch at the frequency, where the phase changes (Chevigné, 2010, Sec. 4.8).

**Pitch is not a linear scale** Though generally we distinguish between low and high pitches, there is no transitivity on the perceived pitches, as can be seen

through Shepard scales (Shepard, 1964). In a rising Shepard scale, the next stimulus appears to have a higher pitch than the previous, even though the whole scale is periodic resulting in a circular scale of rising steps, similar to the Penrose steps.

The observations above reveal that pitch cannot be easily obtained from frequency. Though, in general, counter examples exist, in voice, the pitch usually corresponds to the fundamental frequency. Thus, in voice, we can measure pitch through the fundamental frequency  $f_0$ , by taking a proper pitch scale into account. In the following we will use the terms ‘pitch’ and ‘fundamental frequency’ interchangeably.

### 2.2.2. Pitch scales

Different approaches to quantify pitch exist. Pitch can be quantified as the frequency of the corresponding sinusoidal stimulus. However, since pitch is a perceptive quantity, a proper pitch scale should reflect the perceptive component quantitatively as well. Two approaches exist: either through perceived sizes of intervals, leading to the mel-scale, or through the frequency resolution of the basilar membrane, leading to the ERB-scale.

#### The mel-scale

Observations of non-uniform perception of pitch intervals versus frequency ratios have lead Stevens et al. (1937) to the development of the mel-scale, which maps frequency to a perceptive pitch value in mel. This scale is quantized by assigning 1000 Hz = 1000 mel and fitting the observations from the perceptive measurements. Stevens et al. (1937) asked participants to set specific pitch ratios, e.g. a pitch of 2000 mel would be twice as high as a pitch of 1000 mel. Later formulas have been introduced to fit the measured data and typically the formula

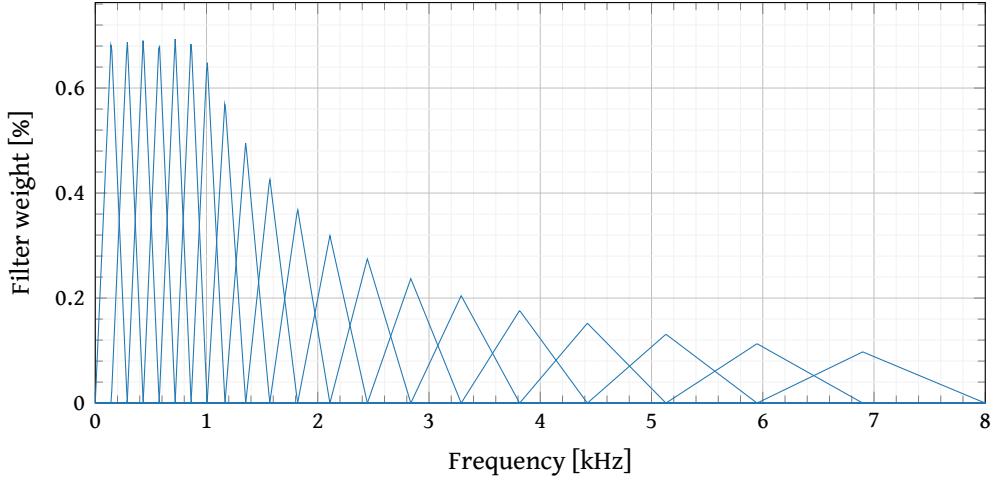
$$m = m_0 \log \left( 1 + \frac{f}{f_m} \right) \quad (2.1)$$

is used to map between values  $m$  on the mel-scale and the frequencies  $f$  in Hz. The cut-off frequency  $f_m$  can be seen as a soft transition where the scale switches from linear to logarithmic. Today normally we use  $f_m = 700$  Hz, while  $f_m = 1000$  Hz is also common in particular in older literature.

The mel-scale has found its application through the *mel-filtered cepstral coefficients (MFCCs)* in speech recognition (Rabiner and Juang, 1993) and is currently very prominent through the *mel-spectrogram* in deep-learning based speech processing (J. Shen et al., 2018).

#### The mel-spectrogram

The mel-spectrogram is a spectrogram with the frequency axis compressed to the mel-scale. To calculate *mel-spectrograms* we need to remap the frequency axis to the mel-scale. If we were dealing with continuous-time signals we could simply



**Figure 2.4.:** Mel filter bank for a configuration with 20 mel-bands. In this thesis we use 80 mel-bands, but this would make the graph unreadable.

apply Eq. (2.1) to the frequency axis. We are, however, dealing with discrete-time signals. That means the spectrograms have discrete frequency bands. Similarly, the mel-spectrogram has discrete mel-bands. Here is really where the strength of the mel-spectrogram comes into play as using the mel-spectrogram allows us to reduce the number of bands dramatically. In a typical application the mel-spectrogram is used to compress 1025 frequency bands to 80 mel-bands. Thus, one mel-band represents multiple frequency bands, especially for higher frequencies. To reflect the information from all the frequency bands it does not suffice to interpolate the frequency axis at the centre frequencies of the mel-bands. Since each mel-band represents multiple short time Fourier transform (STFT) bands, we average over the bandwidth of each mel-band with triangular weights. This creates the mel-filter bank; an example is shown in Fig. 2.4. Since each filter is only used around one centre-frequency we can write all the filters into a matrix and shift them to the position of the mel-band. Thus mapping from a spectrogram  $S$  to its mel-spectrogram  $M$  can be achieved by the linear operator  $B$  as

$$M_{\text{lin}} = B |S| . \quad (2.2)$$

In practice often we use the log-amplitude mel-spectrograms in our processing pipeline rather than the linear-amplitude mel-spectrogram which changes Eq. (2.2) to

$$M_{\log} = \log(B |S|) . \quad (2.3)$$

Since  $\log / \exp$  are bijective these representations are equivalent, and therefore we will use the term *mel-spectrogram* to refer to either of these representations.

### The equivalent rectangular bandwidth filter bank

An alternative to the mel-scale are equivalent rectangular bandwidth (ERB) filter banks. The ERB scale is based on the study of the filter characteristics of the basilar membrane. Every point along the basilar membrane acts as a band pass filter with a different centre frequency and a different bandwidth. The bandwidth can be given as a function of centre frequency which is approximated by Glasberg and Moore (1990) as

$$w_{\text{ERB}} = \alpha (1 + \beta f_c), \quad (2.4)$$

where  $f_c$  is the centre frequency,  $w_{\text{ERB}}$  is the equivalent rectangular bandwidth, and  $\alpha = 24.7 \text{ Hz}$  and  $\beta = 4.37 \text{ ms}$  are heuristic constants. Equation (2.4) can be used to create the ERB-scale  $s_{\text{ERB}}$  through

$$s_{\text{ERB}} = \int_0^f \frac{1}{w_{\text{ERB}}(f_c)} df_c \quad (2.5)$$

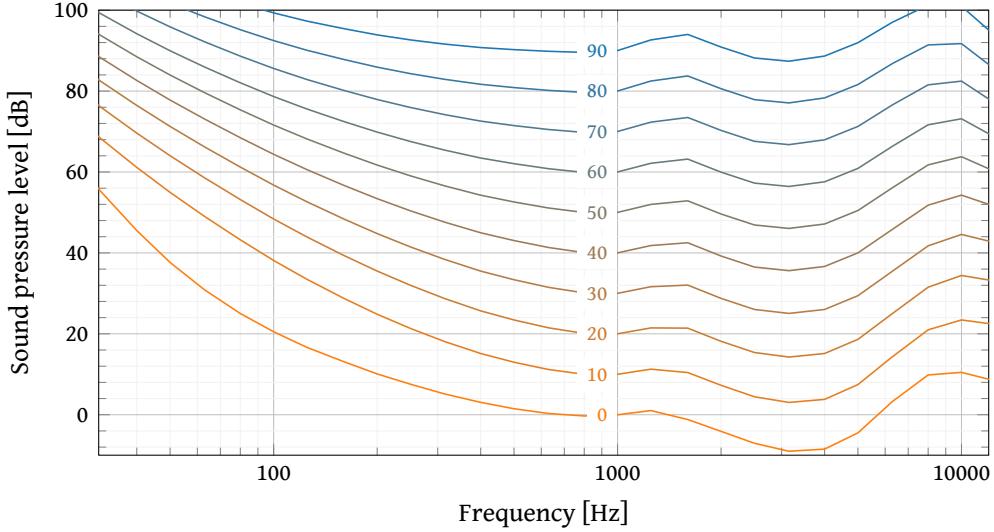
$$= \frac{1}{\alpha\beta} \log(1 + \beta f) \quad (2.6)$$

$$\approx 9.265 \log\left(1 + \frac{f}{229 \text{ Hz}}\right). \quad (2.7)$$

Thus, the ERB scale resembles the mel-scale in structure. Note that on the ERB scale the transition from linear to exponential is much lower (230 Hz instead of 700 Hz). The other significant difference is that ERB bands have exactly width 1 on the ERB scale and are constant height rather than triangles. The width of the mel-bands depends on the number of mel-bands and at any frequency exactly two mel-bands overlap. The width of the ERB-bands is always the same and how many bands overlap depends on the overlap value  $r$ . Given an overlap value  $r$ , an ERB filter bank can be constructed by taking intervals of length 1 with step size  $1 - r$  on the ERB scale and mapping the boundaries back to the frequency scale in Hz.

#### 2.2.3. Loudness

The amplification through the middle ear does not occur with flat transfer function (Glasberg and Moore, 2002) and the basilar membrane reacts different to each frequency (Moore, 2012, Ch. 1.6). As a consequence our hearing has different sensitivity for different frequencies. First measurements were done by Fletcher and Munson (1933) where the authors measured how loud people perceive sounds of different frequencies. For this, they introduced the phon-scale. For 1 kHz sinusoids the phon-scale corresponds to the sound pressure level in dB. A sinusoid at a different frequency gets assigned the loudness level of a 1 kHz sinusoid that is perceived as equally loud. Using the phon-scale we can obtain equal-loudness contours. An example can be seen in Fig. 2.5. From these curves we can visualize how our hearing is dependent on the frequency, especially for lower levels. Later the equal-loudness contours were revised by D. W. Robinson and Dadson (1956) to create the initial ISO 226 standard. The standard was revised in 2003.



**Figure 2.5.: Equal loudness contours from ISO 226: 2003 standard for phon values 0, 10, …, 90.** The 0 phon value is the minimal audible threshold. Hearing is most sensitive between 1 kHz to 5 kHz.

These equal-loudness contours are obtained using sinusoids at different frequencies. However, a more complex frequency spectrum and temporal changes have an effect on the perceived loudness. These issues are addressed by Glasberg and Moore (2002) who provide a loudness model for non-stationary sounds.

The phon-scale is measured in units of dB. However, observations of Stevens et al. (1937) have shown that a sound is perceived twice as loud with each increase by 10 dB. Thus, Stevens et al. (1937) introduced the sone scale to capture the perceived loudness. We assign the value of 1 sone to the phon value of 40 dB and doubles with each increase by 10 dB. Thus, the loudness in sone can be obtained through the formula

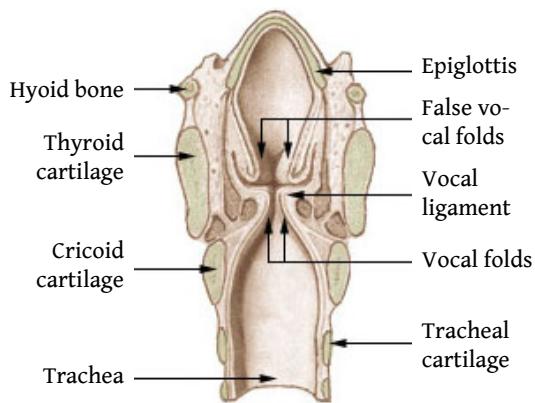
$$L_s = 2^{\frac{L_p - 40 \text{ dB}}{10 \text{ dB}}} \text{ sone} \quad (2.8)$$

$$= (10^{-4} L_e)^{0.30103} \text{ sone}, \quad (2.9)$$

where  $L_s$  is the loudness in sone and  $L_p$  is the loudness in phon and  $L_e$  is the signal power. Thus, the perceived loudness increases with the power of 0.3 of the signal's power.

### 2.3. Fundamentals of the voice

To transform human voice we need to understand how human voice is produced. In this section we will have a look at speech production. We will introduce the source filter model and the glottal pulse model of Fant, Liljencrants, et al. (1985)



**Figure 2.6:** Cross-section of the larynx, the main speech production organ. Located in the neck, the bottom part connects through the trachea to the lungs. The other end is connected to the nasal and oral cavities. [Adaption from Wikimedia Commons Illu\_larynx.jpg. Public Domain]

which provides us with various parameters of the voice that we can use to describe different aspects of it.

### 2.3.1. The theory of speech production

During the production of human voice, three principal components influence its acoustic appearance: the glottal source, the noise component and the vocal tract filter. Voiced speech is produced in the glottis, which generates periodic or sometimes irregular pulses. Additionally, noise originates in various places of the vocal tract. Finally, the glottal pulse and the noise is shaped acoustically through the vocal tract.

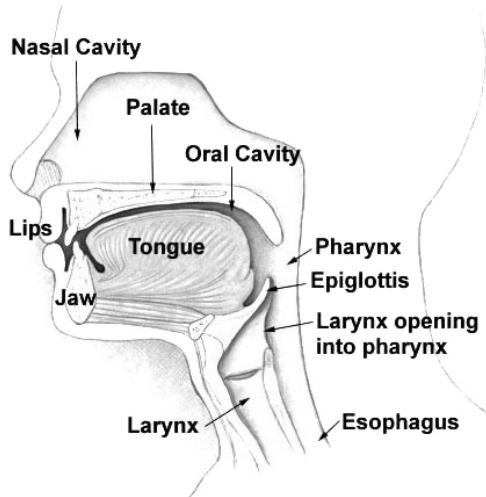
#### Phonation and the glottal source

The act of producing voiced speech is called phonation. Voiced speech originates in the glottis and is formed in the nasal and oral cavities. An illustration of the Larynx can be found in Fig. 2.6. During phonation the vocal folds are tensed such that they are pushed together and close the larynx. At the same time the lungs build up air pressure that pushes the vocal folds aside. Once the air pressure is high enough, the vocal folds open and air bursts out. As the air leaks through the now opened vocal folds, the air pressure pushing the vocal folds apart drops, allowing the vocal folds to close again, and the process repeats anew. This happens periodically and creates a pulse signal, called *glottal pulse signal*.

#### Noise component

In phonated voice the glottal pulse is the dominant sound source, but it is not the only source. Turbulences in the air flow through the vocal tract create noise which can be most prominently heard in unvoiced fricatives like [f], [θ], [s] or [ʃ], but is in fact an audible component in all other modes of speech, including vowels.

We can divide the noise component in speech into two categories (Richard and Alessandro, 1996):



**Figure 2.7:** Cross-section of the vocal tract. Glottal pulses originate in the larynx and are filtered by travelling through the nasal and oral cavities. The filter is highly dependent on the position of the tongue. [Adaption from Wikimedia Commons *Il lu01\_head\_neck.jpg*. Public Domain]

**Additive noise** which is generated independently of the glottal source and can be assumed to be superimposed additively.

**Structural noise** which originates from variations in the glottal pulse.

The additive noise further be subdivided into *transient noise*, *quasi-stationary noise* and *modulated noise*. Transient noise occurs through plosive consonants like [ p ], [ t ] or [ k ] or through mouth noise, lip noise and clicks. Quasi-stationary noise is generated through turbulent airflow in the vocal tract or by the glottis if the vocal folds remain open. It is prominently dominant in unvoiced fricatives. Quasi-stationary noise is used as the excitation signal in whispering. Modulated noise is generated from turbulent airflow which is modulated by the vocal fold vibration and occurs most notably in breathy vowels and voiced fricatives. The structural noise can be subdivided into *jitter* (random variation in the pulse positions), and *shimmer* (random variation in the pulse amplitudes) (Richard and Alessandro, 1996).

### Vocal tract filter and spectral envelopes

After the glottal pulse signal originates in the glottis, it has to leave the body to be heard by the listener. This happens mainly through the oral and nasal cavities (see Fig. 2.7) though a small fraction of the sound may also be transported through the body itself. Acoustically, the oral and nasal cavities can be thought of as long tubes that reflect and absorb the sound waves. As a result they don't transmit all frequencies equally well and thus act as filters on the glottal pulse signal. The superposition of the different paths that the signal can take creates the *vocal tract filter* (VTF). This filter depends very much on the geometry of the oral cavity which in turn depends on the position of the tongue and the opening of the mouth. This allows the speaker to create different sounds from the same glottal source signal such as vowels and voiced consonants.

Similarly, the noise component of the voice is filtered through the vocal tract. Since parts of the noise can be created in the vocal tract after the glottis, the mech-

anism is slightly different. Depending on where the noise component originates only a fraction of the vocal tract filter acts on the noise component.

The combination of these different effects creates a filter that is characterized by resonances and zeros. The resonances are called *formants*, and it was shown (e.g. by Stewart (1922)) that the formants are the key acoustic feature that allows us to distinguish different vowels.

### 2.3.2. Glottal flow models

The glottal pulse signal has been studied extensively in the last 50 years and several glottal flow models have emerged. Common glottal pulse models are the Rosenberg C model (Rosenberg, 1971), the Liljencrants-Fant (LF) model (Fant, Liljencrants, et al., 1985), the KLGLOTT88 model (D. H. Klatt and L. C. Klatt, 1990) and the R++ model (Veldhuis, 1998). A discussion on them can be found in Doval et al. (2006). Of particular interest for this work is the Liljencrants-Fant (LF) model (Fant, Liljencrants, et al., 1985) as we will be using it in Chapters 4 and 5. Here it shall serve as an example how the glottal pulse can be parametrized, allowing us to introduce various pulse qualities like the *open quotient (OQ)* or the *glottal closure instant (GCI)*.

The classical Liljencrants-Fant (LF) model of Fant, Liljencrants, et al. (1985) describes the glottal pulse by three parameters (four if you count the amplitude  $E_e$ , or five if you also count the period  $t_0$ ). A simplified model has later been given by Fant, 1995 using only one parameter. The glottal pulse of the LF model alternates between two phases, the open- and the closing phase. A visualization can be found in Fig. 2.8. The three LF pulse parameters are:

$t_p$  the time when the maximal flow occurs,

$t_e$  the time when the vocal folds close,

$t_a$  a measure of how long it takes for the vocal folds to close, the *return phase*. As the glottal flow in the LF model is never truly zero, we define the closing duration as the time between the closure instant  $t_e$  and the time  $t'_a$  where the tangential at  $t_e$  (red dashed line in Fig. 2.8) crosses the time-axis.

The pulse parameters follow the restriction

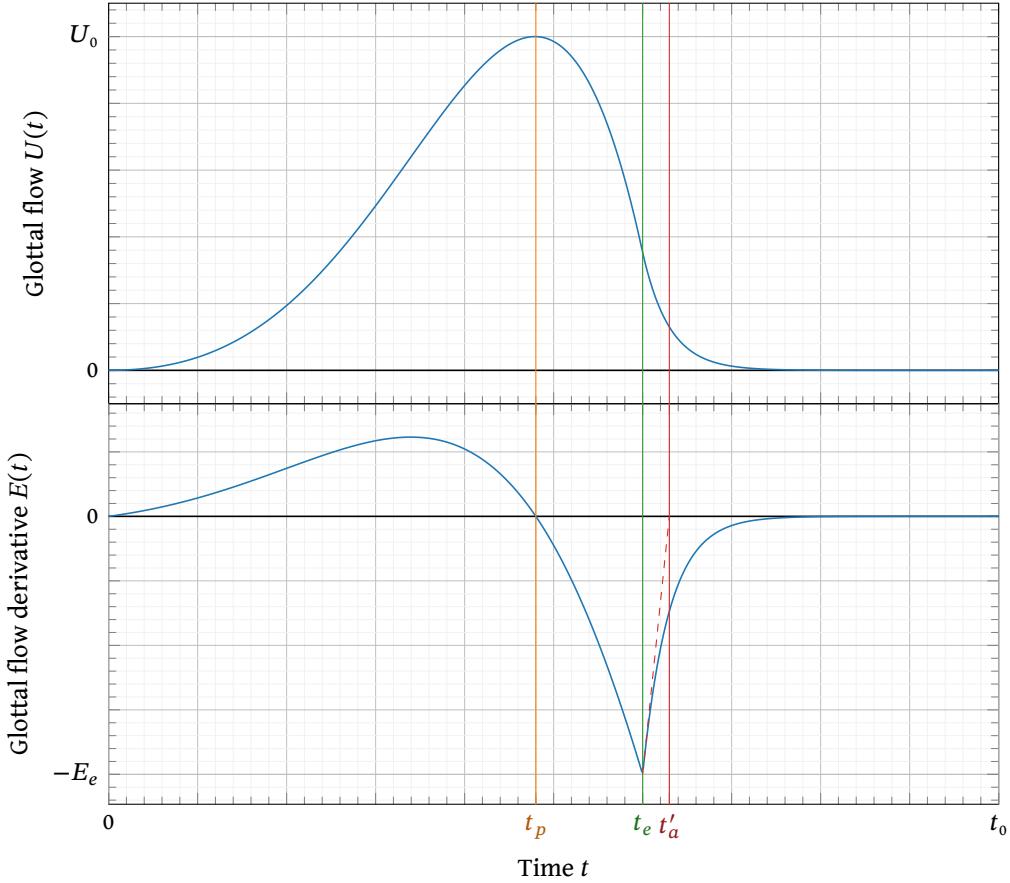
$$0 < \frac{t_e}{2} \leq t_p < t_e \leq t_e + t_a \leq t_0. \quad (2.10)$$

Furthermore, we identify the parameters

$t_0$  the period of the pulse,

$U_0$  the amplitude of the glottal flow, and

$E_e$  the amplitude of the glottal flow derivative.



**Figure 2.8.**: Glottal flow and glottal flow derivative of the *Liljencrants-Fant* glottal pulse model, annotated with the defining parameters  $t_p$ ,  $t_e$ , and  $t'_a = t_e + t_a$ . The curves are normalized in time by the period and in value by their maxima.

The LF model defines the glottal flow derivative  $E : [0, t_0) \rightarrow \mathbb{R}$  using two separate cases for open- and closing phase:

$$E(t) = \begin{cases} E_o(t) & t \leq t_e \\ E_c(t) & t > t_e \end{cases} \quad (2.11)$$

with

$$E_o(t) = E_0 e^{\alpha t} \sin \frac{\pi t}{t_p} \quad (2.12)$$

during the open phase ( $t \leq t_e$ ) and

$$E_c(t) = -E_1 (e^{-\varepsilon(t-t_e)} - e^{-\varepsilon(1-t_e)}) \quad (2.13)$$

during the closing phase ( $t > t_e$ ). Equations (2.12) and (2.13) define the pulse only implicitly. Furthermore, we require the following two conditions:

$$E_o(t_e) = E_c(t_e) \quad (2.14)$$

to ensure  $E$  is continuous (except for the special case where  $t_a = 0$ , in which case  $E_c(t) = 0$  for all  $t$ ), and

$$\int_0^{t_0} E(t) dt = 0 \quad (2.15)$$

to ensure  $E$  is a derivative of a periodic function. There is no closed form solution for  $E_0$ ,  $E_1$ ,  $\alpha$  and  $\varepsilon$  but an algorithm to obtain the necessary parameters numerically can be found in Fant, Liljencrants, et al. (1985).

Equivalently the glottal pulse model can be described by

$O_q$  (*open quotient*) The ratio of duration of the open phase and the whole pulse duration.

$$O_q = \frac{t_e}{t_0} \quad (2.16)$$

$\alpha_m$  (*open phase skewness*) The ratio between  $t_p$  and  $t_e$ .

$$\alpha_m = \frac{t_p}{t_e} \quad (2.17)$$

$f_a$  (*glottal formant*) The frequency where the spectral tilt changes from 20 dB per decade to 40 dB per decade.

$$f_a = \frac{1}{2\pi t_a} \quad (2.18)$$

or by the ratios

$$R_a = \frac{t_a}{t_0} \quad (2.19)$$

$$R_g = \frac{t_0}{2t_p} \quad (2.20)$$

$$R_k = \frac{t_e - t_p}{t_p} \quad (2.21)$$

### The simplified LF model

The full LF model allows a flexible parametrization of the glottal pulse but in fact these parameters are not independent. Thus, not all combinations of pulse parameters make sense even if they stay within their mathematically permitted bounds. Furthermore, estimation of the pulse parameters from voice recordings is unreliable as the glottal pulse cannot be observed directly. Thus, Fant (1995) provides a heuristic that defines a simplified LF model in which the relevant parameters are derived from one meta-parameter  $R_d$ . The  $R_d$ -parameter is defined as

$$R_d := \frac{f_0 U_0}{110 E_e}, \quad (2.22)$$

where  $U_0$  is the amplitude of the glottal flow and  $E_e$  is the amplitude of the glottal flow derivative.

Using the statistical relations from Fant and Kruckenberg (1994) the other ratios can be predicted as

$$R_a = \frac{-1 + 4.8R_d}{100} \quad (2.23)$$

$$R_k = \frac{22.4 + 11.8R_d}{100} \quad (2.24)$$

$$R_g = \frac{0.125R_k + 0.3R_k^2}{0.11R_d - 0.5R_a - 1.2R_aR_k}. \quad (2.25)$$

The numerical range for  $R_d$ , is provided empirically by Fant (1995) as  $R_d \in [0.3, 2.7]$ .

Different pulses from the simplified LF pulse model are visualized in Fig. 2.9. For high  $R_d$  values the pulse becomes more and more smooth, and for the highest value, 2.7, the pulse closely resembles a sine curve except for a small dent towards the end. In the spectrum we can see that with higher  $R_d$ , the energy moves more and more towards the lower Fourier coefficients. For the highest  $R_d$  value most energy is located in the fundamental frequency and already the second partial (coefficient index 2) is attenuated by 20 dB. Conversely, with lower  $R_d$  value, the ‘dent’ becomes sharper and turns into a sharp peak for the lower  $R_d$  values. This peak creates a rich overtone spectrum and moves the energy more and more to the higher Fourier coefficients so that the higher partials actually carry more energy than the fundamental frequency.

## 2.4. Neural networks

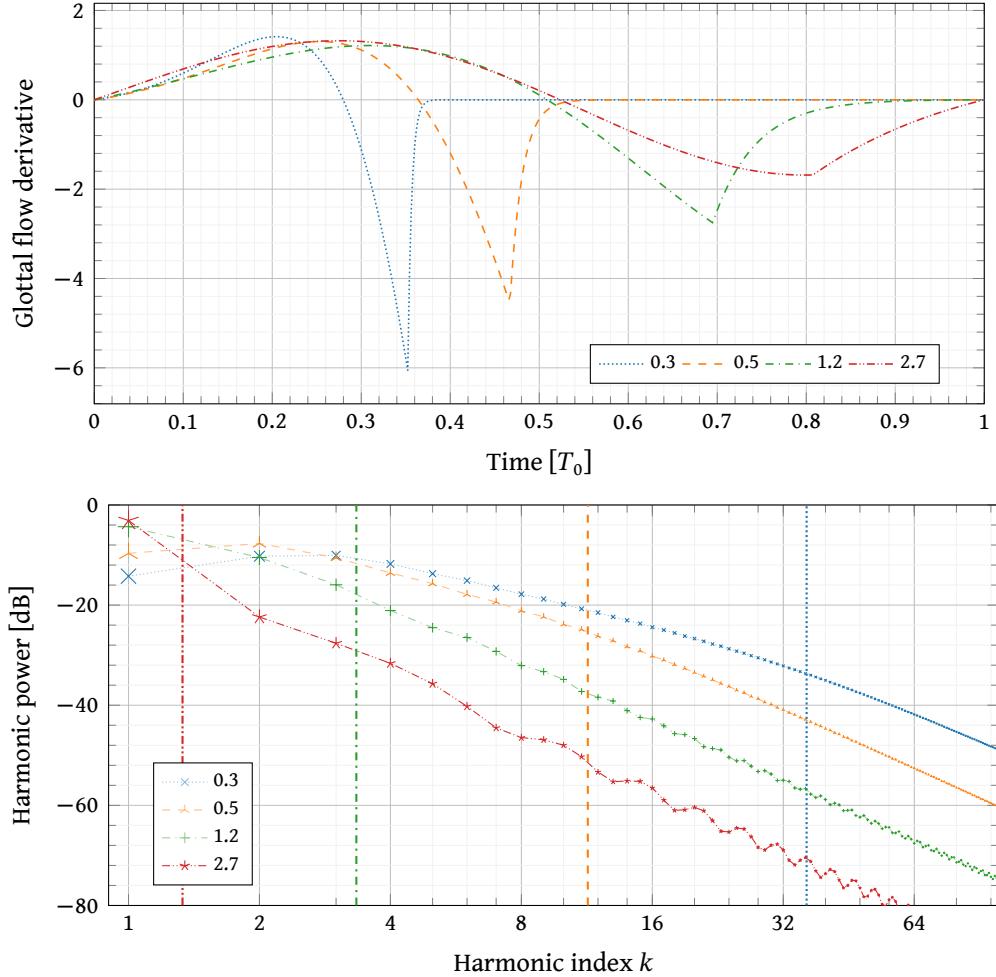
The main tool for voice treatment in this thesis are neural networks. Although we’re not studying neural networks in this work explicitly, it is an important tool for modelling the voice, so a deep understanding of the underlying principles is essential. Neural networks are universal approximators (Hornik et al., 1989), which means that any<sup>1</sup> mathematical function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  can be approximated arbitrarily well using a sufficiently complex neural network  $\hat{f}_\theta$ . The parameters of a neural network are called *weights* and are typically denoted by  $\theta$ .

Neural networks can have very complex structures (called *architecture*) but generally a neural network is characterized through alternating applications of affine operations  $\Phi_{\theta_k}^k : \mathbb{R}^{n_k} \rightarrow \mathbb{R}^{n_{k+1}}$  and non-linear scalar functions  $a : \mathbb{R} \rightarrow \mathbb{R}$  applied to each dimension individually ( $a(\mathbf{x}) := (a(x_1), \dots, a(x_n))$ ). In its most basic form the neural network function can be written as

$$\hat{f}_{\theta_1, \dots, \theta_n} = \Phi_{\theta_n}^n \circ a \circ \Phi_{\theta_{n-1}}^{n-1} \circ \dots \circ a \circ \Phi_{\theta_1}^1, \quad (2.26)$$

---

1. The original paper of Hornik et al. (1989) proves the approximation theorem for any Borel-measurable function between finite dimensional spaces. The claim has since been extended to more functional spaces. In practice this means we can approximate any function that is not specifically designed as a counter example.



**Figure 2.9.: Liljencrants-Fant glottal flow for different  $R_d$  values.** Top: flow derivative. Bottom: Fourier coefficients and glottal formants. The pulses are normalized to constant energy. For low  $R_d$  values, the glottal closure instant increasingly becomes a sharp peak. The sharp peaks create a rich overtone spectrum which we can see in lower graph. For high  $R_d$  values the energy moves more towards the lower harmonics. In the extreme case of  $R_d = 2.7$ , already the second partial (coefficient index 2) is attenuated by 20 dB and apart from a small dent at the end, the pulse shape looks much like a sinusoid. Since for all  $R_d$  values the glottal flow is continuous but not differentiable, eventually all Fourier coefficients decay with 40 dB per decade. The transition from 20 dB decay to 40 dB decay is called glottal formant (marked by the vertical lines) and occurs at higher frequency for lower  $R_d$  values.

where  $\circ$  denotes functional composition ( $[f \circ g](x) = f(g(x))$ ). An iteration of  $a \circ \Phi^k$  is called a *layer*. The linear components of the affine operations  $\Phi$  do not necessarily need to be dense matrices. For example, convolutional neural networks (LeCun et al., 1989) use convolutions instead of matrix multiplications which yields much fewer parameters. Likewise, other operations may be interwoven between layers such as combinations of previous layers (called skip connections (Long et al., 2015; Ronneberger et al., 2015)) or DSP operations (called differentiable digital signal processing (DDSP) (Engel et al., 2019))

Normally the target function  $f$  is not available explicitly, which is why we want to approximate it by a neural network. Instead, the function  $f$  is described implicitly thorough samples

$$D = \{(x, y) : y = f(x), x \in X\} \quad (2.27)$$

at specific points  $X$ . The set of known points  $D$  is called dataset. Typically, we divide the dataset into two subsets:

$$D_{\text{train}} = \{(x, y) : y = f(x), x \in X_{\text{train}} \subseteq X\}, \quad (2.28)$$

the training set, and

$$D_{\text{test}} = \{(x, y) : y = f(x), x \in X \setminus X_{\text{train}}\}, \quad (2.29)$$

the test set.

We optimize the weights  $\theta$  such that the approximation error  $\mathcal{L}$  is minimal on the training set:

$$\mathcal{L}(\theta) = \sum_{(x,y) \in X_{\text{train}}} d(y, \hat{f}_\theta(x)), \quad (2.30)$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta), \quad (2.31)$$

for some appropriate function  $d$  that quantifies how well  $x$  matches<sup>2</sup>  $\hat{f}_\theta(x)$ . This optimization is called *training*. To estimate how well the trained network  $\hat{f}_\theta$  approximates the true function  $f$  we can evaluate its ability to generalize on the test set.

To train the neural network, a convenient algorithm exists, the *backpropagation* algorithm (Rumelhart et al., 1986), which is based on gradient descent. Backpropagation allows efficient calculation of the gradient for all the weights by propagating the error through the neural network in reverse order.

While neural networks are a simple and powerful tool, their success depends on the proper application. In many cases the dataset  $D$  can be easily compiled by collecting the sufficiently many data pairs  $(x, y)$ . From the machine learning standpoint, these cases are considered solved and are hardly interesting research questions. Many

---

<sup>2</sup>. We can most intuitively think of  $d$  as a ‘distance’ function though in strict mathematical terms not all requirements for a distance in the mathematical sense are necessary. The Kullback-Leibler divergence, for instance, is not a distance due to its asymmetry but is a very common choice for  $d$ .

applications of neural networks, however, are not as straightforward. Often the relationship we want to model cannot be easily expressed with input-output pairs required to create the training data  $D$ .

Consider, for example, the task of transposing voice. To train a neural network transpose directly as discussed above, we would need to compile a dataset  $D$  containing pairs of original and transposed voice. Firstly, the amount of data required to sufficiently densely sample this mapping is unpractical. Creating a dataset that represent voice well, already requires large amounts of data. For the task of transposition, however, we would need a representative amount of pairs for every transposition value. Even assuming that we would have enough time to compile such a dataset, the attempt would still be futile, since proper pairs of original and transposed voice do not exist: To transpose voice, different equally likely solutions exist. Nevertheless, as we will see in Chapter 6, we are able to use neural networks to do transpositions. The trick is, not to model the transposition directly, but to train multiple neural networks on more efficient sub-tasks. Finding the right relationships to model with neural networks is what makes the field of machine learning challenging and interesting.

#### 2.4.1. Neural arithmetic on voice signals

Voice signals, as a sub-category of audio signals, are functions that map  $\mathbb{Z} \rightarrow \mathbb{R}$ , though we usually only look at a finite length section of the signal.<sup>3</sup> We assume shift invariance for the underlying distributions in the sense that it does not matter at which time we look at the signal. Audio recordings are almost never of fixed length and the network architecture should be able to process audio recordings of arbitrary length.<sup>4</sup> The prior knowledge about shift invariance can be hard-wired into the neural network architecture by convolutional layers (LeCun et al., 1989), recurrent layers (Werbos, 1990; R. J. Williams and Zipser, 1995; Sutskever et al., 2014) or transformers (Vaswani et al., 2017), together with pooling and resampling. We will very rarely see fully connected layers that stretch out across the whole time axis of the signal.

Inside the neural network the voice signal can be represented in different ways. The most obvious signal representation is the time-domain signal as it is provided through audio recording and required for audio playback.<sup>5</sup> The time-domain representation, however, has some undesired properties that make processing in a neural network difficult. Time-domain audio signals have an unwieldy shape: As they are

---

3. While ultimately sound is a continuous-time signal in this thesis we address only the signal processing part that happens after digitalization and do not concern ourselves with analogue-digital conversion.

4. Computational limitations apply for large durations, where memory constraints might prevent us from processing a whole recording at once. In this case the recording must be split, and each subsection must be processed individually. Thanks to the shift invariance assumption this is possible, but a sufficient overlap must be provided to avoid boundary effects.

5. 100 years of electronic signal processing may make this point seem obvious but as we have seen in Section 2.2, the human brain works completely different. In the brain the signal never appears in the time domain as the frequency decomposition occurs in the ear before the signal is passed to the brain.

one-dimensional, unlike for two-dimensional images, convolutional layers are not as efficient. Audio signals encode their information through oscillation which creates dependencies spread far across the signal. In audio, signals can have significant correlation across multiple seconds. In high fidelity audio, this would require the neural networks to have receptive fields of several 10 000 audio samples which is unfeasible with stacks of one-dimensional convolutional layers. As discussed in Section 2.2, human hearing does not perceive the temporal evolution of sound waves explicitly but rather the frequency components within the sound waves. Thus, the time-domain signal does not only represent the information that arrives in the brain but also perceptually unimportant additional information. Furthermore, the information from different frequency bands is superimposed in the time-domain signal. In particular low amplitude frequency components can be easily perceived but might be almost invisible in the time domain. Finally, comparing audio signals in the time domain is difficult. Two sinusoids with the same frequency might arguably be equivalent but might produce a large difference if the phases are different. While phase information is not completely irrelevant, it emphasizes problems that usually have little significance if comparisons are made in the time domain.

All these issues can be addressed through spectral representations by using the magnitude spectrogram and similar representations. Nowadays, the mel-spectrogram is a very popular choice to representing voice recordings (J. Shen et al., 2018). As a positive side effect, spectral representations are two-dimensional data, like images. We can often borrow ideas from image processing by applying the same (or slightly modified) systems to spectral representations of audio. Ultimately, however, we need to generate a time-domain signal if we want to listen to the result. Therefore, several strategies exist to efficiently model raw-audio.

### Strided, transposed and dilated convolutions

*Strided convolutions* can be thought of<sup>6</sup> as a convolution with a sub-sampling applied afterwards: A strided convolution with stride  $n$  would shift the filter in steps of  $n$ . Thus, the number of required computations is reduced by  $n$  in the first layer and cumulatively in the deeper layers. A *strided transposed convolution* can be thought of as first upsampling by filling  $n - 1$  zeros between each input value and then performing the convolution. Consequently, a stride in a transposed convolution increases the sample rate again which allows us to obtain the required sample rate. *Dilated convolutions* can be thought of as a convolution where the filter is upsampled by filling  $n - 1$  zeros between each filter coefficient. Dilated convolutions do not change the sample rate. Using stacks of varying dilation rates, allows creating large receptive fields using only few filter parameters. Typically, the dilation rate is increased by a factor of two after each layer (Oord, Dieleman, et al., 2016). This way, Prenger et al. (2019), for example, can use only filters of size 3 but have effective receptive fields of several seconds at 24 kHz.

---

6. We describe the operations in this paragraph through illustrative but inefficient implementations. Practical implementations avoid unnecessary operations like multiplications by zero which makes these operations much faster than their non-sparse counterparts.

### Sample rate reduction

To decrease the sample rate of the synthesized signal and to utilize the benefit of parallel computation, we can use a reshape operation to group short time frames into a single time step. This way we are creating an  $n$ -dimensional signal of length  $t/n$  containing  $n$  sub-sampled versions of the 1-dimensional original signal of length  $t$  (Prenger et al., 2019). Since no filtering occurs before the re-sampling the sub-sampled signals contain aliasing from the higher frequency bands. This aliasing is necessary to represent the higher frequency bands and can properly be reproduced by neural networks in the synthetic signals. If combined with a convolution, this representation is equivalent to using a strided convolution with filter size  $s \cdot n$  and stride  $n$  instead of reshape and filter with filter size  $s$ . Similarly to map back from the  $n$  channels to the original sample rate a strided transposed convolution can be used instead of a regular convolution followed by a reshape.

Alternatively, we can use quadrature mirror filters to represent the different frequency bands in separate signals (C. Yu et al., 2020). In this approach the signal is split into  $n$  signals, each representing one frequency band. The frequency bands are obtained by filtering with a specific filter for each frequency band that removes all frequency components outside the specific band. The band-limited signals are then sub-sampled by the factor  $n$ . The frequency bands are of equal width such that the whole frequency band fits into the bandwidth of the sub-sampled signal. To recreate the original signal, the sub-bands need to be up-sampled and modulated on the centre frequency of their respective band. The lowest band is filtered by a low pass and does only require resampling. The band-pass filters can be approximated by pseudo quadrature mirror filter bank (PQMF). These PQMF will be used in the neural vocoder of Chapter 5.

Finally, the complex short time Fourier transform (STFT) can be used to create a signal with multiple frequency bands and reduced sample rate. Kaneko, Tanaka, et al. (2022) showed that an STFT with 8 or 16 frequency bands can provide a good reconstruction with significantly reduced computational cost. While an STFT with 8 or 16 frequency bands is not meaningful in the way we would usually expect from a spectral representation, this representation can work well for sample rate reduction.

### Spectral losses

The short time Fourier transform (STFT) is a linear operation and thus differentiable. The differentiability remains when removing the phase by taking the magnitudes of each frequency bin.<sup>7</sup> Therefore, the magnitude spectrogram can be used to define a loss between two time-domain signals allowing us to circumvent issues with unknown phase information. This is useful for example if we want to synthesize a periodic signal whose pulse positions are unknown. Furthermore, we can use

---

<sup>7</sup>. Note that in terms of complex analysis, many common operations like the absolute value  $|\cdot|$  and real- and imaginary part ( $\Re$  and  $\Im$  resp.) are not holomorphic (complex differentiable) and thus holomorphy is lost if these operations are involved. Usually, however, we only require differentiability with respect to real variables, in which case differentiability is preserved through the Wirtinger derivatives (Caracalla and Roebel, 2017).



**Figure 2.10:** A deep dream transformed image of the Mona Lisa using the VGG16 classifier trained on the ImageNet dataset [From Wikimedia Commons, File: "Mona\_Lisa"\_with\_DeepDream\_effect\_using\_VGG16\_network\_trained\_on\_ImageNet.jpg. Creative Commons CC0 1.0 Universal Public Domain Dedication.]

multiple STFTs with different window lengths to obtain a multi-resolution spectral loss (X. Wang et al., 2019) to match the spectral composition on different time scales.

#### 2.4.2. Common network architecture types for voice

There are different network architectures which are being used in signal synthesis and transformation, notably for images and sound, all with their own advantages and drawbacks. Since voice can be parametrized by spectral representations, in many applications we can treat it as an image. Thus, architectures for image transformation and synthesis can often be easily adapted to voice signals. We will discuss the most common ones here as the choice of the right tool depends on the application and should be carefully made depending on the requirements.

#### Optimization based transformations

Signals can be transformed by synthesizing new signals that are constraint to contain specific information from a source signal. Synthesis can be performed by the same technique that we use to train neural networks, with gradient descent. In this strategy we initialize a prototype signal with zeros or noise and minimize the distance between some feature metric of the target signal and the synthetic signal.

Optimization based transformation has been used for style transfer (Gatys et al., 2016), where the content of a photograph is matched with the ‘artistic style’ of famous paintings. This allows creating a version of the photograph that looks like a painting of famous painters like van Gogh, Picasso, or Kandinsky. Another application of this technique is in deep dream (Olah et al., 2017, see Fig. 2.10). Here a classifier is used to increase the response to specific classes, making these objects appear everywhere in the image, creating a dream-like hallucinogenic image. This principle has been applied by Caracalla and Roebel (2020) to sound texture synthesis.

The drawback of these methods is that an optimization is required during the synthesis. This results in long synthesis durations as thousands of optimization steps are required. (Gatys et al. (2016) report about one hour for a  $512 \times 512$  image on a dedicated GPU.)

### Generative adversarial network

Generative adversarial network are systems of two neural networks that are trained adversarially (Goodfellow et al., 2020). Generative adversarial network are typically applied in synthesis tasks. A generator creates samples that resemble a set of target samples and a discriminator that is trained to distinguish between real samples from the example set, and fake samples generated by the generator. The training procedure is adversarial in the sense that a low loss value for the generator creates a high loss value for the discriminator and vice versa. Since in purely adversarial setups the loss is only provided by the discriminator it is impossible to deduce the quality of the generator from the loss value. Convergence may occur at any loss value and there is no relationship between final loss value and quality of the generated samples. Thus, when training generative adversarial network, convergence issues are likely to occur. These issues can be addressed through the use of auxiliary classifiers (Odena et al., 2017).

GANs have been applied to voice in different ways. MelGAN (K. Kumar et al., 2019) is used to generate audio from mel-spectrograms. An extension MelGAN, Universal MelGAN (Jang et al., 2020), uses multi-resolution time-domain and spectrogram discriminators, multiple discriminators operating on the time-domain signal at different sample rates and on different spectrograms with different window lengths. Similarly, multiple discriminators may be used to operate on different sections of an image or different frequency bands of a spectrogram (C. Li and Wand, 2016). Multiple variants exist for signal transformation, in particular stemming from identity transformation where we wish to map a signal between a set of discrete variants (different speaker). Here we wish to ensure consistency when mapping back and forth between different classes. CycleGAN (J.-Y. Zhu et al., 2017) maps between two classes using two generative adversarial network that work as inverses of each other. A cycle consistency loss ensures that the generators act as inverses. CycleGAN has been applied to voice identity conversion by Kaneko and Kameoka (2018). StarGAN (Y. Choi et al., 2018) allows mapping between multiple classes using a single generator conditioned on the input image and the target domain. An auxiliary classifier guides the convergence. StarGAN has first been applied to voice by Kaneko, Kameoka, et al. (2019). AttGAN (He et al., 2019) combines the ideas of StarGAN and auto-encoders by using an auto-encoder as the generator. Nowadays, we often see discriminators applied to other network architectures combining adversarial losses and losses with known target values (Roebel and Bous, 2022; Bous, Benaroya, et al., 2022).

### Auto-encoders

An auto-encoder is a system of two neural networks that are trained to be inverses of each other (Lange and Riedmiller, 2010). An encoder produces an abstract representation (typically called hidden representation or latent representation) whereas a decoder maps back from the abstract representation to the original input. The point of this encoder-decoder loop could be to create a compressed representation of the input if the latent space has fewer dimensions than the input space. Of interest here, however, is signal transformation. Through the use of additional inputs the auto-encoder may be able to create an abstract representation of its input that is disentangled from a specific feature. For example Lample et al. (2017) allows interpolation in pictures of faces between binary classes like young / old, smile / no smile or glasses / no glasses. The disentanglement can be achieved through a bottleneck auto-encoder, like AutoVC (Qian, Yang Zhang, Chang, X. Yang, et al., 2019), or an adversarial classifier operating on the latent space (Lample et al., 2017). The auto-encoder with information bottleneck is the central tool of Chapter 6 and will be discussed in more detail there.

Generally, auto-encoders have no constraint for their latent space, so the organization of it tends to be highly irregular and elements close to each other in the latent space may correspond to completely different inputs. To regularize the latent space, variational auto-encoders (Kingma and Welling, 2013) encode their input by a probability distribution on the latent space rather than a single point. In practice this means that the encoder returns the parameters of a parametric probability distribution, usually mean vector and covariance matrix of a multi variate Gaussian. During training, we sample from this distribution before mapping back to the original space for reconstruction. This forces the auto-encoder to create smooth transitions in its latent space and allows interpolating between samples by following a path in the latent space, for example between the timbre of different instruments (Esling, Bitton, et al., 2018).

### Generative flow

The idea of a generative flow is to create a complex probability distribution from a simple elementary distribution, like the multivariate Gaussian distribution. This can then be used to model the probability distribution of images of faces (Kingma and Dhariwal, 2018) or time domain representation of speech (Prenger et al., 2019). To obtain samples from this distribution we sample from the multivariate Gaussian distribution and use the generative flow to obtain a sample of the desired distribution.

The flow is trained by applying the inverse of the flow to the training samples and using a loss that constraints the output to be Gaussian distributed. For this the flow needs to be invertible, which is not the case for arbitrary neural networks. A flow consists of (invertible) affine operations and (bijective) activation functions, but the weights of the affine operations are not trainable weights — the weights of the affine operations are calculated through a convolutional neural network from a subset of the input. The subset that is used to calculate the affine transformation is

passed unchanged to the next layer. Thus, the affine operation can be reconstructed on the other side, making each layer invertible.

The drawback of this architecture is that each layer in the flow requires a whole neural network to calculate its weights. Therefore, neural networks based on generative flows tend to be extremely large and training and inference are relatively slow.

### Differentiable digital signal processing

Many elements of digital signal processing are differentiable and can thus be used as elements in a neural network. In particular the parameters of filters, windows, wavetable synthesis and additive- and subtractive synthesis can be learnt as weights of a neural network or be computed by a neural network. During training, the gradient can be transported through these operations allowing us to apply losses on synthesized signals. This hybrid approach between digital signal processing and deep learning is called *differentiable digital signal processing (DDSP)* (Engel et al., 2019). DDSP allows integrating prior knowledge about the signal into the architecture. For example, melody instruments like the flute can only play one note at a time, so it makes sense to restrict the synthesis to use only one fundamental frequency. At the same time we get interpretable parameters. In the example with the flute, we would obtain the fundamental frequency, which corresponds to the note that is being played. The imposed restriction can guide convergence of the models while also being very expressive. We can learn hundreds of filter coefficients or wavetables with hundreds of points as functions of the input parameters.

A drawback of this approach is that the assumptions may be too restrictive for special cases. In speech, we generally assume that the glottal pulses happen periodically following a fundamental frequency. This is true during stable phonation but is violated in rough speech, creaky voice and even at the boundary between voiced and unvoiced speech. In these modes of speech production glottal pulses may happen irregularly either through varying amplitude, varying pulse positions, or missing of some pulses. By means of over-parametrisation,<sup>8</sup> a DDSP model could in theory model many of these effects as well, but convergence is unlikely as these re-parametrisations tend to lie in different local minima.

## 2.5. Datasets

Deep learning relies on the availability of large datasets. Since all models are learnt from data we cannot expect to produce much more than what is contained in the data. Any deep learning model reflects the data it has been trained with. Thus, it is worth to pay closer attention to the datasets that were available during the time this research carried out.

---

<sup>8</sup>. We could for example construct an equivalent representation for wavetable synthesis with wavetable  $w$  and frequency  $f$  by concatenating  $w$  to itself (and resampling) and dividing  $f$  by 2.

Most of the time we treat singing voice and speech separately due to their differences. Voice conversion systems for singing voice have rather different requirements as voice conversion systems for speech. As both forms of voice rarely occur together it is justified to develop different models for either voice type if the quality of both models can be increased by separating the voice types. Thus, we will have different datasets for singing voice applications and for speech applications. A lot of datasets with voice are publicly available for both singing voice and speech.

### 2.5.1. Speech datasets

Speech datasets are available in large quantities. Since consumer microphones are widely available these days, anyone can record themselves reading a book or a newspaper. Projects like [LibriVox](#)<sup>9</sup> create large corpus of audiobooks by crowdsourcing the recording process, which can be used to generate large speech datasets like LibriSpeech (Panayotov et al., 2015). Furthermore, vast amounts of voice recordings can be easily found on video hosting platforms and as podcasts. Projects like Vox-Celeb (Nagrani et al., 2019) collect voice from celebrities from the internet resulting in thousands of hours of speech.

The result, however, of these large scale efforts are huge amounts of data with poor quality. In the case of amateurs reading audiobooks the recording conditions may be suboptimal resulting in high background noise or distortions from cheap recording hardware. Additionally, through audio compression further artefacts are introduced. In the case of videos from the internet we face the difficulty that voice is often not isolated. Additional signals such as sound effects, background music, background noise or other speakers interrupting each other result in unclean data. Furthermore, the same difficulties as before arise if the videos are created by amateurs. Due to the huge size of these datasets it is impossible to ensure consistent quality.

Depending on the application, similar arguments may apply to the input data processed by the user. If a user wishes to transform a recording that has been made under poor recording conditions, the transformation might fail if the transformation system has never seen unclean data. Thus, to increase robustness it might actually be desirable to include noisy or poor quality data.

On the other hand, deep learning approaches reproduce the data they are trained with. Unclean training data degrades the quality of the derived models as it may teach the model to generate noisy sounds as a result of a transformation. Thus, if the goal is to process and produce high-quality audio, these crowdsourced large-scale datasets may not be suitable.

There are a few high-quality datasets available for speech. The datasets that satisfy our quality demands and were thus used in this thesis are summarized in [Table 2.1](#) and described below:

**Att-HACK** Database for French expressive speech recorded by Moine and Obin (2020). The database includes 20 speakers speaking a set of phrases in four

---

9. <https://librivox.org>

**Table 2.1.: Speech datasets used in this thesis.**

| Name       | Language | #Spk. | Dur.     | Credit                    |
|------------|----------|-------|----------|---------------------------|
| Att-HACK   | French   | 20    | 21:51:55 | (Moine and Obin, 2020)    |
| BREF120    | French   | 120   | ~ 100 h  | (Gauvain et al., 1990)    |
| BREF80     | French   | 80    | 10:28:45 | (Gauvain et al., 1990)    |
| CMU Arctic | English  | 3     | 2:42:56  | (Kominek and Black, 2004) |
| TIMIT      | English  | 630   | 5:11:03  | (Zue et al., 1990)        |
| VCTK       | English  | 110   | 33:47:56 | (Yamagishi et al., 2019)  |
| LJ Speech  | English  | 1     | 23:55:17 | (Ito and Johnson, 2017)   |

different attitudes, friendly, seductive, dominant and distant.

**BREF120** Large commercial database containing 100 hours of French speech from 120 speakers. The speakers read from a newspaper.

**BREF80** Smaller subset of the BREF120 database, containing only 80 speakers and less material per speaker.

**CMU Arctic** A dataset created for speech synthesis at Carnegie Mellon University by Kominek and Black (2004). The dataset is accompanied with electroglottograph (EGG) measurements which allow detection of pulse positions.

**TIMIT** A dataset created for speech recognition in 1990 by Zue et al. (1990). The dataset contains 630 native English speakers covering a large variety of American dialects.

**VCTK** Multispeaker dataset recorded by Yamagishi et al. (2019). The dataset includes speech data from 110 English speakers. Note that in this dataset each utterance is recorded with two microphones. Thus the total available audio is twice as much as the total duration.

**LJ Speech** Single speaker dataset by Ito and Johnson (2017). The speaker is reading 7 non-fiction books which amounts to approximately 24 hours of speech.

### 2.5.2. Singing datasets

Singing datasets are much rarer than speech datasets. There are multiple reasons behind this. First, the research community around singing voice is much smaller than for speech. Thus, much fewer datasets are produced with singing voice than there are for speech. Recording high quality databases requires a lot of money to pay for the artists, technical equipment and staff to operate the equipment. Although more and more commercial applications of singing voice technology (and music technology in general) have emerged in recent years, the market for speech technology is still much bigger (and probably always will be). Thus also funding from the private sector is much larger for speech databases.

Another reason is how we encounter singing voice in the wild. While we generally appreciate isolated speech, as it maximizes intelligibility, the opposite is true for singing voice. Solo a cappella singing is very rare in music. Singing voice is in most cases accompanied by other instruments, other voices or sound effects. Thus finding isolated singing on the internet yields much fewer data than for speech and thus makes projects like VoxCeleb unfeasible.

Finally, singing requires much more physical effort than speaking. While it can be exhausting to speak non-stop for hours, it is almost impossible for singing. It takes years of practice for professional singers to be able to sing through an opera of several hours. Even then, scores are deliberately written in a way that allows enough breaks for each role so that a singer does not have to sing for too long. Thus recording the same duration of singing requires much more effort than speech which means that singing databases generally tend to be shorter.

