THE UNIVERSITY *of* EDINBURGH

Fast and Controllable Speech Generation

Jacob Josiah Webber

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2025

Abstract

The work described in this thesis was completed shortly after the emergence of deep learning as the state-of-the art for speech generation tasks. In 2016 WaveNET showed that autoregressive generative modelling could generate high quality speech waveforms, and in 2017 the emergence of Tacotron showed the power of cutting edge sequence-to-sequence models (then recurrent encoder-decoder nets with attention) for speech synthesis.

However, while these models showed a large improvement in output quality when compared with the statistical-parametric models that preceded them, they came with new problems of their own. Firstly these models were extremely expensive in terms of the compute required at both training and synthesis time. Secondly, while these models excelled at creating samples similar to those upon which they were trained, the lack of interpretable intermediate features, such as F_0 or durations, meant that it was no longer possible to have fine-grained control over synthesis.

This thesis tackles these two problems in turn, presenting approaches which generate speech in a way that is fast on low-end hardware or controllable by arbitrary parameters. In the case of the former, a range of techniques are presented that combine digital signal processing techniques with machine learning to deliver high-quality audio with considerably less computational expense than state-of-the-art neural vocoders. In order to enable controllable speech generation, I present Hider-Finder-Combiner: a system of adversarial information hiding used to derive disentangled representations of speech. The common thread uniting these approaches is a reliance on learned representations of speech.

By developing the Hider-Finder-Combiner architecture, which was introduced in a MScRes thesis also derived from the same project as this thesis, I show the power of learned representations to control speech prosody and enable privacy-preserving uses of speech technology. Through Autovocoder I show that representations of speech can be learned that are efficiently converted into audio waveforms. Together these streams of work show that learned representations, derived using deep learning techniques, can enable fast, controllable speech generation.

Acknowledgements

I would like to first thank my primary supervisor, Professor Simon King for his help and guidance throughout this work. Simon rescued me when I was flailing for a PhD subject, and I couldn't have hoped for a more interesting field or a better supervisor. I would also like to thank my additional supervisors, Dr Cassia Valentini Botinhao and Professor Junichi Yamagishi.

I would also like to thank all my coauthors from work I completed during my studies, Dr Olivier Perrotin, Dr Gustav Henter, Evelyn Williams, Dr Oliver Watts, Dr Jennifer Williams, Samuel Lo and Professor Isaac Bleaman. Working with so many interesting people was a highlight of completing a PhD. So to was working, and eating lunch with, so many wonderful colleagues in the CSTR. Special thanks go to Zack Hodari who helped me so much with many aspects of this thesis and CSTR student life.

Speak Unique have agreed to employ me after I finish this document and have been a delight to work with, and just about as flexible an employer as anyone could hope to have. Thanks to Oliver, Alice and Cassia for taking the stress about my future life off my shoulders so I could focus on writing this thesis.

I would also like to thank my family and especially Evelyn, who as well as being an excellent coauthor and proofreader has been a consistent source of moral support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jacob Josiah Webber*)

Table of Contents

1 Introduction **1**

I Background **3**

2 Speech signal processing and Fourier methods **5**

2.1 Digital audio 5

2.2 The frequency domain 6

2.3 Digital filtering 9

2.4 The human voice 9

3 Machine learning, controllability and generative models **11**

3.1 Deep learning 11

3.1.1 Optimisation 14

3.2 Probabilistic machine learning and generative modelling 14

3.2.1 Generative Adversarial Nets 17

3.2.2 Autoregressive models 18

3.2.3 Diffusion and flow matching 19

3.3 A note on computational performance 21

4 Speech Synthesis **23**

4.1 Introduction 23

4.2 A note on computational performance 24

4.3 Text-to-speech 24

4.4 Vocoding and waveform synthesis 27

4.4.1 DSP-based waveform generation 27

4.4.2 Autoregressive vocoders 29

4.4.3 Non-autoregressive generative vocoders 30

4.4.4 Neural DSP 30
4.5 Evaluation 31
4.5.1 Subjective evaluation 32
4.5.2 Objective evaluation 32
4.6 Summary 33

II Controllable speech synthesis 35

5 Hider-Finder-Combiner 37
5.1 Defining controllability 38
5.2 Motivation and approach 39
5.3 Training the combiner on WORLD features 40
5.4 The Hider-Finder-Combiner System 43
5.4.1 Hider 45
5.4.2 Finder 45
5.4.3 Combiner 46
5.4.4 End-to-end adversarial training 46
5.4.5 Experimental Setup 48
5.5 Evaluation 50
5.5.1 The Baseline DSP System 51
5.5.2 Objective Evaluation 51
5.5.3 Subjective Evaluation 52
5.6 Controlling arbitrary parameters 53
5.6.1 Formant control 53
5.6.2 Controlling DSP-extracted vocal effort correlates 55
5.6.3 Control parameter specifics 57
5.6.4 Output from GFM-IAIF features 58
5.6.5 Analysis of GFM-IAIF features for Lombard and quiet speech 58
5.7 Chapter summary and motivating a change of emphasis 61

6 HFC — New architectures, a global parameter and Voice Conversion-based Privacy 65
6.1 Introduction 66
6.2 Background 67
6.3 Method 69

6.3.1 Losses . . . 69
6.3.2 Data . . . 71
6.3.3 Model and training . . . 71
6.3.4 Component networks . . . 72
6.3.5 Hyper-parameters . . . 73
6.3.6 Anonymisation . . . 74
6.4 Results . . . 74
6.5 Conclusion . . . 77

III Efficient audio synthesis 79

7 Phase and the sign spectrogram 81
7.1 The sign spectrum . . . 82
7.1.1 Definition . . . 83
7.1.2 Recovery algorithm . . . 84
7.2 Fast sign spectrogram inversion . . . 85
7.2.1 The improved algorithm . . . 85
7.2.2 FSSIA Results . . . 86
7.2.3 Recovering from a noisy signal . . . 87
7.2.4 A motivation for generative modelling . . . 89
7.3 Sign prediction . . . 90
7.3.1 SignGAN . . . 91
7.3.2 SignLM . . . 91
7.3.3 SignCFM . . . 92
7.4 Conclusions and summary . . . 93

8 Autovocoder 95
8.1 Introduction and background . . . 95
8.1.1 Parallelism and autoregression . . . 97
8.1.2 Vocoding based on explicit harmonic synthesis . . . 98
8.2 Baseline Autovocoder . . . 99
8.2.1 Architecture . . . 99
8.2.2 Training setup . . . 101
8.2.3 Evaluation . . . 101
8.2.4 Listening tests . . . 102

8.3 Extending Autovocoder with an adversarial loss 103
8.3.1 Evaluating the improved model 105
8.4 Computational cost 106
8.5 Future application to TTS 108
8.6 Conclusion 108
8.7 Frontground: Learned representations and fast waveform synthesis . . 109

9 Conclusion and future work **111**

Bibliography **115**

Chapter 1

Introduction

The work in this thesis was undertaken with the aim of generating speech both controllably and efficiently. These goals were developed in response to the limitations of state-of-the-art models at the time this thesis was started. At that time, statistical-parametric speech synthesis (SPSS) had been usurped by deep sequence models. These models, such as Tacotron, from Wang et al. (2017), were introduced with the aim of generating speech in an *end-to-end* fashion, meaning, unlike SPSS, there was a lack of interpretable intermediate features, such as F_0 or formants, that could be explicitly controlled. The use of deep sequence models, often autoregressive deep sequence models, meant that computational performance was often very poor (van den Oord et al., 2016). This thesis aims to address these challenges in turn, applying novel approaches to specific parts of the speech generation pipeline. This thesis was completed at a time when very rapid advances were happening in the field of speech synthesis. More recent models such as FastPitch from Łańcucki (2021) delivered massive improvements in computational performance, while also allowing some control of F_0. However, this model, which was released after many of the innovations introduced in this thesis, still requires a *neural vocoder* to achieve waveform synthesis. More recently, models such as Parler, from Lyth and King (2024), use performant neural codecs for waveform synthesis, which, following the work introduced in Chapter 8, synthesise from learned representations of speech.

In this dissertation, I propose two main *theses* that are examined in detail.

1. If you can disentangle it, you can control it. That is, *disentangling* a feature from all others is enough, when combined with known machine-learning (ML) techniques, to control that feature.