Additional difficulty arises from the variability in singing techniques. Different singing styles have emerged in different cultures and for different purposes. Each of these styles typically consists of a set of different possible singing techniques. Some of these techniques may overlap between different styles while some may be unique to a specific style and sometimes the same technique is used in a different context in different singing styles. To represent singing voice in general a comprehensive dataset needs to cover all different singing styles in addition to the variety of voices. Thus, a representative singing voice dataset should be even larger than a speech dataset with a similar ambition. Alternatively a singing voice dataset may focus on only one singing style. Most singing voice datasets opt for the second choice while other datasets try to cover a large variety of singing styles and consequently under-represent all of them.

Many of the existing singing voice datasets have been recorded for different purposes than deep learning. Concatenative synthesizers like Vocaloid or the [IRCAM Singing Synthesizer](#) (Ardaillon, 2017) require all diphones of a language to be recorded by one singer. The desired phonetic sequence is generated by concatenating the required diphones and the desired melody is obtained by transposition. The required dataset is recorded on a constant pitch and with minimal expression. These datasets are relatively large and common but are not optimal for deep learning as they do not represent real singing; they are samples of how singing voice sounds like but do not reflect how singers would interpret a piece and lack expressive features like  $f_0$  contours and articulations.

The datasets that were available for this thesis are summarized in [Table 2.2](#) and described below.

**CREL** A Research Database recorded by Tsirulnik and Dubnov (2019). It contains singing voice from three professional singers, singing scales and the song *Twinkle Twinkle Little Star* in different singing styles.

**NUS-SMC** The NUS-48E Sung and Spoken Lyrics Corpus, a dataset from Duan et al. (2013) at the Sound and Music Computing Lab at National University of Singapore. It consists of singing and speech from 9 amateur singers.

## 2. History and fundamentals

---

**Table 2.2.: Singing datasets used in this thesis.**

| Name      | Language | #Spk. | Dur.    | Credit                       |
|-----------|----------|-------|---------|------------------------------|
| CREL      | English  | 3     | 1:48:45 | (Tsirulnik and Dubnov, 2019) |
| NUS-SMC   | English  | 9     | 1:18:34 | (Duan et al., 2013)          |
| ICH       | Greek    | 1     | 2:49:27 | (Grammalidis et al., 2016)   |
| PJS       | Japanese | 1     | 30:44   | (Koguchi et al., 2020)       |
| JVS-MuSiC | Japanese | 100   | 1:36:24 | (Tamaru et al., 2020)        |
| Tohoku    | Japanese | 2     | 3:44:46 | (Ogawa and Morise, 2021)     |
| VocalSet  | Multiple | 20    | 8:35:17 | (Wilkins et al., 2018)       |
| ISiS      | French   | 3     | 4:53:08 | (Ardaillon, 2017)            |
| Farinelli | Italian  | 7     | 6:30:37 | (Deschamps, 2022a)           |

**ICH** Byzantine singing from the i-Treasures Intangible Cultural Heritage dataset created by Grammalidis et al. (2016). It consists of 4 hours of Byzantine singing sung by a single singer in baritone range.

**PJS** Phoneme-balanced Japanese singing-voice corpus by Koguchi et al. (2020). The dataset consists of 100 phrases both sung and spoken by the same singer. The singing is pop tenor voice.

**JVS-MuSiC** Japanese multispeaker singing-voice corpus by Tamaru et al. (2020). The dataset contains singing voice from 100 amateur singers singing two songs. One song is shared by all singers and the other may be different for each singer.

**Tohoku** Two voices, Tohoku Kiritan and Tohoku Itako, from the singing synthesisers Voiceroid and Utaw. Recordings of the Tohoku Kiritan voice include singing of real pop songs which have been made available by Ogawa and Morise (2021).

**VocalSet** A singing voice dataset with 20 professional singers made available by Wilkins et al. (2018). The dataset includes vocalises as well as songs from folklore, opera and clerical repertoire.

**ISiS** Datasets recorded by Ardaillon (2017) for the IRCAM Singing Synthesizer. It consist of singing from three French professional singers from opera, jazz and pop. Most of the singing is performed on constant pitch but several real songs are also included.

**Farinelli** Dataset recorded by Judith Deschamps for her artistic residency at IRCAM. Seven singers were recorded singing the song *Quell'usignolo* by Giacomelli as well as vocalises and songs from their own repertoire.

### 2.5.3. Combining datasets

Some datasets presented here are rather small and in particular for singing, none of those datasets is large enough to train models that are general enough for consumer

purposes. To increase the effective dataset size we combine datasets to obtain a wider coverage of different singing styles, languages, voice identities etc. We can even combine singing and speech to train universal voice models that work on any type of voice.

There is no uniform standard in dataset organization thus each dataset comes in its own format. The simplest form is a folder that contains all the recordings of the dataset split into short audio files. Often, however, datasets come with additional metadata. Multi-speaker and multi-singer datasets typically annotate their files with the speaker / singer information. Especially in speech, phonetic or word transcriptions are common and can be provided in a separate file for each audio file or in a single file containing all the text. Some datasets are even unprocessed or contain long files that are unhandy and thus need to be cut into smaller files.

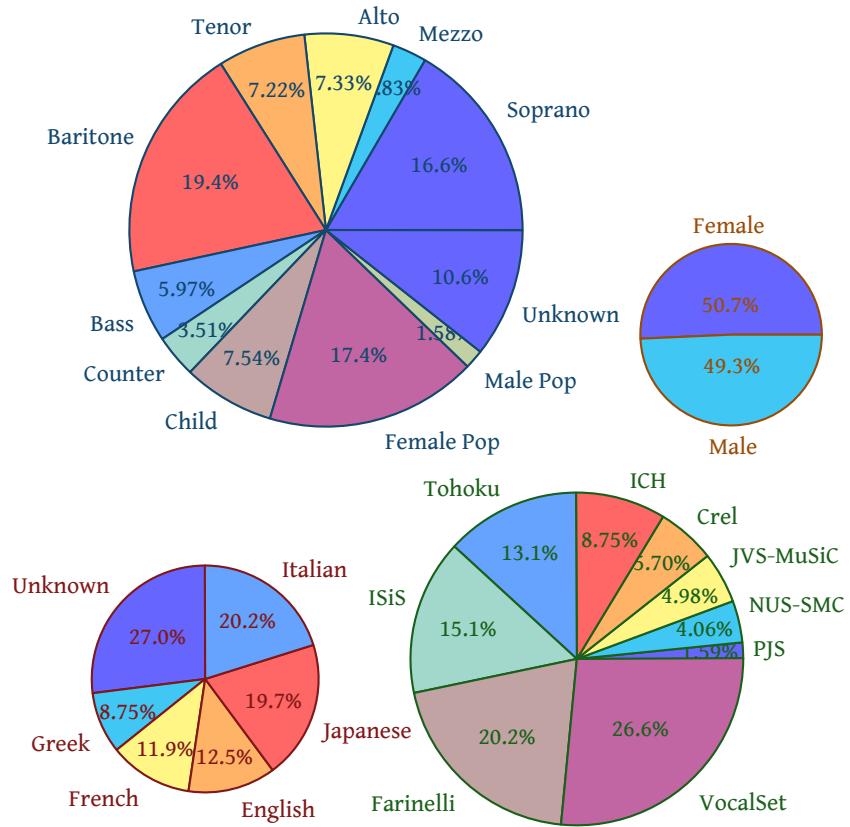
To standardize all the metadata we developed an ecosystem for databases. Maintaining this framework is a significant part of necessary housework for the researcher in machine learning and an ongoing chore as every new database needs to be integrated in the unified framework. The standardized framework can be seen as a contribution of this thesis to other research within our research group as many of the preprocessing scripts have been reused in other research projects. We extended the existing database structure with import scripts, notably with the possibility to automatically cut long files during silences and remove excessive leading and trailing silences around the imported audio files.

### Sampling

When working with combined datasets we have to address the question of sampling. If we combine for instance a very large dataset in one language with another not so large dataset in another language we might over-represent the language of the larger dataset if we sample uniformly from all training samples. Similarly, some datasets have shorter phrases and thus consist of many short audio files while others might consist of longer audio files. If we uniformly sample from all audio files we will over-represent the dataset which contains short audio files. The effect of the latter can be mitigated by cutting all audio files into the same length, but this might not be desired as it would require cutting within a speech- or musical phrase.

In other cases we want to over-represent some samples, for instance, to ensure that even rare cases are synthesized reasonably well. For example when we were developing the mel-inverter we had the problem that singing voice above 700 Hz (F5) would not work well (or at all) although typical soprano ranges go up to 1200 Hz (D6) or even higher and samples of this were included in the training data. However, inspecting the training data for these models revealed that far less than 1% of all the audio frames in the training set contained singing with a fundamental frequency above 700 Hz. Thus, the error on these frames was insignificant for the training loss. The issue was resolved by drawing audio samples that contained *high-pitched* singing far more often.

Whether we need to equalize or emphasize some properties in the training data, a good sampling strategy is essential. Which sampling strategy is appropriate will



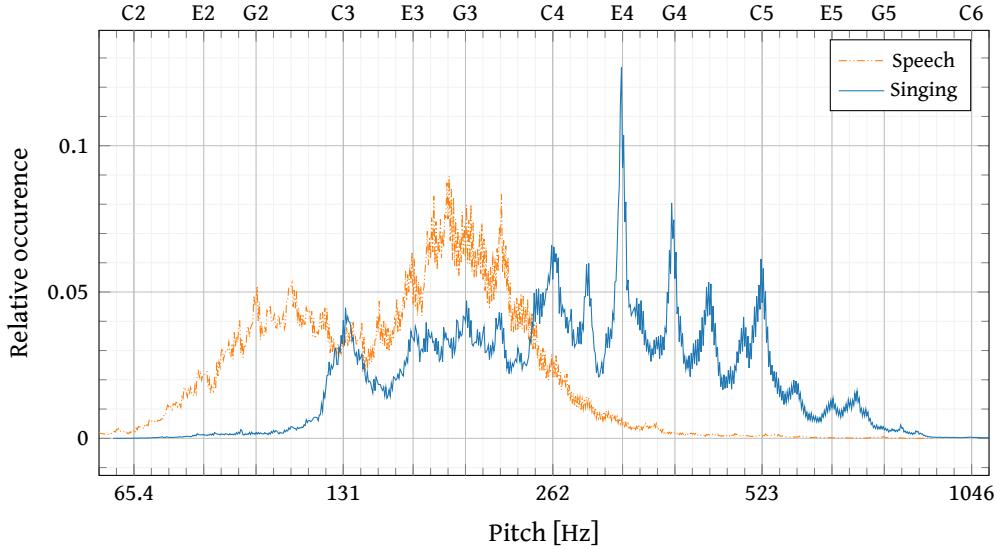
**Figure 2.11.:** Pie charts showing the distribution of different attributes in the combined singing dataset. Clockwise from top left: **Voice type**, **gender**, **database**, and **language**.

depend on the application and needs to be found through experimentation.

#### 2.5.4. Dataset analysis

The neural models that we train are reflections of the data we train them with. They distil information from the training set. It is therefore essential to be very familiar with the datasets that we use and be aware of their properties.

We start our analysis with the singing databases. As we will almost always train our effects on the combined database of the databases from Table 2.2, we will analyse how the databases mix together. The distribution of different attributes over the duration can be seen in Fig. 2.11. We see that the dataset is astonishingly well-balanced between male and female. This is surprising as most of the databases themselves are not well-balanced, not even the ones containing many singers. We can see that the dataset is dominated by soprano, baritone and female pop. Within pop singing, the dataset is strongly biased against female singers. This is because we have 3 female but only one male pop singer and the database with the male pop singer (**PJS**) is not very large. The soprano range is dominating against other female classical voices as it is the most common female classical voice and is represented by

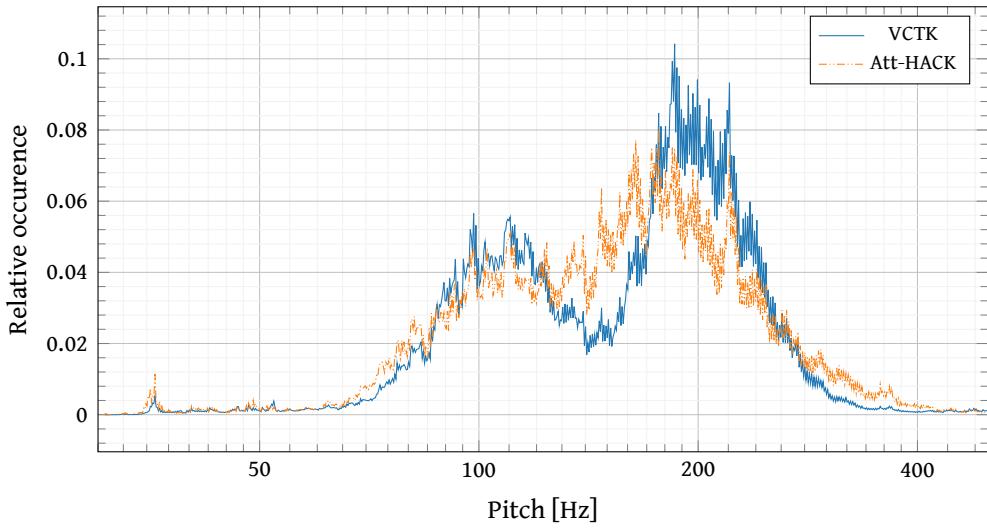


**Figure 2.12.:** Histogram of fundamental frequency in different datasets. The singing dataset consists of all the databases from [Table 2.2](#). The speech dataset is a combination of [VCTK](#) and [Att-HACK](#) as used in [Chapter 6](#). The histogram has a resolution of 6.25 cent and is normalized by width.

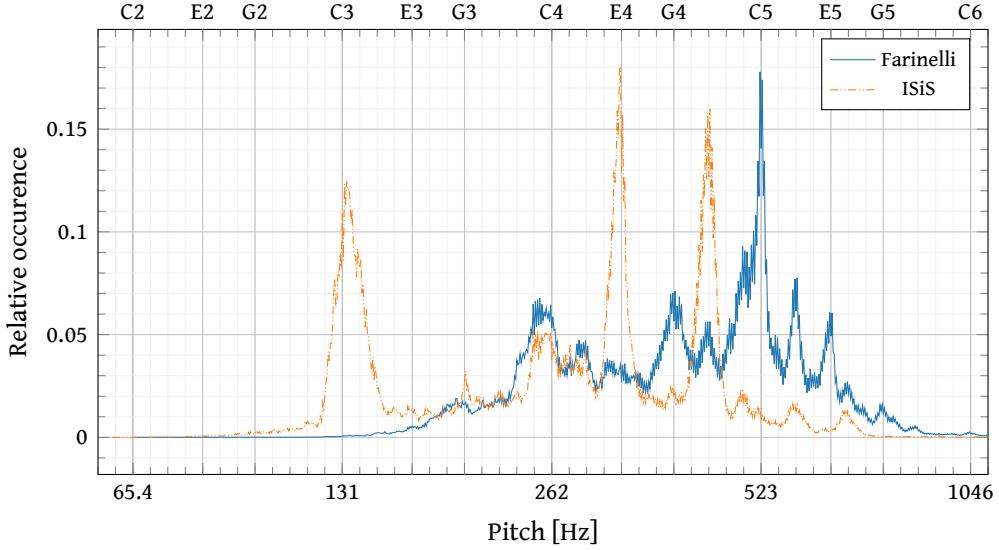
11 distinct singers while Alto and Mezzo only have 2 each. Similarly, in our dataset the baritone range is most common among male classical voices. In this case, this is due to two singers with a large number of recordings, (from [ICH](#) and [ISiS](#)) who together already amount for 17% of the total voices. Finally, we can see that the dataset is well-balanced between 5 languages, Japanese, English, Italian, Greek and French, while a large portion is unknown. However, this distribution should not be taken too serious as much of the singing is in coloraturas or vocalises which is rather different from the actual language but was not treated separately in this analysis.

A lot of time will be spent discussing the fundamental frequency of the voiced segments in particular as we will be changing the fundamental frequency in [Chapter 6](#). Thus, it is worth to have a look at the distribution of the fundamental frequency inside the dataset. We obtain the fundamental frequency annotation by using the analysis method proposed by [Ardaillon and Roebel \(2019\)](#). Histograms of the fundamental frequency for different datasets can be seen in [Figs. 2.12 to 2.14](#). One big difference between speech and singing is the distribution of the  $f_0$ , as can be clearly seen in [Fig. 2.12](#): In singing we generally observe higher  $f_0$  than in speech. For both speech datasets (see [Fig. 2.13](#)) we can identify two distinguishable modes corresponding to male and female speakers. The modes for male speakers are very similar between [VCTK](#) and [Att-HACK](#), while the distribution of  $f_0$  for the female speaker has a lower average for the [Att-HACK](#) database and a larger variance.

In the histogram of the full singing voice dataset, [Fig. 2.12](#), a striking feature are the peaks at different locations, most prominently at E4, G4, C4 and C5. Further peaks can be found at D4, A4, C3 and closer inspection also reveals peaks that can



**Figure 2.13.:** Histogram of fundamental frequency in speech different datasets. Both histograms show two modes for male and female voices respectively.



**Figure 2.14.:** Histogram of fundamental frequency in different singing datasets. The **ISiS** dataset was recorded for concatenative singing synthesis and shows the sparse distribution of pitches for those datasets. The **Farinelli** dataset consists of singing of mostly the same song, *Quell'usignolo*, the song is in C-major, and we see peaks for many notes from the C-major scale.

be associated with other notes from the 12-tone scale. All large peaks are part of the C major scale. Furthermore, the most prominent peaks are from the C major triad (C, E, G). This can be explained by the fact that most of the material in the dataset is in fact recorded in C major or some related key that shares most of the scale with C major. A significant amount of data comes from datasets which were recorded specifically for concatenative synthesis (most of **ISiS**, in part **Tohoku**) where coverage of phonetic material is far more important than coverage of pitches. Thus, in those datasets only one or a few pitches are represented, which explains the most prominent peaks (C3, D4, E4, G4, A4, C4). The difference between a dataset for concatenative singing synthesis and a dataset of real singing is demonstrated in Fig. 2.14 where we compare **ISiS**, a dataset recorded for concatenative singing synthesis, versus **Farinelli**, a dataset containing real singing (mostly in C major).

Another interesting observation from Fig. 2.12 is that although the soprano range typically exceeds C6 (1047 Hz) and frequently reaches up to F6 (1400 Hz), fundamental frequencies above G5 (784 Hz) rarely occur in the dataset, even though 1/6th of the recordings come from actual sopranos. Similarly, for the lower notes we observe that below B2 (124 Hz) there are only very few recordings. As discussed in the previous section, the lack of representation of extreme pitches within the dataset is an issue that we will have to overcome for both mel-inversion ([Chapter 5](#)) and pitch transformation ([Chapter 6](#)).

## 2.6. Perceptive tests

When studying synthesis methods, we investigate how to create something new. This can lead to fascinating results and gives us new possibilities in creativity. The drawback of this kind of discipline is that the problem is often not defined exactly which leads to multiple equally valid but very different solutions. As a consequence evaluating the results and by extension evaluating the solution methods is subjective. To allow for scientific rigour we rely on reliable metrics that allow comparing one method to another. This is typically done in perceptive tests. A subjective quality can be measured by asking human participants for their subjective preference. Most of the time we want to evaluate the audio quality, but we might also ask about other qualities like how loud someone is singing, as we did in [Chapter 7](#), or the similarity to a certain speaker, as is common for voice identity conversion.

### 2.6.1. Conception

The International Telecommunication Union (ITU) provides a recommendation for subjective testing in the ITU-T P.800 and ITU-R BS.1534-3 recommendations. ITU-T P.800 gives guidelines for evaluating telecommunication systems and measuring the subjective degradation. ITU-R BS.1534-3 give recommendations how to subjectively evaluate the intermediate quality of audio systems. We can borrow many of the recommendations from ITU-T P.800 and ITU-R BS.1534-3 for our own subjective test design.

### Common test designs

In a perceptive test we let participants (test subjects) evaluate randomly drawn audio samples (test samples) generated by the synthesis methods we want to evaluate. The provided scores are averaged over all test subjects and test samples to generate a mean opinion score (**MOS**). Recommendations ITU-T P.800 and ITU-R BS.1534-3 provide a few common test designs:

**Absolute Category Rating (ACR)** According to ITU-T P.800 the quality of a sound recording can be quantified by translating the following subjective values into the corresponding numbers:

| Quality of the speech | Score |
|-----------------------|-------|
| Excellent             | 5     |
| Good                  | 4     |
| Fair                  | 3     |
| Poor                  | 2     |
| Bad                   | 1     |

**Degradation Category Rating (DCR)** The vocoder can be seen as a communication channel which introduces degradation with respect to the original recording. From this perspective one can employ a degradation scale instead of the quality scale using a Degradation Category Rating (**DCR**) (ITU-T P.800, Annex D)

| Degradation                             | Score |
|---|-------|
| Degradation is inaudible                | 5     |
| Degradation is audible but not annoying | 4     |
| Degradation is slightly annoying        | 3     |
| Degradation is annoying                 | 2     |
| Degradation is very annoying            | 1     |

**Comparison Category Rating (CCR)** When comparing two synthesis methods against each other it can be advisable to compare samples from both methods directly through a Comparison Category Rating (**CCR**) (ITU-T P.800, Annex E):

| Comparison      | Score |
|-----------------|-------|
| Much better     | 3     |
| Better          | 2     |
| Slightly better | 1     |
| About the same  | 0     |
| Slightly worse  | -1    |
| Worse           | -2    |
| Much worse      | -3    |

A CCR is particularly useful if we want to evaluate the impact of a specific hyperparameter to our model by comparing two versions of the model with only the hyperparameter changed.

**MUSHRA** The MUlti Stimulus test with Hidden Reference and Anchor (MUSHRA) test is recommended by the ITU-R BS.1534-3. Grading is done on a continuous scale (typically discretized by steps of one) from 0 to 100 with the following nominal correspondences (ITU-R BS.1534-3)

| Quality of the speech | Score  |
|-----------------------|--------|
| Excellent             | 80–100 |
| Good                  | 60–80  |
| Fair                  | 40–60  |
| Poor                  | 20–40  |
| Bad                   | 0–20   |

Test subjects are asked to rate the samples from the models that we want to evaluate together with original, unprocessed recordings (hidden reference) as well as two hidden anchors that undergo standardized degradation. The degradation is achieved by low-pass filtering at 3.5 kHz and 7 kHz respectively (ITU-R BS.1534-3, Section 5.1). The test subjects are not told which sample comes from which source.

### 2.6.2. Procedure

Standards ITU-T P.800 and ITU-R BS.1534-3 both recommend a controlled test environment with the same equipment for each participant. While a controlled test environment yields the most consistent results, this requirement makes the evaluation expensive and thus impractical for our purposes. The progress in our domain is fast-paced with new models being created every week. We need to perform perceptive tests regularly to obtain indications which methods really provide improvements. The ITU recommendations are designed for testing consumer products with anticipated longer life-spans. In that context it makes sense to invest more money to obtain a reliable rating of the finished product. In our domain, new improvements can appear every few weeks making results of perceptive test viable for much shorter time.

For us the trade-off between cost of testing vs. frequency of testing is leaning more towards frequency rather than quality. Therefore, we permit ourselves to deviate from the recommendations with regard to testing environment. This requires more strict post-screening which will be discussed in [Section 2.6.3](#). To be able to reach as many participants as possible our test are carried out online, thus allowing many people to participate in a short time for relatively low cost.

Our online test consist of a web-page that presents audio samples and requires the participant to give a rating with respect to the quality to be evaluated. [Fig. 2.15](#) shows a screenshot of the perceptive test from [Chapter 7](#). The test starts with an explanation about what is to be tested together with some instructions how to carry out the test. To evaluate the quality of the synthesized samples from [Chapters 6](#) and [7](#), we perform [ACR](#) tests and let the test subjects evaluate audio samples on the perceptive scale from 1 to 5. The order of the samples is randomized according to ITU-R BS.1534-3, Section 3. Furthermore, [MUSHRA](#) is performed to evaluate the quality of the audio samples generated by the model in [Chapter 5](#), and we use a modified [CCR](#) to compare evaluate the effect of the voice transformation in [Chapter 7](#) (depicted in [Fig. 2.15](#)).

The ITU-R BS.1534-3 standard recommends expert listeners as test subjects. Using the same argument as before we permit ourselves to deviate from this recommendation to allow faster and more frequent testing. The ITU-T P.800 standard recommends ‘Subjects [...] are chosen at random from the normal telephone using population’ (ITU-T P.800, Section A.4.1). Thus, according to ITU-T P.800, any person that is able to hear would be a suitable test subject. To recruit many participants in a short time we use an online platform to hire paid test subjects. The platform allows targeting specific demographics and hiring people with specific skills. This is especially useful when evaluating samples containing speech as a high proficiency is required in the language that is being synthesized. According to ITU-T P.800 in order to get people to do the tests carefully the test should not be too long so that the participants are finished before they get tired. It is recommended that ‘Ideally no session should last for more than 20 minutes [...]’ (ITU-T P.800, Section B.3). However, we found that for internet based tests even short durations are advisable and believe durations between 5 and 10 minutes are best to get valuable answers.

### 2.6.3. Evaluation

#### Post-screening

The studies are carried out online by anonymous participants. Carrying out the study online allows us to reach a lot of people from a wide range of demographics for a relatively cheap price. On the other hand we have no control over the test environment. Participants are advised to take the test in a quiet environment using good quality headphones, but we have no means to make sure these instructions are properly honoured. Since test participants earn more money the more surveys they participate in, there is an incentive for them to finish the test quickly. As a consequence some submissions may be of bad quality. This could be either because participants did not read the instructions properly, because they didn’t take the

## Perceptive Test

### Transformation of intensity in singing voice

contact: [frederik.bous@ircam.fr](mailto:frederik.bous@ircam.fr)

Thank you for your time! This is a perceptive test to evaluate the quality of our new intensity transformation algorithm.

**Instructions**

- Make sure you have the time to take the test in one sitting. It will take around 5 minutes.
- Do the test in a quiet place.
- Use headphones.
- Verify that the sound level is **loud enough** to hear the sound details properly. Use this test sound to adjust your volume now.

- If there is any **technical problem** (e.g. sound not playing), tick the box in the row "Problem". (please note that it may take a few seconds for all sounds to load)
- You will hear pairs of short samples of singing. For each pair please rate which one you think was sung louder / with a **stronger voice**. For your response, please take into account **only the timbre of the sound not the volume**. If you think the left sound was sung in a louder voice please select a number on the **left** according to the degree of the difference. If you think the right sound was sung in a louder voice please select a number on the **right** accordingly.
- Take your time and listen carefully. Sometimes the differences are very subtle. If you are unsure, you can listen to the samples as much as you like.

| Pair | File1 | left is stronger            |                             | equal                      | right is stronger           |                             | File2 | Prob                     |
|------|-------|-----------------------------|-----------------------------|----------------------------|-----------------------------|-----------------------------|-------|--------------------------|
| 1    |       | +2<br><input type="radio"/> | +1<br><input type="radio"/> | 0<br><input type="radio"/> | +1<br><input type="radio"/> | +2<br><input type="radio"/> |       | <input type="checkbox"/> |
| Pair | File1 | left is stronger            |                             | equal                      | right is stronger           |                             | File2 | Prob                     |
| 2    |       | +2<br><input type="radio"/> | +1<br><input type="radio"/> | 0<br><input type="radio"/> | +1<br><input type="radio"/> | +2<br><input type="radio"/> |       | <input type="checkbox"/> |
| Pair | File1 | left is stronger            |                             | equal                      | right is stronger           |                             | File2 | Prob                     |
| 3    |       | +2<br><input type="radio"/> | +1<br><input type="radio"/> | 0<br><input type="radio"/> | +1<br><input type="radio"/> | +2<br><input type="radio"/> |       | <input type="checkbox"/> |
| Pair | File1 | left is stronger            |                             | equal                      | right is stronger           |                             | File2 | Prob                     |
| 4    |       | +2<br><input type="radio"/> | +1<br><input type="radio"/> | 0<br><input type="radio"/> | +1<br><input type="radio"/> | +2<br><input type="radio"/> |       | <input type="checkbox"/> |

**Figure 2.15.:** Screenshot of the perceptive test website from Chapter 7. The test follows the typical design pattern: First a few instructions are given on how to carry out the test (yellow box). After that the actual test follows. This test does pairwise comparisons and the participant is asked to rate which one has been sung in a “stronger” voice.

time to properly listen, or because they could not hear the audio samples well due to inadequate environment or equipment. Some participants might even give random answers just to finish the test as quick as possible. Thus, to improve the quality of the results, it might be necessary to remove submissions with remarkable abnormality.

The ITU-R BS.1534-3 standard recommends a post-screening of the test subjects by evaluating their response of the hidden reference. Subjects that rated the hidden reference too often too low should be excluded from the aggregated scores if they rated more than 15% of the hidden references with a score lower than 90 (on a scale from 0 to 100) (ITU-R BS.1534-3, Section 4.1.2). With this approach we found that we can significantly improve the test accuracy in terms of confidence intervals while loosing only few (less than 5%) of the submissions.

### Confidence intervals

A perceptive test is a very unreliable measurement method. Obviously, since responses are subjective and every participant has a different background, different expectations, tastes, and habits responses will vary enormously, and we can expect frequently contradicting responses even from the same person. Thus, it is important to quantify the quality of the estimate itself. The ITU-T P.800 standard recommends the **mean opinion score** to be accompanied by 95%-confidence intervals. Rather than providing a single number as our estimate, we provide an interval within we believe the true value to be in with 95% certainty.

Derivations for 95%-confidence intervals can be found in standard literature on statistics, e.g. in EMS (2020). A typical derivation goes as follows: Assume we get a set of  $n$  ratings  $\mathbf{X} = (X_1, \dots, X_n)$  and assume that  $X_1, \dots, X_n$  are independent and identically distributed Gaussian random variables with distribution  $X_1 \sim \mathcal{N}(\mu, \sigma^2)$  with unknown mean  $\mu$  and unknown standard deviation  $\sigma$ . Thus, the mean  $\mu$  represents the **mean opinion score**, the quantity we want to estimate. The standard deviation  $\sigma$  represents how controversial the opinion is, that is, if  $\sigma = 0$  everyone agrees whereas the higher  $\sigma$  is, the more different the opinion of each individual is. A high standard deviation  $\sigma$  does not necessarily mean that the confidence interval will be large. The confidence interval measures how sure we are that the mean lies within a certain region but makes no statement about how controversial this mean is. As we will see in the derivation, however, the standard deviation  $\sigma$  has an influence on the confidence interval insofar as a higher  $\sigma$  requires more samples to achieve a small confidence interval.

We are thus interested in an estimator  $M_n : \mathbb{R}^n \rightarrow \mathbb{R}$  such that the expectation  $E[M_n(\mathbf{X})] = \mu$  and the variance  $V[M_n(\mathbf{X})]$  is minimal. To estimate  $\mu$  we use the arithmetic mean, thus our estimate  $\bar{X}$  of  $\mu$  is

$$\bar{X} = M_n(\mathbf{X}) = \frac{1}{n} \sum_{k=1}^n X_k. \quad (2.32)$$

The estimate  $\bar{X}$  is itself a random variable and since the Gaussian distribution is

stable,  $\bar{X}$ , as a sum of Gaussian random variables, is Gaussian again. In particular

$$\bar{X} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right). \quad (2.33)$$

Since  $\bar{X}$  is a continuous random variable it is unlikely that the estimate  $\bar{X}$  is exactly correct ( $\mu \neq \bar{X}$  almost certainly). In fact the error can be given as

$$\bar{X} - \mu \sim \mathcal{N}\left(0, \frac{\sigma^2}{n}\right) \quad (2.34)$$

and thus using the 2.5% quantile  $q \approx 2$  of the standard normal distribution we get

$$95\% = \mathbf{P}(\bar{X} - \mu \in [-c, c]) \quad (2.35)$$

$$= \mathbf{P}(\mu \in [\bar{X} - c, \bar{X} + c]), \quad (2.36)$$

where  $c = q\sigma/\sqrt{n}$ . Thus, with 95% probability the real mean  $\mu$  lies within the interval

$$\left[ \bar{X} - \frac{q\sigma}{\sqrt{n}}, \bar{X} + \frac{q\sigma}{\sqrt{n}} \right]. \quad (2.37)$$

In most cases, however, the standard deviation  $\sigma$  is unknown as well and thus the confidence interval from Eq. (2.37) is useless for practical purposes. We can estimate the variance as

$$S^2 = \frac{1}{n-1} \sum_{k=1}^n (X_k - \bar{X})^2, \quad (2.38)$$

but now the estimated variance is itself a random variable and thus Eq. (2.34) does not make sense if we replace  $\sigma^2$  with  $S^2$ . We normalize  $\bar{X} - \mu$  by its estimated standard deviation and observe that

$$T := \frac{\bar{X} - \mu}{S/\sqrt{n}} \quad (2.39)$$

follows Student's t-distribution with  $n - 1$  degrees of freedom (Schmetterer, 2012, Section I.29). Thus, we can obtain the confidence interval from

$$95\% = \mathbf{P}(T \in [-q_{n-1}, q_{n-1}]) \quad (2.40)$$

$$= \mathbf{P}\left(\frac{\bar{X} - \mu}{S/\sqrt{n}} \in [-q_{n-1}, q_{n-1}]\right) \quad (2.41)$$

$$= \mathbf{P}\left(\mu \in \left[\bar{X} - \frac{q_{n-1}S}{\sqrt{n}}, \bar{X} + \frac{q_{n-1}S}{\sqrt{n}}\right]\right), \quad (2.42)$$

where  $q_{n-1}$  is the 2.5% quantile of Student's t-distribution with  $n - 1$  degrees of freedom. Thus, we can say that with 95% probability the real mean  $\mu$  lies within the interval

$$\left[ \bar{X} - \frac{q_{n-1}S}{\sqrt{n}}, \bar{X} + \frac{q_{n-1}S}{\sqrt{n}} \right]. \quad (2.43)$$

## 2.7. Conclusion

Voice synthesis and transformation as a scientific research topic has a history of over 100 years. In this period we came to gain substantial insight about the voice such that today we have a broad and deep understanding of human hearing, the speech production mechanism and acoustical properties of voice. Nevertheless, voice transformation still finds its limitations for extreme invasive changes resulting in poor audio quality or in inaccurate changes.

In the history of voice research we have seen on several occasions how emerging technology can revolutionize the field and allow approaching the issues from a completely different angle, resulting in new possibilities that were unthinkable before: Electronic circuitry allowed synthesis in small compact devices (Stewart, 1922) and allowed the development of the vocoder (Dudley, 1939). Digital signal processing allowed storing the voice in computers and performing complex transformations on the voice. Deep learning allows modelling stochastic relationships and extracting those from collected data. We are still in the process of rethinking our perspective on voice due to the possibilities of deep learning.

In this chapter we had a look at the underlying theories about how we perceive sound, how human voice is produced and what are its physical properties. We have seen a short introduction to neural networks, the main tool used in this thesis, and how they are used in audio signal processing. As deep learning is relying on its training data, we had a closer look at the available data. Finally, we discussed how we can evaluate synthetic voice by means of perceptive tests.

## 3. State of the art

In this chapter we will discuss the state of the art of voice processing. The central tool for processing the voice is the vocoder which is of particular interest for voice transformation. The vocoder identifies voice recordings with parameters in a parametric space: It consists of two conceptually inverse operations. The *analysis* operation extracts voice parameters from the voice recording. The *synthesis* operation uses the extracted voice parameters to reconstruct the original signal perceptually. We can approach the vocoder from different angles. First, in Section 3.1, we will discuss how different voice properties can be extracted from a voice signal and which analysis methods can be used for that task. Then, in Section 3.2, we will discuss how extracted voice properties can be used to resynthesize voice through different vocoders, and we will discuss the strengths and drawbacks of state-of-the-art vocoders. As the parametric space of a vocoder is used to parametrize the resynthesis operation, the parameters tend to describe the acoustic properties of the signal rather than abstract concepts, like text and melody. To synthesize voice from text or a musical score, more elaborate techniques are necessary, which we will discuss in Section 3.3. Similarly, vocoders can be used to transform some properties of the voice directly but find their limitations quickly as they do not automatically model the relationships between different parameters. Thus, in Section 3.4 we will discuss how a vocoder can be used to transform the voice and when it can be avoided.

### 3.1. Analysis of voice

Signal analysis is the extraction of specific information from a signal. The extracted information can be global, as is the case when detecting the speaker of an utterance. Other information changes over time with the signal and is itself a signal (typically with a lower sample rate). An example of a time-varying voice property is fundamental frequency of voiced speech which is estimated on short signal frames. The variety of signal analysis applications is large and ranges from low level parameter extraction for vocoders and parametric voice synthesis to high level voice processing like singing voice transcription or automated speech recognition.

The extracted signal information parametrizes the underlying signal model. Generally, we assume the source-filter model (Fant, 1981; Markel and Gray, 2013) which we discussed in Section 2.3. Differences in signal models originate from how we model the glottal source and the noise component of the voice. The glottal source can be represented either as a sequence of pulses (Huber and Roebel, 2015) or by a combination of harmonically related sinusoids (Stylianou et al., 1995). The noise component can be modelled non-parametrically (leaving it as is), or through a fil-

ter applied to white noise. In the following subsections, we will present the voice parameters that are used in or related to this thesis.

### 3.1.1. Fundamental frequency

The fundamental frequency, often denoted by  $f_0$  or F0, comes from the assumption that an oscillator generates a quasi-periodic excitation signal with frequency  $f_0$ . In the source-filter model this quasi-periodic excitation is then further transformed to produce the observed signal. This model is not constrained to only voice. Many signals follow this pattern so classical signal processing offers a wide range of  $f_0$  analysis tools. These can be categorized into three different approaches, time-based analysis, that typically work on the auto-correlation function (Sondhi, 1968; Rabiner, 1977; Chevigné and Kawahara, 2002; Mauch and Dixon, 2014), analysis on the (log-)spectrum (Schroeder, 1968; Duifhuis et al., 1982; Hermes, 1988; Sun, 2000; Camacho, 2007), or analysis on the cepstrum (Noll, 1967). An overview over the different classical  $f_0$  analysis techniques can be found in Camacho (2007, Chapter 2) and a qualitative study was done by Babacan et al. (2013a).

Recently, neural networks have been used to create domain specific fundamental frequency estimators for musical instruments (J. W. Kim et al., 2018; Singh et al., 2021) and speech (Ardaillon and Roebel, 2019; Roebel and Bous, 2022). As these methods are trained on and applied to specific types of signals (specific musical instruments or voice) they can use additional information, e.g. from the timbre, to improve the accuracy or resolve ambiguities, which could not be done in the general case. The pitch estimation works on raw audio (J. W. Kim et al., 2018; Ardaillon and Roebel, 2019) as well as on the mel-spectrogram (Roebel and Bous, 2022) and might be trained on any other representation of voice.

### 3.1.2. Glottal pulse signal

An alternative characterization of the glottal excitation signal is through the glottal pulses. Here, we don't necessarily assume periodic excitation, but rather simply a set of pulses (with position and possibly shape). The characteristic pointy tip of the glottal closure instant  $t_e$  (see Fig. 2.8 in Section 2.3.2, Page 19) allows us to define the pulse position in terms of the glottal closure instant (GCI). Using a specific glottal pulse model allows parametrizing the glottal pulse shape.

The behaviour of the vocal folds can be analysed by electroglottographic (EGG) measurements, where the electric resistance through the glottis is measured by electrodes placed on the laryngeal prominence. The electric resistance through the glottis depends on the contact area of the vocal folds. Thus, during phonation an oscillation can be measured and the glottal closure instant can be obtained through peak-picking in the electroglottographic derivative. Using parallel recordings of the sound- and electroglottographic signal, we can annotate the voice signal with the glottal closure instants.

### Pulse position

While electroglottographic measurements allow non-intrusive analysis of glottal activity, the requirement for a specific measurement device makes the procedure uncommon. Thus signal processing based analysis methods have been developed to extract the glottal closure instants from the audio recordings. Glottal inverse filtering (Alku, 2011) can be used to estimate the glottal pulse signal from which the glottal closure instants can be extracted. Typically, this is done using the linear prediction residual (Naylor et al., 2007; Prathosh, Ananthapadmanabha, et al., 2013), the zero-frequency filtered signal (Murty and Yegnanarayana, 2008) or through smoothing (Drugman and Dutoit, 2009).

Unlike the fundamental frequency, the glottal closure instants cannot be extracted from audio so easily by means of classical signal processing (Babacan et al., 2013b). Thus, we still observe lots of research activity about GCI extraction in recent years which use more and more data-driven approaches. S. Yang et al. (2018) and Reddy et al. (2018) use a convolutional neural network to classify the peaks in a low-pass filtered speech signal and similar representation derived from low-pass filtering. The signal pre-processing is addressed by Goyal et al. (2019) by training a neural network to create a representation best suited for the classification task upstream. Deepak et al. (2019) and Prathosh, Srivastava, et al. (2019) use generative adversarial network and adversarial approximate inference respectively to generate synthetic electroglottographic measurements, which measures the vibrations of the vocal folds using additional hardware. Ardaillon and Roebel (2020) use the PaN vocoder (Ardaillon, 2017) to create a synthetic database with known pulse signal and a convolutional neural network (based on the  $f_0$  estimator of Ardaillon and Roebel (2019)) to obtain the glottal pulse signal from raw audio. A refinement of Ardaillon and Roebel (2020) will be presented in Chapter 4.

Glottal closure instant detection can be used for speech analysis and processing (Drugman, Alku, et al., 2014) such as analysis of vocal disorders (Reddy et al., 2018; Deshpande and Manikandan, 2018), formant estimation (Anand et al., 2006) or speech synthesis (Drugman and Dutoit, 2012).

### Pulse shape

Glottal closure instant detection is concerned with finding the time instants of the glottal pulse but generally make no statement about the pulse shape. Some glottal closure instant extraction methods, like Deepak et al. (2019), Prathosh, Srivastava, et al. (2019), and Ardaillon and Roebel (2020), generate the non-parametric pulse signal (the latter) or a electroglottographic signal (the former two) which could be used to infer some information about the pulse shape. In fact many approaches to extract the glottal pulse signal from the voice signal have been proposed, known as *glottal inverse filtering* (Alku, 2011). However, obtaining the glottal pulse signal is only half the work. The estimated glottal pulse signal has to be paired with an underlying pulse model to provide a meaningful parametrization. A method to obtain the  $R_d$  parameter from the simplified LF-model (see Section 2.3.2 and Fig. 2.9, Pages 18 and 22 resp.) is provided by Huber, Roebel, and Degottex (2012) which uses phase

minimization (Degottex, Roebel, et al., 2010).

### 3.1.3. Voice timbre

In the classical vocoder model, the glottal source signal and the noise component are filtered by the **vocal tract filter**. We can characterize the result by its spectral envelope. The spectral envelope can be represented by **linear predictive coding (LPC)** (Atal and Hanauer, 1971) which yields filter coefficients that in effect describe the spectral envelope of the signal. More recently the spectral envelope has been parametrized in the frequency domain using the **TrueEnv** algorithm (Roebel and Rodet, 2005) or the **Cheap Trick** algorithm (Morise, 2015). The spectral envelope can be used to whiten the signal, by inverse-filtering the original signal by the spectral envelope. **Parametric vocoders** use the **LPC** coefficients or spectral envelope for resynthesis, by filtering a white excitation. In the past the spectral envelope was also used for speech recognition (Morgan et al., 2005).

Another characterization of the vocal tract and the voice timbre is through the formant frequencies. Formants are the resonance frequencies of the vocal tract and appear as peaks in the spectral envelope. The formant frequencies can be found using group delay (Murthy and Yegnanarayana, 1991; Anand et al., 2006), using peaks of the smoothed spectrum or spectral envelope (Schafer and Rabiner, 1975) or using the linear prediction spectrum by picking the appropriate poles or peaks (McCandless, 1974). Performance can be improved by incorporating the bandwidth of the pole (Evanini et al., 2009).

### 3.1.4. Noise component

Due to its random nature, the noise component can be difficult to describe precisely. To human ears, two different noise signals are almost indistinguishable if the spectral density is the same. Thus, the spectral density could be used to parametrize the noise source. In practice, however, obtaining the spectral density of the noise source, is not easy as it is often masked by sinusoidal components and may change rapidly, violating the necessary stationarity assumptions. The spectral envelope contains an implicit description of the noise component during unvoiced speech. During voiced speech some information about the noise component of the signal may be inferred from the spectral envelope in the higher frequency bands, where the noise component is stronger than the partials of the voiced component. In both cases, however, it is difficult to disentangle the timbre of the voiced component and the noisy component in the spectral envelope alone.

Assuming no other background noise in the signal, the noise component can be obtained by subtracting the deterministic component (Laroche et al., 1993). This approach relies heavily on the quality of the deterministic signal component as a faulty estimation will leave deterministic components in the estimated noise source. This is less of a problem in pitch-synchronous methods (Moulines and Charpentier, 1990; Stylianou et al., 1995) where analysis windows are very small. Alternatively, subtraction can be improved by resampling the time axis depending on the  $f_0$ , to

obtain a deterministic excitation with constant pitch (Kawahara, 2006; Huber, 2015, Sec. 6.3.1.3). Alessandro, Yegnanarayana, et al. (1995) use an iterative procedure in the frequency domain of the **LPC** residual to estimate the noise source directly. Here the noise component is estimated first, and the deterministic component can be obtained by subtracting the noise component.

The extracted noise component can be used to estimate a filter or a spectral envelope to obtain a parametrized description of the noise source. The noise component is used in vocoders and synthesizers. During resynthesis, the original noise source might be replaced by white noise. In breathy voice, however, we observe that a large part of the noise is modulated noise (as discussed in Section 2.3.1). Many models account for this effect by modulating the noise source with the fundamental frequency (Stylianou et al., 1995; Mehta and Quatieri, 2005; Degottex, Lanchantin, Roebel, et al., 2013).

Other methods to extract the noise component have been discussed by and Degottex, Lanchantin, and Gales (2016).

## 3.2. Vocoder

A vocoder is an auto-encoder on the voice signal. Any form of analysis-synthesis cycle can be seen as a vocoder and in the history of vocoders we have seen the vocoder principle taking many forms. The first vocoder was invented by Dudley (1939) who also coined the term ‘vocoder’. Dudley’s vocoder was able to convert an analogue speech signal into an  $f_0$  signal and a multidimensional signal describing the timbre. These parameters could then be used to resynthesize the original phrase.

Generally the vocoder maps between raw-audio and the parametric space of the vocoder. In most cases a vocoder performs lossy encoding and perfect reconstruction is not guaranteed even if the vocoder parameters are not changed. Thus, any application using a vocoder is necessarily bounded by the capability of the vocoder. Vocoder can be categorized by their parametric space. We distinguish between parametric vocoders, like Dudley (1939), that define the signal by some acoustic parameters, and non-parametric vocoders, like the phase vocoder or pitch synchronous overlap and add (PSOLA) (Moulines and Charpentier, 1990), where the signal is cut into frames but not compressed through a parametric description. Furthermore, we can distinguish between fixed rate parametrizations where the parametric space is a (multidimensional) signal of fixed sample rate (typically lower than audio rate), and pitch-synchronous methods (Moulines and Charpentier, 1990) where the step size of analysis frames depends on the pitch. Finally, we need to distinguish between neural vocoders and classical, DSP-based vocoders. An implementation of a vocoder with neural networks allows for parametric spaces fundamentally different from classical parametric space, which allows higher level signal descriptions.

The main three applications of vocoders are transformation, synthesis, and compression. Vocoder designed for voice compression are used in telecommunication to reduce bandwidth and can achieve much higher compression rates than general

purpose audio compression.<sup>1</sup> The main design objective is to achieve a good quality to bitrate ratio. The requirements for vocoders designed for transformation and synthesis are very different. The main design objective is to achieve good resynthesis quality. Furthermore, when generating vocoder parameters through synthesis or transformation it is likely that the vocoder parameters are not generated perfectly, and different parameter groups may contain inconsistent values. Thus, A secondary objective is to achieve robustness towards inconsistent parameter values. We will discuss properties and applications of the vocoder in more detail in [Chapter 5](#). In this section we will discuss the different types of existing vocoders used in voice synthesis and transformation.

### 3.2.1. Pitch synchronous overlap and add

Idea of pitch synchronous overlap and add (PSOLA) is to place an analysis window around each period of the deterministic excitation. If the window placement is correct, consecutive analysis windows are phase-synchronous which allows comparisons in the time domain. Multiple periods are used in one window, typically around 2 to 4 in ([Moulines and Charpentier, 1990](#)). Signals can be stretched in time by inserting additional, repeated frames.

Pitch-synchronous vocoders can be non-parametric ([Moulines and Charpentier, 1990](#)). In this case the frames are treated in the time domain. The non-parametric time-domain PSOLA can be used to change the pitch of a signal, by resampling the audio frames and then adding or removing additional frames to compensate for the change in duration.

Furthermore, the short audio-frames can be parametrized as is done in with harmonic plus noise models (HNMs) ([Stylianou et al., 1995](#)). The HNM assumes that the audio-frames consist of harmonically coupled sinusoids and noise. The amplitudes and phases of sinusoids are estimated through an underdetermined linear set of equations and the noise is obtained through subtraction of the harmonic part. The parametrization of the harmonics allows more flexible transformations than the non-parametric approach.

### 3.2.2. Non-parametric vocoders

Non-parametric vocoders are based around the idea of splitting the voice signal in small analysis windows. The analysis windows can be left in the time-domain, as is the case in PSOLA ([Moulines and Charpentier, 1990](#)) or transformed into the frequency domain as is done by the phase vocoder. To change the durations, additional frames have to be added or removed. In the case of the phase vocoder intermediate frames can be generated by interpolating the amplitudes and adjusting the phase. In general the phase vocoder does not preserve the shape of the sound waves. This creates audible degradation in voice signals ([Quatieri and McAulay, 1992](#)). Thus, until

---

1. Typical compression performed by vocoders used in telecommunication achieve bit-rates between 16 kbit/s to 40 kbit/s ([International Telecommunication Union standard, ITU-T G.726](#)).

the shape-invariant phase vocoder (Roebel, 2010), the phase vocoder was generally used only for transformation of sounds other than voice.

In the time domain, we face the problem that we cannot easily create additional frames from the neighbouring frames. This is addressed by the aforementioned pitch-synchronous step-size. Since the audio frames in PSOLA have a step size of exactly one period, the phases align in the harmonic component. This way individual frames can be dropped or repeated without creating discontinuities in the phase. Since the step size of PSOLA is relatively low, durations can be controlled with sufficient flexibility only by inserting and removing frames.

### 3.2.3. Source-filter parametric vocoder

The source-filter model can be used to parametrize a voice signal by the frequency of the excitation source  $f_0$ , the timbre, the noise source, and possibly additional parameters. A periodic excitation signal is filtered and mixed with noise. The first such approach was through the use of linear predictive coding (LPC) (Atal and Hanauer, 1971). The LPC approach parametrizes the timbre by the filter linear prediction filter. The linear prediction residual can be further parametrized through fundamental frequency and additive noise.

While linear predictive coding has become out of fashion for voice transformation vocoders, the source-filter principal has been used in many recent vocoders. Popular such vocoders are STRAIGHT (Kawahara, 2006; Kawahara et al., 2008), WORLD (Morise et al., 2016), SVLN (Degottex, Lanchantin, Roebel, et al., 2013; Huber and Roebel, 2015) or Pulse and Noise (PaN) (Ardaillon, 2017).

### 3.2.4. Neural vocoder

Neural vocoder allow exploring novel possibilities to parametrize the voice. For example, H.-S. Choi, J. Yang, et al. (2022) use Wav2Vec (Baevski et al., 2020) features to describe the phonetic content and unsupervised speaker embeddings to define the timbre of voice alongside the conventional vocoder parameters  $f_0$  and signal energy. Other neural vocoders completely omit conventional parameters and use the mel-spectrogram as their parametric space instead (see Section 3.2.6). Furthermore, neural networks can be used in vocoding to model the acoustic dependencies within the source filter model.

### 3.2.5. Neural source-filter vocoders

DSP-based source-filter vocoders largely treat the glottal source and the noise excitation independently due to the limited possibilities of DSP. In reality, however, all these properties depend on each other. Therefore, it is desirable to generate the glottal source signal  $e$  and the noise signal  $n$  together to take their co-dependencies into account. Deep learning allows modelling these dependencies, and we find a lot of recent work, where glottal source and noise source are generated by neural networks (Song et al., 2019; Hwang et al., 2020; Oh et al., 2020). Song et al. (2019) use a WaveNet (Oord, Dieleman, et al., 2016) to generate the excitation signal by

conditioning the WaveNet on classical vocoder parameters. Oh et al. (2020) use a WaveGlow (Prenger et al., 2019) to generate the excitation signal. H.-S. Choi, J. Lee, et al. (2021) generate the spectrograms of source and filter separately in their synthesis module. X. Wang et al. (2019) on the other hand generate an excitation from a classical sinusoids-plus-noise model and replace the filter operation by a neural network.

### 3.2.6. Neural vocoder using the mel-spectrogram

Mel-inversion aims at reconstructing a signal from its mel-spectrogram. The first occurrence of a neural mel-inverter was used by J. Shen et al. (2018) in their second version of the text-to-speech engine Tacotron (Y. Wang et al., 2017). That publication synthesized mel-spectrograms from character sequences transcribing the target utterance and then calculated the raw audio using a WaveNet (Oord, Dieleman, et al., 2016) conditioned on the synthesized mel-spectrograms. The audio quality was excellent, but the procedure had some significant drawbacks:

1. Due to its recurrent nature the WaveNet can be trained efficiently, but synthesis can occur only one audio sample after the other. Thus, inference speed is extremely slow (Oord, Y. Li, et al. (2017) report 172 audio samples per second for the classical WaveNet on a Titan V100 GPU, thus 140 times slower than real-time for 24 kHz audio).
2. The mel-inverter is speaker-dependent thus can only be used for a specific voice. Furthermore, the required amount of data to train this mel-inverter is very large, J. Shen et al. (2018) used 24 h of a single speaker. This amount of data is normally not available for a single speaker (see Table 2.2).

The first problem can be addressed by the parallel WaveNet approach (Oord, Y. Li, et al., 2017) where a parallel version of the WaveNet is inferred from a pre-trained classical (non-parallel) WaveNet. This procedure allows fast inference speed in production<sup>2</sup> but is impractical for research purposes due to its very long and complex training procedure.

The above work introduced many possible applications for voice processing on the mel-spectrogram but due to the difficulties above, the mel-inversion was a restrictive element that limited the results. Therefore, a lot of work happened that tried to address these concerns at many places at the same time using different approaches. Lorenzo-Trueba et al. (2018) used recurrent neural networks to create a universal (speaker independent) mel-inverter, Prenger et al. (2019) used generative flows to create a faster mel-inverter. MelGAN (K. Kumar et al., 2019) used generative adversarial network (GAN) based on convolutional neural network (CNN) but was not a universal vocoder, only a single speaker could be properly synthesized with one model. MelGAN has been extended by Jang et al. (2020) using multi-resolution spectrogram discriminators to allow universal vocoding and by G. Yang et al. (2021)

---

2. Parallel WaveNet was developed by a large tech-company that deployed this neural text-to-speech engine on its consumer devices.

using synthesizing the audio signal in multiple bands using pseudo quadrature mirror filter decomposition to reduce model complexity and further increase inference speed. HiFi-GAN (Kong et al., 2020) use Multi-Receptive Field Fusion blocks in the generator of their generative adversarial network. GELP (Juvela et al., 2019) uses linear prediction and adversarial training. iSTFT-Net (Kaneko, Tanaka, et al., 2022) produces a complex valued STFT from the mel-spectrogram that can be inverted using overlap-add. Big-VGAN (S.-g. Lee et al., 2023) uses a large generative adversarial network trained on LibriTTS, and the authors show examples where it can even synthesize music, even though it has never been trained on music (though as discussed in Section 2.5, some music might have been present as background noise). Roebel and Bous (2022) use a source filter model and adversarial training and will be discussed in Section 5.3.