2. Integrating *differentiable DSP* into ML-based waveform synthesis can improve computational performance.

Part 1 of the thesis introduces background to the rest of the work, in turn introducing speech signal processing, machine learning and speech synthesis. In Part 2 of this thesis, I address the challenge of *controllability* using the method proposed in my first thesis, introducing a novel approach, *Hider-Finder-Combiner*, (HFC) that aims to control arbitrary *control parameters* that need only to be labelled on a training set. In Chapter 5 I apply this model to the task of modifying F_0, work that is also published in a peer-reviewed conference paper (Webber et al., 2020), as well as testing how well the model applies to other control parameters. In Chapter 6 I extend the HFC model to be applied to the task of *voice privacy*, in work that has recently been accepted for publication (Webber et al., 2024). F_0 and speaker ID make good choices for control parameters because they are easy to accurately label – either by using a dataset with multiple labelled speakers, or by using a pitch tracking tool. The modification of these parameters have important use cases for prosody modelling (Hodari, 2022) and voice privacy (Tomashenko et al., 2022a).

In Part 3, adressing my second thesis, I analyse and improve the computational performance of speech generation through the integration of digital signal processing into deep learning. This work was strongly motivated by a lack of computational resources available when completing other parts of the thesis, at least relative to industrial research groups. Chapter 7 applies generative modelling to phase prediction, and Chapter 8 introduces *autovocoder*, a novel approach to waveform synthesis from a learned representation. The *autovocoder* model is also discussed in a peer reviewed publication (Webber et al., 2023). Autovocoder successfully demonstrated how applying DSP methods (specifically the inverse Fourier transform) alongside machine learning techniques could yield significant computational performance improvements.

I then conclude the thesis by summarising some of the ways that the field has progressed in the time since parts of this thesis were completed, before a discussion of possible future work. In this thesis, the challenges of fast and controllable speech synthesis are tackled separately. It is left as future work to combine the two strands introduced in this thesis.

Part I

Background

Chapter 2

Speech signal processing and Fourier methods

The aim of this chapter is to introduce traditional Digital Signal Processing (DSP) techniques, with the primary purpose of introducing notation to the reader, alongside references to additional definitions of the techniques. These DSP techniques are those that apply to audio processing, with a particular emphasis on speech.

Throughout this thesis I will draw a distinction between traditional signal processing methods and machine learning methods. In recent times, the latter category of techniques has gained more prominence in speech generation, with learning-based approaches surpassing results achieved using hand-crafted algorithms. In particular, Chapters 7 and 8 aim to advance the state of the art in *neural DSP*, which is introduced in Section 4.4.4 of the following chapter. All chapters of this thesis make use of the fundamentals of digital audio production introduced in this chapter.

2.1 Digital audio

To listen to, analyse, modify, or generate audio on the computer, we first need to convert audio into a format that computers can manipulate. This process involves converting analogue sound waves into digital signals through a process called *sampling*. This happens by taking sound wave pressure values at fixed intervals, with these time intervals corresponding to the inverse of the *sample rate* (f_s). Sample rate is critical to the the amount of information retained in a signal. Frequencies above $f_s/2$ cannot be represented in a digital signal, and attempting to do so leads to *aliasing*. This maximum frequency is known as the *Nyquist limit*. There are two main approaches to

overcoming this issue. The first is to set the sample rate at more than twice the audible range for human ears, using so-called *full band* audio. This typically uses a sample rate of 44.1 kHz or 48 kHz. It is also necessary to use low-pass filtering to remove frequencies over the Nyquist limit. Lower sample rate audio has the advantage of typically requiring less compute to generate and requiring less space in computer memory.

Each sample is assigned a number of bits to represent the possible values of a time-varying signal at a specific time. The nature of this quantisation is not important for this thesis. In addition, there are a number of ways to compress and store audio, in both lossy and lossless ways. These methods are called *codecs*, and are discussed in Section 8.7.

Most of the work introduced in this thesis involves machine learning in some form, and therefore makes use of speech *datasets*. For speech processing tasks I will refer to each unit of audio within a dataset as an *utterance*. These utterances do not have a strict definition in terms of duration, but are generally around one sentence. In all the datasets encountered in this thesis, an utterance will contain speech only from one speaker, although some of the datasets contain utterances from multiple speakers. Collecting, annotating (with text or speaker information) and segmenting speech into datasets are important parts of speech processing that fall outside the scope of this thesis. Specific datasets will be introduced as they are used.

2.2 The frequency domain

Throughout this thesis, I will make use of representations of audio in both the time and frequency domains. The definitions and notation introduced in this section largely follow Smith (2007b). Representing audio in the frequency domain can have a number of advantages. It makes certain manipulations easier. Also, the perception of sound is tied to the frequencies of sound waves.

To transform a time-domain digital signal into the frequency domain we use the Discrete Fourier Transform (DFT) which is defined along with its inverse (iDFT) as

$$X(\omega_k) = \sum_{n=0}^{N-1} x(n) e^{-i\omega_k n}, \qquad k = 0, 1, 2, \ldots, N-1, \tag{2.1}$$

and

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(\omega_k) e^{i\omega_k n}, \qquad n = 0, 1, 2, \ldots, N-1. \tag{2.2}$$

where N is the total number of samples in a signal under analysis, ω is the angular frequency, and e and i are Euler's number and the imaginary unit respectively. Where we use a lower case letter (here x) to define our signal as the function of a sample number (n), we use its upper case equivalent (X) to represent its *spectrum* as a function of angular frequency. Angular frequency is defined as

$$\omega_k = \frac{2\pi k f_s}{N} \tag{2.3}$$

and is related to frequency by

$$\omega_k = 2\pi f_k. \tag{2.4}$$

This shows that, in the general case, the spectrum of a signal will have N values, which correspond to frequencies between 0 and $\frac{(N-1)f_s}{N}$. The work in this thesis is focused on audio signals, which are real in the time domain. The spectrum of a real signal is complex and can be parameterised in two ways — in *Cartesian* or *polar* form. In the case of the latter, a complex value is represented as a *magnitude* and *phase angle*. Although real signals result in complex spectra, such spectra will be *Hermitian*, meaning the magnitude of the spectrum is symmetric around the origin, and the phase angles are antisymmetric around the origin. A fuller proof and exposition of this is given by Smith (2007b). This symmetry and redundancy means that a real signal of length N can be represented in the frequency domain by $N/2+1$ complex values in the frequency domain, where the extra complex number represents ω_0, which is sometimes known as the *D.C. offset*. This thesis does not address complex time domain signals, although this class of signal does sometimes emerge during techniques to generate audio, such as the Griffim-Lim algorithm (discussed in Section 4.4.1.1).

The spectrum as introduced so far will be described as a complex, *linear* spectrum throughout this thesis. A common transformation to apply to the linear spectrum is to use *mel scaling* to reduce the dimensionality of the spectrum. This mel scaling involves transforming the spectrum by pooling linear frequency values into a smaller number of bins, with the range and location of each bin determined by a logarithmic scale that corresponds to perceptual interpretation of pitch. The formula for converting a frequency to its equivalent *mel value* is given by O'Shaughnessy (1987) as

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right). \tag{2.5}$$

It is possible to convert a linear spectrum into a mel spectrum by constructing a mel

basis matrix which maps linear frequencies to their equivalent mel value with linearly spaced ranges in the mel domain defining *bins*. This process is not reversible, but an approximate least-squares solution can be found from the Moore-Penrose (Lawson and Hanson, 1995) pseudo-inverse of the mel basis matrix.

So far, the frequency domain has been described as a being applied to the entirety of an audio signal at once. By doing this we fail to interpretably represent the time-varying nature of audio signals. It is this time-varying nature that allows us to communicate information through speech. It is therefore necessary to use the short-time Fourier Transform (STFT), as introduced by (Allen and Rabiner, 1977). This applies a windowing function to an input signal, cutting the signal into overlapping frames of length M, where the window is only non-zero for M consecutive samples at each hop. The STFT is given by

$$X_m(\omega) = \sum_{n=-\infty}^{\infty} x(n)w(n-mR)e^{-i\omega n}, \tag{2.6}$$

where R is the *hop size*, or the distance in samples between the start of each frame, and w is a window function. The details of window functions are outside the scope of this thesis, but a Hamming window is typically used to avoid spectral artefacts emerging from the edges of windows. The output of the STFT is therefore a time-ordered sequence of spectra that evolve with the underlying time domain signal. The STFT is an easy way to visualise the changing nature of audio signals. It is this diagramatic nature of the STFT that leads it to be known as a *spectrogram*. The STFT is, like the spectrum, complex. However, it is common to plot only the magnitude of the spectrogram in order to visualise it. This form will be described as a *magnitude spectrogram* throughout this thesis. Human auditory perception is approximately on a log-scale in terms of perception of amplitude. It is therefore also common to apply the log to the magnitude spectrogram, giving the log-magnitude spectrogram. Mel scaling is commonly applied to the log of the magnitude (log-magnitude) of the spectrogram, so representations defined in this thesis as *mel spectrograms* can be assumed to be log-magnitude.