### 3.3. Synthesis

Voice synthesis, in the most general sense, is the generation of audio that contains human voice. More specifically when we talk about synthesis as a research question itself we mean *text-to-speech* in the domain of speech or *score-to-singing* in the domain of singing. The former transforms a character sequence representing written text into a recording of someone reading the text. The latter transforms a musical score including notes and lyrics into a recording of someone singing the piece written in the score. Often we would like to have some control over how a text is read, or a piece is performed. Thus, in *expressive* synthesis we use additional control inputs to define the expression and style of the performance. The typical synthesis methods are described below:

#### 3.3.1. Concatenative synthesis

To synthesize voice from a phoneme sequence concatenative synthesis can be used. For a concatenative synthesizer we need a database containing all elementary building blocks, called *units*, of a language. To synthesize a phrase, the text is converted into a sequence of units and the respective samples are concatenated to create a speech signal. Depending on the application different types of units can be chosen, e.g. phonemes, diphones, syllables or whole words. Larger units yield more natural speech but require much larger databases to cover all possible units. In singing, a popular choice is to use diphones (Moulines and Charpentier, 1990) that is, the transitions between two phonemes, as they provide natural transitions between phonemes. Examples of speech synthesizers using concatenative synthesis are Moulines and Charpentier (1990), Sagisaka et al. (1992) or Hunt and Black (1996). Timing and speech melody can be imposed by time-stretch and pitch-shifting respectively.

The concatenation can happen using the audio samples or using the parameters of a parametric vocoder. In the latter case only the parameters that are predominantly responsible for the phonetic content need to be concatenated, while other parameters, in particular the  $f_0$ , can be freely chosen. This is particularly useful in singing synthesis where the  $f_0$  is dictated independently of the phonetic content. Since to

impose the  $f_0$  a vocoder is needed regardless whether the concatenation occurs on the audio samples or on the vocoder parameters, the parametric concatenative approach is more efficient. Examples of parametric concatenative singing synthesizers are Umbert, Bonada, and Blaauw (2013), Bonada et al. (2016) and Ardaillon (2017).

### 3.3.2. Statistical parametric speech (singing) synthesis

Similar to parametric concatenative synthesis, statistical parametric speech synthesis (SPSS) (Zen, Tokuda, et al., 2009) aims to synthesize vocoder parameters for a given utterance. In the case of statistical parametric speech synthesis, however, the vocoder parameters are not found by concatenating pre-recorded units of speech, but rather statistical models are used to generate the vocoder parameters. The statistical model is typically obtained from hidden Markov models (HMM) (Yoshimura, 2002) or more recently by neural networks (Zen, Senior, et al., 2013).

The statistical model obtained for statistical parametric speech synthesizers is much smaller than the whole dataset required for concatenative synthesizers. Thus, disk space requirements are much smaller for statistical parametric speech synthesizers. Likewise, the statistical models are more flexible as the unit-selection approach of concatenative synthesizers and allow changing ‘voice characteristics, speaking styles and emotions’ (Zen, Tokuda, et al., 2009). The increased flexibility, however, is bought at the cost of synthesis quality. The main weak-points of statistical parametric speech synthesizers are the statistical model and the underlying vocoder.

### 3.3.3. Neural parametric synthesis

Speech- and singing synthesis came to rely more and more on deep learning. Early use of neural networks can be found in statistical parametric speech synthesis, where the statistical model is implemented through a neural network (Zen, Senior, et al., 2013; Zen, Agiomirgiannakis, et al., 2016; Blaauw and Bonada, 2017). These approaches use neural networks to generate parameters of a classical vocoder. Since then, neural networks were used to replace other components of the synthesis system such that today almost all operations rely on neural networks. Neural voice synthesis typically occurs in two steps. First the high-level representation is transformed into an intermediate acoustic representation and then the audio signal is generated from the acoustic representation. The intermediate representation can be vocoder parameters of a parametric vocoder (Zen, Senior, et al., 2013), the spectrogram (Y. Wang et al., 2017) or the mel-spectrogram (J. Shen et al., 2018; Ping et al., 2018). With mel-inversion techniques becoming more and more efficient, the latter has become a popular choice in text-to-speech and related domains.

### 3.3.4. Neural end-to-end synthesis

Neural end-to-end synthesizers generate audio waveforms from the text or score without intermediate representation. Mel-spectrogram based speech synthesizers

can be turned into end-to-end systems by jointly training mel-synthesis and mel-inversion or fine-tuning the mel-inverter to the synthetic mel-spectrograms, as suggested by J. Shen et al. (2018). In this case we obtain an end-to-end system that is guided through the use of the mel-spectrogram as an initial intermediate representation. The advantage of this end-to-end setup is that the mel-inverter can be trained to account for errors produced by the mel-generator.

Donahue et al. (2021) present an end-to-end system that requires no pre-defined intermediate representation at all, using a generative adversarial network. Fast-Speech 2 (Ren et al., 2021) use a mel-spectrogram decoder on an intermediate layer that allows creating a loss based on the mel-spectrogram during training but is disregarded during inference, to avoid generating the mel-spectrogram directly. J. Kim et al. (2021) use normalizing flows and variational auto-encoder (VAE) to guide the end-to-end speech synthesis with the linear spectrogram.

## 3.4. Transformation

Transformation can be seen as a special kind of synthesis that reuses some information extracted from an already existing recording together with some external information. For example in pitch transformation we want to synthesize voice with the phonetic content and the timing of the original phrase together with a freely chosen fundamental frequency. Which properties should be used from the original recording, which properties are imposed explicitly, and which properties are inferred, depends on the specific application.

### 3.4.1. Modelling dependencies

We have already discussed briefly in Section 3.2 how pitch and duration can be changed using different vocoders. Using a vocoder it is not difficult to resynthesize a signal with a different pitch. The difficulty lies in designing transformations where the result still sounds natural.

Most vocoders model the acoustic dependencies of pitch changes, that is how to change the signal without introducing artefacts, but not how the voice of a speaker or singer changes depending on the change. This can work reasonably well for small changes but can quickly create unrealistic results if the new  $f_0$  values are too different from the original. The problem is that the other voice properties also implicitly depend on the  $f_0$ . Thus, any serious attempt of voice transformation needs to address these dependencies as well, either by reducing the dependencies through re-parametrization or by also regenerating the other parameters from the new  $f_0$  values. These points have been addressed by Farner et al. (2009), Degottex, Lanchantin, Roebel, et al. (2013), and Ardaillon (2017). The PaN vocoder (Ardaillon, 2017) reduces the dependency of the timbre parameter on the fundamental frequency by using the vocal tract filter instead of the spectral envelope of the full signal. The vocal tract filter depends much less on the fundamental frequency than the spectral envelope since the spectral colour of the glottal pulse signal which highly depends on the  $f_0$  is removed. Thus, this parametrization allows a higher

range of  $f_0$  changes where the result is still acceptable but still finds its limitation, since the vocal tract itself also depends on the pitch.

Other voice properties that are not directly exposed by vocoders can be manipulated similarly: the relationship between the vocoder parameters and the property that is to be changed needs to be modelled. Degottex, Roebel, et al. (2011); Degottex, Lanchantin, Roebel, et al. (2013) modify the breathiness in voice by changing the pulse shape parameter from the underlying pulse model using a parametric vocoder. Huber (2015) changes the voice identity by adjusting the pitch and timbre to convert between source and target speaker. Gentilucci et al. (2018); Gentilucci et al. (2019) and Liuni et al. (2020) created a rough voice effect for singing voice by introducing sub-harmonics through amplitude modulation.

Voice level manipulations can be achieved by modification of timbre and glottal source (Perrotin and Alessandro, 2016). To increase the perceived voice level, Perrotin and Alessandro (2016) suggest increasing number of harmonics in the glottal source by ‘increasing the source signals impulsiveness using time distortion’ (Perrotin and Alessandro, 2016) and adjust the spectral envelope. The spectral envelope can be adjusted through spectral modelling (Alessandro and Doval, 1998; Alessandro and Doval, 2003) which increases the spectral tilt with increasing voice level. In concatenative synthesis spectral morphing (Turk et al., 2005; Defez et al., 2013) can be used to create intermediate values of voice level between two recordings of different voice levels. Furthermore, voice level affects formant positions, which has been addressed by Molina et al. (2014) by shifting formants according to observed statistics. Ardaillon and Roebel (2017) model the effect of increased mouth opening, due to increased vocal effort, on the vocal tract through the first formant  $f_1$ . Raitio et al. (2020) use the spectral tilt as a control parameter to affect the voice level in text-to-speech.

### 3.4.2. Disentanglement of parameters

The re-parametrization of the spectral envelope  $E$  in the PaN vocoder through the vocal tract filter  $V$  and the glottal pulse parameter  $R_d$ , can be seen as adjusting the spectral envelope to the new pitch. The dependency of the spectral envelope on the pitch is modelled through the pulse shape, which is changed accordingly, while the vocal tract filter is left unchanged. Another perspective is to view this as an attempt to remove the dependency of the fundamental frequency from the other vocoder parameters. The removal of dependency of one parameter from another is called *disentanglement*. In general, disentanglement is a re-parametrization of a parameter set such that its parameter subsets become independent of each other.

Disentanglement is a fundamental discipline in representation learning (Y. Bengio et al., 2013). The goal of disentanglement is to find a representation of the data using independent factors. Due to its many applications it is a research field of high interest (E. Bengio et al., 2017; Higgins, Amos, et al., 2018; X. Liu et al., 2022). Disentanglement using neural networks has been used in many domains; for example, in image processing for face attributes (Lample et al., 2017), face pose (Peng et al., 2017), face identity (Nitzan et al., 2020); in anomaly detection for domain changes

(Dohi et al., 2022) and in voice conversion for voice identity (J.-X. Zhang et al., 2019; Qian, Yang Zhang, Chang, X. Yang, et al., 2019; Qian, Jin, et al., 2020; Qian, Yang Zhang, Chang, Hasegawa-Johnson, et al., 2020), accent (Z. Wang et al., 2021), fundamental frequency (Qian, Jin, et al., 2020; Qian, Yang Zhang, Chang, Hasegawa-Johnson, et al., 2020), speech duration (Qian, Yang Zhang, Chang, Hasegawa-Johnson, et al., 2020) and gender (Benaroya et al., 2023).

### Measuring disentanglement

Disentanglement is difficult to quantify as there is no canonic way to define it. To evaluate different disentanglement methods, however, numerous metrics have been proposed measuring different properties of the disentangled space but sometimes with contradicting indications (Sepliarskaia et al., 2019; Carboneau et al., 2022). The disentanglement metrics are divided by Carboneau et al. (2022) into three categories:

**Intervention-based** metrics measure disentanglement by evaluating how points in the code space generated from the same factor relate to each other. The  $\beta$ -VAE (Higgins, Matthey, et al., 2017), Factor VAE (H. Kim and Mnih, 2018) and R-Factor VAE (M. Kim et al., 2019) belong to this category.

**Predictor-based** metrics rely on a predictor that is trained to predict the factors from the code. This predictor is analysed to quantify the disentanglement. Disentanglement, Completeness, and Informativeness (DCI) (Eastwood and C. K. Williams, 2018), the Explicitness Score (Ridgeway and Mozer, 2018), and the Attribute Predictability Score (A. Kumar et al., 2018) belong into this category.

**Information-based** metrics measure the disentanglement by the mutual information between the factors and the code. Most metrics in this category rely on the Mutual Information Gap (R. T. Chen et al., 2018; Do and Tran, 2020) and derived quantities.

More details and in-depth comparisons between these metrics can be found in Carboneau et al. (2022).

### Bottleneck auto-encoder

Of particular interest for this work is AutoVC Qian, Yang Zhang, Chang, X. Yang, et al. (2019); Qian, Jin, et al. (2020). AutoVC is a bottleneck auto-encoder which serves as a basis of Chapter 6. The bottleneck auto-encoder of Qian, Yang Zhang, Chang, X. Yang, et al. (2019) disentangles the speaker dependent information from the voice through an auto-encoder structure. The encoder produces a speaker-independent representation through the use of an information bottleneck that forces the encoder to prioritize important information and to remove redundant information from its output (the latent code). By supplying the decoder with the speaker identity, the speaker identity becomes redundant in the latent code (the output of the encoder) and is thus removed. Qian, Jin, et al. (2020) follows a hybrid approach, adjusting the

fundamental frequency through remapping first and second order moments but using the auto-encoder for the remaining information. A further development of this approach is given by (Qian, Yang Zhang, Chang, Hasegawa-Johnson, et al., 2020) where speech is split into rhythm, pitch, timbre, and content simultaneously.

### 3.4.3. Higher level voice conversion

In this thesis we introduce a framework that allows transforming low-level, volatile properties of voice. This framework is used to transform pitch and intensity.

Voice conversion has also been applied to more global parameters like voice identity and social attitude. While the timescale of the property of interest is different, we find many similarities, that allow ideas from these applications to be borrowed in our applications and vice versa. It is thus interesting to discuss these related applications here briefly.

#### Identity transformation

In recent years voice identity conversion has become a popular research activity. With the use of neural networks, first introduced by Desai et al. (2009), modelling the influence of the voice identity on the parametric space of a vocoder became feasible. Today voice conversion is typically done using a neural vocoder based on the mel-spectrogram (Kameoka et al., 2018; J.-X. Zhang et al., 2019; Qian, Yang Zhang, Chang, X. Yang, et al., 2019; Proszewska et al., 2022; Y. A. Li et al., 2023) while end-to-end approaches are becoming popular as well (Nguyen and Cardinaux, 2022; Xie et al., 2022; H.-S. Choi, J. Yang, et al., 2022).

There are two different types of properties that constitute voice identity, physical constraints and habitual features (Bous, Benaroya, et al., 2022). Physical constraints, like the size and shape of the vocal tract, influence the voice's acoustic features. Habitual features are effects that happen not because of the physical setup of the vocal tract but rather from habits. Examples of this are accent, speech rate, speech melody and the way how the speaker accentuates. Voice that defies these constraints is not impossible but improbable for the given speaker.

Which parameters need to be transformed depends on the application. Thus, we see a wide range of sub-disciplines of voice conversion: From complete conversion (Kameoka et al., 2018; J.-X. Zhang et al., 2019) where only the phonetic content is preserved, and the speech is resynthesized from that, to voice reenactment (Bous, Benaroya, et al., 2022) which preserves all habitual features to allow actors to impersonate someone in film and theatre. In between we find voice conversion with preserved timing (Qian, Yang Zhang, Chang, Hasegawa-Johnson, et al., 2020; Ferro et al., 2020) that allows temporal alignment or conversions that preserve the speaker but change their habitual features like accent conversion (Zhao et al., 2018). In Chapters 6 and 8 we will observe that the pitch transformation introduced in this thesis is also implicitly capable of changing the voice identity.

### Transformation of emotion and attitude

Another type of voice conversion is the transformation of emotion and social attitude. In this flavour we want to preserve the voice identity and phonetic content but change the emotion or attitude behind the utterance. While social attitudes influence many parameters of speech production (Salais et al., 2022) its most prominent influence is on the fundamental frequency (F. Chen et al., 2004; Puts et al., 2007; Goupil et al., 2021). Emotional voice conversion can be achieved by converting the speech melody encoded through the fundamental frequency  $f_0$ , and several studies have been proposed to model the dependency between  $f_0$  and emotion (Veaux and Rodet, 2011; Luo, Takiguchi, et al., 2016; Luo, Jinhui Chen, et al., 2017; C. Robinson et al., 2019). More holistic approaches can be found in K. Zhou et al. (2021); K. Zhou et al. (2022) using generative adversarial network (GAN) and Moine, Obin, and Roebel (2021) using transformer networks.

## 3.5. Conclusion

This chapter has provided an overview of the state of the art in voice signal processing technology related to the transformation of pitch and voice level. The different analysis methods were introduced, that allow us to obtain the voice properties necessary to model the voice sufficiently (Section 3.1). Then we have discussed how these extracted properties are used in vocoders that provide a parametric space of the voice (Section 3.2). The vocoder was a central tool in voice transformation in the past, and it remains at the heart of many voice transformation systems through the latest generation of neural vocoders. We have discussed applications of the vocoder such as voice synthesis (Section 3.3) and voice transformation (Section 3.4). Regarding voice transformation we have discussed different approaches, using classical vocoders, and more recent approaches that rely more and more on deep learning. In this context briefly discussed disentanglement, which is the central tool of voice transformation used in this work. We have discussed several applications of voice transformation, including pitch transformation and other voice properties, voice identity conversion and social attitude conversion.

## Motivation of research work

The past three years have been fruitful not only for my own research but for the whole voice-research community. Many of the problems we observed at the onset of this thesis have been adequately addressed and at least partially solved. When I started this work in 2019 there were no universal vocoders available. In particular, singing voice was not supported adequately with neural vocoders. In 2019, it was not at all clear whether the mel-spectrogram contained enough information about the voice that it could serve as a sufficient parametrization of the voice in general. Mel-inversion was still very costly, and it was not certain that cheap and high quality mel-inversion was possible. Thus, in this thesis we begin our investigation of the neural vocoder with two complementary approaches.

### *3. State of the art*

---

1. We investigate the possibilities of neural vocoding using classical vocoder parameters with an analysis-synthesis framework in [Chapter 4](#). While this framework has turned out to be too impractical for voice transformation, we have gained valuable insight with this approach about neural vocoding and the modelling capacities of neural networks with respect to voice. The resulting vocoder exposes glottal pulse positions which is useful for voice analysis and might be used in modelling rough voice.
2. In parallel, in the context of the [ARS](#) project, we investigate into possibilities of overcoming the difficulties, that were present at the time, for [neural vocoders](#) based on the [mel-spectrogram](#). The findings of that investigation are found in [Chapter 5](#). There we introduce the vocoder than will provide the voice representation used in the subsequent chapters [Chapters 6](#) and [7](#).

## 4. Glottal closure instance extraction

We begin this thesis with a study on glottal closure instant extraction. This was the first investigation in the context of this thesis and resulted in its first publication (Bous, Ardaillon, et al., 2020). The approach extends recent work of Ardaillon and Roebel (2020) who worked in the same research lab and co-authored with me on the publication that this chapter is based on. Since at the heart of the approach is a neural vocoder, this approach serves well as a starting point to the research around neural vocoding. We will use the insights gained from this study again in Chapter 5 where we will develop a neural vocoder designed for voice transformation.

This chapter been extracted from Bous, Ardaillon, et al. (2020) which has been published in the conference proceedings of the 2020 European Signal Processing Conference (EUSIPCO) and the article and the inference code together with a pre-trained model can be found under the links below.



Article



Inference code

Article: <https://arxiv.org/abs/2003.01220>  
Inference code: <http://thesis.fnab.xyz/ch4/code>

### 4.1. Introduction

Classical parametric neural vocoders typically use the fundamental frequency as one of their vocoder parameters (Kawahara, 2006; Degottex, Lanchantin, Roebel, et al., 2013; Morise et al., 2016; Ardaillon, 2017). This is motivated by the source filter model (Fant, 1981; Markel and Gray, 2013) and allow easy modelling the periodic excitation signal as an interpretable parameter. Using the fundamental frequency, however, finds its limitations in rough speech and vocal fry, which is generated by irregular pulses. In this situation, the fundamental frequency does not exist and vocoders that rely on this parametrization do not reproduce well the voiced excitation with irregular pulses.

An alternative representation of the voiced excitation is through the pulse positions rather than through the fundamental frequency. Some parametric vocoders, like PaN (Ardaillon, 2017), use the pulse positions, derived from the fundamental frequency, internally, but rely on the presence of regular pulses to work properly. To better model speech with irregular glottal pulses, the glottal pulse positions need

to be estimated individually. Unlike fundamental frequency estimation, however, the pulse detection is still unreliable, such that voice transformation of speech with irregular pulses is still uncommon in parametric vocoders.

#### 4.1.1. Glottal closure instant

The position of a glottal pulse can best be characterized by the glottal closure instant (**glottal closure instant (GCI)**) due to its prominent appearance in the glottal flow derivative. The glottal closure instant is the time instant where the glottal pulse transitions from the open phase to the closing phase (see Fig. 2.8, Page 19). This transition is marked by a sharp peak in the glottal flow derivative and can often be recognized visually in the time domain signal. A more detailed introduction to the state of the art on glottal closure instant detection can be found in Section 3.1.2.

#### 4.1.2. Overview

Recently, Ardaillon and Roebel (2020) used a neural network to generate a synthetic glottal source signal (based on Fant (1995)) by using the PaN vocoder (Ardaillon, 2017) to create a synthetic database with perfect annotation (J. W. Kim et al., 2018; Ardaillon and Roebel, 2019). Using the annotated synthetic database, Ardaillon and Roebel (2020) were able to train the glottal source estimator by matching the synthetic audio to the annotated glottal pulse signal. The resulting estimator could then be applied to real speech.

Here, we expand the ideas of Ardaillon and Roebel (2020). While using a synthetic database provides us with perfect annotation, the synthetic data is limited by the generating model and does not cover all the types of phonation that we encounter in speech and singing. In particular at voice/unvoiced boundaries glottal pulses may occur irregularly. Since the PaN vocoder is driven by the fundamental frequency it cannot properly generate data for these cases. To cover these cases we need to train on real data.

In this chapter we will develop a semi-supervised training procedure to train on both synthetic data, using the existing annotations and on real data using a neural vocoder that uses the glottal pulse signal as its source signal. Using a neural vocoder allows us to train both signal analysis and synthesis simultaneously, which allows the analysis module, which extracts the glottal excitation signal from the voice signal, to be trained semi-supervised on data without explicit pulse annotations. (Semi-supervised, because it still needs to be simultaneously trained on synthetic voice as well, where annotation exists.)

One common difficulty in neural vocoding is that time domain signals usually cannot be compared by their difference. While perceptually insignificant, a difference in phase can create an arbitrary difference between two sinusoids. In deep learning, however, we rely on distance metrics that allow us to compare two signals to evaluate how close a result is from the target value. Using the pulse positions of the input signal, allows us to evade this problem, as the vocoder is in fact capable to reproduce the phase of the original signal as well. Thus, we can compare the input

and output signal in the time domain, to create a training loss directly, which is not possible if only the fundamental frequency is used.

## 4.2. Approach

### 4.2.1. The synthetic database

The power of deep learning depends on the available training data. While there exist numerous speech databases with speech, only few of these databases also contain electroglottographic (EGG) signals. Therefore, GCI extraction methods that rely on the presence of EGG signals cannot fully exploit the benefits of deep learning based methods.

Ardaillon and Roebel (2020) proposed to use a synthetic database in order to have glottal flow signals available as ground truth. The model, even though trained on synthetic data, still generalized well to real speech and achieved comparable results with state-of-the-art methods. The dataset consists of the publicly-available BREF80 (Gauvain et al., 1990) and TIMIT (Zue et al., 1990) datasets, that were analysed and re-synthesised using the PaN vocoder (Huber and Roebel, 2015; Ardaillon, 2017). The PaN vocoder uses the LF glottal source model (Fant, Liljencrants, et al., 1985; Fant, 1995) as a source signal. For the resynthesis, the  $f_0$  estimation of Ardaillon and Roebel (2019) was used to generate the source signal by creating pulses which are separated by the fundamental period. Thus, an almost perfect ground truth glottal pulse can be generated from the vocoder model.

By means of gathering the speech recordings available from previous projects we increased the size of the dataset from a few hours to about 165 hours of human voice including normal speech, singing, whispering and shouting. The simplified LF-model as described by Fant (1995), which is used by the PaN vocoder, uses a single parameter to describe the pulse, the  $R_d$  parameter. Ardaillon and Roebel (2020) resynthesized the database multiple times using different  $R_d$ -values in order to create a large variety of different pulse shapes. Assuming that the increased database size would provide a sufficiently large variety of  $R_d$  values, only one resynthesis of the database using the  $R_d$ -values provided by the  $R_d$  extraction algorithm (Degottex, Roebel, et al., 2010; Huber, Roebel, and Degottex, 2012) was used in this work.

The training database thus contains a resynthesis of all speech signals and all real speech signals. The PaN resynthesis is annotated with all PaN parameters: GCI,  $R_d$ ,  $f_0$ , voiced/unvoiced segments, spectral envelops and unvoiced components (see below). For the real speech data a restricted set of annotations will be used:  $f_0$ , voiced/unvoiced segments,  $R_d$  analysis, spectral envelops.

### 4.2.2. The analysis-synthesis framework

To allow the glottal flow extraction module to model the various phenomena from real speech that are not captured by the PaN resynthesis, we need to train the glottal flow extraction module on real data. However, for the real data we do not have corresponding glottal flow annotation. To allow using real voice data

nevertheless, we train the glottal flow extraction module in an analysis-synthesis framework. In this framework, an analyser (the glottal flow extraction module) and a synthesizer (see below) are cascaded and trained simultaneously. We use special loss functions, described in more detail in Section 4.2.3, to ensure the result of the analyser represents the glottal flow.

Following the approach from Ardaillon and Roebel (2020) the modules of this paper also work on audio given at a sample rate of 16 kHz. Due to pooling layers in the analyser, its output sample rate is 2 kHz. To match the required input sample rate of the synthesizer, upsampling using linear interpolation is performed during training. When inferring the GCI, upsampling is performed using cubic splines in order to increase accuracy. The architecture is identical from Ardaillon and Roebel (2020) and can be seen in Table A.1 (Page 160).

### Synthesizer

We use a convolutional neural network (CNN) with dilated convolution based on WaveNet (Oord, Dieleman, et al., 2016) to transform the glottal pulses generated by the analysis module, and an additional white-noise source, to raw audio.

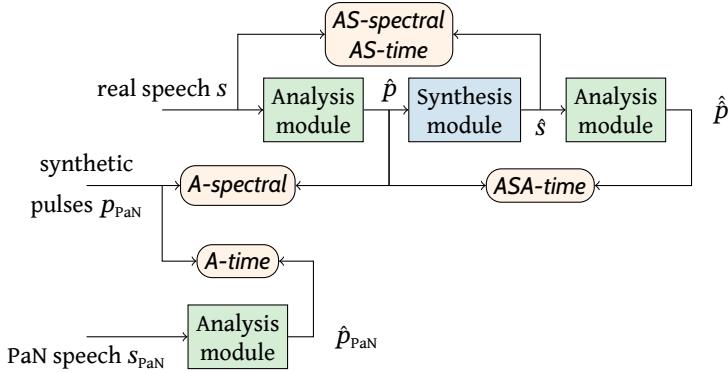
Unlike the original WaveNet, and similar to X. Wang et al. (2019) and parallel WaveNet (Oord, Y. Li, et al., 2017) we create whole frames in one forward pass. Avoiding the recursive structure of the original WaveNet is possible due to the different objective: WaveNet requires recursion to create periodic signals. Our synthesizer has the periodic pulses as input and thus performs not much more than filtering the input according to the spectral envelopes. Furthermore, since the pulses are coherent with the speech signal, the ambiguity in the phase is minimized for the voiced speech component. Consequently, rather than learning a general distribution of raw audio speech signals under their control parameters, here we can use the speech signals as *ground truth*. This allows us to require that the actual audio at hand is properly reproduced according to appropriate loss metrics (see below), which sufficiently models the speech properties when the objective is to infer the pulse positions.

The synthesis task is much simpler than for the original WaveNet, even simpler than for J. Shen et al. (2018), since an oscillation with the correct frequency is already provided at the input. Thus, we follow the observation of J. Shen et al. (2018) and use a reduced size with only two stacks of six layers each (rather than three stacks of ten layers each).

#### 4.2.3. Loss functions

The following section describes the different loss functions that are used to train the networks. Fig. 4.1 gives a schematic overview of the various losses used to update the networks and the flow of the signals used to calculate them.

The principal idea is to train the analysis module jointly with the synthesizer. We use two losses to train this configuration, *AS-spectral* and *AS-time*. These compare the real speech  $s$  and the resynthesized speech  $\hat{s}$  in the frequency and the time



**Figure 4.1.:** Schematic of the analysis-synthesis framework and a visualization of Table 4.1. The **Analysis modules** are in green boxes, the **Synthesis module** is in a blue box, **losses** are in orange boxes. The synthesizer is additionally conditioned on a noise source signal (white noise) and the spectral envelopes of the audio and of the noise signal component. All analysis modules share the same weights.

**Table 4.1.:** The loss function of the analysis-synthesis framework is split into five separate losses that are all calculated on different signals  $\hat{y}$  within the framework, each with its own target value  $y$ . The details for each component are given below. Spectral loss functions are calculated using a spectrogram  $S_w$  with the given window sizes  $w$ .

| Name        | $\hat{y}$                   | $y$              | Weight | Function                           | $w$ [ms]                       |
|-------------|-----------------------------|------------------|--------|------------------------------------|--------------------------------|
| AS-spectral | $\hat{s}$                   | $s$              | 0.1    | $\sum_w \ S_w(y) - S_w(\hat{y})\ $ | 160, 53.3, 32.0, 22.9 and 17.8 |
| AS-time     | $\hat{s} _{\text{noise}=0}$ | $s$              | 10     | $\ y - \hat{y}\ $                  | —                              |
| ASA-time    | $\hat{p}$                   | $\hat{p}$        | 1      | $\ y - \hat{y}\ ^2$                | —                              |
| A-spectral  | $\hat{p}$                   | $p_{\text{PaN}}$ | 0.02   | $\sum_w \ S_w(y) - S_w(\hat{y})\ $ | 80 and 40                      |
| A-time      | $\hat{p}_{\text{PaN}}$      | $p_{\text{PaN}}$ | 10     | $\ y - \hat{y}\ ^2$                | —                              |

domain respectively. However, to ensure that the intermediate signal resembles the glottal pulse signal, we need to employ further regularization. Therefore, we additionally use the losses **AS-time**, **AS-spectral**, and **ASA-time**.

**AS-spectral** To train the analysis-synthesis loop, we follow X. Wang et al. (2019), using multi-resolution spectral loss for the resynthesis. The spectral loss is calculated as the mean absolute error (MAE) between multiple STFT with different window lengths. The STFT are obtained by multiplying the output segment with Hann windows of given sizes, (see Table 4.1), applying an FFT to each of the windowed segments and taking the logarithm of the absolute values. The overlap is 1/2 for each resolution. We call this loss **AS-spectral**.

**AS-time** The STFT considers only absolute values of each frequency bin. Phase differences, therefore, do not affect the spectral loss. This property is required to properly model the noise component of the signal where the phases are random. On

the other hand, the pulse positions are encoded in the phases as well. The spectral loss is thus not suitable to enforce proper pulse positions.

Since the analysis module aims to provide a coherent representation for the excitation signal of both the real and the resynthesized speech signal, there is no ambiguity in phase. We can thus also apply a loss in the time domain between the real audio and the resynthesis to enforce proper reconstruction of the phases, which shall be referred to as *AS-time*. The time domain loss however will not correctly deal with noise. Therefore, when calculating the loss in the time domain, the noise input of the synthesizer is set to zero, enforcing the synthesizer to only produce the deterministic waveform.

### Further regularization losses

When training the GCI analyser in the analysis-synthesis setup, the analyser and synthesizer networks need to be constrained such that they focus on their respective tasks. Notably, the analyser should not start to represent more than the glottal pulse form in its output even if such a strategy clearly would help to improve the synthesis. For imposing these constraints we rely on two strategies: In each training step, the analyser is also trained on the synthetic data, as done by Ardaillon and Roebel (2020), creating the *A-time* loss, and additionally, we include further regularization losses.

**ASA-time** In the analysis-synthesis setup it is desired that the original signal and the resynthesized signal have the same glottal pulses. Thus, we reanalyse the resynthesized audio and apply a loss in the time domain between the reanalysis and the original analysis.

Note that a global optimum for this loss is reached if the analyser does not produce any pulses. In that case all pulse signals are zero and as a result the loss is trivially zero. This behaviour was observed when the synthesizer was not pre-trained and no other regularization was performed on the analysis of the real audio.

**A-spectral** While we do not know the exact pulse positions  $p$  in the real audio  $s$ , we know where pulses are present (the voiced segments), the approximate pulse shape (from the  $R_d$ -analysis), and the pulse frequency (from the  $f_0$ -analysis). All these parameters are also in the synthetic glottal pulse signal  $p_{\text{PaN}}$ . Therefore, the two spectrograms of the unknown and unobservable pulse signal of the real speech  $\hat{p}$  and the pulse signal used in the PaN synthesis  $p_{\text{PaN}}$  should be similar. To enforce this invariant, we apply a spectral loss between the synthetic glottal pulse signal  $p_{\text{PaN}}$  and the analysed glottal pulse signal  $\hat{p}$ . This loss is of major importance of the analyser as it prevents the network to create two distinct behaviours for real and synthetic speech, which will be demonstrated in an ablation study in Section 4.3.

### Total loss

The total loss is obtained as a weighted sum of all the losses. The weights were chosen manually such that the individual losses were in the same order of magnitude. The

weight values can be seen in [Table 4.1](#).

#### 4.2.4. Training procedure

##### Step 1 – Initialization

Both the analysis and the synthesis modules are pre-trained separately on the synthetic database. The analysis module is trained only using the loss *A-time*. The synthesizer is trained to imitate the PaN vocoder by using synthetic pulses as input and applying an L2 loss between the synthesizer output and the output generated by the PaN vocoder using the same synthetic pulses.

For the analysis module we use the same hyperparameters as Ardaillon and Roebel (2020): a batch size of 128, an epoch length of 500 updates, an initial learning rate of  $2 \cdot 10^{-4}$  and decrease the learning rate by a factor of 0.75 if no improvement on the validation loss was observed for 10 epochs. The synthesizer is trained with a batch size of 8 training-samples from 8 different files and a length of 2560 audio-samples, an epoch length of 512 updates, an initial learning rate of  $5 \cdot 10^{-5}$  and the same learning rate schedule as for the analyser with a minimum of  $1 \cdot 10^{-6}$ . After pre-training, the batch-normalisation layers in the analyser are frozen.

##### Step 2 – Fine-tuning with analysis-synthesis

After pre-training, the analysis and synthesis modules are combined in the analysis-synthesis setup. Both real and synthetic data are used to generate the different losses. For details see [Table 4.1](#) and [Fig. 4.1](#). Each training step consists of two updates. First the losses involving real data (*AS-spectral*, *AS-time*, *ASA-time*, *A-spectral*) are calculated according to [Table 4.1](#) and the gradients are propagated to both analyser and synthesizer. Then a *conventional* training step (as in Step I) is performed for the analyser on synthetic data to ensure that the output of the analyser does not diverge too far from the *LF*-model.

For the analysis-synthesis we use a batch size of 8 with a length of 3553 audio samples (2560 remain as input to the synthesizer due to valid-padding in the analyser), an initial learning rate of  $1 \cdot 10^{-5}$  with the usual schedule (cf. Step 1) with no minimal learning rate.

### 4.3. Evaluation

We evaluate the performance of the analyser on the CMU arctic (Kominek and Black, 2004) databases using the metrics from Ardaillon and Roebel (2020), which are based on the metrics defined by Thomas and Naylor (2009). The results are summarized in [Table 4.2](#). The GCI are obtained using a simple peak-picking procedure on the negative peaks of the derivative of the predicted glottal flow, as was done by Ardaillon and Roebel (2020).

### 4.3.1. Test setup

To evaluate the proposed method we compare the following five models:

**SEDREAMS** For comparison, we include the results from SEDREAMS (Drugman and Dutoit, 2009) as a baseline.

**FCN-BT** As another baseline we use the configuration ‘FCN-synth-GF’ from Ardaillon and Roebel (2020), which is trained on the BREF80 and TIMIT dataset. This was assumed to be the state-of-the-art at the time of the study.

**FCN-UNI** The same network and training procedure as FCN-BT but trained on the synthetic (not augmented) part of the new, universal voice database, described in Section 4.2.1. This is the same as Step 1 from Section 4.2.4 and without the joint training of Step 2.

**AS-UNI** The analysis module from the analysis-synthesis as introduced in the previous section. We do not pre-train a new analysis module for this configuration but use FCN-UNI as initialization to Step 2.

**AS-UNI\*** The analysis module from the analysis-synthesis, however trained without *A-spectral*. We do not pre-train a new analysis module for this configuration but use FCN-UNI as initialization to Step 2.

As baselines, we use SEDREAMS from Drugman and Dutoit (2009) and FCN-BT from Ardaillon and Roebel (2020). To compare the effect of the new database we include FCN-UNI which has the same architecture and is trained the same way as FCN-BT but is trained on the universal voice dataset proposed in Section 4.2.1. Furthermore, FCN-UNI allows us to isolate the effect of the analysis synthesis training framework by comparing it to AS-UNI, which uses FCN-UNI as the pre-trained initialization to Step 2. Finally, we include the model AS-UNI\* which is trained without using the *A-spectral* loss, to investigate the effect of the *A-spectral* loss.

### 4.3.2. Evaluation metrics

We use the same evaluation metrics that were used by Ardaillon and Roebel (2020) for Table 2, which are as follows:

**IDR** (*identification rate*) The number of unique identifications between generated pulses and pulses in the ground truth divided by the total number of pulses in the ground truth.

**MR** (*miss rate*) The number of pulses in the ground truth that were not detected by the system divided by the total number of pulses in the ground truth.

**FAR** (*false alarm rate*) The number of generated pulses that could not be associated with any pulse from the ground truth divided by the total number of pulses in the ground truth.

**Table 4.2.: Results on CMU arctic with metrics considering all GCI.**

| Model     | IDR [%] | MR [%] | FAR [%] | IDA [ms] |
|-----------|---------|--------|---------|----------|
| AS-UNI    | 94.21   | 1.66   | 9.71    | 0.51     |
| AS-UNI*   | 92.55   | 0.87   | 15.21   | 0.64     |
| FCN-UNI   | 91.90   | 0.88   | 22.33   | 0.75     |
| FCN-BT†   | 92.73   | 2.86   | 10.87   | 0.47     |
| SEDREAMS‡ | 90.74   | 0.19   | 61.38   | 0.30     |

† Ardaillon and Roebel (2020)

‡ Drugman and Dutoit (2009)

**IDA** (*identification accuracy*) Standard deviation of the misalignment between the associated detected and ground truth **GCI**.

For each model we calculate the metrics for each file separately and create an average for each of the speakers, *bdl*, *jmk* and *slt*. We then create total averages, as the average over each speaker, i.e. equally representing each *speaker*, and display the result in Table 4.2.

## 4.4. Results

### Having a large variety of pulse shapes in the synthetic data is important

Although the database for FCN-UNI was much larger than the database for FCN-BT, the identification rate of FCN-UNI is slightly worse than the one of FCN-BT. This can be explained by the larger variety of  $R_d$  values in the synthetic database for FCN-BT obtained by the data augmentation (multiple resynthesis with different  $R_d$ -values).

It is worth noting that although here we focus on estimating the glottal closure instant (GCI), the analyser learns much more than just the **GCI**: It learns the whole pulse shape, that is, in the context of the underlying pulse model (LF-model (Fant, 1995)) it learns the proper  $R_d$ -value. Thus augmenting the variety in  $R_d$ -values / pulse-shapes should improve generalization. Furthermore, the pulses generated by the analysis module could potentially be used to infer the  $R_d$ -value, which might improve the  $R_d$ -analysis and consequently the resynthesis of the synthetic database.

### Fine-tuning the analysis as part of the analysis-synthesis framework does improve the GCI detection

We see that the identification rate of the fine-tuned model AS-UNI is higher than the model it was initialized with, FCN-UNI. We can conclude that fine-tuning the analysis in

the analysis-synthesis framework does positively impact the performance of the analysis module.

This can be explained by the fact that the fine-tuning happens on real data and the real data is closer to the evaluation dataset, CMU arctic, than the synthetic data. This suggests that the analysis-synthesis setup is able to force the analysis module to generate a more expressive pulse model. Consequently, the model generalizes better to real data,

Observe that none of the losses used for training explicitly represents the GCI. Nevertheless, our model is able to detect the GCI better than any other method. This suggests that the model does not only learn the pulse positions, it really learns the pulse shapes from which the GCI can be derived.

### Regularization of the pulse signal is crucial

The improvement of AS-UNI\* is smaller than for AS-UNI compared to FCN-UNI and still does not allow surpassing the results of FCN-BT. This shows that the regularization in terms of the *A-spectral* is a crucial element of this set-up. One problem with the analysis-synthesis method is the fact that the synthesizer learns only to use the pulse input in voiced speech. Consequently, the predicted pulse signal has no effect at the synthesizer output during unvoiced speech and silence and thus does not affect the AS losses. Since most of the error happens in the unvoiced segments, the improvement in configuration AS-UNI\* is minimal.

In an additional experiment it was shown that training the analyser on real data with *A-spectral* and on synthetic data with *A-time*, hence avoiding the use of a synthesizer, does not allow improving over the FCN-UNI baseline.

## 4.5. Conclusion

In order to further improve the results from Ardaillon and Roebel (2020), we proposed a new semi-supervised approach for training a neural GCI detector that allows training with controlled synthetic speech and real speech data. The main idea of this approach is to incorporate the neural network for predicting the glottal flow, introduced by Ardaillon and Roebel (2020), into a constrained analysis-synthesis loop. The end-to-end system trained using a reconstruction loss and a collection of additional loss functions that ensure proper segregation of the analysis and synthesis tasks. To better represent all the variability of voice signals, we also proposed to use here a much larger database with many types of signals.

For the task of GCI detection, evaluation on the CMU arctic database has shown that the fine-tuning using real data significantly improves the identification rate of the estimator with respect to training it solely on the synthetic database. The identification rate also surpasses all previous systems establishing the proposed method as the new state-of-the-art. While we only assessed in this paper the performance of our analyser for GCI detection, the predicted glottal flow may also be used for other tasks like  $R_d$  analysis.

## 5. Neural vocoder

Vocoders are used in many applications. Each application has its own requirements such that different approaches to vocoding exist, designed to accommodate the specific applications. This thesis is concerned with transformation of voice. Thus, in this chapter we will develop a neural vocoder that has a parametric space suitable to transform voice. We will discuss the different requirements that we pose to a vocoder used for transformation and the difficulties to implement such a vocoder. The neural vocoder developed in this chapter, and which is used in the following chapters as well, uses the mel-spectrogram as its parametric space. Due to its many possible applications, today many neural vocoders based on the mel-spectrogram exist and have been developed independently in different places.

The topics described in this chapter have been researched collaboratively with Axel Roebel who worked on the same issues for a different project. My own contribution is experimentation with the parametric space in particular in connection with the previous chapter and improvement on practical issues with MelGAN (K. Kumar et al., 2019). The final vocoder, as described in [Section 5.4](#), has been developed in parallel and cannot be seen as a contribution of this thesis.

### 5.1. Desired properties of a vocoder

We shall first discuss the requirements posed to a vocoder used for transformation. There are two types of requirements: Foremost, we have demands on the properties of the parametric space. The desired properties allow us to design transformations and ensure the limitations of the vocoder do not interfere with our objectives. Furthermore, we have practical requirements on the implementation of the vocoder, that arise, when the voice transformation together with the vocoder is implemented on physical hardware, trained with real data and used in a real application. This allows us to motivate, why we chose to use the mel-spectrogram as the parametric space of our vocoder and why our vocoder is implemented the way it is.

Generally a vocoder can be used to perform two tasks:

1. A vocoder removes unnecessary information.
2. A vocoder restructures and presents the relevant information in a more accessible way.

Which information to remove depends on the application. In voice transformation, we want to remove the perceptually irrelevant information, like certain phase

properties or the exact characteristics of the noise. Removing unnecessary information simplifies the transformation task since this unnecessary information does not have to be treated by the transformation.

Since the time-domain signal is one-dimensional, the contained information is spread across time through oscillations. The fundamental frequency for example, while a time varying property, cannot be localized to a fixed point in time but rather is a property of a window in time in which we can observe an oscillation. (We need to observe at least two periods for a frequency to exist.) A vocoder bundles the information that is spread across time to a degree, by decoding the information that is encoded through the oscillations. This simplifies the transformation task as the information is more explicitly available.

When discussing different vocoders we can evaluate them on the basis of how well they perform these two tasks. Not all vocoders necessarily need to perform both tasks, the phase vocoder, for example, does not remove any information and is still a useful tool for specific kinds of transformation. In the following we will first introduce some mathematical modelling of voice signals and then use the mathematical model to discuss different desirable properties of vocoders.

### 5.1.1. Mathematical model

In this chapter we model signals as finite length sequences. Let  $\mathbf{X}$  be the space of all finite length signals:

$$\mathbf{X} = \bigcup_{n=1}^{\infty} \mathbb{R}^n \quad (5.1)$$

A vocoder is a pair of mappings  $(a, s)$  between the space of time-domain signals  $\mathbf{X}$  and the vocoder parameters  $\mathbf{P}$ .

$$\mathbf{P} \subseteq \mathbf{X}^N \quad (5.2)$$

$$a : \mathbf{X} \rightarrow \mathbf{P} \quad (5.3)$$

$$s : \mathbf{P} \rightarrow \mathbf{X} \quad (5.4)$$

We call  $a$  the analysis operation and  $s$  the synthesis operation. Analysis and synthesis change the lengths uniformly, that is, for all signals  $x \in \mathbb{R}^n$  of fixed length  $n$  the output of  $a$  has always the same lengths  $m_1, \dots, m_N$ .

$$a(x) \in \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_N} \quad (5.5)$$

$$s(a(x)) \in \mathbb{R}^n \quad (5.6)$$

Usually  $m_1, \dots, m_N < n$ , that is, the time-domain signal  $x$  is represented by multiple vocoder parameters with lower sample rate.

We define a perception function  $\pi : \mathbf{X} \times \mathbf{X} \rightarrow [0, \infty)$  with the properties:

$$(\text{symmetry}) \quad \pi(x, y) = \pi(y, x) \quad (5.7)$$

$$(\text{triangle}) \quad \pi(x, z) \leq \pi(x, y) + \pi(y, z) \quad (5.8)$$

for any  $x, y, z \in \mathbf{X}$ . The function  $\pi$  is similar to a metric, except that different elements might have perceptive ‘distance’ zero. The perceptive function measures how similar two signals  $x$  and  $y$  sound like: if they sound very different,  $\pi(x, y)$  will be large, if they sound alike,  $\pi(x, y)$  will be zero.

Using the perception function  $\pi$  we can define the relationship between analysis function  $a$  and synthesis function  $s$  as inverses of each other under the perception function: Let  $\mathbf{V} \subseteq \mathbf{X}$  be the set of voice signals we want to represent with the vocoder  $(a, s)$ . For all voice signals  $x \in \mathbf{V}$  we require

$$\pi(s(a(x)), x) = 0. \quad (5.9)$$

In practice a vocoder can be designed by fixing either  $a$  or  $s$  and then optimizing the other such that

$$s = \operatorname{argmin}_{\tilde{s}} \mathbf{E} [\pi(\tilde{s}(a(V)), V)] \quad (5.10)$$

or

$$a = \operatorname{argmin}_{\tilde{a}} \mathbf{E} [\pi(s(\tilde{a}(V)), V)], \quad (5.11)$$

for some random variable  $V$  that maps to  $\mathbf{V}$  reflecting the distribution of the voice signals that we want to represent with the vocoder  $(a, s)$ .

### 5.1.2. Desired properties of the parametric space

The vocoder exposes relevant information about the voice through its parametric space and presents it in a more accessible way. Below we will discuss what is meant by ‘relevant information’ and ‘accessible’ through more concrete properties.

**Expressiveness** The parametric space of the vocoder should be expressive enough to be able to encode even subtle changes to the voice. In the days of classical parametric vocoders, an insufficiently expressive parametric space would result in degradation for the cases that cannot be expressed in the vocoder’s parametric space. With neural networks the problem can be much more hidden. Neural networks can generate realistically sounding voice without any conditioning (Oord, Dieleman, et al., 2016). Any additional conditioning can guide the result into a desired direction. An under-parametrised neural vocoder might be able to produce realistically sounding voice in any case but the result after an analysis-synthesis cycle might be very different from its input. Thus, it is much more difficult to say whether a parametrization is sufficiently expressive in the context of neural vocoding and can in practice only be shown with certainty by providing examples of synthesizers that can differentiate relevant features.

In mathematical terms, expressiveness can describe how well perceptually relevant details can be distinguished under the analysis function  $a$ . For two voice signals  $x_1, x_2 \in \mathbf{V}$  this property can be expressed as

$$\pi(x_1, x_2) = 0 \Leftrightarrow a(x_1) = a(x_2). \quad (5.12)$$

The left-to-right implication ( $\Rightarrow$ ) requires that the vocoder removes redundant information: if  $x_1$  and  $x_2$  sound alike, there is no need to distinguish between them in the parametric space of the vocoder. The right-to-left implication ( $\Leftarrow$ ) is the actual expressiveness requirement as discussed above. If multiple elements  $x_1$  and  $x_2$  are mapped to the same point in the parametric space, the synthesis function  $s$  is not perceptually unique, and the vocoder cannot distinguish between the acoustic feature that makes  $x_1$  and  $x_2$  perceptually different.

**Comparability** Unless the parametric space of the vocoder consists only of perfectly disentangled parameters, the parametric space must be transformed as a whole to stay coherent. Even if the parametric space would consist of perfectly disentangled parameters, changing a property that is not exposed as a parameter would require to change all parameter simultaneously. We believe this can best be achieved with deep learning. When dealing with neural networks we need to be able to compare different parameter values to define training losses.

Formally, the parametric space of a vocoder needs to form a metric space  $(\mathbf{P}, d)$  such that we can compare two parametrizations  $p, q \in \mathbf{P}$  through the distance function  $d$ . This distance function should reflect the perceptive function  $\pi$  in the underlying signal space. That is, for two voice signals  $x, y \in \mathbf{V}$

$$\pi(x, y) = d(a(x), a(y)). \quad (5.13)$$

If the parametric space consists of many parameter sets that are too different from each other structurally, finding a suitable distance function  $d$ , might require additional hyperparameters, which may add additional difficulties. Thus, for a neural vocoder a single set of closely related parameters is more desirable.

**Interpretability** The parametric space should be meaningful. When designing transformations on the parametric space of a vocoder, we need to have an idea, what the individual parameters mean to make informed decisions about what to change and how.

*The above properties are desired features of the parametric space. The mathematical formulations express ideals but are in practice not enforceable. Thus, we have to interpret the equations as approximations rather than strict requirements. Mathematically Eq. (5.13) implies Eq. (5.12), but they express different concepts such that in an approximative context they can be seen as two separate requirements.*

### 5.1.3. Practical requirements on the vocoder implementation

In addition to the aforementioned features, we face additional difficulties when a neural vocoder is implemented. Since computational hardware is not infinitely fast, the amount of training data is limited and neural networks might not always generalize well. We have to consider the following requirements when creating a neural vocoder after we have decided on a parametric space. The choice of a parametric space, might also be influenced by the possibility of satisfying the requirements below for a specific parametric space.

**Inference speed** For all practical purposes the speed of the neural vocoder was a concern in 2020 (and still is today). WaveNet (Oord, Dieleman, et al., 2016), which was popular in speech synthesis at the time (J. Shen et al., 2018), took minutes to produce a second of audio. Thus, the speed of the vocoder is a serious consideration in any practical application.

**Universality** To perform transformation on the voice a universal vocoder is necessary that allows working on any voice since we cannot know in advance which voice the vocoder will be applied to and what kind of transformations will be used. Historically this was not such a big issue as the vocoders of the past only relied on acoustic models that were speaker independent. With neural networks we are able to learn from speaker specific data and early vocoders worked only on a single speaker (J. Shen et al., 2018) or needed to be conditioned on the specific speaker (Arik, Diamos, et al., 2017). In speech synthesis (where these vocoders were used) this does not pose such a big problem as the synthesized voice identity is known in advance. With voice cloning (Arik, Jitong Chen, et al., 2018), however, this becomes an issue, even for speech synthesis.

**Training data** The availability of training data has grown rapidly in recent years. Still, training data for singing voice is rare with only few publicly available singing voice databases and with many singing styles not covered. Thus, for voice processing that includes singing voice, it is necessary that a neural vocoder can be trained on the limited examples that exist and is able to generalize to singing styles that are only sparsely covered. Similarly, for practical reasons, the same condition applies to speech. Smaller datasets allow results to be reproduced more easily and allow shorter training procedures as iterating over the dataset is faster.

## 5.2. Parametric spaces for voice

In this section we will discuss different representations of the voice in regard to their suitability for neural vocoding and voice transformation. We will argue, why in our opinion the mel-spectrogram is best suited to satisfy the requirements from the previous section.

### 5.2.1. Time-domain signal and end-to-end processing

Though this thesis is concerned with voice transformation using a neural vocoder, it is worth to consider the possibility not to use a vocoder at all and to perform the transformation on the signal directly. End-to-end approaches exist in very recent literature of related fields for text-to-speech synthesis (Donahue et al., 2021; Ren et al., 2021; J. Kim et al., 2021) and for voice identity conversion (Nguyen and Cardinaux, 2022; Xie et al., 2022). When we designed the neural vocoder in 2020, however, the state-of-the-art was very different and end-to-end approaches had serious drawbacks such that an approach with a neural vocoder was the safer choice.

As discussed at the beginning of [Section 5.1](#) the information in the time-domain signal is heavily distributed over time. An end-to-end approach needs to disentangle the information that is spread across time and find an internal representation where these properties are independent. Finding this representation automatically makes sense in an application, where we only want to change one property since computing this representation from the time-domain signal without intermediate steps is the most efficient.

While the parametric space of a vocoder might not be optimal for a specific property, it can be reused in a chain of transformations. Thus, in applications, where we may want to change multiple properties it is more desirable to use a vocoder to avoid the need to reconstruct the raw-audio signal between each transformation step.

### 5.2.2. Classical vocoder parameters

The parametric space of classical vocoders parametrizes signal processing operations used for the signal synthesis. A typical parametrization would be through the fundamental frequency  $f_0$ , which defines the frequency of a source oscillator, the spectral envelope, which defines a filter and the aperiodicity, also defining a filter. Other vocoder parameters have similar origins in signal processing. While not strictly necessary in the context of neural vocoding, we can use the existing parametric spaces as a basis for the parametric space of a neural vocoder. We will discuss possible drawbacks and benefits, according to the properties from [Section 5.1](#), below.

#### Interpretability

Many classical neural vocoders operate on a parametric space that is interpretable. The underlying physical or signal models give meaning to the parameters that can be understood by humans. For our neural vocoder we could borrow some existing vocoder parameters for their interpretability or make some adjustments to make them fit better to processing with neural networks.

Changing properties that are exposed as a parameter can be achieved easily but requires adjusting the remaining parameters to ensure consistency. For example, the fundamental frequency can easily be changed using a classical parametric vocoder if that vocoder uses the fundamental frequency as one of its parameters. To generate

natural voice, however, the remaining vocoder parameters need to be adjusted to remain coherent with the new fundamental frequency contour. Some neural vocoders, like PaN (Ardaillon, 2017), are designed to reduce the influence of the fundamental frequency on the other parameters. In PaN, this is achieved by using the vocal tract filter, rather than the spectral envelope. Nevertheless, we find limitations for large transformations with the PaN approach when listening to the results.

### Comparability

While the parametric space of a classical vocoder exposes some properties that might be of interest, we cannot reasonably expect all interesting properties to be exposed in one parametric space. Thus, we are faced with two possibilities: Either use a different vocoder for different sets of properties, or fix one parametric space and resynthesize the vocoder parameters depending on the changed property. The former approach has the same drawbacks as the end-to-end approach with the additional problem that now the parametric space might not even be optimal for the specific transformation. If using the parametric space of a classical vocoder we would have to implement the transformation on each of the parameter sets and ensure consistency between the different parameter sets.