A set of fast and efficiently parallelisable algorithms exists for the application of the DFT and iDFT to signals. These are collectively known as the Fast Fourier Transform (FFT) and its inverse (iFFT). The FFT is not discussed in detail in this thesis, although it does facilitate widespread use of the DFT and STFT in performance sensitive settings. It also affects which window sizes are selected. I typically use window sizes that are a power of 2, for which the FFT performs well, although some FFT algorithms

exhibit good performance with other values (Frigo and Johnson, 2005).

The use of spectral features is widespread for a number of reasons. The human auditory system interprets sounds by performing a sort of analogue version of frequency analysis, meaning representations in the frequency domain can be used to understand, modify, and generate perceptually relevant aspects of speech.

2.3 Digital filtering

Although modifying and generating frequency domain signals form a large part of this thesis, there exist a number of tools for doing the same in the time domain. A key approach taken to do this is *digital filtering*. Applying a digital filter involves the *convolution* of one a signal with a filter (Smith, 2007a). Due to *convolution theorem*, the application of a convolution in the time domain is equivalent to the element-wise multiplication of the spectra of the signal and filter, meaning the spectrum of a signal can be scaled arbitrarily by a well-chosen filter signal.

2.4 The human voice

The speech production system in the human body is a periodic sound source-filter system, where the glottis generates a source signal at a given frequency, before the vocal tract and lips apply transformations to the signal. The *fundamental frequency* (F_0) of a speech signal is given by the lowest frequency at which the glottis is vibrating with at a given time. The signal generated by the glottis can be approximated as a *pulse train* signal. Because it is not sinusoidal (like the basis function of the Fourier transform), the glottal excitation wave has positive amplitude at frequencies other than F_0. The strongest of these frequences are called *harmonics*, and can be denoted as $H_1, H_2, \ldots$. Systems, including speech, are described as *harmonic* when the harmonics are exact multiples of F_0. In addition to these peaks we can describe *formants*. Formants are given to the speech signal by speakers manipulating their vocal tract, and are essential to our perception of language. Formants are defined by the *spectral envelope* of a signal, which is the frequency envelope with the harmonics smoothed over. These are caused by the resonances of the vocal tract. Formants F_1 and F_2 are given by the first and second peak in this spectral envelope after F_0. A detailed analysis of these as they relate to vowel production is given in Section 5.6.

Chapter 3

Machine learning, controllability and generative models

This section gives a background to the machine learning techniques used in the rest of the thesis. The primary goal is to define terminology and provide citations to more detailed descriptions. Special emphasis is given to some techniques, such as diffusion and flow matching models, that are not necessarily particularly heavily used this thesis, but are less standard in the field and are perhaps less likely to be known to the reader.

Machine learning algorithms, as used in this thesis, differ from other computational methods in that they have *model parameters* that are adjusted during a *training* process. This training occurs with the aim of optimising model parameters to minimise some *loss function* or to maximise some kind of desired outcome on a set of training data, with the hope that these learned parameters will enable the model to perform well when it is tested or used on a separate set later. There are a range of *traditional machine learning* algorithms that exist, such as decision trees or Hidden Markov Models (HMMs) which have, at least until the last decade, been widely used in speech generation tasks (Watts, 2012). This thesis is written at a time where speech technology, and many other areas within the field of informatics and language processing, have become reliant on a range of *deep learning* techniques, for which this chapter aims to provide background.

3.1 Deep learning

Deep learning uses many layers of computational transformations applied in sequence to some input. Between these layers a non-linear function called an *activation function*

is applied. The output of the final layer is optimised during training. It is also possible to use (and optimise) the output of intermediate layers of the networks, whose outputs we can call *intermediate representations*. The use of intermediate representations is important for much of the work done in this thesis. The ability to learn representations that are useful for tasks that are not directly quantified by the output loss function is a particular strength of deep learning approaches. Deep networks are sometimes referred to as *neural networks*. This is because they were initially assumed to mimic the functioning of neurons in the brain. This nomenclature seems to be slightly falling out of fashion. I make no particular claim about the similarity of deep networks to neurons, although the usage of *neural* to designate deep learning is so widespread in some cases that I repeat it in this thesis. For example, *neural vocoders*, introduced in Section 4.4, can be assumed to be vocoders that make use of deep learning, as opposed to more traditional DSP-based vocoders.