This brings us to the requirement of comparability. To change a voice property in the parametric space, each of the sub-sets, e.g.  $f_0$ , spectral envelope and aperiodicity have to be transformed to reflect the change of the modification in the modified parameter. When treating the parameters individually, inconsistencies might occur between the vocoder parameters if the property change can be achieved by different effects. Thus, consistency between the parameters needs to be enforced. Consistency could be achieved by transforming all parameter groups simultaneously. In this case we face the difficulty of defining proper training metrics that give each parameter the correct weight:

The full parametric space  $\mathbf{P}$  can be described as the Cartesian product between the different parameter groups  $\mathbf{P}_1 \times \dots \times \mathbf{P}_n$ . Each parameter group ( $\mathbf{P}_k$ ) forms a metric subspace  $(\mathbf{P}_k, d^k)$ . We can obtain a metric  $d$  on the product of the sub-spaces as a weighted sum over the distance of the metric sub-spaces:

$$d((p_1, \dots, p_n), (q_1, \dots, q_n)) := \sum_{k=1}^n \lambda_k d^k(p_k, q_k) \quad (5.14)$$

with the individual weights  $\lambda_1, \dots, \lambda_n$ . The problem of this approach is that it introduces additional hyperparameters ( $\lambda_k$ ) which can have a large influence on the result. There may not be a straightforward optimal solution, especially if the metric sub-spaces  $(\mathbf{P}_k, d^k)$  are structured very differently, which is the case between fundamental frequency and spectral envelope.

Alternatively, consistency could be achieved, by successively changing one parameter after the other, taking all the previously changed parameters into account. For example changing the  $f_0$  based on the property  $x$  first, then changing the spectral envelope  $E$  based on  $x$  and the new value of  $f_0$  and finally changing the aperiodicity

based  $x$ ,  $f_0$  and  $E$ . While the discussed issues could be overcome somehow, it is worth to consider whether other parametrizations could be easier to use. The parametric space of classical vocoders is motivated by the synthesis operation. With neural networks we are not restricted in the same way and are free to explore different parametrizations, possibly inspired by classical vocoder parameters.

### Expressiveness

Many classical vocoders suffer from limited expressiveness. The use of the fundamental frequency  $f_0$ , in particular, poses serious restrictions on the possibility to represent voice with irregular pulses, like glottal fry. In the context of neural vocoding it might be possible to infer the existence of irregular pulses, but there is no explicit control. In Chapter 4 we experimented with a neural vocoder that uses a non-parametric glottal pulse signal instead of the fundamental frequency. This approach allows irregular pulses, which is why we investigated into its possibilities. Originally the idea was that this vocoder could also be used for transformation. Using a non-parametric glottal pulse signal, however, poses difficulties with interpretability: To change pulse positions, additional tools are required to infer the pulse positions and to resynthesize the pulse signal from the new positions. Furthermore, coherency issues arise, as the glottal pulse signal is more strongly correlated with the spectral envelope and the aperiodicity. Finally, the experiments with the neural vocoder of Chapter 4 suggested that the spectral envelopes of full signal and noise signal might miss certain characteristics of the voice thus making its parametric space not expressive enough. For these reasons the vocoder from Chapter 4 was found to be unsuitable as a general vocoder on which to base the voice transformations on.

#### 5.2.3. Mel-spectrogram

When we started our work on the neural vocoder in 2020, the mel-spectrogram was an emerging representation for voice used in text-to-speech (J. Shen et al., 2018; Arik, Diamos, et al., 2017; Ping et al., 2018) and voice identity conversion (J.-X. Zhang et al., 2019).

In a signal processing context, the spectral envelope is a handy description of a signal as it describes a filter that can be easily applied to any input signal. We can see it as a means to remove the direct<sup>1</sup> dependency of the fundamental frequency on the spectrogram by smoothly connecting the partials. Implicit dependency, however, is still contained in the spectral envelope. Another drawback is that the noise source is not described by the spectral envelope in the frequency bands where the sound is dominated by harmonic sounds. The noise source needs to be described with another parameter, like the spectral envelope of the noise source. In classical synthesis this is a desired feature since noise and harmonic sounds are treated independently. In

---

1. From a signal processing point of view, the frequency of an oscillation and the spectral envelope are independent: one describes the position of the partials, the other the amplitude of the partials. In the context of voice, however, dependencies exist between the pitch and the timbre, which manifests itself as correlations between  $f_0$  and spectral envelope.

the context of neural synthesis this is undesired as we want to generate harmonic and noise input at the same time, to properly model their interactions. Furthermore, the parametrization of the noise source depends on analysis methods that may themselves be imprecise.

For the reasons above it might be worth to consider using a property derived from the spectrogram that does not hide its dependency on the fundamental frequency and contains information about the noise source explicitly. The **mel-spectrogram** fulfils these criteria and has successfully been used in speech synthesis (J. Shen et al., 2018). The reduced dimensionality from remapping the frequency axis allows it to be efficiently used in neural networks, and it was a promising candidate for a trade-off between removing redundant information while still describing the signal explicitly. Below we will discuss the strengths and drawbacks of the **mel-spectrogram** as a parametric space based on the requirements defined in [Section 5.1](#).

### Interpretability

The **mel-spectrogram** can be seen as a visual representation of audio. We can inspect the **mel-spectrogram** visually and easily detect regions of voice activity and distinguish between voiced and unvoiced sections. With more experience further details can be inferred from a visual inspection, such as distinction between speech and singing, the rough amount of breathiness and approximate pitch and the change of some voice property can be seen in the **mel-spectrogram**. This allows us to confirm that a transformation is changing the voice in the desired direction — though, quality assessment is not possible through visual inspection alone, it can serve as a first sanity check.

Since the **mel-spectrogram** does not make any attempt in disentangling different voice properties, it can be difficult to inspect the **mel-spectrogram** more systematically. While derived properties, like fundamental frequency and spectral colour, can be inferred from the **mel-spectrogram**, this requires additional tools which may make additional assumptions or introduce estimation errors or biases. Changing the **mel-spectrogram** by hand is impossible due to the high correlation between adjacent frequency bins. Thus, the **mel-spectrogram** is interpretable for informal visual inspection and preliminary checks, but unsuitable for systematic analysis or adjustments by hand.

### Comparability

Having one coherent set of parameters allows comparing results easily which is essential for deep learning to create meaningful gradients. Since all the parameters are on the same scale, we can compute the loss without relying on heuristics how to weight the different loss components. Furthermore, the **mel-scale** reflects some aspects of human perception of sound which naturally assigns higher weight to those frequency components that are perceptually more relevant.

Typically, the values of **mel-spectrogram** are given on a logarithmic scale. The logarithmic scale models well the perception of loudness and allows distinguish-

ing between low amplitude frequency components even in the presence of high amplitude sinusoids in other frequency bands. Calculating differences on the logarithmic scale is robust to changes in total signal gain. An average error over the frequency bins generates a loss with units (typically in dB) that are common in signal processing and thus easily understood and allow meaningful interpretation of the results. Thus, absolute or quadratic differences between log-mel-spectrograms, generate a distance metric, that provides a relevant approximation of Eq. (5.13). The ease of comparing two log-mel-spectrograms through differences yielding a meaningful value is one of the main strengths of the mel-spectrogram.

### Expressiveness

Evaluating the expressiveness of a parametric space is difficult for any parametric space and can in practice only be observed through examples. This is especially the case if deep learning models are involved as these can use information from the parametric space that is not obviously visible. When we evaluated the mel-spectrogram for its suitability as a parametric space, sufficient expressiveness was still an open question, and it still remains uncertain for a few subdomains. The application in text-to-speech (J. Shen et al., 2018; Arik, Diamos, et al., 2017; Ping et al., 2018) showed that the mel-spectrogram was sufficiently expressive to model the speech of a single speaker. Multiple speaker could only be modelled, at the time, through additional speaker conditioning. Further work on universal vocoding by Lorenzo-Trueba et al. (2018) indicated that the mel-spectrogram could be used to differentiate between speech of multiple speaker. Whether the mel-spectrogram capture all perceptually relevant details was still an open question but an essential limitation for this work.

Throughout the work around this chapter we were finally able to show that the mel-spectrogram could also be used for singing voice by presenting a universal vocoder that works on both speech and singing simultaneously. Recent work of S.-g. Lee et al. (2023) has shown examples where the mel-spectrogram could even be used to resynthesize musical accompaniment and other kinds of sounds. Whether the mel-spectrogram can represent perceptually relevant details in rough voice, glottal fry and other modes of irregular phonation may be likely but remains to be seen. More and more examples where the mel-spectrogram is a suitable representation have been appearing in the literature lately. Still, it might be possible that eventually we will find limitations of the modelling capacity of the mel-spectrogram which may necessitate using a different representation of the voice. Nevertheless, for the tasks set out in this thesis, all the indications are there that the mel-spectrogram is sufficiently expressive.

### Practical concerns

In 2020, it was not clear at all from the literature, whether universal vocoding using the mel-spectrogram was practical. The mel-spectrogram was already commonly used in text-to-speech applications like J. Shen et al. (2018), Arik, Diamos, et al. (2017)

and Ping et al. (2018). However, mapping from the mel-spectrogram to raw-audio was either achieved by single-speaker WaveNet (J. Shen et al., 2018), or a WaveNet with speaker conditioning (Arik, Diamos, et al., 2017). No universal vocoder existed that could be used for both speech and singing voice. Since transparent resynthesis could only be achieved using WaveNet, inference speed was very slow or took a very elaborate refinement process (Oord, Y. Li, et al., 2017). Furthermore, training single-speaker neural vocoders required large amounts of training data for a single speaker. Thus, practical concerns existed with regard to inference speed, universality and required amount of training data.

With progress towards speaker independent universal vocoders (Lorenzo-Trueba et al., 2018) and faster network architectures through WaveGlow (Prenger et al., 2019) and MelGAN (K. Kumar et al., 2019) it became probable, that computationally efficient universal vocoding on the mel-spectrogram would be feasible. Today, all these practical concerns are overcome by various neural vocoders (Jang et al., 2020; Jiao et al., 2021; Roebel and Bous, 2022; S.-g. Lee et al., 2023), one of which will be presented in Section 5.4.

#### 5.2.4. Conclusion

Neural synthesizers are able to generate audio by extracting implicit information from its parametric space. Thus, new possibilities arise for the way how we represent voice and when designing a neural vocoder we have the chance to rethink the parametric space. The choice of parametric space is important as it influences the limits of what kind of transformation is possible. In this section we have discussed different representations of voice, from classical vocoder parameters to the mel-spectrogram and have contemplated the possibilities of end-to-end approaches.

From the discussion above, we can conclude that the mel-spectrogram is a promising representation of the voice and could provide interesting possibilities for voice transformation. Most of the apparent drawbacks that were unclear when we first began working on the neural vocoder in 2020, have been resolved in recent years. Today we find that the mel-spectrogram has established itself as the *de facto* standard for neural vocoding.

### 5.3. Neural vocoding using the mel-spectrogram

Even though there were a few open questions about the expressiveness of the mel-spectrogram and practical concerns with regard to the implementation of such a neural vocoder, the mel-spectrogram was a promising parametrization of the voice. Thus, much of the early work of my thesis, was dedicated to addressing the existing concerns. Since other projects depended on an efficient neural vocoder as well (Benaroya et al., 2023), means to optimize the existing mel-inversion approaches and to allow unconditioned multi speaker synthesis were investigated in parallel work by me and others. While I concentrated on singing voice and universal vocoding using MelGAN, others also investigated into WaveGlow and other architectures. Finally, the MelGAN approach was dropped due to insufficient quality, and we started using

a multi speaker version of WaveGlow which was developed for a project on gender conversion (Benaroya et al., 2023). This approach was later replaced by a custom architecture that more closely implemented the source-filter model. Since the question of suitability of the mel-spectrogram was solved at that point, I started to work on transformations on the mel-spectrogram. The final mel-inverter is summarized in Section 5.4 for completeness. My own contribution, however, was only minor, thus insufficient to constitute an independent contribution for this thesis. This section summarizes the research on the mel-spectrogram inversion I was involved in.

### Initial experiments with state-of-the-art vocoders

Our first experiments on creating a mel-inverter were based on WaveGlow (Prenger et al., 2019) and MelGAN (K. Kumar et al., 2019). The WaveGlow approach consists of creating a bijective mapping between the data that we want to synthesize and white noise. During training, the mapping is trained to generate white noise from audio recordings, during inference, the inverse mapping is used to generate audio recordings from white noise. These mappings can be conditioned on the mel-spectrogram to enforce that the generated audio follows the provided mel-spectrogram. One problem with this approach is that the mapping requires huge receptive fields<sup>2</sup> (several seconds) to create oscillations with consistent phase and even then often struggles with incoherent phases at different times. This phase inconsistency would sometimes cause the phase to jump suddenly, spreading the peaks of the partials and creating audible artefacts. Thus, any attempt to reduce the size of the models to make them more efficient was futile.

Similarly, with MelGAN we observed that oscillations would not come naturally as well. During early training stages one could clearly see periodic patterns along the frequency axis that emerged not from partials of the sound but rather from the upsampling operations. The resulting sound resembled the classic robot-like vocoder effect typically found in pop music of the ‘80. With increasing iterations the patterns would become more fine-grained and follow the desired spectral shape until the static periodicity was (barely) audible.

### Difficulties with oscillations

What both these approaches have in common is that they try to generate oscillating signals from non-oscillating data. This can easily be achieved by recursive structures like WaveNet, as an autoregressive system can have an oscillating impulse response with only two recursive coefficients. Non-recursive, finite impulse response systems on the other hand, cannot naturally produce oscillations and therefore have to compensate with large receptive fields to fake an oscillating impulse response. As a result inconsistencies in the phase may occur which introduce artefacts as the phase differences cause cancellation of harmonics.

---

2. Receptive field here means that audio samples seconds apart share a large common fraction of the same noise input.

In Chapter 4 we discussed a vocoder that used the estimated glottal pulse signal as its input. This vocoder did not face the difficulty of creating an oscillation as it was already provided to it. We adopted a similar approach for the mel-spectrogram based vocoder of this chapter. However, as the pulse positions cannot be extracted from the mel-spectrogram, we will be using the fundamental frequency to generate the oscillation. This fundamental frequency is extracted from the mel-spectrogram and then used to operate a wavetable synthesizer. With this approach we can resynthesize the signal with a relatively small neural network.

## 5.4. Multi-Band Excited WaveNet

*The remainder of this section has only minor contribution from myself. It is a summary of Roebel and Bous (2022) and is provided for completeness since the remaining chapters build on the existence of a universal vocoder and audio sample are generated, using this mel-inverter. It cannot be seen as an independent contribution for this thesis.*

The Multi-Band Excited WaveNet (MBExWN) reflects the insights we gained in this chapter. The core features are:

**Externally generated oscillation** The Multi-Band Excited WaveNet generates its oscillation by estimating the fundamental frequency from the mel-spectrogram and synthesizing an excitation using a wavetable.

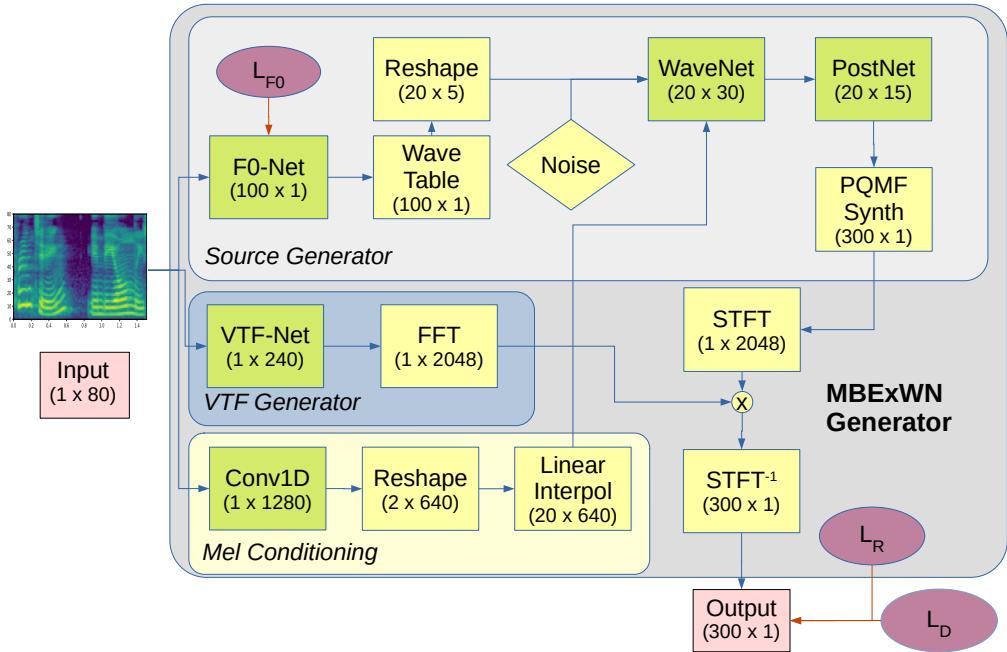
**Sample-rate reduction via PQMF** Computations with neural networks in the time domain are costly but are often redundant. Using the pseudo quadrature mirror filter bank (PQMF) the Multi-Band Excited WaveNet reduces the sample rate of the main component from 24 kHz to 1.6 kHz.

**Spectral domain filter** The final output is obtained through a filter implemented on the STFT. The filter is obtained through a neural network and allows creating narrow resonances without requiring long time consistency of the model.

A summary of the full architecture can be found in Fig. 5.1.

### 5.4.1. Network architecture

First a neural network (the *F0-Net* in Fig. 5.1) estimates the fundamental frequency from the mel-spectrogram. The network performs an upsampling by a factor of 100 to match the sampling rate of the later elements. The estimated  $f_0$  is used in a wavetable generator to produce the necessary oscillation. To prevent aliasing but provide enough harmonics at the same time the wavetable generator can choose from 13 band-limited wavetables with increasing number of harmonics, according to the input  $f_0$ , and averages over two adjacent wavetables to create smooth differentiable transitions. The wavetable generator generates a signal at a sampling rate of 8 kHz. To improve efficiency, the signal is reshaped such that each time step of the output signal contains five time steps of the input signal. Finally, random



**Figure 5.1.:** Schematic of the Multi-Band Excited WaveNet (MBExWN). Green boxes are neural networks, yellow boxes are differentiable DSP operations, red ovals are losses. The dimensions inside the boxes are denoting temporal  $\times$  spectral dimensions, where the temporal dimension is given as upsampling with regard to the input sample rate (here 80 Hz). [Illustration by Axel Roebel, from Roebel and Bous (2022)]

noise is concatenated with the pulse signal to allow mixing and switching between harmonic and noise components of the source.

The periodic excitation, together with the noise source are shaped through a WaveNet (Oord, Dieleman, et al., 2016) to create the glottal source signal. Like in Chapter 4, the WaveNet does not have to create an oscillation and thus neither needs to be recursive nor requires a large receptive field. Here the WaveNet consists of two stacks with five layers each. A final convolution ('PostNet' in Fig. 5.1) maps the output of the WaveNet to the necessary number of channels required for the next step. The WaveNet is conditioned on the mel-spectrogram. To match the number of channels inside the WaveNet, the mel-spectrogram is first mapped through a convolution that maps the mel-frequency axis to the required number of features. The WaveNet does not change the sample rate of its input. To map back from the 1.8 kHz to the required 24 kHz at the output, a pseudo quadrature mirror filter bank with 15 bands is used.

The synthesized source is filtered by a vocal tract filter. The vocal tract filter is estimated from the mel-spectrogram. The vocal tract filter estimator is a neural

network with five layers. It produces the cepstral coefficients of the vocal tract filter. Using the cepstral representation allows us to enforce minimum phase for the resulting filter by interpreting the coefficients as causal cepstral coefficients and to control the formant bandwidth by the number of cepstral coefficients. The filter allows creating narrow resonances without requiring long time consistency of the model.

#### 5.4.2. Training

Training the Multi-Band Excited WaveNet relies on three losses; a multi-resolution spectral loss (X. Wang et al., 2019) matches the signal to the target perceptually, the  $f_0$ -loss guides the  $f_0$  estimation and a discriminator improves the quality of the noise component.

When calculating the mel-spectrogram the phase is disregarded removing all time information inside a frame and in particular the pulse positions of voiced speech. Though some information about pulse positions might be inferred through inter-frame relations, in particular due to frame overlap, the information does not suffice to properly restore the pulse positions. Perceptually, this is not necessary as individual pulses of regular phonation are not perceived in voiced speech. The problem arises when we want to compare the synthesized signal with the original: As the phase difference between the two could be arbitrary, comparing the synthesized signal to the target signal in the time domain will not yield any useful information. Thus, we rely on a spectral loss to properly compare the two signals. When calculating the spectral loss, again, the phase is disregarded before comparing the magnitudes of the spectra. To best capture the differences, the spectrogram is calculated with different window lengths thus providing us with multiple resolutions.

The estimated fundamental frequency can and must be trained with existing annotations. This approach is a double-edged sword. Ideally we would like the network to figure out the best matching  $f_0$  by itself: First of all the  $f_0$  annotations are not perfect but above all the existing  $f_0$  models do not work well for speech boundaries where pulses happen irregularly. One of the reasons why we do not simply use a parametric vocoder is because the underlying assumptions are inaccurate. We utilize deep learning not to rely on parameters like the  $f_0$  and its analysis tools that rely on these assumptions and let the dependencies be modelled by the data instead. On the other hand it is necessary to provide the fundamental frequency as it will not be able to find the correct  $f_0$  from random initialization. The problem is that using a spectral loss does not produce a useful gradient for the  $f_0$  if the  $f_0$  is too far off: The spectral loss defines how spectrum needs to be adjusted along the energy axis, the  $f_0$ , however, is in a sense orthogonal to this, as changes in the  $f_0$  move the spectrogram along the frequency axis. If a partial of the synthesized signal is close to a partial in the target signal, the spectral loss may move the  $f_0$  such that the partials move closer together, regardless whether these are the same partials or not. Thus, the spectral loss matches harmonics from the synthetic signal to harmonics of the target signal that do not necessarily belong together. In particular, the direction to the closest partial in the target signal could be different for each partial, such

that the net gradient is cancelled out. As a compromise we train the F0-Net using the auxiliary  $f_0$ -loss, but apply the  $f_0$ -loss only on segments of stable phonation. In this context, anything more than 50 ms away from voice/unvoiced boundaries, is assumed to have a stable fundamental frequency. For any other regions only the spectral loss has an influence on the generated  $f_0$ .

The multi-resolution spectral loss can perceptually capture the deterministic part of the signal. The noise component, on the other hand, cannot be evaluated so well with this method. Perceptually we cannot distinguish two noise signals well if the spectral density is the same. The spectrogram of a noise signal, however, does not estimate the spectral density well. Instead, we obtain a noisy version of the spectral density which is different for each realization of the random signal. Due to the unpredictability of the noise input, the synthesizer is inclined to disregard the noise input and to simulate the noise part with complex but deterministic patterns instead. To increase the quality of the noise, an adversarial strategy is used to teach the synthesizer to generate realistic noise signals. A discriminator, operating on two different time-resolutions (Jang et al., 2020), is trained to distinguish synthesized and real voice on their waveforms. The two discriminators work on different sample-rates to provide different viewpoints to the audio waveform. The discriminators help to make a difference between somewhat acceptable quality and almost transparent synthesis but during the initial stages of training the information provided by the discriminators is rather misleading and slows down or even prohibits convergence. We thus perform training in three steps: First, the F0-Net is pre-trained on the mel-spectrogram with the  $f_0$  annotations alone. Then the whole synthesizer (without the discriminator) is pre-trained, using only the spectral- and the  $f_0$ -loss initializes the weights to already produce proper voice. Afterwards, during the fine-tuning, the synthesizer is trained together with the discriminators using all losses.

#### 5.4.3. Discussion

##### Perceptive evaluation

The Multi-Band Excited WaveNet was evaluated against Multi-Band MelGAN (G. Yang et al., 2021) and Universal MelGAN (Jang et al., 2020) in a perceptive test for Roebel and Bous (2022). An excerpt of the results is shown in Tables 5.1 and 5.2. From these tables we can observe the following: On speech, the voice model MW-VO, trained on both speech and singing, performs worse than the domain specific speech model MW-SP, trained only on speech. On singing however, training on both speech and singing, improves the quality compared with the model MW-SI, trained only on singing. For singing voice, the voice model MW-VO performs best on both seen and unseen singing, while on speech the speech model MW-SP performs comparable with existing state-of-the-art neural vocoders. The perceptive tests have shown that the quality loss through analysis-synthesis is very small for regular speech and singing, even for unseen voices. The mel-inverter struggles with rough and creaky voice and in particular with growling. These are cases where the assumption of a fundamental frequency is violated. For these special cases further treatment is necessary to generate irregular pulses or allow the use of sub-harmonics.

**Table 5.1.:** Mean opinion score for MBExWN and other baselines on speech: Evaluation on LJ Speech (see Table 2.1, Page 32), and the evaluation data from the demo website<sup>†</sup> of Jang et al. (2020) using the samples from LJ Speech (UMG(LJ)) and other samples (UMG(other)). Values on a scale from 0 to 100. [From Roebel and Bous (2022), Table 4]

|             | HREF         | MW-SP        | MW-VO         | UMG-LJ+LT     | MMG-LJ       |
|-------------|--------------|--------------|---------------|---------------|--------------|
| LJ          | $94 \pm 1.9$ | $85 \pm 4.3$ | $82 \pm 5.2$  | —             | $86 \pm 4.1$ |
| UMG(LJ)     | $93 \pm 3.7$ | $86 \pm 6.8$ | $77 \pm 11.4$ | $80 \pm 10.6$ | —            |
| UMG(others) | $94 \pm 2.2$ | $78 \pm 6.4$ | $74 \pm 7.5$  | $81 \pm 6.3$  | —            |

**Table 5.2.:** Mean opinion score for MBExWN and other baselines on singing: Evaluation on ICH(see Table 2.2, Page 34), samples of pop singing from MusDB (Rafii et al., 2017) (Pop(MusDB)) and samples of rough singing voice (Rough). Values on a scale from 0 to 100. [From Roebel and Bous (2022), Table 5]

|            | HREF         | MW-SI         | MW-VO         | MMG-ICH      |
|------------|--------------|---------------|---------------|--------------|
| ICH        | $94 \pm 2.7$ | $83 \pm 8.4$  | $86 \pm 6.2$  | $85 \pm 5.5$ |
| Pop(MusDB) | $93 \pm 2.7$ | $73 \pm 8.3$  | $79 \pm 8.1$  | —            |
| Rough      | $96 \pm 2.7$ | $60 \pm 12.7$ | $68 \pm 11.3$ | —            |

**HREF** Hidden reference

**MW-SP** Multi-Band Excited WaveNet trained only on speech

**MW-SI** Multi-Band Excited WaveNet trained only on singing

**MW-VO** Multi-Band Excited WaveNet trained on speech and singing

**UMG-LJ+LT** Universal MelGAN (Jang et al., 2020) trained on LJ speech and Libri-TTS

**MMG-LJ** Multiband MelGAN (G. Yang et al., 2021) trained on LJ speech.

**MMG-ICH** Multiband MelGAN (G. Yang et al., 2021) trained on ICH dataset

<sup>†</sup> <https://kallavinka8045.github.io/icassp2021/>

#unseen-domains-speaker-emotion-language-1

### Computational speed

The Multi-Band Excited WaveNet allows us to efficiently map between time-domain audio and mel-spectrogram. It is able to generate 50 000 audio samples per second on a single CPU core thus allowing real-time processing (with latency).

## 5.5. Conclusion

In this chapter we have developed the vocoder framework that we will use to perform voice transformations in. The development of a new vocoder model was motivated by the new possibilities in synthesis due to neural networks. These new possibilities allowed us to reinvent the vocoder using a parametric space, the mel-spectrogram that has long been thought to be insufficient to fully describe the voice. However, more and more work appeared presenting methods to resynthesize voice from mel-spectrogram with almost no perceptual difference. Together with the results of Section 5.3, it has become clear that the mel-spectrogram can be used in a universal neural vocoder for a sufficiently large variety of voice signals.

Today a large quantity of mel-inversion algorithms exist, all with different strengths and drawbacks. We will keep using the Multi-Band Excited WaveNet throughout the remainder of this thesis because it is available to us, relatively easy to retrain on new data and extremely fast during synthesis. Due to its universality, however, the voice transformations proposed in the next chapters, are independent of the mel-inverter which would permit using any other mel-inversion technique that allows inversion of arbitrary mel-spectrogram containing voice.

For the remainder of this thesis we may assume time-domain audio and mel-spectrogram to be equivalent.

## 6. Bottleneck auto-encoder

After extensive considerations we finally have settled for the mel-spectrogram as the parametric space of our neural vocoder. Having a means to freely map between mel-spectrogram and raw audio, it remains to develop transformations on the mel-spectrogram. We will see in the following chapters that the mel-spectrogram is indeed a very suitable representation of the voice for the task of transformation.

The tool of choice for the voice transformations will be the **bottleneck auto-encoder**. The auto-encoder is a type of neural network architecture that is trained to reproduce its own input. Two neural networks, the Encoder, and the Decoder, are cascaded and trained to act as inverses of each-other. The auto-encoder produces a latent space between encoder and decoder where the input signal is re-parametrised. The study of auto-encoders is concerned with influencing the auto-encoder in a way that the latent space inhibits certain desirable properties. In our particular case, we will train the auto-encoder to produce a latent space that is independent (disentangled) from the pitch (this chapter) and also from the voice level (Chapter 7).

We will develop the bottleneck auto-encoder with the transformation of the pitch. Pitch, as encoded through the fundamental frequency is easily available for any voice recording thanks to extensive previous work on the subject (see Section 3.1.1, Page 48). These methods work reasonably well so that we have reliable annotations which will allow us to concentrate on the auto-encoder architecture itself without having to worry too much about the underlying data. Furthermore, the result can be evaluated both systematically by the existing  $f_0$  estimation tools and subjectively by looking at the mel-spectrogram or listening to the resynthesis: The fundamental frequency is visible in the mel-spectrogram so that we can see whether and where the method is working or not. Similarly, our hearing is well-trained on recognizing pitches to that we can evaluate the result by hearing. Applications for pitch transformations are plentiful and the two motivating applications for us were the artistic residency with Judith Deschamps and the IRCAM Singing Synthesizer.



Article



Audio samples

Article: <https://www.mdpi.com/2078-2489/13/3/102>  
Audio Samples: <http://thesis.fnab.xyz/ch6/demo>

This chapter was originally published under the title *A Bottleneck Auto-Encoder for F0*

*Transformations on Speech and Singing Voice* (Bous and Roebel, 2022) in the open-access journal *Information* by the Multidisciplinary Digital Publishing Institute (MDPI).

## 6.1. Introduction

Since the invention of the vocoder over 80 years ago by Dudley (1939), people have been studying means to transform the acoustical properties of speech recordings. Since then, enabling transparent pitch transformations has been a prominent topic inside the speech processing research community and attempts to achieve this were plentiful. Prominent approaches include the phase vocoder (Flanagan and Golden, 1966), pitch synchronous overlap and add (Moulines and Charpentier, 1990), shape-invariant additive models (Quatieri and McAulay, 1992), shape-invariant phase vocoder (Roebel, 2010), harmonic plus noise model (Laroche et al., 1993; Stylianou et al., 1995), parametric vocoders (Kawahara, 2006; Morise et al., 2016), or extended source-filter models (Degottex, Lanchantin, Roebel, et al., 2013; Huber and Roebel, 2015). Recently, as speech signal processing has become dominated by deep-learning based techniques, the community has focused on more abstract, high-level voice properties, like speaker identity (Desai et al., 2009; Kameoka et al., 2018; J.-X. Zhang et al., 2019; Qian, Yang Zhang, Chang, X. Yang, et al., 2019; Qian, Jin, et al., 2020; Ferro et al., 2020) but also emotion (C. Robinson et al., 2019; Moine, Obin, and Roebel, 2021) and accent (Zhao et al., 2018). These methods may change the fundamental frequency implicitly as part of their transformation. Nevertheless, explicit transformation of pitch remains an interesting and challenging research topic, in particular in artistic applications.

Musical pitch and speech melody are determined by the fundamental frequency  $f_0$  of the quasi-periodic oscillation during phonation. In singing voice, the  $f_0$  does not only carry the melody but also the singer's individual expression used to interpret the piece which is highly dependent on the genre of the piece (Umbert, Bonada, Goto, et al., 2015). In speech the  $f_0$  plays a different role than in singing. It carries important information such as mood, intent, and identity (Salais et al., 2022). The  $f_0$  contour is strongly related with expressivity (Veaux and Rodet, 2011), and editing the  $f_0$  contour may therefore be used to manipulate the expressivity of a given voice signal.

### 6.1.1. Applications

For singing synthesis, it does not suffice to simply map the notes to their corresponding frequencies and durations. Singing synthesizers need to translate the required melody to an  $f_0$  curve to convey the expression and style required by the score (Umbert, Bonada, and Blaauw, 2013; Ardaillon, Chabot-Canet, et al., 2016). A common approach is unit-selection with  $f_0$  modification as is done by Ardaillon, Degottex, et al. (2015) or Bonada et al. (2016). Thus, transforming the  $f_0$  of singing voice is crucial for parametric concatenative singing synthesis and could also help refine existing recordings. The quality of these synthesizers is thus limited by the quality of the transposition algorithm. Combining singing units in mel-spectral

representation with the neural  $f_0$  transformation introduced in this chapter, along with our mel-inverter MBExWN described in [Chapter 5](#) could allow creating a neural singing synthesizer.

As for voice modification,  $f_0$  transformation can also be used on real singing recordings to alter the intonation, expression, and even melody in post-production. The  $f_0$  transformation algorithm presented here could also expand the range of a singer. If pushed far enough, thanks to the implicit adaptation of voice features, it will create a coherent new personality.

One of the most prominent differences between the sexes is the mean  $f_0$ , due to different rates of growth of the vocal folds in puberty. Thus changing the  $f_0$  in speech can result in changing or obfuscating the speaker's gender (Farner et al., 2009), an effect which could be used in scientific studies to research the effect of perceived gender in the voice in social interactions (Arias et al., 2021).

### 6.1.2. Challenges of pitch transformation

The fundamental frequency has an effect on most other voice properties. A mismatch during synthesis is often perceived as unnatural (Degottex, Roebel, et al., 2011). Due to the inherent complexity of the relationships between  $f_0$ , glottal pulse and vocal tract filter, we think that these effects can best be modelled with data-driven approaches. While deep-learning has successfully been applied to numerous similar domains, the transformation of pitch has not been addressed adequately since widespread accessibility of deep-learning. In this chapter we will address the issues that cannot be addressed adequately by classical parametric vocoders. As discussed in [Chapter 5](#), using the mel-spectrogram as parametric space allows us to train a neural network to model the dependency between  $f_0$  and other voice properties more efficiently than using a classical parametric vocoder space.

### 6.1.3. Mathematical description of disentanglement

In the literature, disentanglement is commonly approached by a full disentanglement of explaining factors (Higgins, Amos, et al., 2018). In this approach independent factors are learnt unsupervised. The resulting parameters can be manipulated independently but might not correspond to meaningful parameters or to parameters of interest. In our applications we have a specific property that we want to change. We are not interested in a complete description using disentangled factors but rather disentangling individual factors from the remaining parameter set. Thus, here we approach disentanglement from a slightly different angle.

We define disentanglement through stochastic independence on a parametric feature space  $\mathbf{F} = \mathbf{F}_1 \times \dots \times \mathbf{F}_n$ . Let  $\mathbf{X}$  be the space of all finite length signals

$$\mathbf{X} = \bigcup_{n=1}^{\infty} \mathbb{R}^n. \quad (6.1)$$

Furthermore, let  $(\Omega, \mathcal{F}, \mathbf{P})$  be a probability space, and let  $(F_k : \Omega \rightarrow \mathbf{F}_k, k = 1, \dots, N)$

be a finite set of voice features.<sup>1</sup> Assume there exists a synthesis function  $s$  such that

$$s : \mathbf{F}_1 \times \cdots \times \mathbf{F}_N \rightarrow \mathbf{X} \quad (6.2)$$

$$V(\omega) = s(F_1(\omega), \dots, F_N(\omega)). \quad (6.3)$$

Assuming there exist probability densities for  $(F_1, \dots, F_N)$ , we can evaluate a signal by the likelihood of its features. This defines the signal's likelihood

$$L : \mathbf{X} \rightarrow \mathbb{R}_0^+ \quad (6.4)$$

$$L(s(f_1, \dots, f_N)) := p_{F_1, \dots, F_N}(f_1, \dots, f_N). \quad (6.5)$$

If all features  $(F_k)$  are mutually independent, we say that  $\mathbf{F}$  is *disentangled*. Then the likelihood  $L$  can be written as the product of the likelihoods of the individual features:

$$L(s(f_1, \dots, f_N)) = \prod_{k=1}^N p_{F_k}(f_k) \quad (6.6)$$

In this case changing one feature changes the overall likelihood of the voice signal only with regard to changes in the likelihood of that feature. Assume without loss of generality we change the value of the first feature from  $f_1$  to  $f'_1$ . Then the voice signal changes from  $v = s(f_1, \dots)$  to  $v' := s(f'_1, \dots)$ . Accordingly, it's likelihood changes as

$$L(v') = L(v) \cdot \frac{p_{F_1}(f'_1)}{p_{F_1}(f_1)}. \quad (6.7)$$

This property ensures that changing a feature to a reasonable value will not result in improbable, unnatural voice.

Generally, however, different voice features are not independent, and thus the feature space is not disentangled. This means that (6.7) does not hold, and often we find

$$L(v') \ll L(v) \cdot \frac{p_{F_1}(f'_1)}{p_{F_1}(f_1)}. \quad (6.8)$$

In this case, for a reasonable change in the voice property, the changed signal can become very unlikely ( $L(v')$  may be even 0) and may no longer represent a real world voice signal. This is a problem as changing one voice feature may have unpredictable and undesirable results, which is an issue with the parametric spaces of classical vocoders. Therefore, disentanglement aims at a re-parametrization of  $\mathbf{F}$  through the invertible mapping  $d : \mathbf{F} \rightarrow \mathbf{D}$ . The goal of disentanglement is to find such a mapping  $d$ , such that Eq. (6.7) holds for its subcomponents  $\mathbf{D}_1, \dots, \mathbf{D}_M$ . Note that the number  $M$  of subcomponents of  $\mathbf{D}$  does not necessarily need to be the same as the number  $N$  of subcomponents of  $\mathbf{F}$ .

---

1. Here we use the term ‘feature’ instead of ‘property’ to match the variable name  $f$ . We use  $f$  instead of  $p$  (for property) to avoid a naming clash with variables related to probabilities. The terms ‘voice property’ and ‘voice feature’ can be thought of synonyms in this context.

If we want to change a feature, it has to be a subspace of the disentangled space  $\mathbf{D}$ . Due to the re-parametrization  $d$ , however, features may be lost in  $\mathbf{D}$ . Thus, for disentanglement to be useful, we pose the additional constraint that the features we want to change must be still present in  $\mathbf{D}$ . If two features  $F_1$  and  $F_2$  are not independent, both cannot be in  $\mathbf{D}$ . Thus, for practical reasons, unless two features are already independent, we disentangle into the space  $\mathbf{D} = \mathbf{F}_1 \times \mathbf{D}'$  with the independent subcomponents  $F_1$  and  $d'(F_1, \dots, F_N)$ . In this case we say, *feature 1 has been disentangled (from the other features)*.

If we still want to be able to change two features, two options arise.

1. Find two disentanglements  $d_1$  and  $d_2$  that disentangle feature 1 and feature 2 respectively, and transform the features one after the other. In this case the feature that was transformed first will not be preserved, but we may hope that it will still remain close to what we wanted.
2. Model  $F_1$  and  $F_2$  as a joint feature  $F_{1,2} = (F_1, F_2)$  and disentangle  $F_{1,2}$  from all the remaining features. In this case Eq. (6.7) holds for the joint distribution  $p_{F_1, F_2}$ , and it is up to the user to provide combinations of  $f_1$  and  $f_2$  that make sense in the context of voice.

#### 6.1.4. Goals of this chapter

This chapter introduces the neural voice transformation framework, which is the central tool of this thesis. The neural voice transformation framework is a framework that allows disentanglement and consequently transformation of aligned signal properties. We introduce the neural voice transformation framework using the fundamental frequency  $f_0$  as an example property as transformation of the  $f_0$  is a well-established and well-studied problem. Thus, we will address the free modification of the fundamental frequency  $f_0$  in recordings of speech and singing voice. The algorithm requires only minimal input from the user and allows one to change the  $f_0$  by means of providing a fixed transposition value or redrawing the  $f_0$  curve.

The transformation system is based on the bottleneck auto-encoder of AutoVC (Qian, Yang Zhang, Chang, Hasegawa-Johnson, et al., 2020). Qian, Yang Zhang, Chang, Hasegawa-Johnson, et al. (2020) use the part of speech melody that is independent of rhythm, timbre, and content. In particular, the  $f_0$  is normalized by mean and variance for each speaker. Contrary to that, here we use the non-normalized, absolute  $f_0$  as control parameter. Therefore, the control parameter for our auto-encoder contains some information about the speaker identity and timbre. Changing the raw  $f_0$  signal may change the component of speaker identity and timbre that depends on the  $f_0$ . The system should apply the given  $f_0$  contour onto the output signal such that the output signal contains exactly the same  $f_0$  contour. This implies that parts of the identity will change if the change is incompatible with the speaker identity.

Two main goals arise for a potential  $f_0$  transformation system:

1. The desired  $f_0$  contour should be followed exactly.

2. The result should sound like a human.

The first goal, the  $f_0$  accuracy, can be evaluated objectively by resynthesizing the transformed **mel-spectrogram** and checking that the  $f_0$  in the resynthesized audio matches the target  $f_0$ . Our findings are stated in [Section 6.3.1](#). The second objective, the naturalness of the syntheses, can only be evaluated by listening to the generated samples. Therefore, we have performed a perceptive study by asking a selected group of participants to listen to synthesized and non-synthesized audio and give a perceptive rating. The results of the perceptive test are presented in [Section 6.3.2](#).

One crucial hyperparameter of the bottleneck auto-encoder is the size of the bottleneck. We therefore perform a systematic analysis of the dependence between disentanglement and bottleneck size and present the results in [Section 6.3.1](#).

### 6.1.5. Outline

The remainder of the chapter is structured as follows. In [Section 6.2](#) we introduce the auto-encoder setup, its architecture and how to train it, the datasets used in the experiments and the experimental setup itself. In [Section 6.3](#) we present the results from a perceptive test and a transformation accuracy analysis. We compare the proposed system to other  $f_0$  transformation algorithms, as well as a visual analysis of the latent code for the case where the bottleneck size is 2. A conclusion and an outlook for future work will be found in [Section 6.4](#).

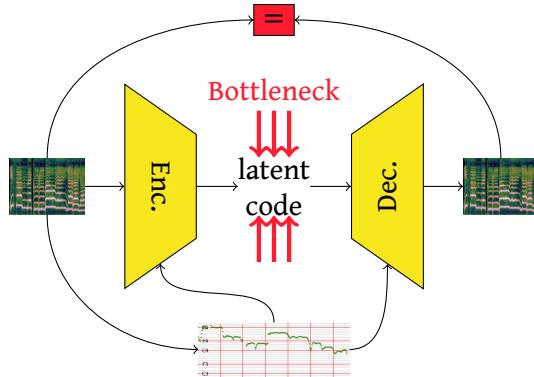
## 6.2. Methodology

### 6.2.1. Proposed model

The auto-encoder operates on the **mel-spectrogram**. To change the  $f_0$  we first extract the mel-spectrogram and the  $f_0$  from the original audio recording. Then we use the encoder to produce a latent code from the **mel-spectrogram** and the original  $f_0$ . We resynthesize the **mel-spectrogram** by applying the decoder to the latent code and the target  $f_0$ . Finally, the transformed audio is obtained by inverting the transformed mel-spectrogram using the neural vocoder **MBExWN**, which we have presented in [Chapter 5](#). [Figure 6.1](#) gives an overview of the idea of the bottleneck auto-encoder setup.

#### Input data

The mel-spectrograms are computed on 24 kHz audio. We use 80 mel-bands to represent the frequencies between 0 and 8 kHz. Frequencies above 8 kHz are discarded and expected to be completed coherently by the final mel-inverter. The mel-spectrogram has a hop size of 12.5 ms resulting in a 80-dimensional signal with 80 Hz sample rate. The **mel-spectrogram** is given with logarithmic amplitudes. The



**Figure 6.1.:** Schematic overview of the *bottleneck auto-encoder* idea. The encoder and the decoder are trained jointly to regenerate the original mel-spectrogram using a reconstruction loss. The decoder receives the  $f_0$  as additional information. The information bottleneck ensures that the encoder removes the  $f_0$  from the latent code and the decoder uses the  $f_0$  from the conditional input. Additionally conditioning the encoder on the  $f_0$  helps the encoder to identify the information that needs to be removed.

$f_0$  curve is provided as a number pair ( $\log(f_0)$ , voiced/unvoiced mask) and is generated from the audio signal by the  $f_0$  estimation algorithm, F0-FCN from Ardaillon and Roebel (2019).

To obtain normalized input data we rescale the input data such that the interval  $[-1, 1]$  represents  $[-120 \text{ dB}, -20 \text{ dB}]$  for the mel-spectrogram and  $[45 \text{ Hz}, 1400 \text{ Hz}]$  for the fundamental frequency  $f_0$ .

### Network architecture

The auto-encoder is implemented as a fully CNN where the encoder and decoder are resembling mirrored structures. To benefit from the natural topology of the frequency axis we treat the mel-spectrogram as an image and perform 2D-convolutions on them, which facilitates training according to Bous and Roebel (2019). Through strides in frequency direction, the encoder reduces the frequency-axis to length 1 but keeps the time-axis unchanged. The decoder uses transposed convolutions to incrementally rebuild the frequency-axis. The exact architecture can be taken from Tables A.3 and A.4 (Page 161).

In the encoder, we start with an *image* with the structure *time*  $\times$  *frequency*. This is stacked with the  $f_0$  signal, which is resampled and extended on the frequency axis to match the size of the mel-frequency axis with a trainable linear projection (cf. Table A.3, Page 161). We then reduce the size of the frequency axis step-by-step through down-sampling until it has length 1. Down-sampling is performed with stride convolutions across the frequency dimension, and the down-sampling operations are intercalated with classic  $3 \times 3$  convolutions. The last layer of the encoder, a  $3 \times 1$  convolution, maps the features to the desired bottleneck size.

The decoder has the inverse structure with respect to the encoder. We start

with the concatenation of the latent code and the target  $f_0$  curve and broaden the frequency axis step-by-step by up-sampling. Up-sampling is performed through transposed convolutions. The final convolution maps the number of features back to 1 recreating the shape of the input mel-spectrogram.

All convolutions use zero-padding to keep the shapes consistent. Activation functions are ReLU (Nair and Hinton, 2010). Encoder and decoder have 10 and 11 convolutional layers respectively, where each layer has 512 filters. The models have roughly 12.5 M parameters each.

Contrary to AutoVC (Qian, Yang Zhang, Chang, X. Yang, et al., 2019), there is no temporal down-sampling in our approach. As AutoVC does speaker identity conversion, the conditional input is constant in time. Thus, there a precise temporal resolution is not important. Furthermore, the temporal down-sampling allows the bottleneck auto-encoder of Qian, Yang Zhang, Chang, X. Yang, et al. (2019) to adjust timings, which may render the voice identity conversion more natural. Here, however, we condition on the  $f_0$  signal, which varies over time, and aim to preserve the temporal structure implied by the  $f_0$  signal as precise as possible. Similarly, we use convolutional layers instead of recurrent, to focus the neural network to use more local features.

We also experimented with 1D-convolutions but found the results' quality to be inferior with respect to the 2D-versions. We tried growing the number of features after each down-sampling of the frequency-axis in the encoder, while equally decreasing the number of features in the decoder. However, we found no significant difference in the results when keeping the number of parameters and inference speed similar. Thus, we opted to keep the same number of features for each layer, regardless of the size of the frequency-axis.

### Training procedure

The auto-encoder is trained only on self-reconstruction. We apply a point-wise mean-absolute-error loss between the input mel-spectrogram and the output of the decoder. Optimization is performed with Adam (Kingma and Ba, 2014), with a learning rate of  $1 \cdot 10^{-4}$  and the parameters  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . We use a batch size of 16 training-samples per batch with 80 time steps (equivalent to 1 s) per sample. All models were trained for 250 k updates.

#### 6.2.2. Datasets

In our experiments we use four different datasets:

- si** A pure singing-voice dataset, obtained by combining several publicly available datasets and our own proprietary datasets (**CREL**, **NUS-SMC**, **ICH**, **PJS**, **JVS-MuSiC**, **Tohoku**, **VocalSet**, **ISiS**, and **Farinelli** (see Table 2.2, Page 34).)
- sp** A pure speech dataset consisting of the English dataset **VCTK** (Yamagishi et al., 2019) and the French dataset **Att-HACK** (Moine and Obin, 2020)
- vo** A hybrid dataset of speech and singing, consisting of the two datasets above

**sp-xs** A smaller speech dataset, consisting of a reduced number of **VCTK** speakers, to match the duration of the singing-voice dataset

The singing-voice dataset is a combined dataset of **CREL**, **NUS-SMC**, **ICH**, **PJS**, **JVS-MuSiC**, **Tohoku**, **VocalSet**, **ISiS**, and **Farinelli** (see Table 2.2, Page 34). In total, the singing-dataset contains 26.5 h of singing voice from 136 different singers, however 100 of them are from **JVS-MuSiC** and 94% of the total duration is sung by the remaining 36 singers. Most of the singers in the dataset are professionals, but a few of them are also amateurs. For datasets which also contained speech, only the singing recordings were used. The dataset contains singing in English, French, Greek, Italian, Japanese, and Latin.

For speech, we train on both **VCTK** (Yamagishi et al., 2019) and **Att-HACK** (Moine and Obin, 2020) which amounts to a combined dataset of English and French speech. The speech dataset contains a total of 134 speakers (109 from **VCTK**, 25 from **Att-HACK**) and 54 h of speech (32 h from **VCTK**, 22 h from **Att-HACK**).

To better compare the results between speech and singing voice, we also train models on a smaller speech dataset, consisting of only a subset of the **VCTK** speakers, resulting in 26.5 h of speech.

### Dataset splits

For evaluation, we withhold four speakers from the **VCTK** dataset in the following section, two female and two male speakers. These are p361, p362, p374 and p376. Thus, all evaluation of the speech models is performed on unseen speakers.

Test set for singing voice, consists of five samples each from a set of different singers, aimed to include a representative of each voice class. In particular the test set includes baritone, tenor, mezzo-soprano, soprano, child voice and countertenor, and singing style, including classical, pop, j-pop and Byzantine singing.

#### 6.2.3. Nomenclature

To distinguish the models we use the following nomenclature: A model is named after the dataset it is trained on, e.g. **si** (singing), **sp** (speech) or **vo** (voice, speech + singing). The bottleneck size is given in the subscript. Additionally, the size of the model may be given by the number of features in the hidden layers indicated in the superscript. If no superscript is given, the default model size of 512 is implied. Thus, the model **si<sub>3</sub><sup>256</sup>** is a singing model with bottleneck size  $n_b = 3$  and 256 features in the hidden layers. The model **vo<sub>6</sub>** has been trained on both speech and singing, uses a bottleneck size of  $n_b = 6$  and has 512 features in the hidden layers. A table with all models trained for this chapter is given in Table 6.1, and explained in more details in the section below.

#### 6.2.4. Experimental setup

The variety of sounds that can be produced with the human voice is rich and human voice is used in different contexts. Often a distinction is made between speaking

**Table 6.1.:** Final loss values (training (validation)) for the models used in Section 6.3. All values in dB. Nomenclature according to Section 6.2.3.

| $n_b$ | $\text{si}_{n_b}$ | $\text{sp}_{n_b}$ | $\text{sp-xs}_{n_b}$ | $\text{vo}_{n_b}$ | $\text{sp}_{n_b}^{256}$ | $\text{sp}_{n_b}^{128}$ |
|-------|-------------------|-------------------|----------------------|-------------------|-------------------------|-------------------------|
| 1     | 3.13 (3.46)       | 3.52 (3.55)       |                      |                   |                         | 3.68 (3.79)             |
| 2     | 2.47 (2.91)       | 2.84 (2.89)       | 2.90 (3.36)          | 3.06 (3.17)       | 2.99 (3.03)             | 3.09 (3.12)             |
| 3     | 2.41 (2.70)       | 2.51 (2.55)       |                      | 2.61 (2.72)       |                         | 2.85 (2.84)             |
| 4     | 2.06 (2.44)       | 2.36 (2.39)       | 2.42 (2.74)          | 2.54 (2.64)       | 2.48 (2.51)             | 2.68 (2.68)             |
| 5     | 2.13 (2.45)       | 2.19 (2.21)       |                      |                   |                         | 2.47 (2.46)             |
| 6     | 2.03 (2.29)       | 2.08 (2.11)       | 2.14 (2.38)          | 2.30 (2.39)       | 2.16 (2.19)             | 2.32 (2.33)             |
| 7     |                   | 1.96 (1.99)       |                      |                   |                         | 2.24 (2.24)             |
| 8     |                   | 1.87 (1.90)       | 1.94 (2.17)          |                   | 1.97 (2.00)             | 2.18 (2.21)             |
| 9     |                   | 1.83 (1.85)       | 1.88 (2.14)          |                   | 1.91 (1.93)             | 2.10 (2.10)             |
| 10    |                   | 1.75 (1.76)       | 1.81 (1.98)          |                   | 1.83 (1.85)             | 1.97 (1.96)             |
| 11    |                   | 1.69 (1.71)       |                      |                   | 1.76 (1.77)             | 1.95 (1.95)             |
| 12    |                   | 1.64 (1.66)       | 1.67 (1.87)          |                   | 1.71 (1.73)             | 1.92 (1.92)             |
| 13    |                   | 1.56 (1.58)       |                      |                   |                         | 1.89 (1.89)             |
| 14    |                   | 1.52 (1.54)       | 1.56 (1.75)          |                   | 1.63 (1.65)             | 1.85 (1.85)             |
| 16    |                   |                   |                      |                   | 1.53 (1.54)             |                         |

$n_b$  Bottleneck size

**si** Singing model (trained on singing dataset)

**sp** Speech model (trained on speech dataset)

**sp-xs** Speech model (trained on the small speech dataset)

**vo** Voice model (trained on speech and voice)

**<sup>256, 128</sup>** Number of features in the hidden layers (default is 512)

and singing voice due to their different acoustic properties and applications. We compare two different approaches: either to train separate models for speech and singing or to train a combined voice model using training data of speech and singing voice. As speech and singing have their differences, training a model only on one domain may improve its quality as the complexity of the task is reduced. On the other hand, speech and singing share many similarities; training a universal model might allow using overlapping information and improve overall quality as a consequence. We will see, however, that for our application this is not the case neither for  $f_0$  accuracy nor for audio quality.

Since this is the first time the  $f_0$  is used as the only feature that is manipulated in a neural network, we cannot compare the proposed method to other neural-based baselines. However, the repertoire of classical signal processing based  $f_0$  transformation algorithms is rich, and so we compare our auto-encoder with the PaN vocoder (Ardaillon, 2017), a variant of the parametric vocoders built upon the **Liljencrants–Fant pulse model** (Fant, Liljencrants, et al., 1985; Fant, 1995).

Since the size of the information bottleneck is the crucial part of the disentangle-

ment, we train models with different values for the bottleneck size. In our study about the effect of bottleneck size we also vary the count of filters in the convolution operations from the set {128, 256, 512} Unless explicitly stated, the count of filters is 512.

The loss values for the models we trained for this chapter can be found in [Table 6.1](#). We can see that for every configuration the final loss decreases with the increase of bottleneck size. For *singing* and *speech, small dataset* we observe substantial overfitting due to the smaller dataset size; however, while the validation error is significantly larger than the training error, it did not increase over time. For the configurations trained on the full speech dataset almost no over-fitting can be observed.

To allow the reader to get their own perceptive impression on the proposed method, we created a [website<sup>2</sup>](#) with audio examples from the models used in the studies discussed below.

## 6.3. Experimental results

### 6.3.1. $f_0$ accuracy

To check if the proposed method actually changes the  $f_0$  we apply the transposition algorithm to our test sets and measure the difference between requested the  $f_0$  and the  $f_0$  extracted from the synthesized audio.