The constituent layers of the deep networks used throughout this thesis are mostly in widespread use and need no particular introduction in this chapter. The most ubiquitous of these is the *fully connected* or *linear* layer. This is a simple affine transformation using a matrix with learnable weights. The sizes of the input and outputs of such layers must be defined ahead of training. Another typical layer is the *convolutional* network. The work described in this thesis often takes spectrograms as input and outputs, which are two dimensional. It is possible to use 2d convolutions on these, as with images. However it is also common to use 1d convolutions, convolving along the time dimension and treating the frequency dimension as channels. This avoids the assumption that higher and lower frequencies can be modelled using similar convolution kernels. Using a convolution with a width and stride of 1 is equivalent to applying a fully connected layer to each time step of the spectrogram, and this technique is applied many times in this thesis. Convolutional networks can be applied to sequences of arbitrary length but the number of channels must be fixed before training.

As well as these basic building-block layers, there are many more sophisticated composite networks that are used in this thesis. Recurrent Neural Networks (RNNs) are sequence-processing networks that processes sequences step-by-step, encorporating previous *hidden states*, which are learned representations, in computing their output. Modern versions of recurrent networks, such as LSTMs or Gated Recurrent Units, extend RNNs by adding components that better model long-term dependencies. These types of recurrent network gained widespread use in tasks such as machine translation. For this task, it was important to be able to map inputs of arbitrary length to

outputs of arbitrary length, in order to model cases where the ordering of words in a source language differs from that of the target language. For these cases true *sequence-to-sequence* models are used. These are typically encoder-decoder models where the input sequence is reduced to a fixed dimensionality, while the decoder generates an arbitrary length sequence from the fixed width encoded representation. *Attention* was introduced by Sutskever et al. (2014), which summarises encoder output by learnably *attending* to specific timesteps of the input by computing a weighted sum of encoder timestep outputs, with weights that are a function of learned network. This work was further developed into *Transformers* by Vaswani et al. (2017), which use self-attention in a way which is now ubiquitous in the many areas of research that require the use of sequence processing.

Recurrent and Transformer sequence processing networks, as described above, can be used in a variety of ways. As well as the sequence-to-sequence encoder-decoder setup, it is also possible to map from one sequence to another of the same length by applying a recurrent network or Transformer. In the case of a Transformer, this is known as an *encoder only* Transformer. It is necessary to disambiguate this use case of the Transformer, as Transformer as a term was initially applied to encoder-decoder systems. In the case of recurrent networks, it is assumed in this thesis that the default usage is for what I will call *sequence processing*, where the input and output have similar lengths and there is a strong semantic link between equivalent timesteps in the input and output. A typical example where these assumptions hold true is in speech enhancement, where the output is the same speech as the input, with noise or artefacts removed. It is sometimes necessary to summarise sequences into fixed dimensions other than for sequence-to-sequence models, for example classifying utterances by speaker or accent. As well as attention-based methods, there are simpler ways to reduce the time dimension, for example taking a final output state, or using a mean to summarise over the time dimension. The these methods are described as and when they are used throughout the thesis. Another use of both convolutional networks (specifically *causal convolutions*), and the other sequence networks described above, is in *autoregressive* generative models. These are described more fully below in Section 3.2.2.

3.1.1 Optimisation

The training processes used for deep nets throughout this thesis are all a variant, usually Adam (Kingma and Ba, 2015), of mini-batch stochastic gradient descent. This involves taking pairs of inputs and outputs, computing a loss function and minimising this using the backpropagation algorithm. This backpropagation algorithm requires taking the derivative of the loss with respect to all learnable parameters in the network. It is therefore necessary that all components of our deep networks, including learnable layers, activation functions and losses are *differentiable*, at least to a reasonable approximation. The word differentiable occurs frequently in this thesis, as a range of algorithms and techniques are applied within deep models. The use of this word implies that we can use these techniques inside or between trainable layers and still backpropagate losses to train layers closer to the input.