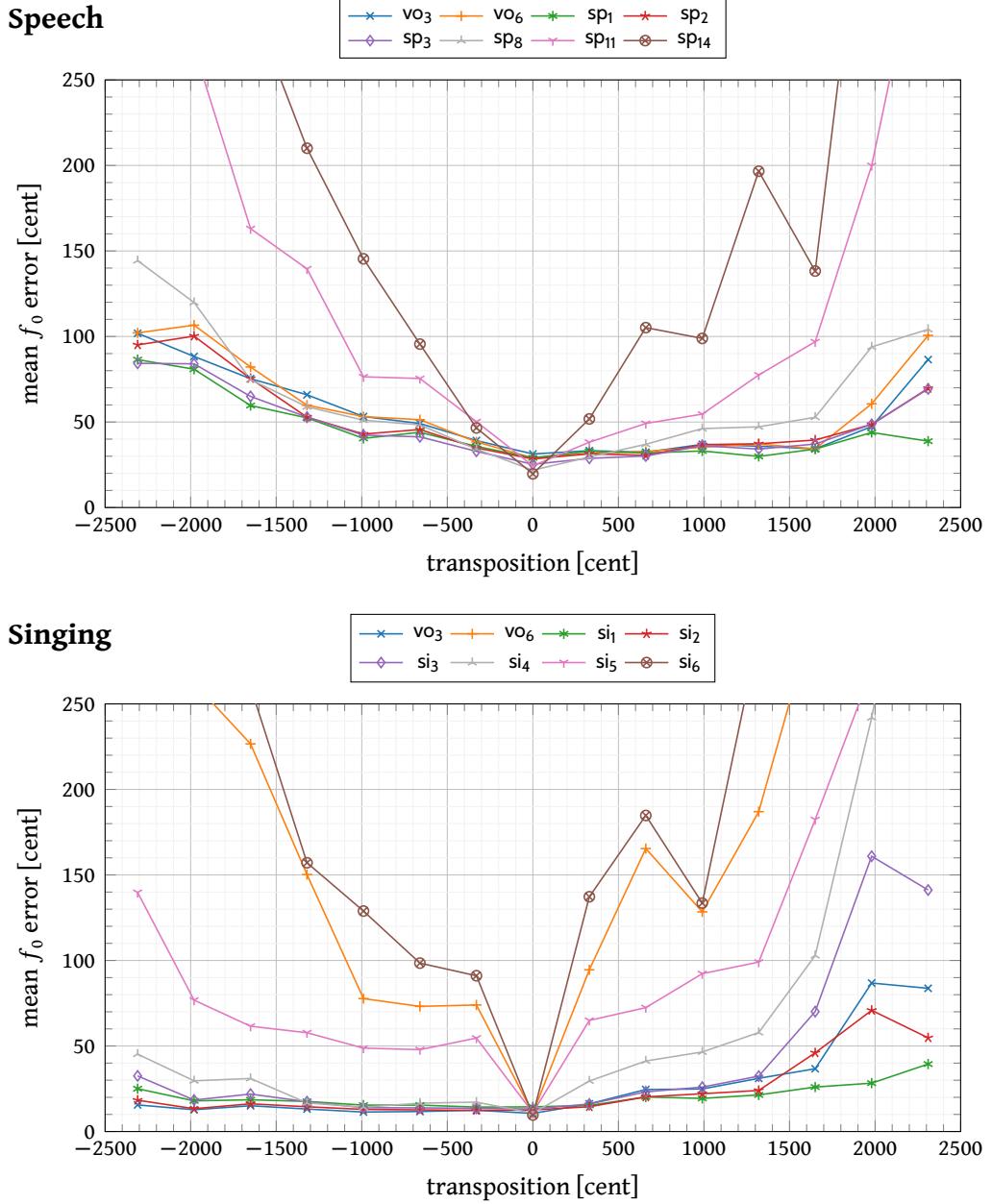
Since our algorithm learns the correspondence between  $f_0$  and mel-spectrogram from the training data, it is not able to produce mel-spectrograms with  $f_0$  values that are not contained in the training dataset. We therefore limit the transpositions to files from the test set such that the target  $f_0$  lies between 53.8 Hz and 377 Hz for speech and between 105 Hz and 746 Hz for singing. These ranges were chosen based on the distribution of  $f_0$  in the dataset, such that only 1% of the  $f_0$  values from the training set lie below the lower limit and only 1% above the upper limit.

[Figure 6.2](#) plots the mean absolute  $f_0$  error against the transposition in cent for speech and singing separately for selected models. As one might expect, the larger the transposition (in magnitude), the higher the mean  $f_0$  error in most of the cases, although the error does not increase much for many of the presented models. In particular for singing, for models for which the bottleneck is sufficiently small the error seems almost constant across a range of about -2300 to +1300 cent. It is worth noting that for singing down-transpositions seems to work better, whereas for speech up-transpositions seem to be slightly more stable. The transposition seems to be more accurate for singing in general.

The models which were trained on both speech and singing provide a special insight into how speech and singing are treated inside the model, as they appear in both plots of [Fig. 6.2](#). One might expect that they work similarly for both speech and singing, but this is not at all the case for the model with bottleneck size 6, which seems to work relatively well for speech but effectively fails for singing. In fact for

---

2. <http://thesis.fnab.xyz/ch6/demo>



**Figure 6.2.:** Accuracy of the  $f_0$  transformation given as mean absolute error between requested  $f_0$  and synthesized  $f_0$ . Nomenclature according to Section 6.2.3. (Top) speech, (Bottom) singing.

singing, bottleneck size 6 has roughly the same  $f_0$  error, regardless whether it was trained only on singing or both singing and speech. One might expect that due to the higher complexity of the problem, the channel capacity needs to increase to properly accommodate all possibilities from the presented data. Instead, the model operates differently on speech and singing, successfully disentangling the former from the  $f_0$  while keeping the  $f_0$  in the latent code for the latter.

### Bottleneck size analysis

To choose the right dimensionality for the latent code we evaluate the disentanglement for various sizes of the bottleneck, by measuring the accuracy of the  $f_0$  transformation. Let  $f_t$  be the target  $f_0$ , let  $f_s$  be the  $f_0$  measured in the synthesized signal and  $\tau$  the transposition, then we define the **normalized mean  $f_0$  error (NMFE)** as

$$\text{NMFE} = \mathbb{E} \left[ \frac{|\log f_t - \log f_s|}{|\log \tau|} \right]. \quad (6.9)$$

If the  $f_0$  is perfectly disentangled from the latent code, the auto-encoder will produce a signal with the  $f_0$  provided to the decoder and the **NMFE** will be zero.<sup>3</sup> If, on the other hand, the decoder only uses the  $f_0$  information contained in the latent code, then the  $f_0$  error will be equal to the transposition and thus the **NMFE** will be one. Any value in between should give us a measure for the degree of entanglement.

We plot the **NMFE** against the bottleneck size for different configurations in Fig. 6.3.

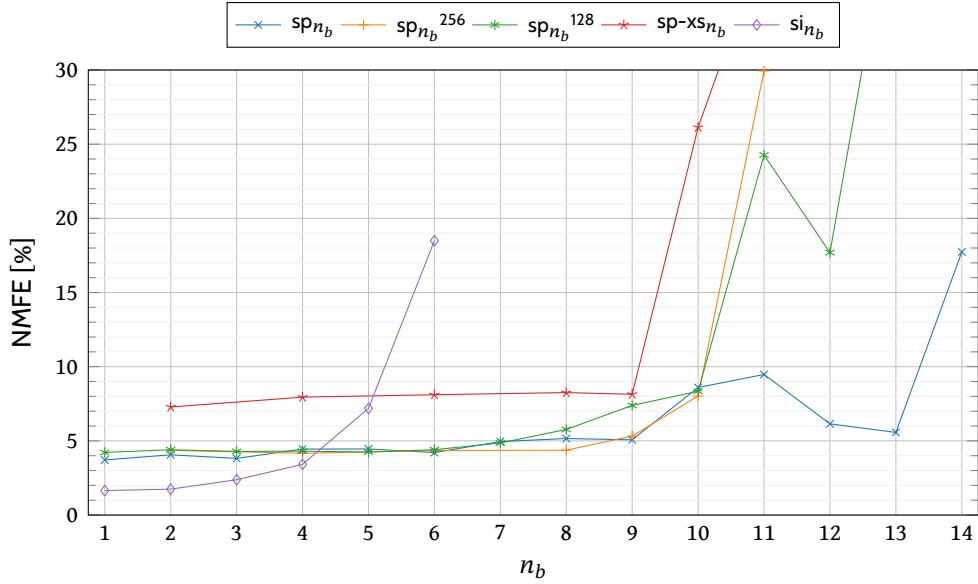
For all configurations on speech we see a drastic increase in **NMFE** at around 8 or 9. For the *speech* configuration with 512 filters per convolutional layer (the default configuration) the **NMFE** increases between 9 and 10 but drops again for 12 and 13 to almost the same value as for the lower bottleneck sizes. At 14 the **NMFE** is finally multiple times worse. There is no reason to believe that models with bottleneck size 12 or 13 should generally perform better than bottleneck sizes 10 or 11 in terms of **NMFE**. This may be an effect of network initialization, since each of the data points is generated by training only one neural network and measuring its **NMFE**. It might well be that here for bottleneck size 13 the model by chance did not find the way to entangle the  $f_0$  into the 14 dimensions preserving the small reconstruction error. Regardless of the explanation, for bottleneck sizes above 9 the disentanglement becomes unlikely.

For speech the critical point seems to be always the same, regardless of the modelling capacity of the neural network or the size of the training dataset. This is counter to the intuition that the computational capacity of the model is responsible to create the **information bottleneck**. However, as we do not actually measure the modelling capacities of the neural networks, no clear conclusions can be drawn from this observation. It seems that the critical point is rather stable for a wide range

---

<sup>3</sup>. Of course due to the approximations with neural networks, the **NMFE** will never be zero and some fraction of the **NMFE** will be due to inaccuracies of the model. The **NMFE** should be seen more as an indication rather than a proper measure of disentanglement.

## 6. Bottleneck auto-encoder



**Figure 6.3.:** Normalized mean  $f_0$  error (NMFE) averaged over different transpositions  $\pm 660$ ,  $\pm 1320$  and  $\pm 1980$  as a function of the bottleneck size  $n_b$  for various configurations (see Table 6.1). The NMFE is calculated according to Eq. (6.9). Nomenclature according to Section 6.2.3.

of neural networks with practical size, the networks in the study here have 50 M, 12.5 M and 3.13 M parameters respectively.

For singing voice the NMFE behaves quite differently. The NMFE is much lower for low bottleneck sizes (1-4) but already for a bottleneck size of 4 the NMFE is more than twice as high as at the beginning.

### 6.3.2. Synthesis quality

A low mean  $f_0$  error for a wide range of transpositions implies disentanglement between  $f_0$  and latent code. This does not necessarily mean that the model is in

**Table 6.2.:** Quality rating of the algorithms for speech averaged over all transposition values. Value is given with 95% confidence interval. Nomenclature according to Section 6.2.3.

| transp. | GT              | PaN             | sp <sub>2</sub> | sp <sub>3</sub> | vo <sub>3</sub> | sp <sub>8</sub> |
|---------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 1760    |                 | 1.41 $\pm$ 0.31 | 2.28 $\pm$ 0.54 | 2.50 $\pm$ 0.65 | 2.61 $\pm$ 0.62 | 2.11 $\pm$ 0.40 |
| 1320    |                 | 1.74 $\pm$ 0.32 | 3.17 $\pm$ 0.52 | 3.65 $\pm$ 0.49 | 3.26 $\pm$ 0.46 | 3.00 $\pm$ 0.46 |
| 880     |                 | 2.50 $\pm$ 0.53 | 3.72 $\pm$ 0.52 | 3.61 $\pm$ 0.50 | 3.33 $\pm$ 0.62 | 3.47 $\pm$ 0.42 |
| 440     |                 | 2.48 $\pm$ 0.55 | 3.73 $\pm$ 0.50 | 3.41 $\pm$ 0.53 | 3.55 $\pm$ 0.48 | 4.09 $\pm$ 0.44 |
| 0       | 4.79 $\pm$ 0.09 | 2.90 $\pm$ 0.37 | 3.67 $\pm$ 0.37 | 4.10 $\pm$ 0.22 | 3.83 $\pm$ 0.31 | 4.10 $\pm$ 0.38 |
| -440    |                 | 3.06 $\pm$ 0.56 | 3.28 $\pm$ 0.49 | 3.83 $\pm$ 0.56 | 3.50 $\pm$ 0.60 | 4.18 $\pm$ 0.51 |
| -880    |                 | 2.22 $\pm$ 0.42 | 3.17 $\pm$ 0.57 | 3.41 $\pm$ 0.48 | 3.35 $\pm$ 0.58 | 3.61 $\pm$ 0.47 |
| -1320   |                 | 1.65 $\pm$ 0.43 | 2.83 $\pm$ 0.63 | 3.39 $\pm$ 0.58 | 3.72 $\pm$ 0.54 | 3.61 $\pm$ 0.53 |
| -1760   |                 | 1.61 $\pm$ 0.35 | 2.50 $\pm$ 0.53 | 2.68 $\pm$ 0.51 | 2.82 $\pm$ 0.53 | 2.50 $\pm$ 0.51 |
| average | 4.79 $\pm$ 0.09 | 2.25 $\pm$ 0.16 | 3.21 $\pm$ 0.18 | 3.47 $\pm$ 0.16 | 3.38 $\pm$ 0.17 | 3.48 $\pm$ 0.17 |

**Table 6.3.:** Quality rating of the algorithms for singing averaged over all transposition values. Value is given with 95% confidence interval. Nomenclature according to Section 6.2.3.

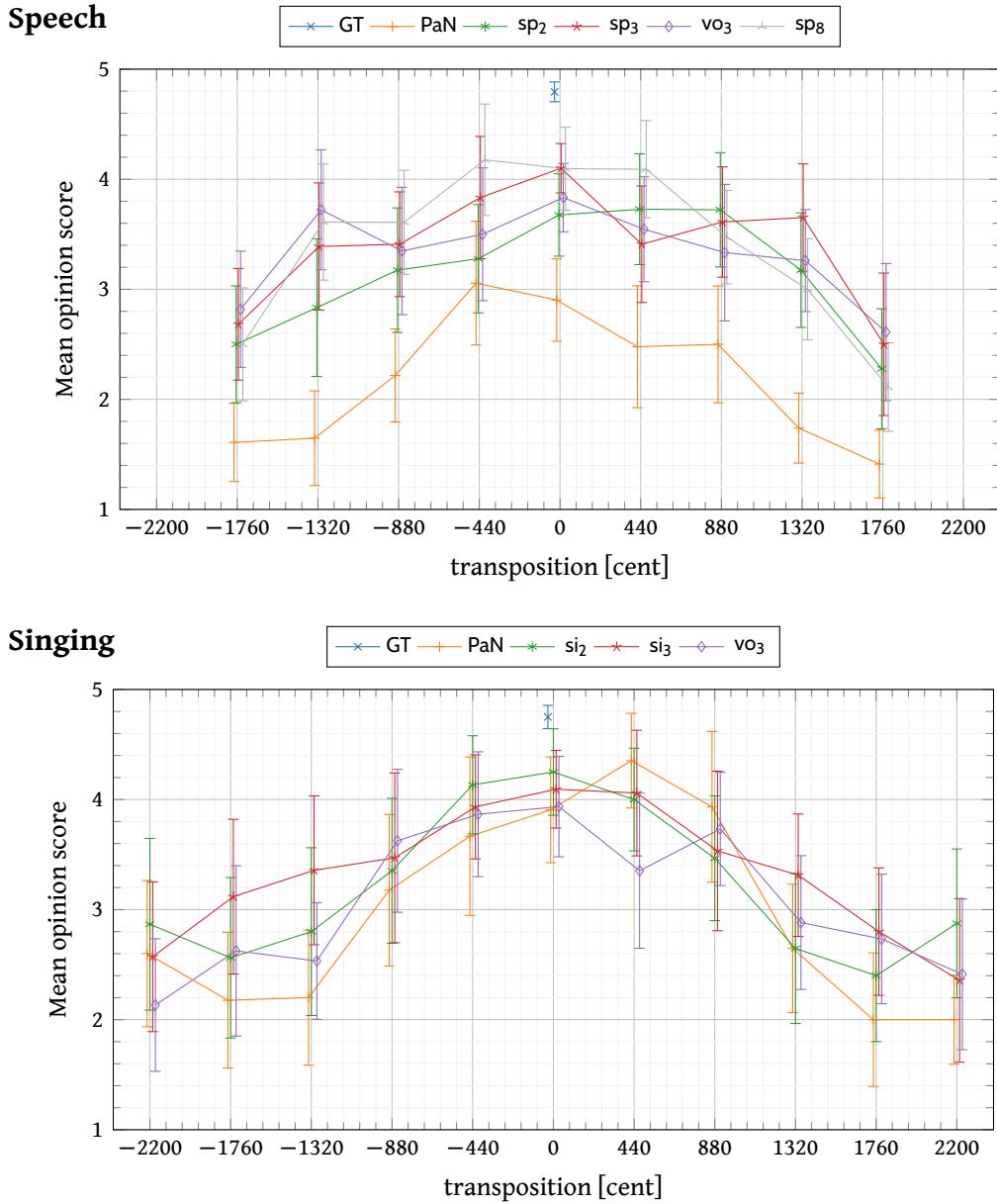
| transp. | GT          | PaN         | si <sub>2</sub> | vo <sub>3</sub> | si <sub>3</sub> |
|---------|-------------|-------------|-----------------|-----------------|-----------------|
| 2200    |             | 2.00 ± 0.40 | 2.88 ± 0.68     | 2.41 ± 0.68     | 2.36 ± 0.74     |
| 1760    |             | 2.00 ± 0.61 | 2.40 ± 0.60     | 2.73 ± 0.59     | 2.80 ± 0.58     |
| 1320    |             | 2.65 ± 0.58 | 2.65 ± 0.68     | 2.88 ± 0.61     | 3.31 ± 0.56     |
| 880     |             | 3.93 ± 0.68 | 3.47 ± 0.57     | 3.73 ± 0.51     | 3.53 ± 0.73     |
| 440     |             | 4.35 ± 0.43 | 4.00 ± 0.47     | 3.35 ± 0.70     | 4.06 ± 0.57     |
| 0       | 4.75 ± 0.11 | 3.91 ± 0.48 | 4.25 ± 0.39     | 3.94 ± 0.46     | 4.09 ± 0.35     |
| -440    |             | 3.67 ± 0.72 | 4.13 ± 0.45     | 3.87 ± 0.57     | 3.93 ± 0.47     |
| -880    |             | 3.18 ± 0.69 | 3.35 ± 0.66     | 3.62 ± 0.65     | 3.47 ± 0.77     |
| -1320   |             | 2.20 ± 0.61 | 2.80 ± 0.76     | 2.53 ± 0.53     | 3.36 ± 0.68     |
| -1760   |             | 2.18 ± 0.62 | 2.56 ± 0.73     | 2.62 ± 0.77     | 3.12 ± 0.70     |
| -2200   |             | 2.60 ± 0.66 | 2.87 ± 0.78     | 2.13 ± 0.60     | 2.57 ± 0.68     |
| average | 4.75 ± 0.11 | 3.06 ± 0.21 | 3.31 ± 0.19     | 3.14 ± 0.19     | 3.41 ± 0.19     |

fact useful for performing actual transpositions. The accuracy of the  $f_0$  could be achieved by generating an arbitrary signal with the target  $f_0$  unrelated to speech. In the case where the bottleneck size is 1, phonetic content is greatly reduced and the audio quality of the resynthesized mel-spectrogram actually degrades quite noticeably. Therefore, to analyse the synthesis quality, we conduct a perceptive test.

In the perceptive test we asked 96 participants to rate audio samples on a scale from 1 to 5, 1 being the worst and 5 being the best possible quality. The samples were randomly drawn from the test set, the models, various transpositions, as well as from the ground truth. The results are given in Table 6.2 for speech, Table 6.3 for singing and in Fig. 6.4 as a plot over the transposition amount.

Tables 6.2 and 6.3 contain 46 and 45 data points respectively. Thus, the coverage for each of the data points is rather thin, despite the large number of participants. Consequently, the confidence intervals are rather large.

For speech, the auto-encoder outperforms the classical vocoder with a strong margin and for all transpositions. For singing the classical PaN vocoder is not as bad. For transpositions up of +440 and +880 cent the classical vocoder is even slightly outperforming the auto-encoder, while for transpositions down of -440 and -880 cent the auto-encoder is slightly better. The auto-encoder shows an approximately symmetric performance for transpositions up and down. For the classical vocoder the performance is asymmetric, which is due to the fact that internally the vocoder preserves the spectral envelope (the formant structure) of the original signal. For higher  $f_0$  the harmonics are spread further apart, which decreases the details in the spectral envelope (the formant structure). Therefore, preserving the spectral envelope for upwards transpositions does not pose a problem. On the other hand, for lower  $f_0$  the harmonics are spread more densely, and more information about the spectral envelope (the formant structure) is perceptually expected. As the classical vocoder cannot create more details its perceptual result is worse. Note that the difference in this intermediate range of transpositions remains



**Figure 6.4.: Mean opinion score plotted against transposition amount for speech (Top) and singing (Bottom). The curves are slightly offset horizontally to prevent overlapping of the error bars. The error bars denote the 95% confidence interval. All data points are integer multiples of 440. The exact values can be read from Tables 6.2 and 6.3. Nomenclature according to Section 6.2.3.**

below 1 point in the MOS scale, and it is smaller than the 95% confidence interval observed in the perceptual tests.

For transposition above  $+/- 880$  cent the auto-encoder considerably outperforms the classical vocoder by about 1 point on the MOS scale. Here the signal needs to change more dramatically to remain coherent, a change that the auto-encoder can create, while the classical vocoder fails.

Averaging over all transpositions, as shown in the bottom row of [Table 6.3](#), reveals a significant dominance of the auto-encoder models over the classical vocoder (with a confidence of over 95%). One weak-point of the PaN vocoder is synthesizing consonants like plosives and fricatives. Those are much more frequent in speech than in singing and could explain the larger perceptual difference we found for speech.

As expected, the models tend to receive the highest rating for transposition 0, with a few exceptions that lie within the 95% confidence interval. For speech, most of the auto-encoder models manage to keep a perceptive rating above 3 by a large margin over the interval  $[-1320, 1320]$ . No auto-encoder model stands out among the others. In the average ratings bottleneck size 3 and 8 seem almost equally good. Only the model with bottleneck size 2 has a significantly lower overall rating. In the same way for singing there is no clear dominance of one model over all transpositions. The model with bottleneck size 2 seems to perform better for smaller transpositions while the model with bottleneck size 3 seems to perform better for larger transpositions. The model with bottleneck size 3 manages to keep a perceptive rating above 3 for over an interval of  $[-1760, 1320]$ .

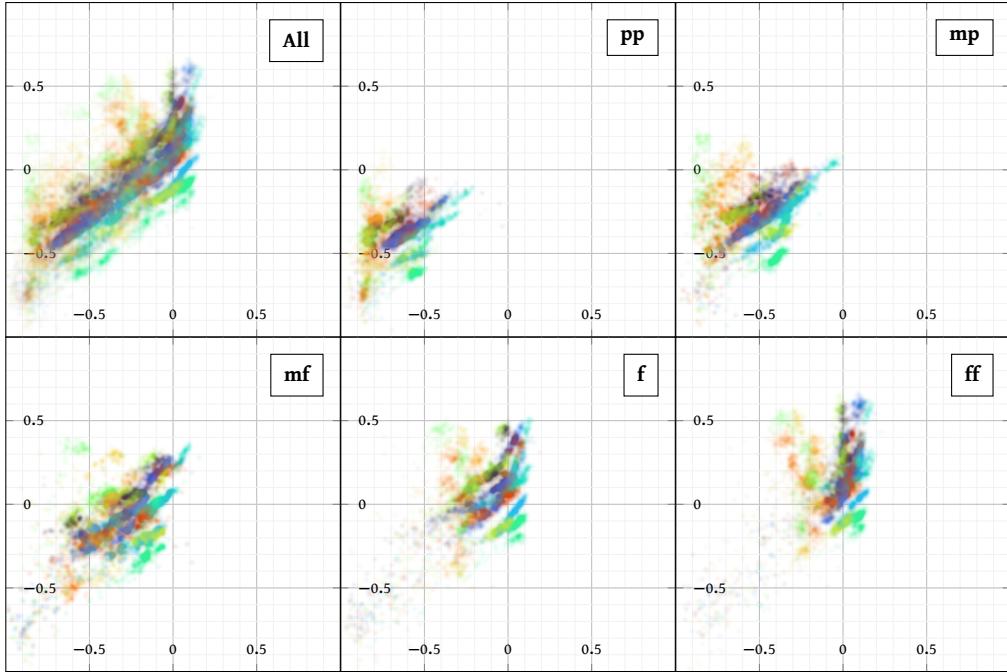
For both singing and speech we can observe that the universal model (trained on both speech and singing) generally performs worse than its domain specific counterpart with the same bottleneck size.

### 6.3.3. Investigation of the latent code

Since the model with bottleneck size 2 has achieved a rather good rating for singing voice in both mean  $f_0$  error and mean opinion score, we can reasonably assume that two dimensions suffice to represent singing voice signals. Since the latent code in that model is only two-dimensional we can visualize it with a 2d-plot. This allows an investigation on how the model uses its two dimensions to represent the features of human voice internally.

In one of our singing datasets we have recordings from one singer with isolated notes for each vowel of the French language and for five different dynamics (pp, mp, mf, f, ff). Plotting the latent code of these notes allows us to see how the dynamics and the phonemes are encoded in the latent code.

The latent code vectors typically have a norm of less than 4 and outliers with a Euclidean norm above 10 are extremely rare. However, for better visualization we train a special model with a tanh activation at the end of the encoder to force all the samples into the unit square. Loss-wise, and in terms of objective ( $f_0$ ) and perceptual accuracy, there is no difference with the equivalent model that uses no activation at the output of the encoder.



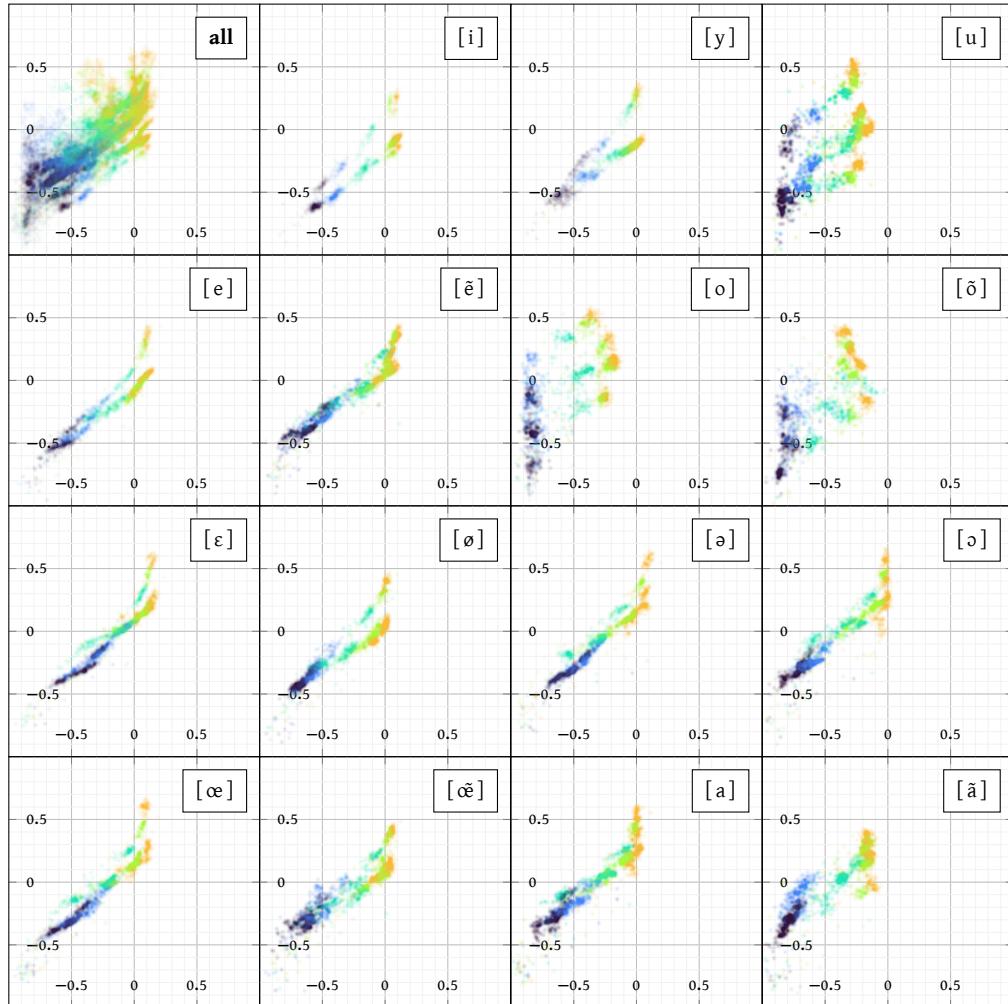
**Figure 6.5.: Distribution of vowels in the latent code for various intensities.** Top row: all intensities, pp, mp. Bottom row: mf, f, ff. Each vowel is represented in a different colour. The distributions are estimated as a Gaussian mixture model with one mode for each sample fixed standard deviation  $\sigma = 0.015$  placed at the location of the sample.

Figure 6.5 shows the latent codes for each frame in a scatter plot with each vowel in a different colour. The subplots show only the codes of frames of a fixed intensity. Fig. 6.6 shows the latent codes for each frame with each intensity in a different colour and with a subplot for each individual vowel. From both figures we can clearly see that intensity is ordered along a diagonal axis.

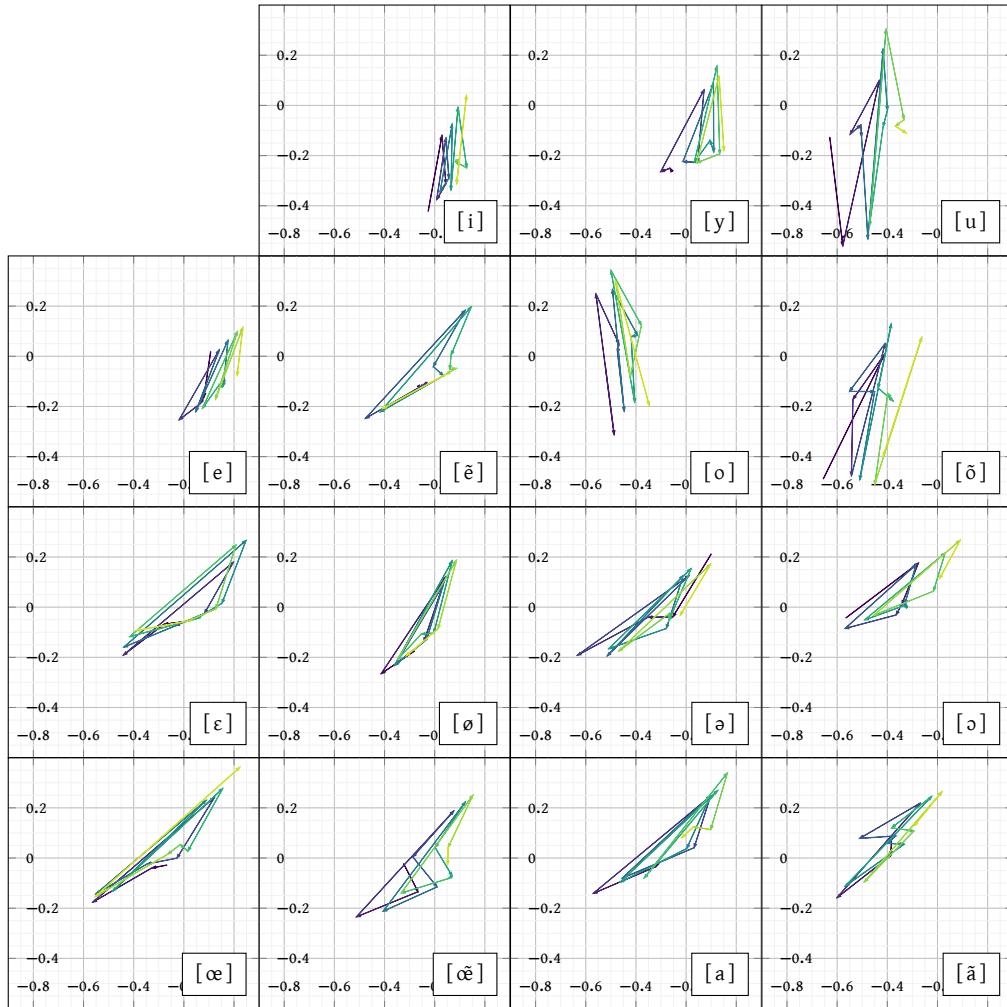
From Fig. 6.6 we can see that the phonemes are arranged in clusters. However, each phoneme forms more than one cluster, typically three to four. Looking at each code as a temporal sequence as plotted in Fig. 6.7, we notice that the code values hop periodically from one cluster to the other. More specifically, each phoneme is represented by a specific gesture usually repeating with a periodicity of 3 to 4 time-steps. Two effects could be used to explain this phenomenon. Firstly, due to the overlapping analysis windows, two consecutive frames are highly correlated. Here, we use an overlap of 75%, which coincides perfectly with the periodicity of 3 to 4. Furthermore, as human voice (and especially singing) normally does not change that quickly over time, there is further redundancy across consecutive frames. In our implementation, the encoder and decoder have rather large receptive fields – 11 and 13 time-steps respectively, which translates to 138 ms and 163 ms. Using their large receptive fields the networks in the auto-encoder seem to be able to discard this redundancy under the large pressure of the information bottleneck: in addition to the two feature dimensions, they also take advantage of the temporal

## 6. Bottleneck auto-encoder

---



**Figure 6.6.: Distribution of intensities in the latent code space for all vowels in the French language. Colour brightness translates to musical loudness: Dark blue is pp, yellow is ff. The distributions are estimated as a Gaussian mixture model with one mode for each sample fixed standard deviation  $\sigma = 0.015$  placed at the location of the sample.**



**Figure 6.7:** A few consecutive steps of latent code for each vowel in the French language showing how the latent code progresses through the latent code space. The arrows point from one point to the next, progression goes from dark to bright. Each vowel forms a specific gesture.

dimension to represent phonemes.

## 6.4. Conclusion

In this chapter we have applied an auto-encoder with information bottleneck (Qian, Jin, et al., 2020) to speech and singing voice to disentangle the  $f_0$  from the latent code. With this auto-encoder we were able to perform high quality transpositions of speech and singing voice.

Since it is possible to measure the  $f_0$  in the transposed samples we were able to define a disentanglement measure and perform a study over the required size of the information bottleneck. We have seen that for singing voice disentanglement requires a bottleneck size of less than 4 ( $n_b \leq 3$ ). For speech, however, disentanglement can be achieved with a much larger bottleneck ( $n_b \leq 9$ ). With the appropriate bottleneck size we were able to perform coherent transpositions over a range of more than an octave for both speech and singing voice.

A perceptive test was carried out to study the quality of the transposed audio. While there is still a substantial difference between ground truth and transposed audio, the proposed method surpassed the state-of-the-art classical vocoder PaN Ardaillon, 2017 in perceptual ratings for both singing and speech. For speech the difference between auto-encoder and classical vocoder was severe.

Finally, we performed a deeper analysis of the latent code for the case of  $n_b = 2$ . We have seen that, while voice intensity corresponds to a clear direction in the latent space, the phonemes were not only encoded in position but rather through temporal gestures within the latent code.

## 6.5. Future work

While the auto-encoder has disentangled the  $f_0$  from the latent code successfully, the resynthesized audio quality is still not en par with the original audio. For singing voice this is partly due to the small dataset size, and we indeed noticed an over fitting phenomenon for the singing models. Audio quality could thus be improved by training on a large singing dataset. For speech, we saw almost no over fitting when training the model on the 54 h large combined dataset. This means that improving audio quality here would not be as straight forward as for singing.

An adversarial approach could help improve audio quality by applying a discriminator either on the latent space as in fader networks Lample et al., 2017 or on the output mel-spectrogram.

The models analysed in Section 6.3 all used non-causal filters. However, informal listening tests showed that the quality does not degrade much if using causal filters instead. A causal model would even allow for real-time processing as the only delay would come from buffering the audio-frames to compute the mel-spectrogram.

As a side effect the proposed method is able to compress the mel-spectrogram to as little as 2 to 8 features per frame. It produces an efficient and visually interpretable representation of speech. A more thorough analysis of the latent code may allow

manipulation of the embedding directly in the latent space, potentially leading to artistic applications in sound design. Currently, using float32 encodings, the model with bottleneck size 2 compresses human voice to a data rate of  $7.76 \text{ kB s}^{-1}$ . Concentrating on the compression could further reduce the data rate and produce a highly compressed high-quality speech codecs.

## 7. Transformation of perceived voice level

In the previous chapter we have developed the bottleneck auto-encoder as a tool for voice transformation and demonstrated its capabilities on transformation of the fundamental frequency. Using the fundamental frequency as a first parameter is an obvious choice since it is easily available as an input parameter through existing analysis techniques. Its effect can easily be evaluated both perceptually and objectively and several other pitch transformation techniques exist that can be used as a baseline. The next challenge is to apply the bottleneck auto-encoder to other voice parameters. The bottleneck auto-encoder has already been applied to speaker identity through AutoVC (Qian, Yang Zhang, Chang, X. Yang, et al., 2019; Qian, Jin, et al., 2020), but numerous other voice properties exist that have never been addressed before, neither with the bottleneck auto-encoder nor with classical digital signal processing methods. In this chapter we will discuss the voice level and try to use the bottleneck auto-encoder to change the voice level in singing voice recordings.



Article



Audio samples



Inference code

Article: <https://arxiv.org/abs/2204.04006>  
Audio Samples: <http://thesis.fnab.xyz/ch7/demo>  
Inference Code: <http://thesis.fnab.xyz/ch7/code>

This chapter is an adaption of Bous and Roebel (2023) which was presented at the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

### 7.1. Introduction

#### 7.1.1. Terminology

In this chapter, voice level shall denote the power with which voice is emitted from the speaker or singer, as defined by Traunmüller and Eriksson (2000). This is related to the *vocal effort* which is a property that describes how much effort is exerted onto the vocal folds to produce speech (or singing) (Traunmüller and Eriksson, 2000; Ford Baldner et al., 2015; McKenna and Stepp, 2018). Vocal effort is different from voice level as can be illustrated by comparing different phonemes: Singing on the vowel

[ a ] allows producing a much higher voice level than humming on the consonant [ m ] using the same vocal effort, or conversely the same voice level requires much more vocal effort for [ m ] than for [ a ]. More generally it has been observed that increase in voice level is a combined effect of ‘increased harmonic energy that is boosted by the vocal tract’ (Sundberg et al., 1993) in addition to increased subglottal pressure. Another related term is the *musical dynamic*, which is a musical term that describes the expressive intention relating to loudness. Musical dynamic is difficult to quantify because its execution is part of the interpretation of a piece and strongly depends on the epoch, genre, composer, and the context inside the piece. A general translation of musical dynamic to voice level or vocal effort is impossible.

Sometimes the term *voice intensity* or *vocal intensity* is used to denote voice level (Titze and Sundberg, 1992; Sundberg et al., 1993). A common confusion occurs when conflating these terms with the recorded level or intensity.<sup>1</sup> Though a higher voice level will produce a recording with higher signal power if the recording conditions remain the same, these quantities are completely independent. The signal power can easily be changed by amplification but changing the volume in the playback device does not change the amount of power that was put into producing the note that is being played. This difference will be obvious to any listener. Similarly, voice level or vocal effort should not be confused with the ratio between voice and other sounds in a recording, for example between singing voice, and its accompaniment in a musical piece. In classical singing, in order to be able to hear the voice over an accompaniment, the two must be balanced through singing- and playing technique of singer and accompaniment respectively. This will give characteristic changes in the timbre of the different instruments. In post-production we might change the signal power of the singing voice if voice and accompaniment were recorded separately, but the effect will be very different and trained ears will notice the difference. Thus, it is important to distinguish between voice level as a property of the source and signal intensity as a property of the representation of the voice.

Finally, we need to distinguish between power and loudness. Power is a pure physical quality that is derived from the square of the signal (either as an averaged frame-wise quantity or instantaneous using the Hilbert transform). Loudness is a perceptual quantity that depends on how loud a sound is perceived. Perceived loudness does not have a canonical definition but depends on the used loudness model. The purpose of a loudness model is to capture empiric observations about human perception of loudness. More details are discussed in Section 2.2.3. Voice level and signal power find their perceptive pendant through *perceived voice level* and *signal loudness* respectively. Signal loudness describes how loud a sound is when it arrives in the ear and can be calculated with a given loudness model. Perceived voice level is a spectral property that describes how loud the source sounds, independent of how loud the source really was or how much power the sound still has in the ear.

---

1. In the literature we see vocal intensity denoted by the abbreviation SPL (sound pressure level) (Titze and Sundberg, 1992; Sundberg et al., 1993) as it is measured by a sound pressure meter at a fixed distance to the mouth. Sound pressure level is a measure of signal power at a specific point. In a controlled experimental set up the SPL can be in fact used to measure the voice level, however, in general these are two different concepts and the distinction is a crucial point in this chapter.

It is the perceived voice level that best reflects what we want to change in voice recordings.<sup>2</sup>

### 7.1.2. Problem formulation

The central challenge addressed in this chapter with voice level and vocal effort is, that we cannot extract these qualities from voice recordings. The voice level could be measured with a calibrated sound pressure level meter close to the singer and several studies investigating the relationship between the voice level and other voice parameters have done so in the past (Titze and Sundberg, 1992; Sundberg et al., 1993; Traunmüller and Eriksson, 2000). Generally however, singing recordings do not come with voice level annotations and none of them are available to us. Vocal effort is even more difficult to measure as it requires knowledge about the effort put into the voice which can only really be evaluated psychologically. Though not exactly the same, vocal effort is highly correlated with the subglottal pressure (Ladefoged and McKinney, 1963; Isshiki, 1964; Sundberg et al., 1993). Measuring subglottal pressure, however, requires special equipment which is not widely available. Thus, to be able to manipulate voice level, first we need to find a method to obtain voice level from voice recordings, which is the main contribution of this chapter.

In speech, we face the additional difficulty that modes of speech production change, when speakers adjust their voice level: When speaking very quiet, people tend to whisper, when speaking very loudly, people tend to scream. Articulation and accentuation, speech rhythm and pitch change when the voice level is changed especially when switching to whispering or screaming. Thus, in speech, voice level, is highly dependent on other factors, like the fundamental frequency and timing. To create a convincing change in voice level these properties need to be adjusted accordingly. The bottleneck auto-encoder, as proposed in Chapter 6, however, is not able to change the timing. Experiments to change the voice level in speech using the method presented below have not yielded naturally sounding changes which suggests that heavier adjustments of the approach from Chapter 6 are required to transform voice level in speech.

For singing voice, pitch, and timing are constrained by the melody and music respectively and are not caused by the intended voice level. The bottleneck auto-encoder from Chapter 6 is well suited in this situation as it preserves the timing and, provided it is conditioned on the fundamental frequency, preserves the pitch as well. Thus, in this chapter we focus the discussion on singing voice.

### 7.1.3. Outline

The remaining chapter is structured as follows: In Section 7.2 we introduce the mathematical modelling and present two ways to estimate the voice level from

---

2. Applying the strict definition of voice level here, changing the voice level in a voice recording is impossible. Changing the voice level would require going back in time and getting the singer to change the way they sing during the recording. What we really want is just to modify the recording in a way such that it sounds like the singer had sung differently, thus the distinction regarding perceived voice level.

voice recordings where no voice level annotation exists. In Section 7.3 we present the adaptions of Chapter 6 to include the voice level as a controllable parameter. We will explain our experimental validation in Section 7.4 and present and discuss the results in Section 7.5. Finally, we will see a short summary and an outlook in Section 7.6.

## 7.2. Extracting voice level from audio recordings

Audio recordings are not calibrated measurements. The goal of a recording is to produce a signal that when played on a speaker will create sound waves that sound like the original source. The scaling of these signals is irrelevant as it is expected to be adjusted by the consumer or the sound engineer that further processes the sound. Thus, each recording has a different relationship between the source's power and the recorded signal's power as microphones have different directivities and transfer functions and the signal gain is adjusted to minimize quantization noise when digitalizing the microphone signal.

Therefore, without explicit annotation it is impossible to infer the voice level from the power of an audio recording. Still, if we look at the speech production mechanism, we notice that humans cannot increase the voice level without changing other properties of their voice (Traunmüller and Eriksson, 2000; Holmberg et al., 1988; Titze and Sundberg, 1992; Henrich et al., 2005). Thus, we should be able to infer the perceived voice level from the signal's timbre.

The perceived voice level is a function of the timbre. The goal of estimating the (perceived) voice level in this chapter is to change the perceived voice level in a recording. Hence, for this purpose, we would like to assign the same perceived voice level to two recording if the voices in both recordings *sound* like they have been sung equally loud.<sup>3</sup> This necessitates using perceptive loudness measures to be able to compare different sounds. Thus, in this derivation we will be using the term 'loudness' rather than 'power' to distinguish between perceptive and objective measures. In the following we will omit the term 'perceived' when talking about perceived voice level, unless it is necessary to distinguish it explicitly from objective voice level.

### 7.2.1. Objective

#### Voice level as a property of the timbre

We want to infer the voice level  $v$  as a perceptive quantity of the signal's timbre. The voice level is independent of the scale of the signal. Let  $x$  be a signal containing voice. Thus, for all factors  $a \in \mathbb{R}$

$$v_x(n) = v_{a \cdot x}(n) \quad (7.1)$$

---

<sup>3</sup>. This illustrative description serves to motivate what we expect from the voice level. In fact, we do not assign a voice level value to a whole recording but rather treat the voice level as a time-varying property.

(the voice level  $v_x$  of  $x$  is the same as the voice level  $v_{a \cdot x}$  of  $a \cdot x$ ).

Since  $v$  is a spectral property we will treat  $v$  as a frame-wise property and assume  $v$  is a function of the signal frames

$$\mathbf{x}(n) := (x(n+k))_{k=-N}^N \quad (7.2)$$

$$= [x(n-N), x(n-N+1), \dots, x(n+N)] \quad (7.3)$$

with frame length  $2N+1$ , with  $N \in \mathbb{N}$ , centred around  $n$ . To ensure the robustness to arbitrary gain changes from Eq. (7.1), any practical implementation  $F$  must remove the scale of the signal during its computation. When using neural networks, the normalization should occur before the frame is fed to the neural network to ensure the arbitrary gain is not used. Thus, we should calculate  $v_x(n)$  on the normalized frame  $\|\mathbf{x}(n)\|^{-1} \mathbf{x}(n)$ .

$$v_x(n) = F(\|\mathbf{x}(n)\|^{-1} \mathbf{x}(n)) \quad (7.4)$$

by some function  $F : \mathbb{R}^{2N+1} \rightarrow \mathbb{R}$ .

In this section we will investigate how  $F$  can be approximated using a neural network. The problem that arises from this objective is that ground truth values for  $v$  are not available. Thus, we need to find a way to train the neural network with the information that is available from the signal.

### Voice level and signal loudness

In this derivation we define voice level  $v$  as the loudness with which the signal is emitted, while the signal loudness  $p$  is the loudness of the signal as it is received. We measure the voice level at a fixed point in relationship to the singer, e.g. at their mouth (or a fixed distance from the singers mouth). When we move around in the room, the signal loudness will change. Similarly, we may change recording settings, like the pre-amplification, which will likewise change the signal loudness. We make the assumption that all these effects can be modelled by a *recording factor*  $a$ , that in the general case has to be assumed time varying. This leads to the relationship

$$p(n) = a(n) \cdot v(n). \quad (7.5)$$

Different ways to estimate the signal loudness  $p$  from the signal exist in the literature and the choice of loudness estimation technique influences the estimation of the perceived voice level. We will discuss practical matters on the loudness estimation in Section 7.4.2. From Eq. (7.5) the voice level could be calculated by solving for  $v$ . The problem is, however, that the recording factor  $a$  is unknown. In the remainder of this section we propose two different techniques to estimate the voice level in the presence of an unknown recording factor  $a$ .

#### 7.2.2. Learnt recording factor

Let  $X = \{x_1, x_2, \dots\}$  be a set of voice recordings. Assume that all recordings from  $X$  have been recorded under the same conditions and have been processed with the

same post-processing. That is, same microphone, pre-gain, spatial positioning of speaker and microphone, normalization, etc. have been used. In this case we can assume that for each recording  $x \in X$  the recording factor  $a$  is a constant such that loudness contour of the recorded signal  $p$  is proportional to the voice level  $v$ :

$$p(n) = a \cdot v(n) \quad (7.6)$$

For multi speaker databases it is highly likely that some of these assumptions are violated; however, it is not unlikely that these assumptions still hold for all recordings of a fixed speaker  $s$ . Therefore, we get the relationship

$$p(n) = a_s \cdot v(n) \quad (7.7)$$

for all recordings generated by a speaker  $s$  with a different recording factor  $a_s$  for each speaker.

The speaker dependent recording factor  $a_s$  can be learnt alongside the weights  $\theta$  of a neural network  $N_\theta$ . As discussed in Section 7.2.1 we have to normalize the frame  $\mathbf{x}(n)$  before feeding it to the neural network. We normalize  $\mathbf{x}(n)$  to ensure the scale invariance of Eq. (7.1) and to make sure the neural network does not use the signal loudness instead of the timbre. This step is crucial for this method to work. Using an  $L_2$  error, we get the following error function

$$\sum_n \left\| p(n) - a_s N_\theta \left( \|\mathbf{x}(n)\|^{-1} \mathbf{x}(n) \right) \right\|^2. \quad (7.8)$$

The resulting neural network  $N_\theta$  allows estimating the voice level from the normalized signal: After training the neural network, we can calibrate the voice level estimator by using the same recording factor  $a$  for all recordings. Once the recording factor has been fixed for all recordings, the relationship between  $p$  and  $v$  is no longer affected by the recording conditions. In this case we can compare the recordings as if they were made under the same recording conditions. To calibrate the estimations to a specific recording we can use the recording factor of that recording. Note that the speaker specific effects are part of the speaker dependent recording factor  $a_s$ .

### 7.2.3. Adaptive recording factor

The assumptions from Section 7.2.2 require the recordings from the training dataset to be grouped by same recording conditions. This works well if the number of speakers is small, and we can be sure that the recordings have not been normalized separately. However, for many databases we don't know what kind of post-processing has been performed or whether the samples for a fixed speaker have been created over multiple recording sessions with slightly different conditions. In this case we would have to assign a different recording factor to each recording. With the previous approach this causes problems as a gradient exists for a specific recording factor only if a sample associated with this recording factor is present in the training batch. Thus recording factors with a small percentage of associated recordings in the dataset are learnt very slowly as they are updated rarely. For the

case where we cannot group the recordings into a reasonable amount of classes we propose an adaptive recording factor:

Let  $\hat{v} := N_\theta(\|\mathbf{x}(n)\|^{-1} \mathbf{x}(n))$  be the output of the neural network and  $p$  the loudness curve associated with the input sample. We assume the recording factor is still constant for a fixed recording but now different for each training sample. Prior to calculating the loss we estimate the recording factor  $a$  such that the  $L_2$  error

$$e_a = \sum_n (p(n) - a \hat{v}(n))^2 \quad (7.9)$$

is minimal for each training sample:

$$\hat{a} = \underset{a}{\operatorname{argmin}} e_a \quad (7.10)$$

For a fixed pair of target and estimate  $(p, \hat{v})$  there exists a closed form solution for  $a$ :

$$\hat{a} = \frac{\sum_n p(n) \hat{v}(n)}{\sum_m (\hat{v}(m))^2}. \quad (7.11)$$

Combining Eqs. (7.9) and (7.11) and normalizing  $e_{\hat{a}}$  by  $\|p\|^2 = \sum_n (p(n))^2$  yields the scalar product loss:

$$e_{\text{scp}} := \frac{e_{\hat{a}}}{\|p\|^2} = 1 - \frac{(\sum_n p(n) \hat{v}(n))^2}{(\sum_m (p(m))^2)(\sum_k (\hat{v}(k))^2)} = 1 - \frac{(p \cdot \hat{v})^2}{\|p\|^2 \|\hat{v}\|^2} \quad (7.12)$$

Using  $e_{\text{scp}}$ , the neural network  $N_\theta$  can be trained without knowing the recording factor  $a$ . To calibrate the estimation during inference to the recording environment of a specific recording, we can calculate the recording factor according to Eq. (7.11) and rescale the network output by that factor.

### Geometric interpretation

Given two vectors  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ , the scalar product  $\cdot$  and their relative angle  $\angle$  are related by the equation

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \angle(\mathbf{a}, \mathbf{b}). \quad (7.13)$$

Thus, the scalar product loss  $e_{\text{scp}}$  can be interpreted geometrically as

$$e_{\text{scp}} = 1 - \frac{(p \cdot \hat{v})^2}{\|p\|^2 \|\hat{v}\|^2} \quad (7.14)$$

$$= 1 - \cos^2 \angle(p, \hat{v}) \quad (7.15)$$

$$= \sin^2 \angle(p, \hat{v}) \quad (7.16)$$

$$\approx (\angle(p, \hat{v}))^2. \quad (7.17)$$

Minimizing  $e_{\text{scp}}$  minimizes the angle between the signal loudness  $p$  and the estimated vocal effort  $\hat{v}$  in the vector space  $\mathbb{R}^N$ , where  $N$  is the length of the signals  $p$  and  $\hat{v}$ . If  $e_{\text{scp}} = 0$ , the signal loudness  $p$  and the vocal effort  $\hat{v}$  are the same up to a scale factor. This reflects the requirement of Eq. (7.5) and the assumption that  $a$  is constant.

The geometric interpretation implies that this method can only work if there is sufficient variation in the training samples. This suggests that the time-dimension of the training samples must be long enough to include enough variation. This method does not learn how to infer the voice level from the timbre itself. The change in voice level that causes a change in timbre is what allows the model to learn the relationship between timbre and voice level.

### 7.3. Proposed voice level transformations

Having a method to infer the voice level from audio recordings could be useful for many applications and in different disciplines. In this chapter we focus our attention on transforming the voice level in singing voice. We use the bottleneck auto-encoder from Chapter 6 to disentangle the voice level from the mel-spectrogram of the voice recordings.

Two possible approaches to include the voice level in the bottleneck auto-encoder arise from the previous chapter: Either replace the  $f_0$  with the voice level, or condition the auto-encoder on both  $f_0$  and voice level. We experimented with both configurations but found a joint conditioning on both  $f_0$  and voice level to work better. Similar to the observations of Qian, Jin, et al. (2020) we noticed that explicitly imposing the  $f_0$  helps to enforce consistency. In music, pitch, and intensity are generally treated independently, so we would not want the  $f_0$  to change due to a change of voice level, and it makes sense to fix the  $f_0$  during the process.

The inverse is not true though: For example, it is very difficult to sing high notes very quietly or low notes with very high voice level. Thus, when changing the pitch it would be acceptable and might even be necessary that the voice level was adjusted accordingly. Thus, to change the voice level, it is recommended to use an auto-encoder conditioned on both voice level and  $f_0$ . To change the  $f_0$ , it is recommended to use an auto-encoder only conditioned on the  $f_0$  as introduced in Chapter 6.

In speech, the instantaneous  $f_0$ , is not a very important speech property, but rather the evolution and contour, e.g. the  $f_0$  going up at the end of questions. Moreover, as discussed by Traunmüller and Eriksson (2000), the  $f_0$  tends to increase for increased voice level and further contour details may change as well. Thus, to transform voice level in speech, it would be necessary to infer a new  $f_0$  curve from the imposed voice level. We tried to automatically adjust the  $f_0$  by only disentangling the voice level and leaving the to be modelled in the latent code. However, with this approach, we were not able to successfully adjust the  $f_0$  in the result. This suggests that the bottleneck auto-encoder trained to disentangle the voice level from the mel-spectrogram in speech was not able to properly disentangle the voice level from the fundamental frequency. A more forced disentanglement between voice level

and fundamental frequency is required for speech. Thus, for the rest of the chapter we focus our attention to the voice level in singing voice and leave the application to speech for future work.

We can use this auto-encoder to validate the proposed voice level estimator that was introduced in Section 7.2 and show that it really represents the (perceived) voice level. If the transformed recordings are perceived to have been sung with a voice level that is coherent with the intended transformation (louder, less loud), we can conclude that the proposed voice level estimators in fact encode the voice level.

## 7.4. Experiments

### 7.4.1. Data

All experiments are trained on the same dataset as in Chapter 6, which is a combined dataset of CREL, NUS-SMC, ICH, PJS, JVS-MuSiC, Tohoku, VocalSet, ISI and Farinelli (see Table 2.2, Page 34). Most of the singing recordings were not made with deliberate variation in voice level. However, since singing naturally contains variations in dynamic and consequently variations in voice level, we can assume that the dataset contains sufficient variations in voice level.

### 7.4.2. Target curves

During the training process, the voice level estimators are trained to match the signal loudness contour  $p$ . We calculate the perceived loudness with a simplified version of Glasberg and Moore (2002). To obtain the loudness, first, we normalize the power spectrum by the 70 dB equal loudness contour  $L_{70\text{dB}}$  of the ISO 226: 2003 standard. Then, we apply an equivalent rectangular bandwidth (ERB) filter bank  $B_{\text{ERB}}$  to the spectrum (Glasberg and Moore, 2002). Finally, the values for each ERB bin are converted to sone by using the exponent  $\alpha = \log_{10}(2) \approx 0.301$  and added together. Given the spectrum  $X$  of a frame of the signal, the corresponding loudness is calculated by the formula

$$p = \sum \left( B_{\text{ERB}} L_{70\text{dB}}^{-1} |X|^2 \right)^\alpha, \quad (7.18)$$

where  $B_{\text{ERB}} \in \mathbb{R}^{n_{\text{ERB}} \times n_{\text{fft}}}$  is the ERB filter bank and  $L_{70\text{dB}} \in \mathbb{R}^{n_{\text{fft}} \times n_{\text{fft}}}$  is a diagonal matrix with the ISO 226: 2003 70 dB equal loudness contour sampled at the fft-frequencies on its diagonal. To calculate the spectrum from the signal given at 24 kHz, we use a window size of 32 ms, an fft-size  $n_{\text{fft}} = 1024$ . To calculate the ERB filter bank  $B_{\text{ERB}}$ , we consider the frequencies between 20 Hz and 12 kHz with an overlap of 4 which results in  $n_{\text{ERB}} = 145$  ERB frequency bands. Significant differences of this approach to Glasberg and Moore (2002) is that we use only a single frame length, use the ISO 226 equal loudness contour and do not take masking and the threshold of hearing into account.

Furthermore, we run a voice activity detection on the voice signal and set the target curve to zero for frames where no voice is present (neither voiced nor unvoiced).

This forces the voice level estimator to produce voice level  $v(n) = 0$  if no voice is produced in frame  $n$  and thus teaches the model to ignore background noise.

### 7.4.3. Architecture

We train two voice level estimators, one with the learnt recording factor from Section 7.2.2 (Le) and the other with the adaptive recording factor strategy from Section 7.2.3 (Ad). Since we do our voice transformations based on the mel-spectrogram, the voice level estimators receive already pre-computed the mel-spectrogram as input. Furthermore, the mel-spectrogram is normalized frame-wise as required by Eq. (7.4). We use the same analysis parameters as in Chapter 6 to generate the mel-spectrogram.

Both networks Le and Ad have the same architecture which is given in Table A.5 (Page 162). The networks are simple convolutional feed-forward networks with 10 layers. Convolutions are 1d, treating the frequency bins as features. The number of filters is 100 in most of the layers, except the first, which has 80, the second to last, which has 50, and the last which has 1. The filter size is 3 in the first two layers and 1 elsewhere. With a step size of 12.5 ms per frame the voice level estimator has thus a receptive field of 5 frames or 50 ms.

We train the models with a batch size of 256 training samples of 80 frames (or 1 s) each. The models are trained for 500 k updates using the ADAM (Kingma and Ba, 2014) optimizer ( $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ) with an initial learning rate of  $1 \cdot 10^{-4}$ . The learning rate is reduced by a factor of  $\sqrt[4]{0.1}$  if the validation loss does not decrease for a period of 16 k updates, with a minimum learning rate of  $1 \cdot 10^{-6}$ .

### 7.4.4. Auto-encoders

We train two auto-encoder configurations, one for each voice level estimator. For the auto-encoder we use almost the same architecture as in Chapter 6 (see Table A.3 and Table A.4) only with the conditional input changed, as we add the voice level to the list of conditional inputs. Thus, the auto-encoders are conditioned on the voice level, the  $f_0$  and the voiced-unvoiced mask. We use the  $f_0$  model of Ardaillon and Roebel (2019).

### 7.4.5. Evaluation methods

Since the dataset that we used to train our models does not include annotations for the voice level, we cannot evaluate the models directly with a ground truth. The hypothesis of this chapter is that we can extract meaningful information from the spectral properties of the mel-spectrogram about the perceived voice level. We will show that the information extracted by our voice level estimators reflects the perceived voice level in two ways. Firstly, by analysing data labelled with musical dynamic and, secondly, by analysing what the auto-encoder that is trained to use the estimated voice level does with the provided information.

We use recordings that are labelled into different dynamics to investigate the relationship between dynamic and the proposed voice level estimator. Labelled

data was obtained by Ardaillon (2017) who asked a singer to sing one note in all combinations of the dynamics pp, mp, mf, f and ff and all French vowels on the same pitch. Thus, we can create histograms over the estimated voice level for different dynamics and vowels. If the proposed method reflects the true voice level, lower dynamics should correspond to lower values in voice level.

Furthermore, we claim that the proposed voice level estimate is useful for changing the perception of how loud a phrase has been sung. Thus, if the proposed auto-encoder indeed succeeds in changing the signals properties in a way that it is perceived as sung with the desired voice level, we can conclude that the proposed voice level estimators indeed encode the voice level. To this end we asked 40 participants in a perceptive online study to rate the perceived voice level change of the transformed audio. Participants were presented pairs of audio where for each pair both files were generated using one of the auto-encoders and where in one file the voice level was changed while for the other the voice level was kept the same. Participants were then asked to rate which recording sounded as if it was sung with a stronger / louder voice and could give an answer of -2, -1, 0, 1 or 2. The order within each pair and the overall order of the pairs were randomized and the volume of each of the files was normalized to the same average loudness according to the loudness model of Glasberg and Moore (2002). In a second test, we asked 26 participants to rate the audio samples for their audio quality and computed a mean-opinion-score for each of the voice level changes of both models and the ground truth reference.

The auto-encoder's precision is evaluated by measuring the average difference between requested voice level and voice level measured in the synthesized mel-spectrograms for various voice level changes. The results are given in Table 7.1. The samples used for the perceptive test are available [online](#)<sup>4</sup>.

## 7.5. Results

### 7.5.1. Classification of dynamic

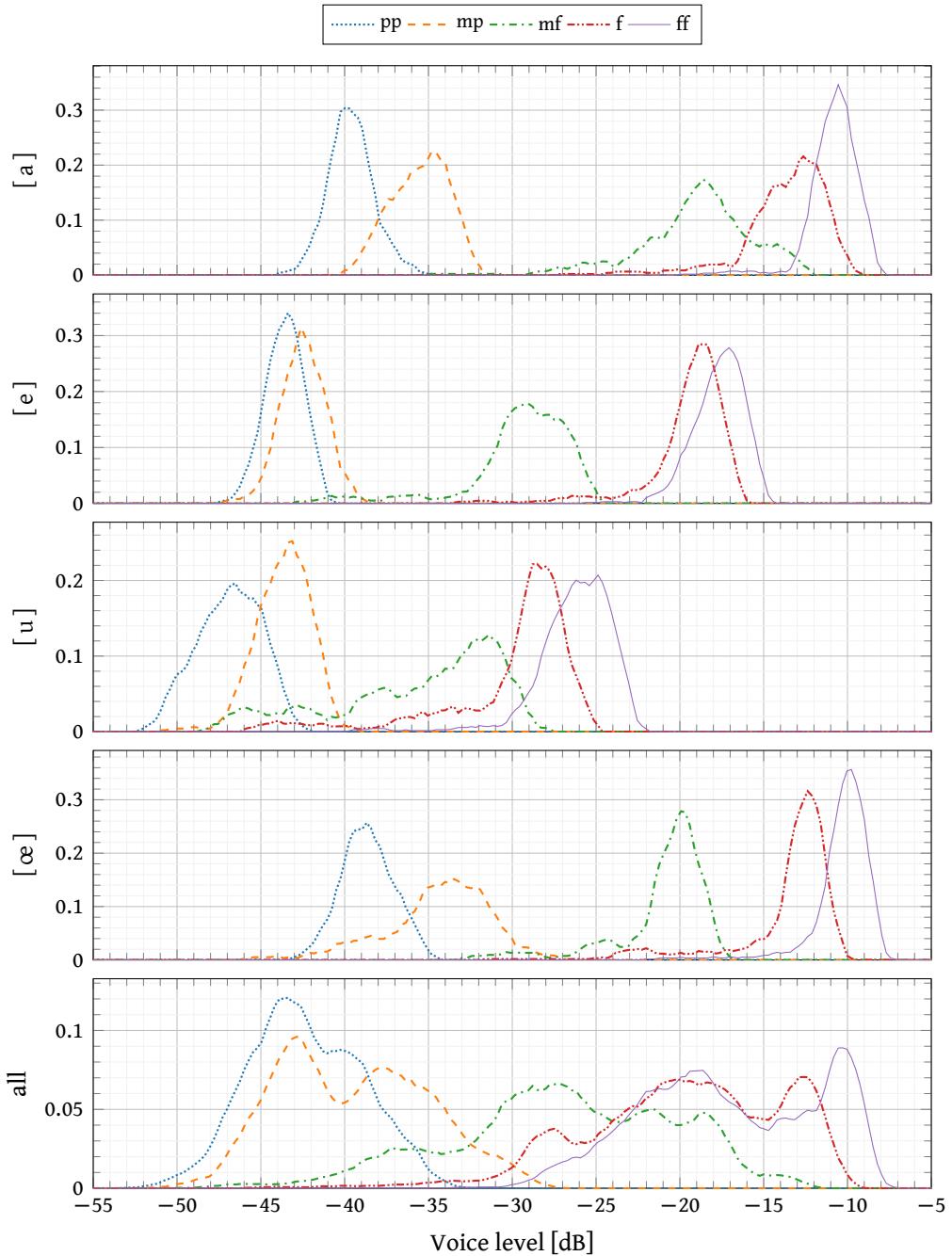
Figure 7.1 visualizes the relationship between estimated voice level estimator of Ad and the symbolic dynamic annotation of the recordings used for this investigation. Similar histograms are produced by estimator Le. First, we observe that louder dynamics produce higher voice level values, so we can conclude that the voice level does indeed include information about how loud a note has been sung. If we look at the individual phonemes, we see that the dynamics create different clusters with increasing mean though for some vowels some dynamics have rather large overlap, e.g. for [e] the pp and mp are largely overlapping: This is because we see the histogram of the voice level for each frame which naturally varies over time for the same dynamic for instance as a side effect of vibrato. Thus while some selected frames from different dynamics may be equally loud, the overall notes still have different loudness and can be measured as averages over larger durations. Furthermore, dynamics are subjective and a matter of interpretation. It could be

---

4. <http://thesis.fnab.xyz/ch7/demo>

## 7. Transformation of perceived voice level

---



**Figure 7.1.:** Estimated voice level values for recordings in different dynamics plotted for the voice level model Ad. The graphs show smoothed relative histograms using a Gaussian kernel with  $\sigma = 0.31$  dB. Only frames inside stable phonation are used, unvoiced frames and frames close (50 ms) to a voiced / unvoiced boundary are ignored. We show histograms for four selected vowels (out of a total of 15 vowels in the French language) and the average over all vowels (bottom). All graphs share the same horizontal axis.

**Table 7.1.**: Transformation precision [dB] of the auto-encoders. Errors are calculated between target voice level (as given to the auto-encoder) and voice level of the output (the voice level estimator applied to the transformed mel-spectrogram) for various voice level changes. The error is calculated on a logarithmic scale to provide meaningful values.

|     | -16 dB | -10 dB | 0 dB | +10 dB | +16 dB |
|-----|--------|--------|------|--------|--------|
| AdA | 4.02   | 2.49   | 1.43 | 3.22   | 4.15   |
| LeA | 4.50   | 2.91   | 1.35 | 4.97   | 7.77   |

that this particular singer does not make such a big difference between pp and mp for an [ e ] in terms of voice level.

Comparing the histograms from different vowels we observe that the voice level can be very different for different phonemes under the same dynamic. This is because not all vowels are in fact equally loud. The vowel [ a ], being an open vowel, can be sung much louder than [ u ], being a close vowel. Thus, for the same dynamic we can expect the resulting voice level to be different, and it is no surprise that the estimated voice level for [ a ] is much higher than for [ u ]. This explains why, if we average over all vowels, adjacent dynamics overlap — because the variations in voice level due to the phoneme are on the same scale as the variations due to the dynamic. Nevertheless, we can still very clearly distinguish between loud dynamics (f, ff) and quiet dynamics (pp, mp).

### 7.5.2. Accuracies

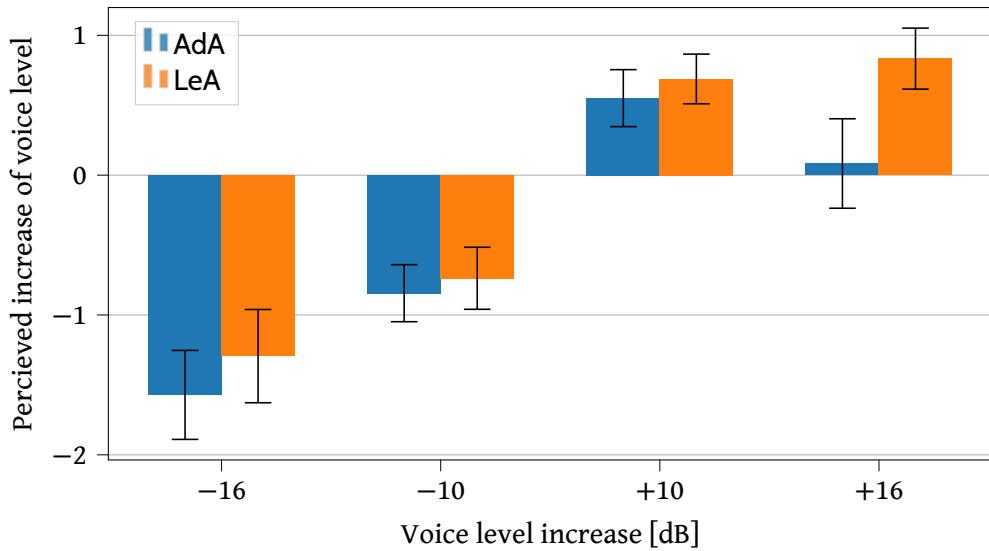
From Table 7.1 we can see that the auto-encoders can use the information provided by the voice level estimator and successfully disentangle the voice level from the mel-spectrogram. The auto-encoder with adapted recording factor (AdA) performs with higher precision than the auto-encoder with learnt recording factor (LeA). This indicates that the adapted recording factor model (Ad) is more robust than the learnt recording factor model (Le). Given that the assumptions for Le are stronger than for Ad, the difference could be explained by some of those assumptions being in fact violated, which caused the voice level estimator Le to provide inconsistent information.

### 7.5.3. Perceptive test

The results of the perceptive test are plotted in Figs. 7.2 and 7.3 and given in Tables 7.2 and 7.3. The perceived voice level increase is given in Table 7.2. We can see, that both models are able to create a noticeable change in voice level for small changes in the target voice level. For the auto-encoder with adaptive recording factor (AdA) we observe that it has trouble creating convincing results for high increases in voice level. Investigating the files that were rated in the opposite direction we noticed that for those files the auto-encoder introduced significant amounts of artefacts which seemed to have created the opposite of the desired effect. The perceptive test suggests that the auto-encoders have a more noticeable impact when decreasing

## 7. Transformation of perceived voice level

---



**Figure 7.2.:** Perceived voice level change caused by changing the voice level with the proposed auto-encoders for various amounts. The values in this figure are averaged over all answers. Error bars represent the 95% confidence interval.

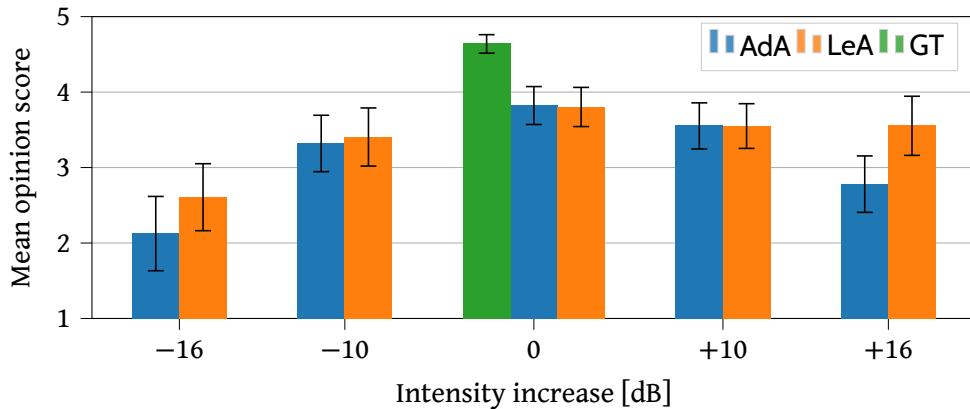
**Table 7.2.:** Change in voice level as perceived by participants of a perceptive test on a subjective scale from -2 to 2 with 95% confidence intervals.

| Voice level | -16 dB       | -10 dB       | 10 dB       | 16 dB       |
|-------------|--------------|--------------|-------------|-------------|
| AdA         | -1.57 ± 0.32 | -0.84 ± 0.20 | 0.55 ± 0.20 | 0.08 ± 0.32 |
| LeA         | -1.29 ± 0.33 | -0.74 ± 0.22 | 0.69 ± 0.18 | 0.83 ± 0.22 |

the voice level, which can be seen for both models and for all amounts of change. The auto-encoder with adaptive recording factor AdA seems to create a stronger effect when decreasing the voice level than the auto-encoder with learnt recording factor LeA. For increasing the voice level LeA seems to be better suited than AdA.

The quality ratings are given in Table 7.3 For self-reconstruction and small amounts of voice level change both models, AdA and LeA, seem to work equally well. For large changes in voice level the auto-encoder with learnt recording factor (LeA) outperforms the auto-encoder with adaptive recording factor (AdA) by a margin. For increases in voice level LeA is able to hold its level of quality even for an increase of 16 dB. On the other hand LeA does make a larger error in Table 7.1. For decrease in voice level both auto-encoder models suffer strong degradation in quality, although both models were successfully able to convince the participants that the recordings had much less voice level. Listening to these samples reveals that the auto-encoders increase the background noise significantly. Since our mel-inverter MBExWN does not handle synthetic noise well, the overall audio quality is poor in these cases although the conversion itself is realistic.

From the test results we can conclude that the given auto-encoders were able



**Figure 7.3.:** Subjective quality as reported by the participants on a scale from 1 to 5. The values in this figure are averaged over all answers. Error bars represent the 95% confidence interval.

**Table 7.3.:** Quality rating given by participants of a perceptive test on a subjective scale from 1 to 5 with 95% confidence intervals.

| Quality | -16 dB      | -10 dB      | 0 dB        | 10 dB       | 16 dB       |
|---------|-------------|-------------|-------------|-------------|-------------|
| GT      |             |             | 4.64 ± 0.12 |             |             |
| AdA     | 2.12 ± 0.49 | 3.32 ± 0.37 | 3.82 ± 0.25 | 3.55 ± 0.31 | 2.78 ± 0.37 |
| LeA     | 2.61 ± 0.44 | 3.40 ± 0.39 | 3.80 ± 0.26 | 3.55 ± 0.30 | 3.55 ± 0.39 |

to change the perceived voice level in singing voice, and therefore the voice level estimators capture the information about the true voice level.

## 7.6. Conclusions

We have introduced a method to estimate the voice level from recordings with unknown amplification factors (recording factor). Two variants to overcome the missing recording factor annotation have been introduced: either by learning the unknown recording factor alongside the weights of the neural network (Le) or by adjusting the loss function to remove scaling and only compare the contours of the network's output and the associated signal loudness (Ad). These voice level estimators have been used to condition a bottleneck auto-encoder to disentangle the voice level from the mel-spectrogram. We have shown that both models produce consistent values and can produce the effect of changed voice level on singing recordings in most cases and with acceptable quality.

While the proposed auto-encoders produce a noticeable change in voice level, the audio quality is still significantly lower than real recordings especially when increasing the voice level. Consequently, this first publication on neural voice level transformation has to be seen as a proof-of-concept rather than a well-polished system. Improvements to the audio quality are required for this method to be used

## *7. Transformation of perceived voice level*

---

in actual musical production. Nevertheless, the estimation method for the voice level opens new ways of voice classification, analysis, and transformation.

## 8. Applications

**S**eeking knowledge through scientific research can be an exciting goal in itself. Yet often the question arises, what all of this is good for. Thus, we conclude this thesis with a discussion of artistic projects that I was involved in during my thesis and that were using the technology presented in the previous chapters. A discussion of the applications is an important part as it gives justification to what we do and can be used to motivate further directions for development. In the case of this thesis much of the work was motivated by the artistic residency of Judith Deschamps who asked us to reinvent the voice of 18th century castrato singer Farinelli.

In general, research at **IRCAM** is largely motivated by creating tools for musical composition. Therefore, it is a welcome confirmation of this work that the pitch and voice level transformations introduced in the previous chapters were used in some compositions created at **IRCAM**, notably by the composers Aïda Shirazi (Shirazi, 2022), Sachie Kobayashi (Kobayashi, 2022) and Omer Barash (Barash, 2023).

### 8.1. Farinelli singing voice

**IRCAM**, as an institution primarily concerned with artistic creation, offers artistic residencies aimed at composers and artists of other domains. Artists in residence can research their artistic ideas as part of one of **IRCAM**'s research groups. This is meant to encourage collaborations between scientific research and artistic creation. Such an artistic residency happened during my thesis between the artist Judith Deschamps and my research group.

One of the reoccurring themes in Deschamps's work is the transience of human and non-human lives and our (futile) attempts to prevent body changes. Since 2018, she started a series of works about the phonatory apparatus, and how History invested this specific part of the body. This led her to discuss the practice of castration in classical music, an attempt to preserve the youth of a boy's voice. In particular the story of 18th century castrato singer Farinelli was a central interest to her. She started her artistic residency with the goal to reinvent the voice of Farinelli using deep neural networks. The voice was supposed to be the product of hybridization between multiple voices covering different parts of Farinelli's singing range. The artistic residency of Judith Deschamps started in January 2021 and yielded a film (Deschamps, 2022b) and a sound installation.

### 8.1.1. The castrato voice of Farinelli

In the musical tradition of the baroque era, castration of young boys was a common procedure to produce high male voices. The castration of young boys prevents puberty due to the lack of testosterone hormones. As a consequence the rapid growth of the vocal folds, that causes the voice break in uncastrated males, never occurs. The vocal fold length remains in the range of a child's or soprano's as has been observed by post-mortem examinations (Tandler and Grosz, 1909). While the larynx of a castrato does not experience a rapid growth in teenage years, the remaining vocal tract does. As a result the voice of a castrato, although similar in pitch, was very different from the voice of a soprano, countertenor, or child (Jenkins, 1998). In particular extensive training and the greatly increased lung capacity allowed castrati to sing extremely long phrases in one breath and with great flexibility and virtuosity. For example, the castrato singer Farinelli was said to be able to 'hold a note for a whole minute without taking breath' (Jenkins, 1998).

Farinelli (1705 – 1782), born as Carlo Maria Michelangelo Nicola Broschi, was one of the most famous castrati and one of the most famous opera singers of all times (Desler, 2014). In 1737, at the age of 32, and at the height of an extremely successful career throughout Europe, he followed a summoning by the Spanish royal house. At the Spanish court, Farinelli was employed as Philip V's private chamber musician. The Spanish king was experiencing severe depression and Farinelli's task was to sing for the king every night to sooth his melancholy. Contemporary accounts mention Farinelli singing the same few songs every night for over ten years, though this extremely repetitive repertoire has been doubted in later literature (Desler, 2014, pp. 142). The ritual between Farinelli and the Spanish king is subject of Deschamps's work on Farinelli. Farinelli remained at the Spanish court after Philip V's death in 1746 and continued to serve for the new King, Ferdinand VI, until his retirement in 1759.

An account on Farinelli's vocal range can be found in Desler (2014, Ch. 3). Farinelli was famous for his large vocal range, one of the distinguishing abilities that set him apart from other singers. At his prime he was able to sing up to D6<sup>1</sup>, but his vocal range was likely diminishing in his later career. By the time he was employed at the Spanish court most of his repertoire was in the range B3–F5 and only rarely exceeded F3–B5. This impressively large vocal range allowed Farinelli in his prime to sing anything written for tenor, alto, or soprano comfortably. This large vocal range is one of the key challenges for anyone trying to recreate the voice of Farinelli in its entirety and was influential to our own design choices.

### 8.1.2. Related work

It is not the first time that **IRCAM** attempted to recreate the voice of Farinelli. Depalle et al. (1994) merged the voices of a countertenor and a soprano for the soundtrack of the 1994 film *Farinelli* (Corbiau, 1994). The soundtrack contains arias sung by the

---

1. Concert pitch at the time was about 100 cent to 200 cent lower than today and varied widely throughout different locations, so an exact frequency cannot be given. D6 today corresponds to 1174 Hz.

fictional Farinelli voice. These arias were recorded by two singers, a countertenor and a soprano, who together covered the whole range of Farinelli's vocal range. To create the impression of a single voice performing the entire repertoire, the female voice was modified through means of voice identity conversion to the voice of the countertenor. The voice identity conversion was achieved by adjusting the amplitudes of the harmonics in the voiced segments to match the recorded statistics of the countertenor. Separate statistics were recorded for each vowel, pitch and three vocal intensities. The adjustment of the partials was achieved by frequency domain filters with separate rectangles around each partial. The filter amplitudes were selected in such a way that the converted voice had the same relative amplitudes between the harmonics as in the target voice reference recordings.

Resurrecting historic voices has many applications in entertainment. When actors die during the time of a production before it can be finished, additional voice recordings of the deceased actors may be required to finish the production and do justice to the actors last work. In documentaries where acoustic material has not been recorded or has been lost, recreating the material artificially can help to give the narrative more realism. In fictional stories it could be desirable that the deceased should make an appearance. If recordings of the target voice exist, voice reenactment (Bous, Benaroya, et al., 2022) can be used to allow an actor to imitate the voice of the deceased. Another possibility is to train a text-to-speech model on the target voice and synthesizing the voice using the script alone. Voice reenactment has been used by the Analysis/Synthesis team of **IRCAM**, to recreate the voice of 20th century chanson singer Dalida in the television series *Hôtel de Temps* for the French television network *France 3* and to recover a lost recording of Charles de Gaulle's *appel du 18 juin 1940*.

Several pieces from Farinelli's repertoire are still being sung regularly by other voice types, e.g. sopranos or countertenors. Historically informed performances can be seen as an imitation of Farinelli's voice, not with computers but with other voices. The piece *Quell'usignolo* which we reinterpret in the following sections has been recorded on numerous occasions.

### 8.1.3. Musical material

The goal of this project was to reinterpret the piece *Quell'usignolo*, by Geminiano Giacomelli using a synthetic reinvention of the voice of Farinelli. *Quell'usignolo* is one of the pieces that is mentioned frequently among the pieces that were sang every night and might thus have been Philip V favourite piece. It is an aria from the opera *la Merope* by Geminiano Giacomelli, premiered in 1734 with Farinelli in the cast. This piece was used as the basis for our virtual voice of Farinelli.

This piece is of particular interest since handwritten notes by Farinelli himself exist (Broschi, ). 'This piece [...] has been transcribed by Farinelli for a manuscript sent to Maria Theresa of Spain, to demonstrate the quality of his voice. The score includes his [Farinelli's] own ornamentations [...]' (Deschamps, 2022a, translation mine), and has been influential to Deschamps's artistic work on Farinelli. The manuscript provides insight on how Farinelli might have interpreted the piece,

## 8. Applications

---

**Table 8.1.: Phrases in the piece Quell'usignolo.**

| Phrase  | Translation  |
|---|--|
| <b>Exposition (A) and recapitulation (A')</b>           |  |
| (Q) <i>Quell'usignolo che innamorato</i>                | The nightingale who fell in love                           |
| (C) <i>Se canta solo</i>                                | When it sings alone among the branches                     |
| (F) <i>Fra fronda e fronda</i>                          | From leaf to leaf  |
| (S) <i>Spiega del fato la Crudeltà</i>                  | Laments the cruelty of fate                                |
| <b>Trio (B)</b>   |  |
| (W) <i>S'ode pietoso nel bosco Ombroso</i>              | Pitifully heard in the dark forest                         |
| (X) <i>Chi gli risponde con lieto core</i>              | Who will answer with a happy heart                         |
| (Y) <i>Di ramo in ramo cantando va</i>                  | Singing it goes from branch to branch                      |
| (Z) <i>Con lieto core nel bosco Ombroso cantando va</i> | With a happy heart it goes singing through the dark forest |

**Table 8.2.: Structure of the piece Quell'usignolo.**

| <i>Exposition (A)</i> | <i>Trio (B)</i> | <i>Recapitulation (A')</i> |
|-----------------------|-----------------|----------------------------|
| Qa Ca Fa Sa           |                 | Qa' Ca' Fa' Sa'            |
| Qb Cb Sb              |                 | Qb' Cb' Sb'                |
| Qc Cc Fc Sc           | W X Y Z         | Qc' Cc' Fc' Sc'            |
| Qd Fd Cd Sd           |                 | Qd' Fd' Cd' Sd'            |

about his vocal range and his abilities. This score was used as the basis for an arrangement created by the composer António Breitenfeld Sá-Dantas.

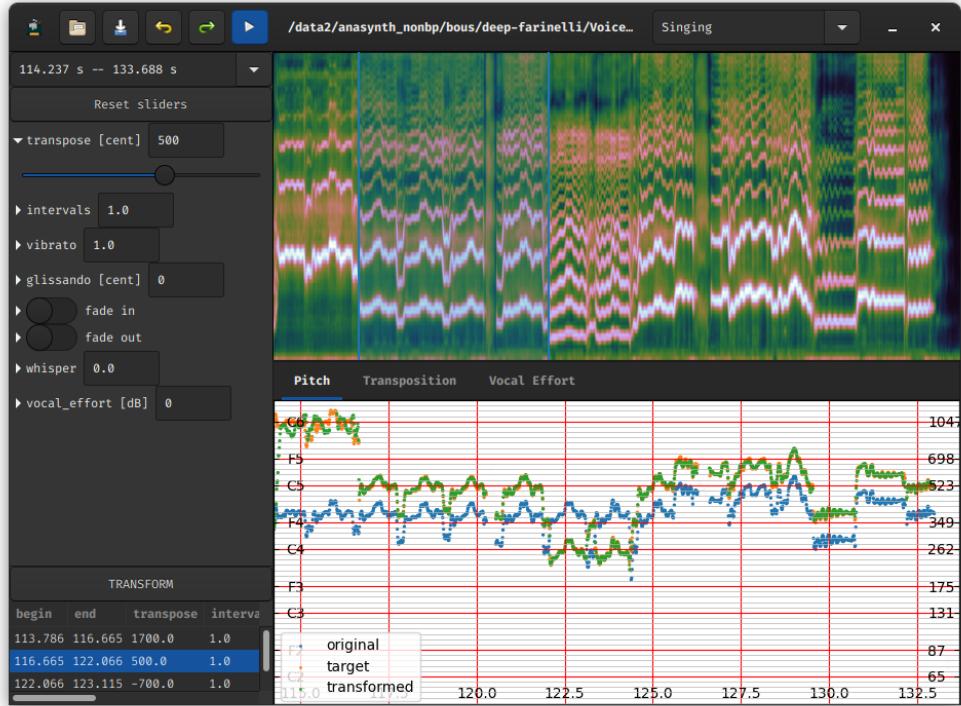
The structure of the original piece is a typical (ABA') structure, with a fast exposition (A), a modified recapitulation (A') and a slower trio (B) in the middle. The exposition (A) consist of 15 phrases, each one beginning with a text part using one line from the lyrics, followed by an optional cadenza. The lyrics of the different lines are given in Table 8.1. The lines (Q), (C) and (S) reoccur four times each. Phrase (F) is omitted once and occurs only three times. The text parts of (A') are slight variations of (A) while the cadenzas are almost completely different between (A) and (A') with much more extended cadenzas in section (A'). The trio takes more liberty in its structure. We can roughly divide the trio into four phrases, (W), (X), (Y) and (Z) with some cadenzas interjected within the phrases. An overview of the structure of the piece is given in Table 8.2.

### 8.1.4. Creating a hybrid voice

To create an artistic impression of Farinelli's voice we combine different voices to generate a hybrid voice that has an unusually large range. This is achieved by the pitch transformer from Chapter 6 utilizing its tendency to convert voice identity depending on the transposition and target pitch.

## 8. Applications

---



**Figure 8.1.:** Screen shot of CIRCE, where pitch transformations are applied to a section of the piece Quell'usignolo. The left panel shows different elementary transformations, the top panel shows the resulting mel-spectrogram and the bottom panel shows the different  $f_0$ -curves for original, target and transformed. In the bottom left corner we see a list of applied transformations.

Due to the unusually high pitch range, only very few singers would be able to perform the whole score. Thus, we use two different versions of Quell'usignolo, arranged by Sá-Dantas: The *target score* follows closely the manuscript of Farinelli, and is our interpretation of how Farinelli might have sung the piece; this is what we want as the result. The *record score* is a simplified version of the *target score*. It follows the same rhythm as the *target score* but has a reduced vocal range, obtained by transposing the higher sections down and the lower sections up. Thus, the *record score*, lies in the alto range and can be sung by any trained singer of that range. The Farinelli voice is created by recording an alto singing the *record score*, and transposing the recording of the *record score* to match the notes of the *target score* using the neural voice transformation framework. The *target score* is in C major while the *record score* is in G major.

To create the correct timings for the transformations and apply the transformation to the recording we created a custom software, called CIRCE, for manipulating the mel-spectrogram interactively. Figure 8.1 shows a screenshot of CIRCE applying the pitch transformations on a section of the piece.

The pitch transformer from Chapter 6 has a tendency to change the voice identity



**Figure 8.2:** The installation *an-other voice* by Judith Deschamps, where the Farinelli voice was exhibited. A mysterious object in the wall invites climbing on the ramp. At the top of the ramp a motion sensor is activated, and the Farinelli Voice can be heard through the use of directional speakers. [© Lynn Theisen]

with increasing amount of transposition. This is necessary to create realistic sounding voice even for pitches that are impossible to sing for the source voice. Thus, with increasing amount of transposition we can observe a gradual transition from one voice identity to another. The target voice can be influenced to some degree by carefully choosing the training dataset. If for a specific pitch range only one voice was recorded, the auto-encoder may overfit to this voice on this particular pitch range. Consequently, any sample synthesized in this pitch range will strongly resemble the voice from the training set. Conversely, this effect is mitigated if more voices cover the target pitch range. If there are sufficiently many voices representing the target pitch, the auto-encoder can approximate the source voice more closely in the target range and create a smooth gradual transition.

To reinvent Farinelli's voice we created a custom dataset with seven voices covering the alleged vocal range of Farinelli: two pre voice-break child voices, one soprano, two altos, one countertenor and one tenor. All singers sang the parts of the *target score* that were within their range, except for the altos who were recorded performing the whole *record score*. Some singers also provided additional material from their own repertoire.

### 8.1.5. Improvisations over *Quell'usignolo*

In our original approach, the piece was produced using the arrangement by Sá-Dantas as described in the previous section (the 'target score'). Parts of this reinterpretation have been used in the Film *La Mue* (Deschamps, 2022b). For artistic reasons we later decided to create random variations of the score to allow the piece to be different every time it is performed. These random variations were exhibited as part of the show *an-other voice* at Casino Luxembourg between 17th of February and 14th of April 2023. A photograph of the installation is depicted in Fig. 8.2. In this installation the voice was improvising new versions of *Quell'usignolo* using rule-based rearrangements of sections of the recording. The recording was split into small

fragments, consisting of only a few notes each, that captured elementary melodic building blocks and individual ornamentations.

The structure of the piece, in particular of the exposition (A) and the recapitulation (A') allow us to reshuffle the structure on multiple levels.

1. The text-parts of each phrase can be subdivided into smaller fragments. The fragments from (A) and (A') can be used interchangeably to create a new version of the text-part.
2. The cadenzas can be split into smaller fragments. For each fragment we define a set of allowed successor fragments. Possible successor are classified into fragments that cause smooth transitions, rough transitions or end the cadenza. Depending on whether we want a smooth transition, a rough transition or exit the cadenza we pick the next fragment from the appropriate class randomly.
3. The different versions of each phrase can be used interchangeably. At the end of each phrase either a random version of a different phrase or the trio may follow. Between two phrases there is a short rest of a few seconds.

This process constitutes a Markov chain over the set of all fragments. Formally we can define a transition matrix that defines the probabilities of going from one fragment to the next. This transition matrix defines the piece and choosing the non-zero elements is the central part of the recomposition, which was done by Sá-Dantas. To obtain simple controls for the value of the non-zero elements we parametrize the matrix by three elementary probabilities. We define probabilities for smooth transitions  $p_s$ , rough transitions  $p_r$  and cadenza exits  $p_e$  and divide the probability by the number of fragments in each transition class.

To further increase the variability of the improvisation we allow key changes to occur:

- Randomly the key can change by going up or down by a semitone. If a maximum or minimum key is exceeded we change the key by a fourth in the opposite direction instead. There is always a larger probability to go back towards the original key rather than to go away from it.
- If a fragment is repeated within the last five fragments the key automatically goes up by a semitone.

Since the key changes depend on the last five fragments, the key changes do not form a Markov chain. We use 11 keys in total, from E to D which further extends the originally-envisioned *target score* by a second up and a minor sixth down.

The improvisation is run on a conventional computer which is part of the installation. The computer can be seen in Fig. 8.3. Since the number of variations is limited, we pre-compute all the required fragments for all the possible keys. Then, during improvisation the program cross-fades between the different fragments. Thus, the neural transformations do not have to be run on the installation hardware which allows for a light-weight setup. During the installation a screen provides information about the state of the improviser through a logging output and a small screen.



**Figure 8.3:** The computer running the improvisation of the Farinelli Voice is part of the installation. A screen shows the state of the improviser and the transition probabilities. Behind the screen the hardware is shown to the visitor in an open case. [© Lynn Theisen]

### 8.1.6. Observations and difficulties

For the proposed approach to work, the *record score* could have theoretically used any arbitrary notes as long as the rhythm matched the *target score*. However, to allow the singer who interpreted the *record score* to sing with a similar overall expression as would be required for singing the *target score*, the melody in the *record score* was mostly the same as the *target score* but in a different key. Most of the time, lower parts of the melody were transposed up by a fifth while higher parts were transposed down by a fourth to create the simplified *record score*. This also made the process of creating the transformation of the recording easier, as large sections could be selected at the same time rather than individual notes.

When we applied the transformation to the recording, however, we noticed that the tenor voice would be very prominent in all sections that were transposed down. This effect occurred regardless of whether the result was in a range exclusively covered by the tenor or not. Especially the transitions were noticeable and sometimes created the impression of another voice taking over. Further investigation revealed the following observation: Suppose we use two notes from the same singer on the same vowel, one high note and one low note. We transform both notes to the pitch of the low note. We had expected that both notes (the note that was originally high and the note that was low all along) would sound similar. This is, however, not the case. Singing a high note requires a different singing technique than singing a low note. The auto-encoder preserves the singing technique as part of the latent code. The transposed note cannot be sung by the source singer using the original singing technique. Instead of adjusting the singing technique, the auto-encoder adjusts the voice identity. This is the same principle as the previously-discussed adjustment of the voice identity in the case where the new pitch cannot be sung by the source singer at all and could have been anticipated. In the case of the Farinelli

Voice, the transposition amount usually changes from +500 cent to –700 cent, thus by a whole octave. This big change of transposition creates an audible change in voice quality even if the absolute pitches are closely together. (Usually the change in transposition occurs, when the melody in the *target score* moves in the opposite direction, such that the melody in the *record score* does not have a large jump.) In the context of the installation this is an interesting artefact of the voice and allowed the artist to showcase the weirdness and imperfections of our method. From the scientific point of view, however, this suggests that more precise control of either voice identity or singing technique is necessary to remove the ambiguity created from providing a target pitch alone.

Another difficulty we observed was concerning extremely high pitches. Using the set-up from [Chapter 6](#) we could never transform the pitch to anything higher than F5 (700 Hz) even if samples containing these pitches were drawn with more than 50% probability. Experimentation suggested that the auto-encoder treated these high pitches not as voiced speech but rather as a separate phoneme with the positions of the partials encoded in the latent code (possibly confusing partials with formants): Transforming from fundamental frequencies below 700 Hz to above 700 Hz saturates at 700 Hz and transforming fundamental frequencies above 700 Hz does not change the result in any systematic way. This suggests that the bottleneck size was not narrow enough for the higher pitches and thus different bottleneck sizes are required for different input pitches.

To solve this problem we implemented an adaptive bottleneck size. During training, we apply dropout to the latent code. The dropout probability is calculated for each time step separately depending on the input fundamental frequency. The dropout probability increases linearly from 0 to 1 starting at 0 for the minimal pitch A2 (45 Hz) until complete erasure at the maximal pitch F6 (1400 Hz). No dropout is applied for unvoiced segments. During inference no dropout is applied. With the adaptive bottleneck even the high-pitched sounds could be disentangled from their fundamental frequency. The adaptive bottleneck allowed us to increase the bottleneck size from 3 dimensions to 8 dimensions. Thus, as a welcome side effect, the larger bottleneck transmits more information through the auto-encoder, in particular for the unvoiced segments, which also increased the audio quality significantly.

### 8.1.7. Conclusion

We created an artistic impression of the voice of Farinelli, by merging different voices covering the large vocal range of Farinelli. Previous work by Depalle et al. (1994) created the final recording by patching together the recordings of two singers performing different sections and a voice identity transformation on one of the voices to match the timbre of the other. In this work the final material was obtained by recording only one singer. This recording was then transformed using the neural pitch transformation from this thesis to increase the vocal range of the singer. While the input voice for the transformations was only provided by one singer, the neural transformation itself was trained on different singers covering the whole range

of Farinelli. The voice was used in a sound installation where the exhibited piece was varied by randomly rearranging the sections and cadenzas. The cadenzas were improvised by rearranging fragments from the cadenzas. Key changes were used to reduce repetitions.

## 8.2. Musical compositions with CIRCE

To implement the transformations required for the Farinelli project, the computer program CIRCE was developed. Though originally only envisioned for a single purpose, the program is itself useful for other artistic creation as well. CIRCE has been available for download on the [IRCAM Forum website](https://forum.ircam.fr/projects/detail/circe/)<sup>2</sup> for free since autumn 2022 and receives updates with new features and new models every few months. Three noteworthy musical compositions have made use of the neural voice transformation framework, developed in this thesis, through CIRCE. The collaborations around these pieces have helped develop the software by uncovering the problems and requirements in a real application.

### 8.2.1. Aïda Shirazi: *Né entre corps* (2022)

Additional media: <http://thesis.fnab.xyz/ch8/shirazi>



The composition *Né entre corps* by the composer Aïda Shirazi is a piece for electronics and dance. The dancer is hidden behind a screen and only their shadow is projected onto the screen. Using different light sources, the shadow can be formed in a variety of ways. (See Fig. 8.4 for a visual example.)

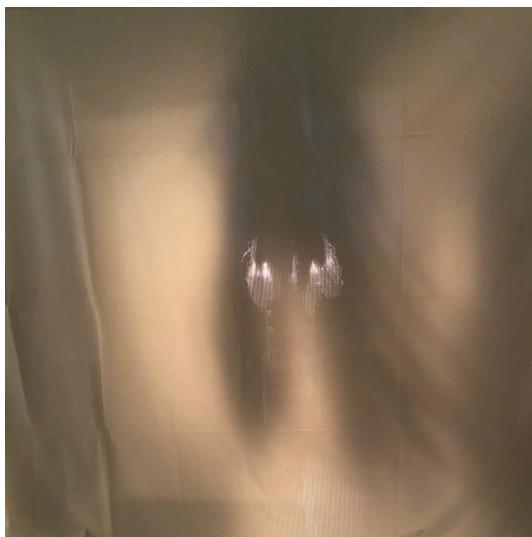
The music is a setting of a poem by Haleh Ghassemi. The poem is read in two different versions, the Persian original and a French adaptation by Irène Gayraud. The voices of the poet and the translator were recorded reading their respective versions of the poem with additional voice recordings of the composer herself. The voice recordings were treated using spectral filtering and reverberation as well as the neural voice transformation framework from this thesis. The accompaniment is composed of analogue synthesizers as well as samples of the sounds from silverware, pots, pans, glass, pepper and salt grinders and knives cutting on glass. Droning granular sounds evoke the general atmosphere of the texts. Additionally, to the fixed media, the breathing noise of the dancer is picked up by a headset microphone and mixed with the other electronics.

The visual aspect of the piece shows a female dancer behind a screen. Thus, only the shadow of the dancer is visible. This effect is used to remove the identity of the performer: to create a neutral character of a woman, reduced to only its ‘timeless features, free from stereotypes and expectations of the physical appearance of a ballerina.’<sup>3</sup> The projected image slowly changes from the abstract to the concrete. Initially we see only abstract shapes that are not recognized as a human body, while later a female body becomes recognizable. The projected shadow obscures the female

---

2. <https://forum.ircam.fr/projects/detail/circe/>

3. This and the other quotes in this section are from my personal correspondence (PC) with the composer.



**Figure 8.4:** Shadow play from Né entre corps by Aïda Shirazi. The body of a female dancer is projected on a screen to produce an abstract representation of the body. This is mirrored in the voice using the neural voice transformation framework. [© Aïda Shirazi]

body and plays with ‘the visual representation of the body’ (Shirazi, PC). It questions ‘the preconceived notion of the body of the female dancer’ by ‘deconstructing and expanding and abstracting the image of human and female body’ (Shirazi, PC). A shadow projection from the piece is shown in Fig. 8.4.

The voice parallels the philosophy of the shadow play. Like the visual appearance of the human, the acoustic appearance is masked and distorted through unusual reading and audio effects. Similar to the staged movements of dance, the composer created ‘distortion of the words through different means of utterance [during the recording], such as stuttering and other exaggerated moods of expression’ (Shirazi, PC). The voice recordings were further distorted by filtering, reverberation, the neural voice transformation framework and other audio effects. The neural voice transformation framework allowed Shirazi to play in the area between whispering and clear speech. To her, the whisper is the acoustic counterpart of the shadow, creating a mystification of the voice that hides otherwise obvious aspects of the voice. Using the voice-unvoiced mask in the control signal creates a ‘nuanced treatment between [whisper and clear speech]’ (Shirazi, PC).

### 8.2.2. Sachie Kobayashi: *Day 0 - trans-instrumentalism* (2022)

Additional media: <http://thesis.fnab.xyz/ch8/kobayashi>



The composition *Day 0 - trans-instrumentalism* by the composer Sachie Kobayashi is a multimedia piece for fixed media and live electronics. It is a theatrical piece for one performer without dialogue. The piece tells the story of a person waking up one day to find themselves turned into a musical instrument, much like in Kafka’s short story *Die Verwandlung* (Metamorphosis). Throughout the piece the character descends further and further into a dream world that could have been the result of imagination or hallucination.

The fixed media part of the piece is composed with synthesizers, field recordings,



**Figure 8.5:** Scene from Day 0 – trans-instrumentalism by Sachie Kobayashi. The character in the piece discovers that they have turned into a musical instrument and their everyday gestures, like scratching, produce music. [© Hervé Véronèse – Centre Pompidou]

recordings of previous works of the composer, and audio samples from daily objects and speech. The speech samples are generated by **text-to-speech** synthesis and from recording the composers and my own voice. The voice plays the role of an announcer on the radio, providing the listener with background information about the topic of the piece by reading excerpts from the Wikipedia article on transhumanism. The voice itself is accompanying the progress into the dream-like state of the metamorphosis. Initially a neutral-emotionless voice from **text-to-speech** is presenting the material. Later the strongly-transformed voices of real speech replace the monotonous announcer. The strong voice transformations are implemented with the neural voice transformation framework using **CIRCE**. The extremeness is achieved by amplifying the pitch variations to range over multiple octaves passing through different voice registers.

The neural transformation introduced an intentional weirdness that was new for the composer. Even within extreme transformation the **neural voice transformation framework** preserves its acoustic naturalness. This is being opposed by the unnatural control curve, the extreme pitch variations, that are used to drive the **bottleneck auto-encoder**. This contradiction between natural sound and unnatural behaviour allowed the composer to enter the region of *uncanny valley* and use it as a narrative device in her piece, which made the neural transformation an attractive tool for her composition.

During this process, the voice identity is obfuscated due to the extreme pitch transformations which move the pitch out of the respective voices range. Kobayashi used the pitch transformations to increase the  $f_0$  range of the speech melody. As a result, the pitch went through different ranges and created intergender voices. This allowed the composer to use her own voice during the composition process to input the voice's expressivity in the most natural way (simply by speaking the way she wants) without inserting herself into the piece explicitly.



**Figure 8.6:** One of the anonymous texts used for the piece G.N.Z. by Omer Barash. Half of the page is missing such that we only know the second half of each line of the poem. (Cambridge University Library T-S 13J10.1) [© Syndics of Cambridge University Library]

### 8.2.3. Omer Barash: G.N.Z. (2023)

Additional media: <http://thesis.fnab.xyz/ch8/barash>



The composition *G.N.Z.* by Omer Barash is a piece for tenor and electronics. It is a musical setting of two anonymous texts from a collection of medieval fragments found in the Cairo Genizah of the Ben Ezra Synagogue. The title *G.N.Z.* refers to the Hebrew root גְּנַז (gnz) meaning to hide, to archive or to store. The name of the Cairo Genizah itself derives from the root גְּנַז. The main theme of the piece is the exploration of ambiguity and betweenness in particular with respect to the question of identity.

Two texts from the Genizah collection form the two parts of the piece. The first text is a poem of which ‘only half of each line has survived’ (Barash, PC). This poem is depicted in Fig. 8.6. The missing half of the poem gave the composer the liberty to fill the gaps as part of the interpretation and plays on the theme of concealment. The music reflects the fragmented nature of the text and fills the missing parts with a heterophonic choir. The second text is a love song which ‘represents a gay experience, not asimilar to a contemporary one’ (Barash, PC), addressing relatable questions such as unachieved love and makes reference to party and drinking. Similarly, the musical setting makes reference to modern rave music, using elements like periodic kicks and the ‘vocoder effect’.<sup>4</sup> Alluding to the holes in the paper, the first poem occasionally shines through during the second poem.

The electronic accompaniment is mostly based on samples of the singer, as well as live effects applied to the voice during the performance. The effects used in the piece are spectral freeze using a phase vocoder, chorus, the ‘vocoder effect’ and the voice transformations introduced in this thesis. Voice transformation is used in this piece in two complementary ways, applied at different times in the piece: First, the electronics are used to create an accompaniment which blends in with the singer and creates a response growing out of the original voice. Second, the voice itself is

4. That is, the effect widely used in pop music of the 1970 and '80s, which has become a stereotype of a ‘robot voice’, thanks to songs like *The Robots* (1978) by Kraftwerk.

## *8. Applications*

---

transformed in real time to make it blend in with its surrounding.

The piece plays with the question of ambiguity and betweenness of identity on several levels. First, the themes of the texts themselves evoke questions about identity, which the composer addresses in his work. Second, the anonymity of the authors leaves room for speculation about the identity of the authors based on content, orthography, and calligraphy of the poems. Barash uses the implicit voice identity transformation of CIRCE to address the question of identity musically. The composer used the pitch transformation to generate new audio samples of the singers voice by imposing new melodies on recordings of long sustained notes. This process allowed Barash to command over a set of derived voices with slightly different identities to be used in ensemble with the original voice. Here, the voice transformation of the neural voice transformation framework was used ‘not so much as an audio effect but rather as a generator’ (Barash, PC). The generated samples act as a new source of singing, which can then be used as an input to the more invasive effects.

# 9. Conclusion

My initial research began with the goal of finding deep learning methods that allow us to transform voice properties in interesting ways. The contemplations around this topic have yielded a neural voice transformation framework with a neural vocoder and a bottleneck auto-encoder. This neural voice transformation framework allows changing fundamental frequency and voice level and could easily be adapted to other voice properties as well.

## 9.1. Recapitulation

After an introduction to the material in [Chapter 2](#) we discussed the fundamental approaches of voice transformation in [Chapter 3](#). In particular, we have introduced the vocoder as a tool to represent voice and how it is used in voice synthesis and transformation. This has led us to more in depth discussion of the vocoder in [Chapters 4](#) and [5](#), which led to the development of an efficient universal neural vocoder as described in [Chapter 5](#). This vocoder was then used in the remaining thesis to provide a practical representation of the voice. In [Chapters 6](#) and [7](#) we have discussed the bottleneck auto-encoder and applied it to fundamental frequency ([Chapter 6](#)) and voice level ([Chapter 7](#)). Finally, we have seen some applications of the voice transformation in [Chapter 8](#).

### 9.1.1. Neural vocoders

In [Chapter 4](#) we have seen a neural vocoder that was based on the classical source filter model. The vocoder of [Chapter 4](#) used a non-parametric glottal pulse signal to define the voiced part of the voice. The glottal pulse signal was estimated from the audio by another neural network. This analysis-synthesis framework allowed jointly training signal analysis and synthesis end-to-end. The vocoder improved state of the art GCI estimation. However, the approach came with considerable drawbacks: While using the glottal pulse signal instead of the fundamental frequency allows modelling irregular phonations more precisely, due to its non-parametric nature, the glottal pulse signal is difficult to transform and thus not well suited for voice transformation. Furthermore, the synthesis quality was still rather poor. Therefore, this approach was ultimately dropped for another parametric representation, the mel-spectrogram.

While [Chapter 4](#) served as a warm-up, in [Chapter 5](#) we developed more systematically a vocoder for voice transformations. We have started the chapter by compiling a list of requirements to a vocoder designed for voice transformations. Then existing approaches were discussed with regard to these requirements. This allowed us to

argue, why in the context of neural voice transformations, the mel-spectrogram was a good representation of the voice. In the remainder of [Chapter 5](#) we discussed the [Multi-Band Excited WaveNet](#), a neural vocoder based on the mel-spectrogram that was a result of the previous contemplations, though the actual development of the [Multi-Band Excited WaveNet](#) was not my own contribution.

### 9.1.2. Bottleneck auto-encoder

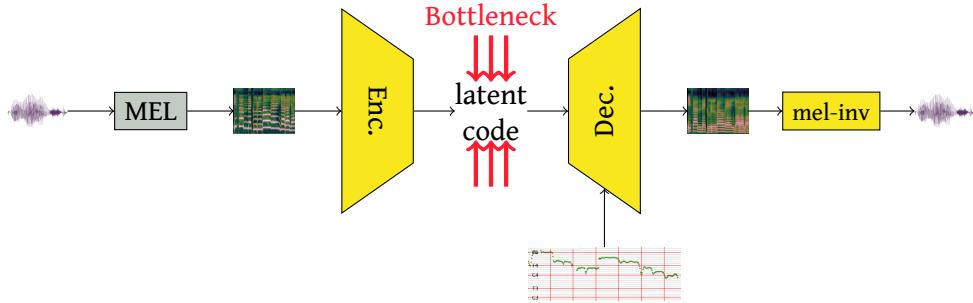
With the neural vocoder providing a practical representation of the voice through the mel-spectrogram, we were able to introduce neural voice transformation framework in [Chapter 6](#). Transformations on the mel-spectrogram were achieved by a bottleneck auto-encoder, which was trained to disentangle the fundamental frequency from the mel-spectrogram. Disentanglement is achieved by an information bottleneck, that forces the auto-encoder to remove information from its latent space. By providing the fundamental frequency to the decoder, the information about the fundamental frequency becomes redundant and is removed. The bottleneck size was studied thoroughly in [Chapter 6](#). We finished the chapter by an analysis of the latent space where we found that information was spread through time through oscillations.

The bottleneck auto-encoder was then applied to the voice level in [Chapter 7](#). With the voice level we have faced the problem that the voice level is not widely available as annotation and no reliable estimation method existed. Thus, in [Chapter 7](#) we have developed an estimation method of perceived voice level that did not rely on explicit voice level annotations. This method assumes that variations in signal loudness are exclusively due to variations in voice level. By matching the loudness contour without regard of scale from the signal timbre alone, the voice level could be successfully estimated. The estimated perceived voice level was used to manipulate the voice level in singing recordings.

## 9.2. The neural voice transformation framework

With all the pieces coming together, we can now summarize the full neural voice transformation framework as depicted in [Fig. 9.1](#):

1. First the mel-spectrogram  $M$  is calculated on the input audio  $a$ .
2. Then we estimate the feature  $F$  that we want to transform from the mel-spectrogram  $M$ .
3. The mel-spectrogram  $M$  and the estimated feature  $F$  are used by the encoder to create the latent code  $c$ .
4. We define the new value  $F'$  for the feature we want to transform either by deriving it from the original value  $F$  or by inventing a completely new curve.
5. The latent code  $c$  and the transformed feature  $F'$  are used by the decoder to resynthesize the mel-spectrogram  $M'$ .



**Figure 9.1:** Full framework during inference.

6. The transformed mel-spectrogram  $M'$  can be further transformed by another auto-encoder or converted to raw audio by the mel-inverter MBExWN.

The neural voice transformation framework consists of two principal components

**Auto-encoder** The auto-encoder disentangles the feature  $F$  from the mel-spectrogram  $M$ . The resulting latent code  $c$  is independent of the feature  $F$ . Resynthesizing the mel-spectrogram using the latent code  $c$  and the transformed feature  $F'$  yields a transformed mel-spectrogram  $M'$ . The auto-encoder was discussed in [Chapters 6 and 7](#).

**Mel-inverter** The mel-inverter converts the transformed mel-spectrogram  $M'$  back to raw audio. In [Chapter 5](#) an implementation of a mel-inverter was introduced, though any other mel-inverter can be used, provided it was sufficiently trained on the voice type that is being transformed.

Additionally, we may require auxiliary neural networks to extract the feature we want to transform from the mel-spectrogram

**Feature estimator** During training and inference, the auto-encoder needs the correct feature value, to work properly. We use a neural network to estimate the feature  $F$  from the same mel-spectrogram  $M$  that is being fed into the auto-encoder. This ensures that the feature  $F$  does not contain additional information that is not present in the mel-spectrogram  $M$ . In [Chapter 7](#) we have discussed extensively how to estimate the voice level from the mel-spectrogram.

### 9.3. Applications

In [Chapter 8](#) we have seen several artistic projects where the neural voice transformation framework has been used. They showed different examples of how the neural voice transformation framework could be used in an artistic application. Generally, there are different categories of applications for the neural voice transformation framework that we will discuss in this section.

### 9.3.1. Audio post-production

After a voice recording is made, for different reasons it may be desired that the original recording would be changed. If the singer of the original recording is not available or unable to make a new recording the existing recording may be altered using the tools developed in this thesis. For example some notes might not have been sung correctly, or we may want to generate alterations of a phrase.

### 9.3.2. Creation of new, hybrid-voices

Similar to the above, but with slightly different implications, we may want to extend the range of a singer. If for example a score requires certain notes that are beyond the singers range, the whole recording might be transposed to map from the original voice to the target range. In this case we benefit from the implicit voice identity conversion that adjusts the voice identity characteristic in a way that allow the higher notes. We will discuss a bit more, the possibilities of controlling the new voice identity in [Section 9.4.2](#) (headline Voice identity).

Furthermore, the voice transformation does not only allow overcoming limitations of a single voice but of what human voice can do in general. In the Farinelli project ([Section 8.1](#)) we have created a voice with an unnaturally large ambitus which singers normally do not have. Similarly, Kobayashi (2022) altered speech recordings by using extreme pitch variations, that pass through pitch registers of multiple voices to create a natural sounding voice with very unnatural behaviour. A human might attempt to imitate the pitch contour, but the voice will betray an extreme exertion. With the auto-encoder the most absurd pitch contours can be followed with ease, creating a super-human voice that is not restraint by the same limits as real speech.

### 9.3.3. Obfuscation and anonymization

In the context of speech, the voice transformation can be used to obfuscate or anonymize the original speaker. This could be used if only one speaker is available, but multiple personalities are required. Another application could be to obfuscate specific aspects of speaker identity, in particular gender by moving the average pitch into a region atypical for that gender. While this does not aim to be a ‘gender transformation’, which is a separate research domain, it allows questioning certain pre-conceptions about voice and identity. This could be used in social or psychological studies to investigate the effect of different voice parameters on the perception of identity.

*It should be noted that the anonymization is not suited to remove all information related to voice identity. It should, therefore, not be used in an application where anonymity is crucial and identification of the original speaker could be consequential.*

### 9.3.4. Singing voice synthesis

The pitch transformations can be applied in parametric concatenative singing synthesis ([Section 3.3.1](#)) in particular the IRCAM Singing Synthesizer. In concatenative

synthesis the phoneme sequence is generated by concatenating samples of diphones (phoneme pairs) from a recorded database. Rhythm and melody are imposed by time-stretch and pitch-shifting. Currently, the IRCAM Singing Synthesizer uses the PaN vocoder of Ardaillon (2017), a parametric vocoder. Using a neural voice transformation framework instead of PaN would increase naturalness especially for the more extreme pitches: the diphone samples are all recorded on only one pitch. For target pitches further away from the original pitch the result can sound unnatural. Currently, the temporal periodicity in the latent code, as observed in Section 6.3.3, prevents time-stretching to be performed on the latent code, because resampling the latent code in time would destroy the characteristic gestures. Two possibilities could be employed: The time-stretching could occur on the mel-spectrogram. Alternatively, the auto-encoder architecture could be modified such that the decoder only sees one frame at a time. In this case the auto-encoder cannot spread information through the time axis and interpolation can be done on the latent code.

Furthermore, the voice level transformation could allow for increased control over the expressiveness of the result. Currently, only the original voice level from the diphone samples can be produced by the IRCAM Singing Synthesizer resulting in everything being sung in *mezzo forte*. While some expressiveness can be controlled using the signal gain alone, the possibilities in the current implementation are limited. Using the neural vocoder with a voice level control would extend the possibilities for expressiveness significantly.

### 9.3.5. Speech enhancement and humanization

Due to the high compression of the auto-encoder, information unrelated to voice is removed. If the auto-encoder has only been trained on clean speech, it is only able to synthesize clean speech. Thus, if during inference, background noise is present, it is suppressed by the auto-encoder. This works to a certain degree but fails completely if the background noise is too strong.

Similarly, if we provide something that isn't voice as the input to the auto-encoder, the auto-encoder has no means to encode it as something else and will conflate it with a voice recording. The result will resemble human voice much more than the input. Though this does not work consistently and reliably the auto-encoder has been observed to transform breakbeats into beatboxing and string sounds into humming.

## 9.4. Limitations and Future work

Possibilities for future work were already discussed in Sections 4.5, 5.5, 6.5 and 7.6 for the specific applications. In the greater scheme, there are a few topics that need to be addressed.

- The disentangled space should ensure that for reasonable input values the transformed result should still sound natural. In many cases, however, we

find that this is not the case. Improvement in audio quality is still necessary.

- While the neural voice transformation framework could be applied to all kinds of voice properties, in this thesis we have only discussed two properties, the fundamental frequency and the voice level. It would be desirable to add more transformable features.
- Although both auto-encoder and mel-inverter are lightweight enough to calculate the audio fast, real-time applications are limited by the non-causal nature of the system. To allow real-time applications with low latency, the architectures have to be adapted not to depend on future values.
- Finally, it would be interesting to understand the latent code better.

#### 9.4.1. Issues with synthesis quality

##### Variable information and bottleneck size

Our experiments have indicated that conditional entropy of the mel-spectrogram under the control parameter  $F$ , is not constant. In Section 6.3.1 we have observed that the required bottleneck size is quite different for singing voice (2–3) and for speech (5–8). Training a model that works well for both speech and singing was thus not feasible. Similarly, in Section 8.1.6 we have observed that the voiced component contains much less information for high pitches than low pitches. This could be explained by the spacing and number of harmonics as in effect the harmonics in the voice sample the vocal tract filter. For high-pitched voice the vocal tract filter is sampled only at few points and might miss significant information like narrow banded formants. This argument only holds for the deterministic component, while the inverse is true for the noise component: The further the harmonics are spaced, the more area is occupied by the noise component between the harmonics. In singing the latter information is far less relevant as, firstly, the most of the information of the noise is perceptually irrelevant and, secondly, the noise component itself is perceptually less relevant as it carries much less energy. In practice this means that for high enough fundamental frequency the information bottleneck is too large to remove the fundamental frequency from the latent code and the decoder will disregard the  $f_0$  input. To force disentanglement for all possible  $f_0$  values, the idea of an adaptive bottleneck size was touched upon in Section 8.1.6. The idea of a variable bottleneck size showed promising results in the case of disentangling  $f_0$  in the presence of a wide range of  $f_0$  values.

The bottleneck size can be varied using dropout on the latent code. By applying dropout with rate  $r \in [0, 1]$  to the latent code, the channel capacity of the residual code is multiplied by  $1 - r$ . The effective bottleneck size can thus be controlled by the dropout rate. By increasing the dropout rate with the  $f_0$ , we were able to achieve much better disentanglement. In preliminary experiments, disentanglement was achieved for the whole pitch range of human voice, including high pitches.

A rigorous evaluation of this method, however, remains due. In particular the optimal bottleneck size needs to be found depending on the fundamental frequency,

voice type (i.e. singing vs. speech) and other voice properties.

### Fundamental problems with the bottleneck approach

Even if the controllable property is disentangled properly, sometimes we observe substantial audio degradation. The degradation of the audio can be explained by a fundamental flaw of the information bottleneck approach: In theory the encoder should remove all the information related to the voice feature that we want to disentangle first, before it removes other information that is not redundant. This could be the case if the reconstruction loss is zero and the bottleneck size is exactly at the optimal size. In practice, however, since the information amount is variable, the optimal bottleneck size cannot be used, even with the aforementioned variable bottleneck size. Furthermore, we observe that in practice, essential information is being removed, before the latent space is completely disentangled. Thus perfect reconstruction of the original mel-spectrogram is not possible and the audio quality suffers especially after transformation.

The degradation of the audio quality manifests itself through two main symptoms:

- The phonetic content is often degraded. Some vowels are changed towards more common ones and some consonants, in particular plosives, are dropped.
- The noise component is over-smoothed rather than having the characteristic texture of noise. As a result the noise component sounds rather poor, especially for voice with high breathiness.

### Phonetic content degradation

In many cases we can observe degradation of the phonetic content. The vowels all become more similar to each other, consonants might be switched, and plosives might be dropped.

During training the network learns to reconstruct the mel-spectrogram, and it can be expected that phonetic content is preserved when the reconstruction error is sufficiently low. On the other hand, the network is not trained to apply transformations explicitly. Since the framework has no mechanism to preserve phonetic content, there is no guarantee that the latent code represents the same phonetic content for different values of the control parameter. The network might choose different coding strategies for different values of the control parameter, as we have observed with pitches above 700 Hz. As a result, changing the control parameter will change the context of the encoded phoneme and thus change its meaning.

Adding additional constraints might be necessary to ensure stability of phonetic content. This could be achieved by forcing consistency through a phoneme recognizer between original and transformed mel-spectrogram as proposed by Moine (2023, Section 7.2.2).

### Noise component

Using the variable bottleneck size, we can improve parts of the noise component by choosing a high dimensionality of the latent space and fully opening the bottleneck during unvoiced segments. Since no disentanglement of the fundamental frequency is necessary during unvoiced segments no bottleneck is required there. This way the unvoiced component can be reconstructed much more easily. The noise component within voiced segments is not affected by this approach. This approach works well for singing voice where the noisy component in the voiced segments is low and even for speech this approach provides an improvement to the overall quality. Particularly breathy voice still finds its limitation with this approach.

Thus, more sophisticated methods are required to force the decoder to produce the right noise texture. Noise is problematic in terms of information theory, as the amount of information is high, but most of the information is perceptually irrelevant. To reconstruct the noisy texture perfectly, many perceptually irrelevant details need to be preserved. When trying to reconstruct the noise component in the mel-spectrogram with insufficient information, the decoder cannot reproduce the noise component. When using any  $\mathcal{L}^P$  loss, the decoder can only approximate these areas with mean values. Over-smoothing is the result and no noisy texture is generated. The MBExWN, however, which is used to calculate audio signal from the mel-spectrogram, has never seen smooth textures like that which results in unpredictable behaviour. Two possible remedies could be followed. Either we force the decoder not to generate smooth surfaces but rather noisy textures, or we teach the mel-inverter to treat the smooth surfaces the same way as noisy textures. The question is thus, who should handle generating the noise, the decoder of the auto-encoder or the mel-inverter.

### Adversarial training

Over-smoothing of the noisy textures is a problem of the loss function since generating arbitrary textures would create a higher loss than creating mean values. In this sense the  $\mathcal{L}^P$  loss does not model the perception well, as noisy textures would be preferred over smooth textures. This could be fixed by using adversarial approaches instead. A typical approach of this would be to train a discriminator to distinguish between real- and synthetic mel-spectrograms. Since smooth textures are rare in real mel-spectrograms this could be a distinguishing feature for the discriminator and the decoder would be incentivized to avoid creating smooth surfaces (Bous, Benaroya, et al., 2022).

### Smoothed mel-spectrogram

For noise signals the perceptually relevant detail is the spectral density, that is, the expected value of the energy in a frequency band. The spectral density is mostly a smooth function with little information. Thus, one might argue that the spectral density might be a more suited parametric space for the neural vocoder as it removes

a lot of perceptually irrelevant information. In this context, the over-smoothing might actually not be such a big problem but rather a welcome side effect.

The big obstacle for this approach is to find the canonic representation. Since the noise component changes quickly, it is impossible to estimate the true spectral density. Simply smoothing the mel-spectrogram is not an option either since narrow resonances in the spectral envelope caused by the formants must be preserved. We only want to smooth the noise component. Therefore, an alternative method to approximate the spectral density is required. A possible approach could be to use a bottleneck auto-encoder without any conditional input. A large advantage of this approach in the context of the proposed neural voice transformation framework would be that we have an upper bound to the total information in the voice representation through the bottleneck size. The auto-encoder that generates the vocoder parameters could be trained jointly with the mel-inverter, further optimizing the parametric space.

### Fine-tuning one model to the other

Another similar approach could be to train the mel-inverter not on the real mel-spectrograms but rather on the synthetic, non-transformed mel-spectrograms. This way we could teach the mel-inversion to invent the missing information and correct possible errors. The biggest drawback of this approach is the loss of universality of the vocoder. For each transformation auto-encoder, a new mel-inverter would have to be trained. Thus, training the mel-inverter on the synthetic mel-spectrograms is impractical. More feasibly, the mel-inverter could be used to provide a loss to the auto-encoder. In this case, the auto-encoder and the mel-inverter would be used simultaneously when training the auto-encoder, but the weights of the mel-inverter would be frozen. This could force the auto-encoder to produce more realistic mel-spectrograms.

### Hybrid information bottleneck

The above ideas are remedies that address the one or the other symptom of the too large information reduction through the information bottleneck. None of these approaches, however, address the underlying issue with the information bottleneck, which is that the information bottleneck inevitably removes more information than it should.

This does not mean, however, that the proposed method of this thesis should not be used at all. The simplicity and the quick disentanglement make the bottleneck auto encoder a compelling and powerful tool. While the information bottleneck itself has its limitation, it can be combined with other disentanglement methods to benefit from multiple approaches. One alternative approach, for example, is the use of GAN. Using a discriminator, we could evaluate transformed voice during training. The problem with this approach is, that convergence is tricky. Using the variable bottleneck size the information bottleneck could be opened and closed whenever necessary. Thus, the information bottleneck could be used to quickly

achieve disentanglement. Then, the bottleneck could be slowly opened to allow more information to be exchanged between encoder and decoder, while a discriminator ensures that the latent space remains disentangled. These approaches could be further combined with content classifiers that ensure that the relevant content is well-preserved.

#### 9.4.2. Controllable voice properties

So far only two voice properties can be controlled, the fundamental frequency and the voice level. We want to be able to add transform more properties, but as with the voice level, many of these parameters are not easily available.

##### Breathiness

Breathiness is the ratio between sinusoidal voiced and noisy unvoiced component. Normally, while still an essential part of the voice, the noisy component is very low in voiced segments. However, due to various reasons, the noise component can become much more dominant, for example when speaking very quietly (in the extreme case for whispering). Thus, breathiness is highly correlated with voice level, though, they can be different: For example we can use varying voice level when whispering or speaking very breathy. Therefore, additional control of breathiness would be a desirable feature.

To control the breathiness in voice, the bottleneck auto-encoder should be conditioned on a breathiness parameter. Finding a useful parametrization of breathiness is key to a successful breathiness transformation. We could use some existing breathiness analysis from the literature. Most classical neural vocoders have some kind of parametrization of the noise. To achieve good breathiness transformation, we should investigate, which noise parametrization is best suited for this task.

The configuration of [Chapter 6](#) already allows modifications of breathiness through the voice-unvoiced mask, which is part of the  $f_0$  conditioning. Since a fundamental frequency is only defined in voiced segments, the conditional input of [Chapter 6](#) contains an additional binary value, the voiced-unvoiced mask, to indicate, whether the  $f_0$  input can be used or not.

During inference, this input can be used to modify the breathiness, by providing intermediate values between 0 and 1. Although the auto-encoder has never seen other values than 0 and 1 during training, it somewhat adds more noise in the voice segments when changing the voice-unvoiced mask from ‘voiced’ to ‘unvoiced’. The result, however, is not very realistic and needs substantial improvement. As previously observed, the auto-encoder generally has a problem with generating the noise component realistically. With the variable bottleneck size, this problem could be improved substantially in the unvoiced regions and in regular non-breathy voice, the poor quality of the noise component did not impact the overall quality much. When synthesizing breathy voice, however, this approach does not work any more. Thus, different methods to generate realistic noise are required for this application.

Furthermore, the approach of interpolating the voiced-unvoiced mask does not

provide very accurate control. The noise increases with increased unvoiced parameter, but it is impossible to predict, which value really should be used. Finally, the voiced-unvoiced mask is used to indicate whether the  $f_0$  input should be used or not. As a consequence, with increased unvoiced parameter the auto-encoder starts to ignore the  $f_0$  input and generates incorrect  $f_0$  contours.

Thus, while the fractional voiced-unvoiced approach allows some control over the breathiness of the voice, further research is required, to create a realistic breathiness modification. For that, a reliable parametrization of the breathiness is required.

### Voice identity

We established that in order to achieve extreme pitch transformation the voice identity needs to be adjusted. If the new pitch is out of the range of the original voice, the synthetic recording needs to borrow the timbre of another voice. Two different behaviours with regard to voice identity adjustment can be observed. Either the voice identity is preserved within its vocal range and only if the new pitch falls outside the original voice's range the voice identity changes rather abruptly. The other possibility is that the voice identity is gradually changed with increasing amount of transposition. Currently, both behaviours are possible, and the outcome is rather unpredictable. Thus, it is desirable to have more control about the implicit voice identity transformation.

Depending on the application, the voice identity should change or not:

- A common application of pitch transformation is in audio editing. In this application we have a recording of one singer and may want to change one or a few notes at the same time. Here, the changes in pitch should not cause a noticeable change in voice identity and thus, the voice identity must be preserved.
- Another application is to transform a recording into another domain. For example a piece could be recorded by one voice and then transformed entirely into a different vocal range. Here the voice identity needs to change.

The deep learning approach allows us to influence how the voice identity changes. It, thus, requires us to consider what kind of pitch transformation do we want: Do we want to have the same person sing the passage in a different pitch and accept the limitations of the original voice or do we want the voice identity (voice type) to change with the pitch, as if it was sung by a voice with a different range?

**Explicit voice identity conditioning** The obvious solution to control the voice identity would be to introduce a voice identity conditioning to the auto-encoder. This way we would quite explicitly enter the domain of voice identity transformation. While voice identity transformation is certainly an interesting research domain in itself, here, this approach has a few undesirable drawbacks: To condition on the voice identity, some additional information about the voice has to be provided. This can be done, either by including the voice in the training set or learning some kind of

space of voice identity and mapping the input voice into this space for conditioning. For our purposes, the former is not feasible, as we do not know in advance on which voice the transformation will be applied. Likewise, it cannot be expected from the user to fine tune the model for each voice they use. Thus, zero-shot methods (the latter), need to be employed if explicit voice identity conditioning is desired.

Another drawback of explicit conditioning is that we loose the implicitness of voice identity adjustment. If we want to transpose a recording into a new range the system should find the voice identity in the new range that is closest to the original voice. Forcing the user to choose a new voice identity themselves might not be desired in all applications.

**Conditioning on the position in the vocal range** Instead of conditioning on the whole voice identity, we could condition only on the characteristics that need to be changed. Here, what needs to be adjusted is only the vocal range of the voice. Thus conditioning on the vocal range alone should suffice to obtain sufficient control over the implicit voice identity change. Two possible ways to convey this information to the auto-encoder exist. The singer's or speaker's vocal range could be provided through some kind of parametrization, e.g. minimum and maximum pitch. Then the auto-encoder can infer where the note lies within the vocal range and synthesize the timbre accordingly. Similar drawbacks as with the explicit voice identity conditioning exist: This method requires the user to know the exact vocal range of the original singer.

Another possibility is to provide information about where the note should lie within the vocal range of the singer. We call this the *relative pitch*, as it is relative to the singers vocal range. This property could be estimated similar to the voice level estimation of [Chapter 7](#) from the voice timbre. Then both absolute pitch (i.e. the pitch in Hz) and relative pitch could be provided simultaneously to the auto-encoder. If we want to preserve the voice identity, the relative pitch needs to change together with absolute pitch. If we want to adjust the voice identity, the relative pitch needs to be preserved while the absolute pitch is changed. While the relative pitch only defines the voice identity implicitly, it describes the voice property that we want to control most explicitly.

**Relative pitch and voice identity** Absolute pitch, relative pitch and voice identity are pairwise independent but not mutually independent. Two properties can be provided while the third will be inferred. Which two properties are provided depends on the application. Above, we have discussed the pros and cons of accompanying absolute pitch with either voice identity or relative pitch. Furthermore, relative pitch and voice identity can be combined to infer the absolute pitch (Qian, Jin, et al., 2020). This is useful in speech, where the absolute pitch bears little significance and most information is contained in the pitch contour. Thus, we often find this parametrization in speaker identity conversion.

### 9.4.3. Understanding the latent space of the auto-encoder

We have studied the latent space of the auto-encoder in [Chapter 6](#) (notably [Section 6.3.3](#)), where we have examined how different vowels are encoded in the latent space of the auto-encoder. While this helped us to understand how the information is spread over time, still many open questions remain.

It would be interesting to investigate, how the remaining information is encoded in the latent space, in particular voice properties related to voice identity. Finding ways to manipulate the voice directly on the latent space would open numerous possibilities of artistic applications.

The variable bottleneck size approach, as discussed in [Section 9.4.1](#), forms a kind of [variational auto-encoder \(VAE\)](#) since the dropout transforms the individual points into a binary probability distribution. Knowledge about [VAEs](#) could be used to reason about the latent space of our auto-encoder. Further research could be conducted around using other [VAE](#) approaches as well to further stabilize the latent space in terms of continuity.

# List of figures

|       |  |     |
|-------|--|-----|
| 2.1.  | Electrical circuit of the first electrical speech synthesizer . . . . .    | 7   |
| 2.2.  | Cross-section of the auditory system . . . . .                             | 9   |
| 2.3.  | Cross-section of the cochlea . . . . .                                     | 10  |
| 2.4.  | Mel filter bank for a configuration with 20 mel-bands . . . . .            | 13  |
| 2.5.  | Equal loudness contours from ISO 226: 2003 standard . . . . .              | 15  |
| 2.6.  | Cross-section of the larynx . . . . .                                      | 16  |
| 2.7.  | Cross-section of the vocal tract . . . . .                                 | 17  |
| 2.8.  | Liljencrants-Fant glottal pulse and it's defining parameters . . . . .     | 19  |
| 2.9.  | Liljencrants-Fant glottal flow derivative for different $R_d$ values . . . | 22  |
| 2.10. | Deep dream transformed image of the Mona Lisa . . . . .                    | 27  |
| 2.11. | Pie charts showing the distribution of different attributes . . . . .      | 36  |
| 2.12. | Histogram of fundamental frequency . . . . .                               | 37  |
| 2.13. | Histogram of fundamental frequency in speech . . . . .                     | 38  |
| 2.14. | Histogram of fundamental frequency in singing . . . . .                    | 38  |
| 2.15. | Screenshot of a perceptive test website . . . . .                          | 43  |
| 4.1.  | Schematic of the analysis-synthesis framework . . . . .                    | 67  |
| 5.1.  | Schematic of the MBExWN . . . . .  | 86  |
| 6.1.  | Bottleneck auto-encoder schematic overview . . . . .                       | 97  |
| 6.2.  | Accuracy of $f_0$ transformation . . . . .                                 | 102 |
| 6.3.  | NMFE as a function of the bottleneck size . . . . .                        | 104 |
| 6.4.  | Mean opinion score plotted against transposition amount . . . . .          | 106 |
| 6.5.  | Distribution of vowels in the latent code space . . . . .                  | 108 |
| 6.6.  | Distribution of intensities in the latent code space . . . . .             | 109 |
| 6.7.  | Progression of the latent code for stable vowels . . . . .                 | 110 |
| 7.1.  | Histogram of voice level for different dynamics . . . . .                  | 124 |
| 7.2.  | Perceived change in voice level over applied voice level change . . .      | 126 |
| 7.3.  | Mean opinion score for transformed recordings . . . . .                    | 127 |
| 8.1.  | Screen shot of CIRCE . . . . .   | 133 |
| 8.2.  | The installation <i>an·other voice</i> by Judith Deschamps . . . . .       | 134 |
| 8.3.  | The computer running the improvisation of the Farinelli Voice . . .        | 136 |
| 8.4.  | Shadow play from <i>Né entre corps</i> . . . . .                           | 139 |
| 8.5.  | Scene from <i>Day 0 - trans-instrumentalism</i> . . . . .                  | 140 |
| 8.6.  | Anonymous text used for <i>G.N.Z.</i> . . . . .                            | 141 |

*List of figures*

---

|  |     |
|--|-----|
| 9.1. Full framework during inference . . . . . | 145 |
|--|-----|

# List of tables

|      |   |     |
|------|---|-----|
| 2.1. | Speech datasets used in this thesis. . . . .                        | 32  |
| 2.2. | Singing datasets used in this thesis. . . . .                       | 34  |
| 4.1. | Losses used for the analysis-synthesis framework . . . . .          | 67  |
| 4.2. | Results on CMU arctic with metrics considering all GCI. . . . .     | 71  |
| 5.1. | Mean opinion scores for MBExWN and other baselines on speech . .    | 89  |
| 5.2. | Mean opinion scores for MBExWN and other baselines on singing . .   | 89  |
| 6.1. | Final loss values for models of Chapter 6 . . . . .                 | 100 |
| 6.2. | Quality ratings of the algorithms for speech . . . . .              | 104 |
| 6.3. | Quality ratings of the algorithms for singing . . . . .             | 105 |
| 7.1. | Transformation precision of the auto-encoders . . . . .             | 125 |
| 7.2. | Perceived change in voice level over applied voice level change . . | 126 |
| 7.3. | Mean opinion score for transformed recordings . . . . .             | 127 |
| 8.1. | Phrases in the piece <i>Quell'usignolo</i> . . . . .                | 132 |
| 8.2. | Structure of the piece <i>Quell'usignolo</i> . . . . .              | 132 |
| A.1. | List of layers in the analysis module. . . . .                      | 160 |
| A.2. | List of layers in the synthesis module. . . . .                     | 160 |
| A.3. | List of layers in the encoder. . . . .                              | 161 |
| A.4. | List of layers in the decoder. . . . .                              | 161 |
| A.5. | List of layers of the voice level estimator. . . . .                | 162 |

## A. List of model configurations and their layers

This appendix section provides the architectures of the models used in this thesis. The tables follow the same nomenclature in addition to the legend below them:

### Layer types:

**BNorm** Batch normalization.

**ConCat** Concatenation of inputs in feature dimension.

**Conv<sub>1</sub>** 1-D convolutional layer.

**Conv<sub>2</sub>** 2-D convolutional layer.

**Conv<sub>2</sub><sup>T</sup>** Transposed 2-D convolutional layer.

**MaxPool** Local max pooling layer.

**WaveNet Stack** A stack of gated **WaveNet** layers with skip connections and exponentially increasing dilated convolutions as described by Oord, Dieleman, et al. (2016).

### Activation functions:

**ReLU** Rectangular Linear Unit  $(x + |x|)/2$ .

**sigmoid** Sigmoid function  $(1 + e^{-x})^{-1}$ .

When listing inputs, external inputs (that are not the output of previous layers) are written in *italics*. The dimensions of the output tensors (out. shape) are given as (time, feature) or (time, frequency, feature).

## A.1. Models from Chapter 4: Pulse detection

**Table A.1.:** List of layers in the analysis module.

| Layer  | Inputs | Type              | $n_h$ | $k$ | Act.    | Out. shape      |
|--------|--------|-------------------|-------|-----|---------|-----------------|
| conv1  | audio  | Conv <sub>1</sub> | 512   | 32  | ReLU    | ( $t$ , 512)    |
| pool1  | conv1  | MaxPool           | —     | 2   | —       | ( $t/2$ , 512)  |
| bn1    | pool1  | BNorm             | —     | —   | —       | ( $t/2$ , 512)  |
| conv2  | bn1    | Conv <sub>1</sub> | 64    | 32  | ReLU    | ( $t/2$ , 64)   |
| pool2  | conv2  | MaxPool           | —     | 2   | —       | ( $t/4$ , 64)   |
| bn2    | pool2  | BNorm             | —     | —   | —       | ( $t/4$ , 64)   |
| conv3  | bn2    | Conv <sub>1</sub> | 64    | 32  | ReLU    | ( $t/4$ , 64)   |
| pool3  | conv3  | MaxPool           | —     | 2   | —       | ( $t/8$ , 64)   |
| bn3    | pool3  | BNorm             | —     | —   | —       | ( $t/8$ , 64)   |
| conv4  | bn3    | Conv <sub>1</sub> | 256   | 32  | ReLU    | ( $t/8$ , 256)  |
| bn4    | conv4  | BNorm             | —     | —   | —       | ( $t/8$ , 256)  |
| conv5  | bn4    | Conv <sub>1</sub> | 512   | 32  | ReLU    | ( $t/8$ , 512)  |
| bn5    | conv5  | BNorm             | —     | —   | —       | ( $t/8$ , 512)  |
| conv6  | bn5    | Conv <sub>1</sub> | 1024  | 32  | ReLU    | ( $t/8$ , 1024) |
| bn6    | conv6  | BNorm             | —     | —   | —       | ( $t/8$ , 1024) |
| output | bn6    | Conv <sub>1</sub> | 1     | 4   | sigmoid | ( $t/8$ , 1024) |

$n_h$  Number of filters in convolution

$k$  Kernel size of convolution

audio Raw-audio input

**Table A.2.:** List of layers in the synthesis module.

| Layer  | Inputs                   | Type          | $n_h$ | $k$ | $n_l$ |
|--------|--------------------------|---------------|-------|-----|-------|
| stack1 | pulses, noise, env, envn | WaveNet Stack | 256   | 2   | 6     |
| stack2 | stack1, env, envn        | WaveNet Stack | 256   | 2   | 6     |

$n_h$  Number of hidden units in stack

$k$  Kernel size of dilated convolutions

$n_l$  Number of layer in stack

pulses Glottal pulse signal as provided by analysis module

noise White noise

env Spectral envelope of the audio signal

envn Spectral envelope of the noise component

## A.2. Models from Chapter 6: Bottleneck auto encoder

**Table A.3.: List of layers in the encoder.**

| Layer  | Inputs      | Type              | $n_h$ | $k$     | $s$     | Act. | Out. shape       |
|--------|-------------|-------------------|-------|---------|---------|------|------------------|
| f0in   | $f_0, vum$  | $\text{Conv}_2^T$ | 2     | (1, 80) | (1, 80) |      | ( $t$ , 80, 2)   |
| concat | $mel, f0in$ | ConCat            |       |         |         |      | ( $t$ , 80, 3)   |
| hid1   | concat      | $\text{Conv}_2$   | 512   | (1, 2)  | (1, 2)  | ReLU | ( $t$ , 40, 512) |
| hid2   | hid1        | $\text{Conv}_2$   | 512   | (3, 3)  |         | ReLU | ( $t$ , 40, 512) |
| hid3   | hid2        | $\text{Conv}_2$   | 512   | (1, 2)  | (1, 2)  | ReLU | ( $t$ , 20, 512) |
| hid4   | hid3        | $\text{Conv}_2$   | 512   | (3, 3)  |         | ReLU | ( $t$ , 20, 512) |
| hid5   | hid4        | $\text{Conv}_2$   | 512   | (1, 2)  | (1, 2)  | ReLU | ( $t$ , 10, 512) |
| hid6   | hid5        | $\text{Conv}_2$   | 512   | (3, 3)  |         | ReLU | ( $t$ , 10, 512) |
| hid7   | hid6        | $\text{Conv}_2$   | 512   | (1, 2)  | (1, 2)  | ReLU | ( $t$ , 5, 512)  |
| hid8   | hid7        | $\text{Conv}_2$   | 512   | (3, 3)  |         | ReLU | ( $t$ , 5, 512)  |
| hid9   | hid8        | $\text{Conv}_2$   | 512   | (1, 5)  | (1, 5)  | ReLU | ( $t$ , 1, 512)  |
| out    | hid9        | $\text{Conv}_2$   | $n_b$ | (3, 1)  |         |      | ( $t$ , 1, 512)  |

**Table A.4.: List of layers in the decoder.**

| Layer  | Inputs      | Type              | $n_h$ | $k$    | $s$    | Act. | Out. shape             |
|--------|-------------|-------------------|-------|--------|--------|------|------------------------|
| concat | $code, f_0$ | ConCat            |       |        |        |      | ( $t$ , 1, $n_b + 2$ ) |
| hid0   | concat      | $\text{Conv}_2$   | 512   | (3, 3) |        | ReLU | ( $t$ , 1, 512)        |
| hid1   | hid0        | $\text{Conv}_2^T$ | 512   | (1, 5) | (1, 5) | ReLU | ( $t$ , 5, 512)        |
| hid2   | hid1        | $\text{Conv}_2$   | 512   | (3, 3) |        | ReLU | ( $t$ , 5, 512)        |
| hid3   | hid2        | $\text{Conv}_2^T$ | 512   | (1, 2) | (1, 2) | ReLU | ( $t$ , 10, 512)       |
| hid4   | hid3        | $\text{Conv}_2$   | 512   | (3, 3) |        | ReLU | ( $t$ , 10, 512)       |
| hid5   | hid4        | $\text{Conv}_2^T$ | 512   | (1, 2) | (1, 2) | ReLU | ( $t$ , 20, 512)       |
| hid6   | hid5        | $\text{Conv}_2$   | 512   | (3, 3) |        | ReLU | ( $t$ , 20, 512)       |
| hid7   | hid6        | $\text{Conv}_2^T$ | 512   | (1, 2) | (1, 2) | ReLU | ( $t$ , 40, 512)       |
| hid8   | hid7        | $\text{Conv}_2$   | 512   | (3, 3) |        | ReLU | ( $t$ , 40, 512)       |
| hid9   | hid8        | $\text{Conv}_2^T$ | 512   | (1, 5) | (1, 5) | ReLU | ( $t$ , 80, 512)       |
| out    | hid9        | $\text{Conv}_2$   | 1     | (3, 3) |        |      | ( $t$ , 80, 512)       |

$n_h$  Number of filters in convolution

$n_b$  Bottleneck size

$k$  Kernel size of convolution

$s$  Stride

$\text{audio}$  Raw-audio input

$f_0$   $f_0$  input

$vum$  Voiced-unvoiced input

$mel$  Mel-spectrogram input

$code$  Latent code input

### A.3. Models from Chapter 7: Voice level transformation

**Table A.5.: List of layers of the voice level estimator.**

| Layer | Inputs     | Type              | $n_h$ | $k$ | Act. | Out. shape   |
|-------|------------|-------------------|-------|-----|------|--------------|
| hid1  | <i>mel</i> | Conv <sub>1</sub> | 80    | 3   | ReLU | ( $t$ , 80)  |
| hid2  | hid1       | Conv <sub>1</sub> | 100   | 3   | ReLU | ( $t$ , 100) |
| hid3  | hid2       | Conv <sub>1</sub> | 100   | 1   | ReLU | ( $t$ , 100) |
| hid4  | hid3       | Conv <sub>1</sub> | 100   | 1   | ReLU | ( $t$ , 100) |
| hid5  | hid4       | Conv <sub>1</sub> | 100   | 1   | ReLU | ( $t$ , 100) |
| hid6  | hid5       | Conv <sub>1</sub> | 100   | 1   | ReLU | ( $t$ , 100) |
| hid7  | hid6       | Conv <sub>1</sub> | 100   | 1   | ReLU | ( $t$ , 100) |
| hid8  | hid7       | Conv <sub>1</sub> | 100   | 1   | ReLU | ( $t$ , 100) |
| hid9  | hid8       | Conv <sub>1</sub> | 50    | 1   | ReLU | ( $t$ , 50)  |
| out   | hid9       | Conv <sub>1</sub> | 1     | 1   |      | ( $t$ , 1)   |

$n_h$  Number of filters in convolution

$k$  Kernel size of convolution

*mel* Mel-spectrogram of analysed audio

# Glossary

## A

|               |  |
|---------------|--|
| <b>ACR</b>    | <b>Absolute Category Rating</b>  |
|               | A perceptive test, where ratings are provided for each sample individually on an absolute scale. <a href="#">40</a> , <a href="#">42</a> , <a href="#">167</a>   |
| <b>ANR</b>    | <b>Agence Nationale de la Recherche</b>  |
|               | The French national research agency. <a href="#">2</a> , <a href="#">3</a> , <a href="#">163</a>   |
| <b>ARS</b>    | <b>Analyse et tRansformation de Style du chant</b><br>(Analysis and tRansformation of singing Style)   |
|               | Research project of the <b>ANR</b> , (ANR-19-CE38-0001-13). This thesis was in part funded by this research project. <a href="#">2</a> , <a href="#">3</a> , <a href="#">62</a>  |
| <b>AutoVC</b> |  |
|               | The original bottleneck auto-encoder by Qian, Yang Zhang, Chang, X. Yang, et al. (2019). Used for voice identity conversion. <a href="#">29</a> , <a href="#">59</a> , <a href="#">95</a> , <a href="#">98</a> , <a href="#">113</a> |

## B

|                                |  |
|--------------------------------|--|
| <b>basilar membrane</b>        |  |
|                                | The part of the inner ear, where the incoming sound is decomposed into frequencies and translated into nervous signals. <a href="#">10</a> , <a href="#">12</a> , <a href="#">14</a> , <a href="#">164</a>   |
| <b>bottleneck auto-encoder</b> |  |
|                                | Auto-encoder with information bottleneck. The information bottleneck ensures disentanglement of the latent code and an additional control parameter. This is the main technology used for disentanglement in this thesis. <a href="#">3–5</a> , <a href="#">29</a> , <a href="#">59</a> , <a href="#">91</a> , <a href="#">95–98</a> , <a href="#">100</a> , <a href="#">103</a> , <a href="#">108</a> , <a href="#">111</a> , <a href="#">140</a> , <a href="#">143</a> , <a href="#">144</a> , <a href="#">148</a> , <a href="#">149</a> , <a href="#">151</a> , <a href="#">152</a> , <a href="#">167</a> |

## C

|              |  |
|--------------|--|
| <b>CCR</b>   | <b>Comparison Category Rating</b>  |
|              | A perceptive test, where pairs of samples are compared against each other. <a href="#">40–42</a>   |
| <b>CIRCE</b> | <b>Circe, the IRCAM voice Encoder</b>  |
|              | A software providing a graphical user interface to the neural voice transformation framework. CIRCE has been developed during this thesis and is available for free on the <b>IRCAM</b> forum at <a href="https://forum.ircam.fr/projects/detail/circe/">https://forum.ircam.fr/projects/detail/circe/</a> . <a href="#">3</a> , <a href="#">133</a> , <a href="#">138</a> , <a href="#">140</a> , <a href="#">142</a> |

|  |  |
|--|--|
| <b>CNN</b>   | <b>Convolutional Neural Network</b>  |
| A neural network architecture, where convolutions are used as linear operations (LeCun et al., 1989). <a href="#">23–26</a> , <a href="#">29</a> , <a href="#">49</a> , <a href="#">54</a> , <a href="#">66</a> , <a href="#">86</a> , <a href="#">97</a> , <a href="#">98</a> , <a href="#">101</a> , <a href="#">103</a> , <a href="#">122</a> , <a href="#">159</a> |  |
| <b>CNRS</b>  | <b>Centre National de la Recherche Scientifique</b><br>(French National Centre for Scientific Research)  |
| French research organization involved in the STMS research lab.  | <a href="#">169</a>  |
| <b>concatenative synthesis</b>   | Synthesis method where elementary units of voice are concatenated to synthesize a whole phrase. <a href="#">3</a> , <a href="#">8</a> , <a href="#">38</a> , <a href="#">39</a> , <a href="#">55</a> , <a href="#">56</a> , <a href="#">92</a> , <a href="#">165</a> |
| <b>D</b>   |  |
| <b>DCR</b>   | <b>Degradation Category Rating</b>   |
| A perceptive test, where degradation is rated.   | <a href="#">40</a>   |
| <b>DDSP</b>  | <b>Differentiable Digital Signal Processing</b>  |
| A framework of neural network components based on digital signal processing, through which gradients can be transported. Developed by Engel et al. (2019).   | <a href="#">23</a> , <a href="#">30</a>  |
| <b>DSP</b>   | <b>Digital Signal Processing</b>   |
| Signal processing implemented in the computer instead of analogue electrical circuits.   | <a href="#">6</a> , <a href="#">9</a> , <a href="#">23</a> , <a href="#">30</a> , <a href="#">46</a> , <a href="#">51</a> , <a href="#">53</a> , <a href="#">164</a>   |
| <b>E</b>   |  |
| <b>EGG</b>   | <b>ElectroGlottograph</b>  |
| A measurement of conductance through the glottis, used to estimate the contact surface between the vocal folds and thus infer vocal fold activity like GCI.  | <a href="#">32</a> , <a href="#">48</a> , <a href="#">49</a> , <a href="#">65</a>  |
| <b>ERB</b>   | <b>Equivalent Rectangular Bandwidth</b>  |
| A pitch scale, modelled after the bandwidth of the filter characteristic on the basilar membrane.  | <a href="#">12</a> , <a href="#">14</a> , <a href="#">121</a>  |
| <b>EUSIPCO</b>   | <b>EUropean Slgnal Processing COnference</b>   |
| A scientific conference specialized in topics around signal processing. The paper on the analysis-synthesis framework for GCI detection from Chapter 4 was published in the conference proceedings of EUSIPCO.   | <a href="#">63</a>   |
| <b>F</b>   |  |
| <b>fft</b>   | <b>fast fourier transform</b>  |
| A computationally efficient algorithm to compute the discrete Fourier transform.   | <a href="#">121</a>  |

**F**

**Frequency Modulation**

A synthesis technique, where the frequency of a carrier is modulated rapidly to create a complex timbre (Chowning, 1989). 8

**G**

**GAN**

**Generative Adversarial Networks**

A set of two neural networks, a Generator and a Discriminator. The Generator produces samples that resemble samples from a training set. The Discriminator distinguishes between real (from the training set) and fake (produced by the generator) samples. Generator and Discriminator are trained adversarially in the sense that a low loss for the Generator results in a high loss for the Discriminator and vice versa. 28, 49, 54, 61, 151, 167

**GCI**

**Glottal Closure Instant**

The time instant where the vocal folds start to close. In the glottal flow derivative, this can be seen as a sharp negative peak (see Fig. 2.8 and Section 2.3.2). 3–5, 18, 22, 48, 49, 64–66, 68, 69, 71, 72, 143, 164

**GPU**

**Graphical Processing Unit**

Our main computational device for neural networks. 28, 54

**H**

**HMM**

**Hidden Markov Model**

Statistical model, where observations are assumed to be generated by a hidden (non-observable) Markov process. 8, 56

**HNM**

**Harmonic plus Noise Model**

A pitch-synchronous parametric vocoder. The each analysis frame is modelled as a sum of harmonically coupled sinusoids (for the voiced component) and noise. 8, 52, 92

**I**

**ICASSP**

**International Conference on Acoustics, Speech and Signal Processing**

A scientific conference specialized in topics around signal processing. The paper on estimation and transformation of voice level from Chapter 7 was published in the conference proceedings of ICASSP. 113

**IRCAM**

*Institut de Recherche et Coordination Acoustique/Musique*

(Institute of Research and Coordination of Acoustics and Music)

An institution for pairing scientific research with artistic musical creation. Scientific research at IRCAM is carried out to enable its application in musical compositions. 1, 2, 8, 34, 129–131, 163, 169

**ISIS**

**IRCAM Singing Synthesizer**

A parametric concatenative singing synthesizer based on the PaN vocoder. Developed by Ardaillon (2017). 2, 3, 33, 34, 91, 146, 147

|                        |  |
|------------------------|--|
| <b>ITU</b>             | <b>International Telecommunication Union</b>   |
|                        | United nations agency responsible for organizing international practices in telecommunication. The ITU provides recommendations and standards regarding practices around topics of telecommunication. Here we use their recommendations on perceptive evaluations. <a href="#">39</a> , <a href="#">41</a> , <a href="#">52</a>  |
| <b>K</b>               |  |
| <b>KTH</b>             | <b>Kungliga Tekniska Högskolan</b><br>(Royal Institute of Technology)  |
|                        | A research institution of importance for research on the fundamentals of voice. Early speech and singing synthesizers were developed at KTH, like the OVE and MUSSE. <a href="#">7</a> , <a href="#">167</a>   |
| <b>L</b>               |  |
| <b>LF</b>              | <b>Liljencrants-Fant</b>   |
|                        | Glottal pulse model that describes the glottal pulse signal either with four (Fant, Liljencrants, et al., 1985) or one (Fant, 1995) parameter. Discussed in Section 2.3.2. <a href="#">8</a> , <a href="#">18–22</a> , <a href="#">49</a> , <a href="#">65</a> , <a href="#">69</a> , <a href="#">71</a> , <a href="#">168</a>   |
| <b>LibriTTS</b>        |  |
|                        | Large speech dataset based on the LibriVox audio book corpus. Used mainly for text-to-speech applications. <a href="#">55</a>  |
| <b>LPC</b>             | <b>Linear Predictive Coding</b>  |
|                        | Reparametrization of a signal, where the next sample is predicted from a linear combination of the $n$ previous samples. This results in a signal parametrization consisting of an auto-recursive prediction filter and a residual signal. <a href="#">8</a> , <a href="#">49–51</a> , <a href="#">53</a> , <a href="#">55</a>   |
| <b>M</b>               |  |
| <b>MBExWN</b>          | <b>Multi-Band Excited WaveNet</b>  |
|                        | The neural vocoder that is used to transform the mel-spectrogram into raw audio. <a href="#">4</a> , <a href="#">85–90</a> , <a href="#">93</a> , <a href="#">96</a> , <a href="#">126</a> , <a href="#">144</a> , <a href="#">145</a> , <a href="#">150</a>   |
| <b>MDPI</b>            | <b>Multidisciplinary Digital Publishing Institute</b>  |
|                        | A publisher for scientific research articles. <a href="#">92</a>   |
| <b>mel-scale</b>       |  |
|                        | A pitch scale modelled after the perception of intervals (see Section 2.2.2). <a href="#">12–14</a> , <a href="#">81</a>   |
| <b>mel-spectrogram</b> |  |
|                        | An amplitude spectrogram where the frequency axis is wrapped to the mel-scale. <a href="#">12</a> , <a href="#">13</a> , <a href="#">25</a> , <a href="#">28</a> , <a href="#">48</a> , <a href="#">54–57</a> , <a href="#">60–62</a> , <a href="#">73</a> , <a href="#">77</a> , <a href="#">80–88</a> , <a href="#">90</a> , <a href="#">91</a> , <a href="#">93</a> , <a href="#">96–98</a> , <a href="#">101</a> , <a href="#">105</a> , <a href="#">111</a> , <a href="#">120</a> , <a href="#">122</a> , <a href="#">123</a> , <a href="#">125</a> , <a href="#">127</a> , <a href="#">133</a> , <a href="#">143–145</a> , <a href="#">147</a> , <a href="#">149–151</a> , <a href="#">161</a> , <a href="#">166</a> , <a href="#">167</a> , <a href="#">169</a> |

**MelGAN**

An architecture of generative adversarial networks for mel-spectrogram inversion. First introduced by K. Kumar et al. (2019). 28, 54, 73, 83, 84, 88, 89

**MFCC**

**Mel-Filtered Cepstral Coefficient**

Signal features obtained by Fourier transforming the log-amplitude mel-spectrogram. 12

**MOS**

**Mean Opinion Score**

A quality rating obtained through a perceptive test by asking participants to rate a recording and averaging their responses. 40, 44, 89, 106, 107

**MUSHRA**

**MUlti Stimulus test with Hidden Reference and Anchor**

A specific kind of Absolute Category Rating perceptive test. 41, 42

**MUSSE**

**Music and Singing Synthesis Equipment**

An early singing synthesizer developed at KTH. 7, 8, 166

**N**

**neural vocoder**

Parametric vocoder implemented with neural networks. The use of neural networks allows usage of unorthodox parametric spaces. 3–5, 51, 60–64, 73, 75–80, 83, 88, 90, 91, 96, 143, 152, 167

**neural voice transformation framework**

A framework to transform voice with neural networks consisting of a neural vocoder using the mel-spectrogram as its parametric space and the bottleneck auto-encoder, which does the actual transformation. The neural voice transformation framework is the central piece of this thesis. 1, 2, 95, 133, 138–140, 142–145, 147, 148, 151, 163

**NMFE**

**Normalized Mean F<sub>0</sub> Error**

A means of analysing disentanglement by measuring how well the external control parameter is used. 103, 104

**O**

**OQ**

**Open Quotient**

The ratio between open phase  $t_e$  and pulse duration  $t_0$  (see Section 2.3.2). 18, 20

**OVE**

**Orator Vox Electrica**

An early speech synthesizer developed at KTH. 7, 166

## P

**PaN****Pulse and Noise**

A classical vocoder that synthesises the voice using the LF glottal pulse model of and Noise. Developed by Ardaillon (2017). [49](#), [53](#), [57](#), [58](#), [63–65](#), [68](#), [69](#), [79](#), [100](#), [105](#), [107](#), [111](#), [147](#), [165](#)

**parametric vocoder**

Vocoder that maps between raw-audio and a parametric space. [3](#), [8](#), [50](#), [51](#), [55](#), [56](#), [58](#), [63](#), [64](#), [75](#), [78](#), [87](#), [92](#), [93](#), [100](#), [147](#)

**PC** Quotes stemming from personal correspondence with the author or composer discussing their piece. [138](#), [139](#), [141](#), [142](#)**phase vocoder**

Non-parametric vocoder that allows signal transformation on the STFT. [8](#), [51–53](#), [92](#)

**phon**

Scale of perceived loudness measured in dB above hearing threshold. [14](#), [15](#)

**PQMF****Pseudo Quadrature Mirror Filter bank**

A set of complementary filters that splits the signal into equally large frequency bands. [26](#), [85](#), [86](#)

**PSOLA****Pitch Synchronous OverLap and Add**

A pitch-synchronous non-parametric vocoder. Short windows are placed around each period of voiced speech. [8](#), [51–53](#), [92](#)

## R

**ReLU****Rectangular Linear Unit**

An activation function that sets the negative components to zero:  $\text{ReLU}(x) = (x + |x|)/2$ . (Nair and Hinton, 2010). [98](#), [159](#)

**RNN****Recurrent Neural Network**

A neural network type, where the output of previous samples is used as additional input. [24](#), [98](#)

## S

**sone**

Scale of perceived loudness that reflects perceived increases. Perceived loudness and thus the sone scale increases with the power of 0.3 of the signal energy. [15](#), [121](#)

**SPL****Sound Pressure Level**

A measure of sound level by measuring the sound pressure at a specific point in space. [114](#)

|                 |  |
|-----------------|--|
| <b>SPSS</b>     | <b>Statistical Parametric Speech Synthesis</b>   |
|                 | Speech (or singing) synthesis method, where vocoder parameters are generated using a statistical model. <a href="#">8</a> , <a href="#">56</a>   |
| <b>STFT</b>     | <b>Short Time Fourier Transform</b>  |
|                 | A spectrogram obtained by splitting the signal in small overlapping frames and applying the discrete Fourier transform to each frame. <a href="#">13</a> , <a href="#">26</a> , <a href="#">27</a> , <a href="#">67</a> , <a href="#">168</a>  |
| <b>STMS</b>     | <b><i>Sciences et Technologies de la Musique et du Son</i></b><br>(Sciences and Technologies of Music and Sound)   |
|                 | Joint research unit of <a href="#">IRCAM</a> , Sorbonne Université, CNRS and the French Ministry of Culture. <a href="#">3</a> , <a href="#">164</a>   |
| <b>T</b>        |  |
| <b>Tacotron</b> | One of the first neural text-to-speech engines using the mel-spectrogram. Developed by Y. Wang et al. (2017) (V1) and J. Shen et al. (2018) (V2). <a href="#">54</a>   |
| <b>TrueEnv</b>  | A spectral envelope estimation algorithm (Roebel and Rodet, 2005). <a href="#">50</a>  |
| <b>TTS</b>      | <b>Text-To-Speech</b>  |
|                 | Speech synthesis, where text is used as input and a voice signal reading the text is produced. <a href="#">54–56</a> , <a href="#">58</a> , <a href="#">140</a> , <a href="#">169</a>  |
| <b>V</b>        |  |
| <b>VAE</b>      | <b>Variational Auto-Encoder</b>  |
|                 | An auto-encoder, where the latent space consists of parametric probability distributions, rather than individual points. <a href="#">57</a> , <a href="#">59</a> , <a href="#">155</a>   |
| <b>VTF</b>      | <b>Vocal Tract Filter</b>  |
|                 | A filter that models all the effects that the vocal tract has on the glottal source signal. <a href="#">16–18</a> , <a href="#">50</a> , <a href="#">57</a> , <a href="#">58</a>   |
| <b>W</b>        |  |
| <b>WaveGlow</b> | A neural network applying generative flow to audio synthesis. First introduced by Prenger et al. (2019). <a href="#">54</a> , <a href="#">83</a> , <a href="#">84</a>  |
| <b>WaveNet</b>  | A neural network architecture type designed for audio synthesis. First introduced by Oord, Dieleman, et al. (2016). <a href="#">53</a> , <a href="#">54</a> , <a href="#">66</a> , <a href="#">77</a> , <a href="#">83</a> , <a href="#">84</a> , <a href="#">86</a> , <a href="#">159</a> |

# Bibliography

- Alessandro, Christophe d' and Boris Doval (1998). 'Experiments in voice quality modification of natural speech signals: the spectral approach'. In: *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis* (cit. on p. 58).
- (2003). 'Voice quality modification for emotional speech synthesis'. In: *Eighth European Conference on Speech Communication and Technology* (cit. on p. 58).
- Alessandro, Christophe d', Bayya Yegnanarayana, and Vassilios Darsinos (1995). 'Decomposition of speech signals into deterministic and stochastic components'. In: *1995 International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE, pp. 760–763 (cit. on p. 51).
- Alku, Paavo (2011). 'Glottal inverse filtering analysis of human voice production—A review of estimation and parameterization methods of the glottal excitation and their applications'. In: *Sadhana* 36.5, pp. 623–650 (cit. on p. 49).
- Anand, J. M., S. Guruprasad, and Bayya Yegnanarayana (2006). 'Extracting formants from short segments of speech using group delay functions'. In: *7th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1009–1012. isbn: 9781604234497 (cit. on pp. 49, 50).
- Ardaillon, Luc (2017). 'Synthesis and expressive transformation of singing voice'. PhD thesis. Université Pierre et Marie Curie. url: <https://hal.archives-ouvertes.fr/tel-01710926/document> (cit. on pp. 8, 33, 34, 49, 53, 56, 57, 63–65, 79, 100, 111, 123, 147, 165, 168).
- Ardaillon, Luc, Celine Chabot-Canet, and Axel Roebel (2016). 'Expressive control of singing voice synthesis using musical contexts and a parametric f0 model'. In: *17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on p. 92).
- Ardaillon, Luc, Gilles Degottex, and Axel Roebel (2015). 'A multi-layer F0 model for singing voice synthesis using a B-spline representation with intuitive controls'. In: *16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on p. 92).
- Ardaillon, Luc and Axel Roebel (2017). 'A mouth opening effect based on pole modification for expressive singing voice transformation'. In: *18th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1288–1292 (cit. on p. 58).
- (2019). 'Fully-convolutional network for pitch estimation of speech signals'. In: *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on pp. 37, 48, 49, 64, 65, 97, 122).
- (2020). 'GCI detection from raw speech using a fully-convolutional network'. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 4, 49, 63–66, 68–72).

- Arias, Pablo et al. (2021). ‘Beyond correlation: acoustic transformation methods for the experimental study of emotional voice and speech’. In: *Emotion Review* 13.1, pp. 12–24 (cit. on p. 93).
- Arik, Sercan, Jitong Chen, et al. (2018). ‘Neural voice cloning with a few samples’. In: *Advances in neural information processing systems* 31 (cit. on p. 77).
- Arik, Sercan, Gregory Diamos, et al. (2017). ‘Deep voice 2: Multi-speaker neural text-to-speech’. In: *arXiv preprint arXiv:1705.08947* (cit. on pp. 77, 80, 82, 83).
- Atal, Bishnu S and Suzanne L Hanauer (1971). ‘Speech analysis and synthesis by linear prediction of the speech wave’. In: *The journal of the acoustical society of America* 50.2B, pp. 637–655 (cit. on pp. 8, 50, 53).
- Babacan, Onur et al. (2013a). ‘A comparative study of pitch extraction algorithms on a large variety of singing sounds’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on p. 48).
- (2013b). ‘A quantitative comparison of glottal closure instant estimation algorithms on a large variety of singing sounds’. In: *14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1702–1706 (cit. on p. 49).
- Baevski, Alexei et al. (2020). ‘wav2vec 2.0: A framework for self-supervised learning of speech representations’. In: *Advances in neural information processing systems* 33, pp. 12449–12460 (cit. on p. 53).
- Barash, Omer (2023). G.N.Z. (Cit. on p. 129).
- Benaroya, Laurent, Nicolas Obin, and Axel Roebel (2023). ‘Manipulating Voice Attributes by Adversarial Learning of Structured Disentangled Representations’. In: *Entropy* 25.2, p. 375 (cit. on pp. 59, 83, 84).
- Bengio, Emmanuel et al. (2017). ‘Independently controllable features’. In: *arXiv preprint arXiv:1703.07718* (cit. on p. 58).
- Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). ‘Representation learning: A review and new perspectives’. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1798–1828 (cit. on p. 58).
- Blaauw, Merlijn (2022). ‘Modeling timbre for neural singing synthesis: methods for data-efficient, reduced effort voice creation, and fast and stable inference’. PhD thesis. Universitat Pompeu Fabra (cit. on p. 6).
- Blaauw, Merlijn and Jordi Bonada (2017). ‘A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs’. In: *Applied Sciences* 7.12, p. 1313 (cit. on p. 56).
- Bonada, Jordi, Martí Umbert, and Merlijn Blaauw (2016). ‘Expressive singing synthesis based on unit selection for the singing synthesis challenge 2016’. In: *17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on pp. 8, 56, 92).
- Bous, Frederik (2019). ‘Generating Spectral Envelopes for Singing Synthesis Using Neural Networks’. MA thesis. Technische Universität Darmstadt (cit. on p. 1).
- Bous, Frederik, Luc Ardaillon, and Axel Roebel (2020). ‘Semi-supervised learning of glottal pulse positions in a neural analysis-synthesis framework’. In: *European Signal Processing Conference (EUSIPCO)*. IEEE (cit. on pp. 4, 63).

- Bous, Frederik, Laurent Benaroya, et al. (2022). ‘Voice Reenactment with F0 and timing constraints and adversarial learning of conversions’. In: *European Signal Processing Conference (EUSIPCO)*. IEEE (cit. on pp. 28, 60, 131, 150).
- Bous, Frederik and Axel Roebel (2019). ‘Analysing deep learning-spectral envelope prediction methods for singing synthesis’. In: *European Signal Processing Conference (EUSIPCO)*. IEEE (cit. on p. 97).
- (2022). ‘A Bottleneck Auto-Encoder for F0 Transformations on Speech and Singing Voice’. In: *Information* 13.3, p. 102 (cit. on pp. 4, 92).
  - (2023). ‘Analysis and transformations of intensity in singing voice’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 4, 113).
- Broschi, Carlo (). *Sammlung von Arien für Gesang mit Instrumentalbegleitung*. url: [https://digital.onb.ac.at/RepViewer/viewer.faces?doc=DTL\\_4711034](https://digital.onb.ac.at/RepViewer/viewer.faces?doc=DTL_4711034) (cit. on p. 131).
- Camacho, Arturo (2007). ‘SWIPE: A sawtooth waveform inspired pitch estimator for speech and music’. PhD thesis. University of Florida. url: <http://www.kerwa.ucr.ac.cr:8080/handle/10669/536> (cit. on pp. 11, 48).
- Caracalla, Hugo and Axel Roebel (2017). ‘Gradient conversion between time and frequency domains using wirtinger calculus’. In: *Digital Audio Effects (DAFx)*, pp. 234–238 (cit. on p. 26).
- (2020). ‘Sound texture synthesis using RI spectrograms’. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 416–420 (cit. on p. 27).
- Carbonneau, Marc-André et al. (2022). ‘Measuring disentanglement: A review of metrics’. In: *IEEE Transactions on Neural Networks and Learning Systems* (cit. on p. 59).
- Chen, Fangxin et al. (2004). ‘Acoustic analysis of friendly speech’. In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE, pp. I–569 (cit. on p. 61).
- Chen, Ricky TQ et al. (2018). ‘Isolating sources of disentanglement in variational autoencoders’. In: *Advances in neural information processing systems* 31 (cit. on p. 59).
- Cheveigné, Alain de (Jan. 2010). ‘Pitch perception’. In: *Oxford Handbook of Auditory Science: Hearing*. Oxford University Press. isbn: 9780199233557. doi: [10.1093/oxfordhb/9780199233557.013.0004](https://doi.org/10.1093/oxfordhb/9780199233557.013.0004) (cit. on p. 11).
- Cheveigné, Alain de and Hideki Kawahara (2002). ‘YIN, a fundamental frequency estimator for speech and music’. In: *The Journal of the Acoustical Society of America* 111.4, pp. 1917–1930 (cit. on p. 48).
- Choi, Hyeong-Seok, Juheon Lee, et al. (2021). ‘Neural analysis and synthesis: Reconstructing speech from self-supervised representations’. In: *Advances in Neural Information Processing Systems* 34, pp. 16251–16265 (cit. on p. 54).
- Choi, Hyeong-Seok, Jinyeok Yang, et al. (2022). ‘NANSY++: Unified Voice Synthesis with Neural Analysis and Synthesis’. In: *The Eleventh International Conference on Learning Representations* (cit. on pp. 53, 60).
- Choi, Yunjey et al. (2018). ‘Stargan: Unified generative adversarial networks for multi-domain image-to-image translation’. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE (cit. on p. 28).

- Chowning, John (1989). ‘Frequency Modulation Synthesis of the Singing Voice’. In: *Current directions in computer music research*. Ed. by Max V. Mathews and John Robinson Pierce. Boston: MIT Press. Chap. 6, pp. 57–64 (cit. on pp. 8, 165).
- Corbiau, Gérard (1994). *Farinelli* (cit. on p. 130).
- Cramer, Elliot M and WH Huggins (1958). ‘Creation of pitch through binaural interaction’. In: *The Journal of the Acoustical Society of America* 30.5, pp. 413–417 (cit. on p. 11).
- Culling, John F, A Quentin Summerfield, and David H Marshall (1998). ‘Dichotic pitches as illusions of binaural unmasking. I. Huggins’ pitch and the “binaural edge pitch”’. In: *The Journal of the Acoustical Society of America* 103.6, pp. 3509–3526 (cit. on p. 11).
- Deepak, KT et al. (2019). ‘Glottal Instants Extraction from Speech Signal Using Generative Adversarial Network’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5946–5950 (cit. on p. 49).
- Defez, Àngel Calzada, Joan Claudi Socoró Carrié, and Robert AJ Clark (2013). ‘Parametric model for vocal effort interpolation with Harmonics Plus Noise Models’. In: *8th ISCA Tutorial and Research Workshop on Speech Synthesis, SSW 2013* (cit. on p. 58).
- Degottex, Gilles, Pierre Lanchantin, and Mark Gales (2016). ‘A Pulse Model in Log-Domain for a Uniform Synthesizer’. In: *9th ISCA Speech Synthesis Workshop*, pp. 230–236 (cit. on p. 51).
- Degottex, Gilles, Pierre Lanchantin, Axel Roebel, et al. (2013). ‘Mixed source model and its adapted vocal tract filter estimate for voice transformation and synthesis’. In: *Speech Communication* 55.2, pp. 278–294 (cit. on pp. 51, 53, 57, 58, 63, 92).
- Degottex, Gilles, Axel Roebel, and Xavier Rodet (2010). ‘Phase minimization for glottal model estimation’. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.5, pp. 1080–1090 (cit. on pp. 50, 65).
- (2011). ‘Pitch transposition and breathiness modification using a glottal source model and its adapted vocal-tract filter’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 58, 93).
- Depalle, Philippe, Guillermo Garcia, and Xavier Rodet (1994). ‘A virtual castrato (!?)’ In: *International Computer Music Conference (ICMC)*, pp. 357–360 (cit. on pp. 130, 137).
- Desai, Srinivas et al. (2009). ‘Voice conversion using artificial neural networks’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 60, 92).
- Deschamps, Judith (2022a). ‘La mue, la mort et le chant d’une IA’. In: *Fabula / Les colloques, Souci de l’autre, souci de soi et création, Pour une littérature du care*. url: <http://www.fabula.org/colloques/document8285.php> (cit. on pp. 34, 131).
- (2022b). *La Mue, un conte vidéographique* (cit. on pp. 2, 129, 134).
- Deshpande, P. S. and M. S. Manikandan (2018). ‘Effective Glottal Instant Detection and Electroglossographic Parameter Extraction for Automated Voice Pathology Assessment’. In: *IEEE Journal of Biomedical and Health Informatics* 22.2, pp. 398–408. issn: 21682194. doi: [10.1109/JBHI.2017.2654683](https://doi.org/10.1109/JBHI.2017.2654683) (cit. on p. 49).
- Desler, Anne (2014). “Il novello Orfeo’ Farinelli: vocal profile, aesthetics, rhetoric”. PhD thesis. University of Glasgow (cit. on p. 130).

## Bibliography

---

- Do, Kien and Truyen Tran (2020). ‘Theory and Evaluation Metrics for Learning Disentangled Representations’. In: *International Conference on Learning Representations*. url: <https://openreview.net/forum?id=HJgKOh4Ywr> (cit. on p. 59).
- Dohi, Kota, Takashi Endo, and Yohei Kawaguchi (2022). ‘Disentangling physical parameters for anomalous sound detection under domain shifts’. In: *30th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 279–283 (cit. on p. 59).
- Donahue, Jeff et al. (2021). ‘End-to-end Adversarial Text-to-Speech’. In: *International Conference on Learning Representations*. url: <https://openreview.net/forum?id=rsf1z-JSj87> (cit. on pp. 8, 57, 78).
- Doval, Boris, Christophe d’ Alessandro, and Nathalie Henrich (2006). ‘The spectrum of glottal flow models’. In: *Acta acustica united with acustica* 92.6, pp. 1026–1046 (cit. on p. 18).
- Drugman, T., Paavo Alku, et al. (2014). ‘Glottal source processing: From analysis to applications’. In: *Computer Speech and Language* 28.5, pp. 1117–1138. issn: 10958363. doi: [10.1016/j.csl.2014.03.003](https://doi.org/10.1016/j.csl.2014.03.003) (cit. on p. 49).
- Drugman, T. and T. Dutoit (2009). ‘Glottal Closure and Opening Instant Detection from Speech Signals’. In: *10th Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cit. on pp. 49, 70, 71).
- (2012). ‘The Deterministic plus Stochastic model of the residual signal and its applications’. In: *IEEE Transactions on Audio, Speech and Language Processing* 20.3, pp. 968–981. issn: 15587916. doi: [10.1109/TASL.2011.2169787](https://doi.org/10.1109/TASL.2011.2169787) (cit. on p. 49).
- Duan, Zhiyan et al. (2013). ‘The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech’. In: *6th Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE (cit. on pp. 33, 34).
- Dudley, Homer (1939). ‘Remaking speech’. In: *The Journal of the Acoustical Society of America* 11.2, pp. 169–177 (cit. on pp. 6, 7, 46, 51, 92).
- Duifhuis, Hendrikus, Lei F Willems, and RJ Sluyter (1982). ‘Measurement of pitch in speech: An implementation of Goldstein’s theory of pitch perception’. In: *The Journal of the Acoustical Society of America* 71.6, pp. 1568–1580 (cit. on p. 48).
- Eastwood, Cian and Christopher KI Williams (2018). ‘A framework for the quantitative evaluation of disentangled representations’. In: *International Conference on Learning Representations* (cit. on p. 59).
- EMS, European Mathematical Society (2020). ‘Confidence intervals’. In: *Encyclopedia of Mathematics*. Accessed 2023-05-07. EMS Press. url: [http://encyclopediaofmath.org/index.php?title=Confidence\\_estimation&oldid=44400](http://encyclopediaofmath.org/index.php?title=Confidence_estimation&oldid=44400) (cit. on p. 44).
- Engel, Jesse, Chenjie Gu, Adam Roberts, et al. (2019). ‘DDSP: Differentiable Digital Signal Processing’. In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 23, 30, 164).
- Esling, Philippe, Adrien Bitton, et al. (2018). ‘Generative timbre spaces: regularizing variational auto-encoders with perceptual metrics’. In: *21st International Conference on Digital Audio Effects (DAFx)* (cit. on p. 29).
- Evanini, Keelan, Stephen Isard, and Mark Liberman (2009). ‘Automatic formant extraction for sociolinguistic analysis of large corpora’. In: *10th Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cit. on p. 50).

## Bibliography

---

- Fant, Gunnar (1981). ‘The source filter concept in voice production’. In: *STL-QPSR* 1.1981, pp. 21–37 (cit. on pp. 47, 63).
- (1995). ‘The LF-model revisited. Transformations and frequency domain analysis’. In: *Speech Trans. Lab. Q. Rep., Royal Inst. of Tech. Stockholm* 2.3, p. 40 (cit. on pp. 18, 20, 21, 64, 65, 71, 100, 166).
- Fant, Gunnar and Anita Kruckenberg (1994). ‘Notes on stress and word accent in Swedish’. In: *Proceedings of the international symposium on prosody*. Vol. 18, pp. 2–3 (cit. on p. 21).
- Fant, Gunnar, Johan Liljencrants, and Qi-guang Lin (1985). ‘A four-parameter model of glottal flow’. In: *STL-QPSR* 4.1985, pp. 1–13 (cit. on pp. 8, 15, 18, 20, 65, 100, 166).
- Fant, Gunnar and J Martony (1962). ‘Speech synthesis instrumentation for parametric synthesis (OVE II)’. In: *Speech Transmission Laboratory Quarterly Progress and Status Report (KTH)* 2, pp. 18–24 (cit. on p. 7).
- Farner, Snorre, Axel Roebel, and Xavier Rodet (2009). ‘Natural transformation of type and nature of the voice for extending vocal repertoire in high-fidelity applications’. In: *Audio Engineering Society Conference: 35th International Conference: Audio for Games*. Audio Engineering Society (cit. on pp. 57, 93).
- Ferro, Rafael, Nicolas Obin, and Axel Roebel (2020). ‘CycleGAN Voice Conversion of Spectral Envelopes using Adversarial Weights’. In: *European Signal Processing Conference (EUSIPCO)*. IEEE (cit. on pp. 60, 92).
- Flanagan, James L and Roger M Golden (1966). ‘Phase vocoder’. In: *Bell system technical Journal* 45.9, pp. 1493–1509 (cit. on pp. 8, 92).
- Fletcher, Harvey and Wilden A Munson (1933). ‘Loudness, its definition, measurement and calculation’. In: *Bell System Technical Journal* 12.4, pp. 377–430 (cit. on p. 14).
- Ford Baldner, Elizabeth, Emerald Doll, and Miriam Ruth van Mersbergen (2015). ‘A Review of Measures of Vocal Effort With a Preliminary Study on the Establishment of a Vocal Effort Measure’. In: *Journal of Voice* 29.5, pp. 530–541. issn: 0892-1997. doi: <https://doi.org/10.1016/j.jvoice.2014.08.017> (cit. on p. 113).
- Gatys, Leon A, Alexander S Ecker, and Matthias Bethge (2016). ‘Image style transfer using convolutional neural networks’. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE (cit. on pp. 27, 28).
- Gauvain, J. L., L. F. Lamel, and M. Eskenazi (1990). ‘Design considerations and text selection for BREF, a large French read-speech corpus’. In: *ICSLP*, pp. 1097–1100 (cit. on pp. 32, 65).
- Gentilucci, Marta, Luc Ardaillon, and Marco Liuni (2018). ‘Vocal distortion and real-time processing of roughness’. In: *International Computer Music Conference (ICMC)* (cit. on p. 58).
- (2019). ‘Composing vocal distortion: A tool for real-time generation of roughness’. In: *Computer Music Journal* 42.4, pp. 26–40 (cit. on p. 58).
- Glasberg, Brian R and Brian CJ Moore (1990). ‘Derivation of auditory filter shapes from notched-noise data’. In: *Hearing research* 47.1-2, pp. 103–138 (cit. on p. 14).
- (2002). ‘A model of loudness applicable to time-varying sounds’. In: *Journal of the Audio Engineering Society* 50.5, pp. 331–342 (cit. on pp. 14, 15, 121, 123).

## Bibliography

---

- Goodfellow, Ian et al. (2020). ‘Generative adversarial networks’. In: *Communications of the ACM* 63.11, pp. 139–144 (cit. on p. 28).
- Goupil, Louise et al. (2021). ‘Listeners’ perceptions of the certainty and honesty of a speaker are associated with a common prosodic signature’. In: *Nature communications* 12.1, p. 861 (cit. on p. 61).
- Goyal, Mohit, Varun Srivastava, and Prathosh A. P. (2019). ‘Detection of Glottal Closure Instants from Raw Speech Using Convolutional Neural Networks’. In: *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Ed. by Gernot Kubin and Zdravko Kacic. ISCA, pp. 1591–1595. doi: [10.21437/Interspeech.2019-2587](https://doi.org/10.21437/Interspeech.2019-2587) (cit. on p. 49).
- Grammalidis, Nikolaos et al. (2016). ‘The i-treasures intangible cultural heritage dataset’. In: *3rd International Symposium on Movement and Computing (MOCO)* (cit. on p. 34).
- He, Zhenliang et al. (2019). ‘Attgan: Facial attribute editing by only changing what you want’. In: *Transactions on Image Processing* 28.11, pp. 5464–5478 (cit. on p. 28).
- Henrich, Nathalie et al. (2005). ‘Glottal open quotient in singing: Measurements and correlation with laryngeal mechanisms, vocal intensity, and fundamental frequency’. In: *The Journal of the Acoustical Society of America* 117.3, pp. 1417–1430 (cit. on pp. 8, 116).
- Hermes, Dik J (1988). ‘Measurement of pitch by subharmonic summation’. In: *The journal of the acoustical society of America* 83.1, pp. 257–264 (cit. on p. 48).
- Higgins, Irina, David Amos, et al. (2018). ‘Towards a definition of disentangled representations’. In: *arXiv preprint arXiv:1812.02230* (cit. on pp. 58, 93).
- Higgins, Irina, Loic Matthey, et al. (2017). ‘beta-vae: Learning basic visual concepts with a constrained variational framework’. In: *International conference on learning representations* (cit. on p. 59).
- Holmberg, Eva B, Robert E Hillman, and Joseph S Perkell (1988). ‘Glottal airflow and transglottal air pressure measurements for male and female speakers in soft, normal, and loud voice’. In: *The Journal of the Acoustical Society of America* 84.2, pp. 511–529 (cit. on pp. 8, 116).
- Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). ‘Multilayer feedforward networks are universal approximators’. In: *Neural networks* 2.5, pp. 359–366 (cit. on p. 21).
- Huber, Stefan (2015). ‘Voice Conversion by modelling and transformation of extended voice characteristics’. PhD thesis. Université Pierre et Marie Curie-Paris VI (cit. on pp. 51, 58).
- Huber, Stefan and Axel Roebel (2015). ‘On glottal source shape parameter transformation using a novel deterministic and stochastic speech analysis and synthesis system’. In: *16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on pp. 8, 47, 53, 65, 92).
- Huber, Stefan, Axel Roebel, and Gilles Degottex (2012). ‘Glottal Source Shape Parameter Estimation Using Phase Minimization Variants’. In: *Thirteenth Annual Conference of the International Speech Communication Association* (cit. on pp. 49, 65).
- Hunt, Andrew J and Alan W Black (1996). ‘Unit selection in a concatenative speech synthesis system using a large speech database’. In: *Acoustics, Speech, and Signal*

- Processing*, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on. Vol. 1. IEEE, pp. 373–376 (cit. on pp. 8, 55).
- Hwang, Min-Jae et al. (2020). ‘LP-WaveNet: Linear prediction-based WaveNet speech synthesis’. In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, pp. 810–814 (cit. on p. 53).
- Isshiki, Nobuhiko (1964). ‘Regulatory mechanism of voice intensity variation’. In: *Journal of speech and hearing research* 7.1, pp. 17–29 (cit. on pp. 7, 115).
- Ito, Keith and Linda Johnson (2017). *The LJ Speech Dataset*. <https://keithito.com/LJ-Speech-Dataset/> (cit. on p. 32).
- Jang, Won, Dan Lim, and Jaesam Yoon (2020). ‘Universal MelGAN: A Robust Neural Vocoder for High-Fidelity Waveform Generation in Multiple Domains’. In: *arXiv preprint arXiv:2011.09631* (cit. on pp. 28, 54, 83, 88, 89).
- Jenkins, James S (1998). ‘The voice of the castrato’. In: *The Lancet* 351.9119, pp. 1877–1880 (cit. on p. 130).
- Jiao, Yunlong et al. (2021). ‘Universal neural vocoding with parallel wavenet’. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6044–6048 (cit. on p. 83).
- Juvela, Lauri et al. (2019). ‘GELP: GAN-Excited Liner Prediction for Speech Synthesis from Mel-Spectrogram’. In: *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. International Speech Communication Association, pp. 694–698 (cit. on p. 55).
- Kameoka, Hirokazu et al. (2018). ‘Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks’. In: *Spoken Language Technology Workshop (SLT)*. IEEE (cit. on pp. 60, 92).
- Kaneko, Takuhiro and Hirokazu Kameoka (2018). ‘Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks’. In: *26th European Signal Processing Conference (EUSIPCO)*. IEEE (cit. on p. 28).
- Kaneko, Takuhiro, Hirokazu Kameoka, et al. (2019). ‘StarGAN-VC2: Rethinking Conditional Methods for StarGAN-Based Voice Conversion’. In: *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on p. 28).
- Kaneko, Takuhiro, Kou Tanaka, et al. (2022). ‘iSTFTNet: Fast and lightweight mel-spectrogram vocoder incorporating inverse short-time Fourier transform’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6207–6211 (cit. on pp. 26, 55).
- Kawahara, Hideki (2006). ‘STRAIGHT, exploitation of the other aspect of VOCODER: Perceptually isomorphic decomposition of speech sounds’. In: *Acoustical science and technology* 27.6, pp. 349–353 (cit. on pp. 8, 51, 53, 63, 92).
- Kawahara, Hideki et al. (2008). ‘TANDEM-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, F0, and aperiodicity estimation’. In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 3933–3936 (cit. on p. 53).
- Kelly Jr, John L. and Carol C. Lochbaum (1962). ‘Speech synthesis’. In: *Proceedings of the fourth international congress on acoustics*. Vol. 1. G42, pp. 1–4 (cit. on p. 8).

- Kempelen, Wolfgang von (1791). *Mechanismus der menschlichen Sprache*. Degen (cit. on p. 6).
- Kim, Hyunjik and Andriy Mnih (2018). ‘Disentangling by factorising’. In: *International Conference on Machine Learning*. PMLR, pp. 2649–2658 (cit. on p. 59).
- Kim, Jaehyeon, Jungil Kong, and Juhee Son (2021). ‘Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech’. In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 5530–5540 (cit. on pp. 9, 57, 78).
- Kim, Jong Wook et al. (2018). ‘Crepe: A convolutional representation for pitch estimation’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 48, 64).
- Kim, Minyoung et al. (2019). ‘Relevance factor vae: Learning and identifying disentangled factors’. In: *arXiv preprint arXiv:1902.01568* (cit. on p. 59).
- Kingma, Diederik P and Jimmy Ba (2014). ‘Adam: A method for stochastic optimization’. In: *3rd International Conference on Learning Representations (ICLR)* (cit. on pp. 98, 122).
- Kingma, Diederik P and Prafulla Dhariwal (2018). ‘Glow: Generative flow with invertible 1x1 convolutions’. In: *Advances in neural information processing systems 31* (cit. on p. 29).
- Kingma, Diederik P and Max Welling (2013). ‘Auto-encoding variational bayes’. In: *arXiv preprint arXiv:1312.6114* (cit. on p. 29).
- Klatt, Dennis H and Laura C Klatt (1990). ‘Analysis, synthesis, and perception of voice quality variations among female and male talkers’. In: *the Journal of the Acoustical Society of America* 87.2, pp. 820–857 (cit. on pp. 8, 18).
- Kobayashi, Sachie (2022). *Day 0 – Trans-instrumentalism* (cit. on pp. 129, 146).
- Koguchi, Junya, Shinnosuke Takamichi, and Masanori Morise (2020). ‘PJS: phoneme-balanced Japanese singing-voice corpus’. In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE (cit. on p. 34).
- Kominek, John and Alan W Black (2004). ‘The CMU Arctic speech databases’. In: *Fifth ISCA workshop on speech synthesis* (cit. on pp. 32, 69).
- Kong, Jungil, Jaehyeon Kim, and Jaekyoung Bae (2020). ‘Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis’. In: *Advances in Neural Information Processing Systems* 33, pp. 17022–17033 (cit. on p. 55).
- Kumar, Abhishek, Prasanna Sattigeri, and Avinash Balakrishnan (2018). ‘Variational Inference of Disentangled Latent Concepts from Unlabeled Observations’. In: *International Conference on Learning Representations*. url: <https://openreview.net/forum?id=H1kG7GZAW> (cit. on p. 59).
- Kumar, Kundan et al. (2019). ‘Melgan: Generative adversarial networks for conditional waveform synthesis’. In: *Advances in neural information processing systems* 32 (cit. on pp. 28, 54, 73, 83, 84, 167).
- Ladefoged, Peter and Norris P McKinney (1963). ‘Loudness, sound pressure, and subglottal pressure in speech’. In: *The journal of the Acoustical Society of America* 35.4, pp. 454–460 (cit. on pp. 7, 115).
- Lample, Guillaume et al. (2017). ‘Fader networks: manipulating images by sliding attributes’. In: *International Conference on Neural Information Processing Systems (NeurIPS)* (cit. on pp. 29, 58, 111).

- Lange, Sascha and Martin Riedmiller (2010). ‘Deep auto-encoder neural networks in reinforcement learning’. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8 (cit. on p. 29).
- Laroche, Jean, Yannis Stylianou, and Eric Moulines (1993). ‘HNM: A simple, efficient harmonic+ noise model for speech’. In: *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, pp. 169–172 (cit. on pp. 8, 50, 92).
- Larsson, Bjorn (1977). ‘Music and singing synthesis equipment (MUSSE)’. In: *Speech Transmission Laboratory Quarterly Progress and Status Report (STL-QPSR)* 1.1977, pp. 38–40 (cit. on p. 7).
- LeCun, Yann et al. (1989). ‘Backpropagation applied to handwritten zip code recognition’. In: *Neural computation* 1.4, pp. 541–551 (cit. on pp. 23, 24, 164).
- Lee, Sang-gil et al. (2023). ‘Bigvgan: A universal neural vocoder with large-scale training’. In: *12th International Conference on Learning Representations (ICLR)* (cit. on pp. 55, 82, 83).
- Li, Chuan and Michael Wand (2016). ‘Precomputed real-time texture synthesis with markovian generative adversarial networks’. In: *European conference on computer vision*. Springer, pp. 702–716 (cit. on p. 28).
- Li, Yinghao Aaron, Cong Han, and Nima Mesgarani (2023). ‘Styletts-VC: One-Shot Voice Conversion by Knowledge Transfer From Style-Based TTS Models’. In: *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 920–927 (cit. on p. 60).
- Liljencrants, Johan (1968). ‘The OVE III speech synthesizer’. In: *IEEE Transactions on Audio and Electroacoustics* 16.1, pp. 137–140 (cit. on p. 7).
- Linggard, Robert (1985). *Electronic synthesis of speech*. CUP Archive (cit. on p. 6).
- Liu, Xiao et al. (2022). ‘Learning disentangled representations in the imaging domain’. In: *Medical Image Analysis*, p. 102516 (cit. on p. 58).
- Liuni, Marco et al. (2020). ‘ANGUS: Real-time manipulation of vocal roughness for emotional speech transformations’. In: *arXiv preprint arXiv:2008.11241* (cit. on p. 58).
- Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). ‘Fully convolutional networks for semantic segmentation’. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440 (cit. on p. 23).
- Lorenzo-Trueba, Jaime et al. (2018). ‘Robust universal neural vocoding’. In: *arXiv preprint arXiv:1811.06292* (cit. on pp. 54, 82, 83).
- Lu, Peiling et al. (2020). ‘Xiaoicesing: A high-quality and integrated singing voice synthesis system’. In: *21st Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cit. on p. 9).
- Luo, Zhaojie, Jinhui Chen, et al. (2017). ‘Emotional voice conversion using neural networks with arbitrary scales F0 based on wavelet transform’. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2017, pp. 1–13 (cit. on p. 61).
- Luo, Zhaojie, Tetsuya Takiguchi, and Yasuo Ariki (2016). ‘Emotional voice conversion using deep neural networks with MCC and F0 features’. In: *15th International Conference on Computer and Information Science (ICIS)*. IEEE/ACIS, pp. 1–5 (cit. on p. 61).
- Markel, John D and AH Jr Gray (2013). *Linear prediction of speech*. Vol. 12. Springer Science & Business Media (cit. on pp. 47, 63).

- Mauch, Matthias and Simon Dixon (2014). ‘pYIN: A fundamental frequency estimator using probabilistic threshold distributions’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on p. 48).
- McCandless, Stephanie (1974). ‘An algorithm for automatic formant extraction using linear prediction spectra’. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 22.2, pp. 135–141 (cit. on p. 50).
- McKenna, Victoria S and Cara E Stepp (2018). ‘The relationship between acoustical and perceptual measures of vocal effort’. In: *The Journal of the Acoustical Society of America* 144.3, pp. 1643–1658 (cit. on p. 113).
- Mehta, Daryush and Thomas F Quatieri (2005). ‘Synthesis, analysis, and pitch modification of the breathy vowel’. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005*. IEEE, pp. 199–202 (cit. on p. 51).
- Moine, Clément Le (2023). ‘Neural conversion of social attitudes in speech signals’. PhD thesis. Sorbonne Université (cit. on p. 149).
- Moine, Clément Le and Nicolas Obin (2020). ‘Att-HACK: An Expressive Speech Database with Social Attitudes’. In: *Speech Prosody* (cit. on pp. 31, 32, 98, 99).
- Moine, Clément Le, Nicolas Obin, and Axel Roebel (2021). ‘Towards end-to-end F0 voice conversion based on Dual-GAN with convolutional wavelet kernels’. In: *European Signal Processing Conference (EUSIPCO)*. IEEE (cit. on pp. 61, 92).
- Molina, Emilio et al. (2014). ‘Parametric model of spectral envelope to synthesize realistic intensity variations in singing voice’. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 634–638 (cit. on p. 58).
- Moore, Brian CJ (2012). *An introduction to the psychology of hearing*. Brill (cit. on pp. 9–11, 14).
- Morgan, Nelson et al. (2005). ‘Pushing the envelope-aside [speech recognition]’. In: *IEEE Signal Processing Magazine* 22.5, pp. 81–88 (cit. on p. 50).
- Morise, Masanori (2015). ‘CheapTrick, a spectral envelope estimator for high-quality speech synthesis’. In: *Speech Communication* 67, pp. 1–7 (cit. on p. 50).
- Morise, Masanori, Fumiya Yokomori, and Kenji Ozawa (2016). ‘WORLD: a vocoder-based high-quality speech synthesis system for real-time applications’. In: *IEICE TRANSACTIONS on Information and Systems* 99.7, pp. 1877–1884 (cit. on pp. 8, 53, 63, 92).
- Moulines, Eric and Francis Charpentier (1990). ‘Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones’. In: *Speech communication* 9.5–6, pp. 453–467 (cit. on pp. 8, 50–52, 55, 92).
- Murthy, Hema A and Bayya Yegnanarayana (1991). ‘Formant extraction from group delay function’. In: *speech communication* 10.3, pp. 209–221 (cit. on p. 50).
- Murty, K. S. R. and Bayya Yegnanarayana (2008). ‘Epoch Extraction From Speech Signals’. In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.8, pp. 1602–1613 (cit. on p. 49).
- Nagrani, Arsha et al. (2019). ‘Voxceleb: Large-scale speaker verification in the wild’. In: *Computer Science and Language* (cit. on p. 31).
- Nair, Vinod and Geoffrey E Hinton (2010). ‘Rectified linear units improve restricted boltzmann machines’. In: *International Conference on Machine Learning (ICML)*. PMLR (cit. on pp. 98, 168).

## Bibliography

---

- Naylor, Patrick A et al. (2007). ‘Estimation of glottal closure instants in voiced speech using the DYPSA algorithm’. In: *IEEE Transactions on Audio, Speech and Language Processing* 15.1, pp. 34–43. issn: 15587916. doi: [10.1109/TASL.2006.876878](https://doi.org/10.1109/TASL.2006.876878) (cit. on p. 49).
- Nguyen, Bac and Fabien Cardinaux (2022). ‘Nvc-net: End-to-end adversarial voice conversion’. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7012–7016 (cit. on pp. 60, 78).
- Nitzan, Yotam et al. (2020). ‘Face identity disentanglement via latent space mapping’. In: *ACM Transactions on Graphics (TOG)* 39.6, pp. 1–14 (cit. on p. 58).
- Noll, A Michael (1967). ‘Cepstrum pitch determination’. In: *The journal of the acoustical society of America* 41.2, pp. 293–309 (cit. on p. 48).
- Odena, Augustus, Christopher Olah, and Jonathon Shlens (2017). ‘Conditional image synthesis with auxiliary classifier gans’. In: *International Conference on Machine Learning (ICML)*. PMLR (cit. on p. 28).
- Ogawa, Itsuki and Masanori Morise (2021). ‘Tohoku Kiritan singing database: A singing database for statistical parametric singing synthesis using Japanese pop songs’. In: *Acoustical Science and Technology* 42.3, pp. 140–145 (cit. on p. 34).
- Oh, Suhyeon et al. (2020). ‘ExcitGlow: Improving a WaveGlow-based neural vocoder with linear prediction analysis’. In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, pp. 831–836 (cit. on pp. 53, 54).
- Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). ‘Feature Visualization’. In: *Distill*. <https://distill.pub/2017/feature-visualization>. doi: [10.23915/distill.00007](https://doi.org/10.23915/distill.00007) (cit. on p. 27).
- Oord, Aaron van den, Sander Dieleman, et al. (2016). ‘Wavenet: A generative model for raw audio’. In: *arXiv preprint arXiv:1609.03499* (cit. on pp. 25, 53, 54, 66, 75, 77, 86, 159, 169).
- Oord, Aaron van den, Yazhe Li, et al. (2017). ‘Parallel WaveNet: Fast high-fidelity speech synthesis’. In: *arXiv preprint arXiv:1711.10433* (cit. on pp. 54, 66, 83).
- Panayotov, Vassil et al. (2015). ‘Librispeech: an asr corpus based on public domain audio books’. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 5206–5210 (cit. on p. 31).
- Patel, Aniruddh D and Evan Balaban (2001). ‘Human pitch perception is reflected in the timing of stimulus-related cortical activity’. In: *Nature neuroscience* 4.8, pp. 839–844 (cit. on p. 11).
- Peng, Xi et al. (2017). ‘Reconstruction-based disentanglement for pose-invariant face recognition’. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1623–1632 (cit. on p. 58).
- Perrotin, Olivier and Christophe d’ Alessandro (2016). ‘Vocal effort modification for singing synthesis’. In: *17th Annual Conference of the International Speech Communication Association (INTERSPEECH 2016)*, pp. 1235–1239 (cit. on p. 58).
- Ping, Wei et al. (2018). ‘Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning.’ In: *7th International Conference on Learning Representations (ICLR)* (cit. on pp. 56, 80, 82, 83).

- Prathosh, A. P., T. V. Ananthapadmanabha, and A. G. Ramakrishnan (2013). ‘Epoch Extraction Based on Integrated Linear Prediction Residual Using Plosion Index’. In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.12, pp. 2471–2480 (cit. on p. 49).
- Prathosh, A. P., Varun Srivastava, and Mayank Mishra (2019). ‘Adversarial approximate inference for speech to electroglottograph conversion’. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.12, pp. 2183–2196 (cit. on p. 49).
- Prenger, Ryan, Rafael Valle, and Bryan Catanzaro (2019). ‘Waveglow: A flow-based generative network for speech synthesis’. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3617–3621 (cit. on pp. 25, 26, 29, 54, 83, 84, 169).
- Proszewska, Magdalena et al. (2022). ‘GlowVC: Mel-spectrogram space disentangling model for language-independent text-free voice conversion’. In: *23rd Annual Conference of the International Speech Communication Association (INTERSPEECH)*. doi: [10.21437/Interspeech.2022-322](https://doi.org/10.21437/Interspeech.2022-322) (cit. on p. 60).
- Puts, David Andrew et al. (2007). ‘Men’s voices as dominance signals: vocal fundamental and formant frequencies influence dominance attributions among men’. In: *Evolution and Human Behavior* 28.5, pp. 340–344 (cit. on p. 61).
- Qian, Kaizhi, Zeyu Jin, et al. (2020). ‘F0-consistent many-to-many non-parallel voice conversion via conditional autoencoder’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 4, 59, 92, 111, 113, 120, 154).
- Qian, Kaizhi, Yang Zhang, Shiyu Chang, Mark Hasegawa-Johnson, et al. (2020). ‘Unsupervised speech decomposition via triple information bottleneck’. In: *International Conference on Machine Learning (ICML)*. PMLR (cit. on pp. 59, 60, 95).
- Qian, Kaizhi, Yang Zhang, Shiyu Chang, Xuesong Yang, et al. (2019). ‘Autovc: Zero-shot voice style transfer with only autoencoder loss’. In: *International Conference on Machine Learning (ICML)*. PMLR (cit. on pp. 4, 29, 59, 60, 92, 98, 113, 163).
- Quatieri, Thomas F and Robert J McAulay (1992). ‘Shape invariant time-scale and pitch modification of speech’. In: *IEEE Transactions on Signal Processing* 40.3, pp. 497–510 (cit. on pp. 8, 52, 92).
- Rabiner, Lawrence (1977). ‘On the use of autocorrelation analysis for pitch detection’. In: *IEEE transactions on acoustics, speech, and signal processing* 25.1, pp. 24–33 (cit. on p. 48).
- Rabiner, Lawrence and Biing-Hwang Juang (1993). *Fundamentals of speech recognition*. Prentice-Hall, Inc. (cit. on p. 12).
- Rafii, Zafar et al. (Dec. 2017). *The MUSDB18 corpus for music separation*. doi: [10.5281/zenodo.11117372](https://doi.org/10.5281/zenodo.11117372) (cit. on p. 89).
- Raitio, Tuomo, Ramya Rasipuram, and Dan Castellani (2020). ‘Controllable neural text-to-speech synthesis using intuitive prosodic features’. In: *21st Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cit. on p. 58).
- Reddy, M. G., T. Mandal, and K. S. Rao (2018). ‘Glottal closure instants detection from pathological acoustic speech signal using deep learning’. In: *International Conference on Neural Information Processing Systems (NeurIPS). Machine Learning for Health* (cit. on p. 49).

## Bibliography

---

- Ren, Yi et al. (2021). ‘FastSpeech 2: Fast and High-Quality End-to-End Text to Speech’. In: *International Conference on Learning Representations*. url: <https://openreview.net/forum?id=piLPYqxtWuA> (cit. on pp. 8, 57, 78).
- Richard, Gaël and Christophe d’ Alessandro (1996). ‘Analysis/Synthesis and Modification of the Speech Aperiodic Components’. In: *Speech Communication* 19.3, pp. 221–244 (cit. on pp. 16, 17).
- Ridgeway, Karl and Michael C Mozer (2018). ‘Learning deep disentangled embeddings with the f-statistic loss’. In: *Advances in neural information processing systems* 31 (cit. on p. 59).
- Robinson, Carl, Nicolas Obin, and Axel Roebel (2019). ‘Sequence-to-sequence modeling of f0 for speech emotion conversion’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 61, 92).
- Robinson, Derek W and R So Dadson (1956). ‘A re-determination of the equal-loudness relations for pure tones’. In: *British Journal of Applied Physics* 7.5, p. 166 (cit. on p. 14).
- Rodet, Xavier, Yves Potard, and Jean-Baptiste Barriere (1984). ‘The CHANT project: From the synthesis of the singing voice to synthesis in general’. In: *Computer Music Journal* 8.3, pp. 15–31 (cit. on p. 8).
- Roebel, Axel (2010). ‘A shape-invariant phase vocoder for speech transformation’. In: *13th International Conference on Digital Audio Effects (DAFx)* (cit. on pp. 8, 53, 92).
- Roebel, Axel and Frederik Bous (2022). ‘Neural Vocoding for Singing and Speaking Voices with the Multi-band Excited WaveNet’. In: *Information* 13.3, p. 103 (cit. on pp. 28, 48, 55, 83, 85, 86, 88, 89).
- Roebel, Axel and Xavier Rodet (2005). ‘Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation’. In: *International Conference on Digital Audio Effects*, pp. 30–35 (cit. on pp. 50, 169).
- Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). ‘U-net: Convolutional networks for biomedical image segmentation’. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, pp. 234–241 (cit. on p. 23).
- Rosenberg, Aaron E (1971). ‘Effect of glottal pulse shape on the quality of natural vowels’. In: *The Journal of the Acoustical Society of America* 49.2B, pp. 583–590 (cit. on pp. 8, 18).
- Ruggero, Mario A (1992). ‘Responses to sound of the basilar membrane of the mammalian cochlea’. In: *Current opinion in neurobiology* 2.4, pp. 449–456 (cit. on p. 10).
- Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). ‘Learning representations by back-propagating errors’. In: *nature* 323.6088, pp. 533–536 (cit. on p. 23).
- Sagisaka, Yoshinori et al. (1992). ‘ATR  $\mu$ -talk speech synthesis system.’ In: *Second International Conference on Spoken Language Processing*. Vol. 92, pp. 483–486 (cit. on pp. 8, 55).
- Salais, Léane et al. (2022). ‘Production Strategies of Vocal Attitudes’. In: *23rd Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, pp. 4985–4989 (cit. on pp. 61, 92).
- Schafer, Ronald W and Lawrence Rabiner (1975). ‘Digital representations of speech signals’. In: *Proceedings of the IEEE* 63.4, pp. 662–677 (cit. on p. 50).

- Schmetterer, Leopold (2012). *Introduction to mathematical statistics*. Vol. 202. Springer Science & Business Media (cit. on p. 45).
- Schroeder, Manfred R (1968). ‘Period histogram and product spectrum: New methods for fundamental-frequency measurement’. In: *The Journal of the Acoustical Society of America* 43.4, pp. 829–834 (cit. on p. 48).
- Sepliarskaia, Anna, Julia Kiseleva, and Maarten de Rijke (2019). ‘How to not measure disentanglement’. In: *arXiv preprint arXiv:1910.05587* (cit. on p. 59).
- Shen, Jonathan et al. (2018). ‘Natural tts synthesis by conditioning wavenet on mel spectrogram predictions’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 12, 25, 54, 56, 57, 66, 77, 80–83, 169).
- Shen, Kai et al. (2023). ‘NaturalSpeech 2: Latent Diffusion Models are Natural and Zero-Shot Speech and Singing Synthesizers’. In: *arXiv preprint arXiv:2304.09116* (cit. on p. 9).
- Shepard, Roger N (1964). ‘Circularity in judgments of relative pitch’. In: *The journal of the acoustical society of America* 36.12, pp. 2346–2353 (cit. on p. 12).
- Shirazi, Aïda (2022). *Né entre corps* (cit. on p. 129).
- Singh, Satwinder, Ruili Wang, and Yuanhang Qiu (2021). ‘DeepF0: End-to-end fundamental frequency estimation for music and speech signals’. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 61–65 (cit. on p. 48).
- Sondhi, Mohan (1968). ‘New methods of pitch extraction’. In: *IEEE Transactions on audio and electroacoustics* 16.2, pp. 262–266 (cit. on p. 48).
- Song, Eunwoo, Kyungguen Byun, and Hong-Goo Kang (2019). ‘ExcitNet vocoder: A neural excitation model for parametric speech synthesis systems’. In: *27th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1–5 (cit. on p. 53).
- Stevens, Stanley Smith, John Volkmann, and Edwin Broomell Newman (1937). ‘A scale for the measurement of the psychological magnitude pitch’. In: *The journal of the acoustical society of america* 8.3, pp. 185–190 (cit. on pp. 12, 15).
- Stewart, John Q (1922). ‘An electrical analogue of the vocal organs’. In: *Nature* 110.2757, p. 311 (cit. on pp. 6, 7, 18, 46).
- Stylianou, Yannis, Jean Laroche, and Eric Moulines (1995). ‘High-quality speech modification based on a harmonic+ noise model’. In: *Fourth European Conference on Speech Communication and Technology* (cit. on pp. 8, 47, 50–52, 92).
- Sun, Xuejing (2000). ‘A pitch determination algorithm based on subharmonic-to-harmonic ratio’. In: *2nd Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on p. 48).
- Sundberg, Johan (2006). ‘The KTH synthesis of singing’. In: *Advances in cognitive Psychology* 2.2, p. 131 (cit. on pp. 7, 8).
- Sundberg, Johan, Ingo Titze, and Ronald Scherer (1993). ‘Phonatory control in male singing: A study of the effects of subglottal pressure, fundamental frequency, and mode of phonation on the voice source’. In: *Journal of Voice* 7.1, pp. 15–29 (cit. on pp. 8, 114, 115).
- Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). ‘Sequence to sequence learning with neural networks’. In: *Advances in neural information processing systems* 27 (cit. on p. 24).

- Tamaru, Hiroki et al. (2020). ‘JVS-MuSiC: Japanese multispeaker singing-voice corpus’. In: *arXiv preprint arXiv:2001.07044* (cit. on p. 34).
- Tandler, Julius and Siegfried Grosz (1909). ‘Über den Einfluß der Kastration auf den Organismus’. In: *Archiv für Entwicklungsmechanik der Organismen* 27.1, pp. 35–61 (cit. on p. 130).
- Thomas, Mark RP and Patrick A Naylor (2009). ‘The SIGMA algorithm: A glottal activity detector for electroglottographic signals’. In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.8, pp. 1557–1566 (cit. on p. 69).
- Titze, Ingo R and Johan Sundberg (1992). ‘Vocal intensity in speakers and singers’. In: *the Journal of the Acoustical Society of America* 91.5, pp. 2936–2946 (cit. on pp. 8, 114–116).
- Traunmüller, Hartmut and Anders Eriksson (2000). ‘Acoustic effects of variation in vocal effort by men, women, and children’. In: *The Journal of the Acoustical Society of America* 107.6, pp. 3438–3451 (cit. on pp. 8, 113, 115, 116, 120).
- Tsirulnik, Liliya and Shlomo Dubnov (2019). ‘Singing Voice Database’. In: *International Conference on Speech and Computer (ICSC)*. Springer (cit. on pp. 33, 34).
- Turk, Oytun et al. (2005). ‘Voice quality interpolation for emotional text-to-speech synthesis’. In: *6th Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cit. on p. 58).
- Umbert, Martí, Jordi Bonada, and Merlijn Blaauw (2013). ‘Generating singing voice expression contours based on unit selection’. In: *Stockholm Music Acoustics Conference (SMAC)* (cit. on pp. 8, 56, 92).
- Umbert, Martí, Jordi Bonada, Masataka Goto, et al. (2015). ‘Expression control in singing voice synthesis: Features, approaches, evaluation, and challenges’. In: *IEEE Signal Processing Magazine* 32.6, pp. 55–73 (cit. on p. 92).
- Vaswani, Ashish et al. (2017). ‘Attention is all you need’. In: *Advances in neural information processing systems* 30 (cit. on p. 24).
- Veaux, Christophe and Xavier Rodet (2011). ‘Intonation conversion from neutral to expressive speech’. In: *Twelfth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA (cit. on pp. 61, 92).
- Veldhuis, Raymond (1998). ‘A computationally efficient alternative for the llijencrants-fant model and its perceptual evaluation’. In: *The Journal of the Acoustical Society of America* 103.1, pp. 566–571 (cit. on pp. 8, 18).
- Wang, Chengyi et al. (2023). ‘Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers’. In: *arXiv preprint arXiv:2301.02111* (cit. on p. 9).
- Wang, Xin, Shinji Takaki, and Junichi Yamagishi (2019). ‘Neural source-filter waveform models for statistical parametric speech synthesis’. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28, pp. 402–415 (cit. on pp. 27, 54, 66, 67, 87).
- Wang, Yuxuan et al. (2017). ‘Tacotron: Towards End-to-End Speech Synthesis’. In: *18th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 4006–4010. doi: [10.21437/Interspeech.2017-1452](https://doi.org/10.21437/Interspeech.2017-1452) (cit. on pp. 54, 56, 169).

## Bibliography

---

- Wang, Zhichao et al. (2021). ‘Accent and speaker disentanglement in many-to-many voice conversion’. In: *12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, pp. 1–5 (cit. on p. 59).
- Werbos, Paul J (1990). ‘Backpropagation through time: what it does and how to do it’. In: *Proceedings of the IEEE 78.10*, pp. 1550–1560 (cit. on p. 24).
- Wilkins, Julia et al. (2018). ‘VocalSet: A Singing Voice Dataset’. In: *International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR (cit. on p. 34).
- Williams, Ronald J and David Zipser (1995). ‘Gradient-based learning algorithms for recurrent networks and their computational complexity’. In: *Backpropagation: theory, architectures, and applications*, pp. 433–486 (cit. on p. 24).
- Wu, Jie and Jian Luan (2020). ‘Adversarially trained multi-singer sequence-to-sequence singing synthesizer’. In: *21st Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cit. on p. 9).
- Xie, Qicong et al. (2022). ‘End-to-End Voice Conversion with Information Perturbation’. In: *arXiv preprint arXiv:2206.07569* (cit. on pp. 60, 78).
- Yamagishi, Junichi, Christophe Veaux, and Kirsten MacDonald (2019). *CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92)*. doi: [10.7488/ds/2645](https://doi.org/10.7488/ds/2645) (cit. on pp. 32, 98, 99).
- Yang, Geng et al. (2021). ‘Multi-band MelGAN: Faster waveform generation for high-quality text-to-speech’. In: *Spoken Language Technology Workshop (SLT)*. IEEE, pp. 492–498 (cit. on pp. 54, 88, 89).
- Yang, S. et al. (2018). ‘Detection of glottal closure instants from speech signals: A convolutional neural network based method’. In: *19th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 317–321. doi: [10.21437/Interspeech.2018-1281](https://doi.org/10.21437/Interspeech.2018-1281) (cit. on p. 49).
- Yoshimura, Takayoshi (2002). ‘Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for HMM-based text-to-speech systems’. In: *PhD diss, Nagoya Institute of Technology* (cit. on pp. 8, 56).
- Yu, Chengzhu et al. (2020). ‘DurIAN: Duration Informed Attention Network For Speech Synthesis’. In: (cit. on p. 26).
- Zen, Heiga, Yannis Agiomyrgiannakis, et al. (2016). ‘Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices’. In: *17th Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cit. on pp. 8, 56).
- Zen, Heiga, Andrew Senior, and Mike Schuster (2013). ‘Statistical parametric speech synthesis using deep neural networks’. In: *international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 7962–7966 (cit. on pp. 8, 56).
- Zen, Heiga, Keiichi Tokuda, and Alan W Black (2009). ‘Statistical parametric speech synthesis’. In: *speech communication* 51.11, pp. 1039–1064 (cit. on pp. 8, 56).
- Zhang, Jing-Xuan, Zhen-Hua Ling, and Li-Rong Dai (2019). ‘Non-parallel sequence-to-sequence voice conversion with disentangled linguistic and speaker representations’. In: *Transactions on Audio, Speech, and Language Processing* 28, pp. 540–552 (cit. on pp. 9, 59, 60, 80, 92).

## Bibliography

---

- Zhao, Guanlong et al. (2018). ‘Accent conversion using phonetic posteriograms’. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE (cit. on pp. 60, 92).
- Zhou, Kun et al. (2021). ‘Seen and unseen emotional style transfer for voice conversion with a new emotional speech dataset’. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 920–924 (cit. on p. 61).
- (2022). ‘Emotional voice conversion: Theory, databases and ESD’. In: *Speech Communication* 137, pp. 1–18 (cit. on p. 61).
- Zhu, Jun-Yan et al. (2017). ‘Unpaired image-to-image translation using cycle-consistent adversarial networks’. In: *International Conference on Computer Vision (ICCV)*. IEEE (cit. on p. 28).
- Zue, V., S. Seneff, and J. Glass (1990). ‘Speech database development at MIT: TIMIT and beyond’. In: *Speech communication* 9.4, pp. 351–356 (cit. on pp. 32, 65).