There are a number of commonly used losses for training. The most common of these is perhaps the Mean Squared Error (MSE) loss. This is simply the mean squared difference between the output of a deep network and the *ground truth*, that is the observed output in training data associated with the given input. Similar to MSE is *L1* loss, which is the absolute difference from a target, as opposed the the square. This is slightly less punitive of larger errors. In addition to these losses which are used to predict continuous valued variables, it is also possible to use neural networks to predict a particular *class* to associate with a specific input. The most common way to do this is to construct a network that outputs the same number of real numbers as there are classes, and to optimise the network using a *categorical loss*. This categorical loss associates each of the real outputs of the network with a class, or category, and optimises the output for the correct class for the training input to be as high as possible relative to the incorrect classes. For reasons that are explained in the section below, the typical loss used in this setup is called the *negative log likelihood*, and the values generated for each of the classes by the deep network are called *logits*.

3.2 Probabilistic machine learning and generative modelling

Probabilistic machine learning is as much a perspective on machine learning as it is a particular subset of machine learning. Nevertheless, as a perspective, it is extremely useful in understanding many of the techniques used throughout this thesis. A

probabilistically-informed approach has led to the development of some of the most powerful and useful machine learning models that exist. This section aims to give a very brief definition of probabilistic models before introducing *generative models*, which are used extensively in this thesis. Excellent and thorough background on the subjects in this section are given by Murphy (2022, 2023).

It can be seen as natural that probabilities are a good framework for analysing models that are trained and evaluated on very large numbers of samples, given the frequentist interpretation of statistics. A typical use of probabilistic machine learning is in the training of classifiers. As described in the previous section, we can output a set of scores, with a score for each class. A probabilistic approach converts these scores to assigned probabilities, using a softmax function, and then trains by aiming to minimise a *negative log likelihood* score. This refers to the *likelihood* that a model explains the data in the training set. This can also be referred to as the *cross-entropy* between the predicted probability distribution across classes and the observed distribution. The log of the likelihood is used for numerical stability.

While the probabilistic interpretation of classification models comes very naturally, it is also possible to view tasks involving predicting continuous variables through this lens. A typical way to do this is to *quantise* continuous variables into bins, and then model the probability of each of these bins. Although linear or log scale binning is sometimes used for this task, there are a range of quantising methods, including sophisticated learned approaches, such as codebooks and vector quantisation (Du et al., 2022), that apply the same basic principle. Generating a probability distribution allows us a range of ways to draw samples. It is possible to select the value with highest assigned probability or to take a weighted average of the output class values, weighted by probability. In addition, we can use *sampling*, where a pseudo-random number generator used to generate values according to a predicted discrete probability distribution. An analysis of the comparative merits of these approaches as applied to TTS is given by Henter et al. (2014).

True continuous variable prediction tasks, where a real number is directly generated by a model, are sometimes called *regression* tasks. These do not escape a probabilistic analysis. Indeed, it can be shown that training with a MSE loss is equivalent to maximising the likelihood of a model where an assumption is made that the output follows a Gaussian distribution (Murphy, 2022, p. 9).

Generative modeling is an important application of some of the deep learning techniques described so far. Generative modelling can be thought of as a statistical ap-

proach to generating samples. These samples are designed to be representative of a known training set. To do this we take a training set and treat it as samples from an unknown distribution. Using these samples, and some training algorithm and model, we estimate a distribution. Sometimes this distribution only exists in theory, and the model only generates samples from it. In other cases, the deep network predicts a probability distribution explicitly, and this is then sampled from to generate output. For example, autoregressive models and variational autoencoders generate an explicit distribution, whereas for diffusion models and GANs these distributions are implicit (Murphy, 2023).

As a very simple use-case, we take the task of generating characters for a video game. To populate the height field for each person, we can train a model by constructing a histogram across height for a population of real people. We can convert this histogram into a probability distribution by applying a softmax to the counts for each bin, and then sample from this discrete probability distribution. While this is straightforward for very low dimensionality data fields, the situation is much more complex for high dimensional data such as images, text sequences, spectrograms or waveforms. For these media there are strong and complicated dependencies between values at different output coordinates. Taking the image case as an example, whether the face is towards the left or right of a generated image may have little difference to the perceived quality of an output, but has a very large bearing on individual pixel values. These dependencies pose a particular challenge when generating waveforms, as audio information is communicated through the presence of *frequencies* rather than through the particular value of individual samples. We can describe this problem of complex relations between sample values as a *curse of dimensionality*. Much of the challenge of generative modelling is in overcoming this, and allowing structures to emerge in ouput in a way that is coherent but also computationally tractable. A deeper explanation of this, particularly as it relates to speech signals, where the *frequency content* of the signal is important as opposed to what individual sample levels of the output signal, is given in Chapter 7.

Although generative models can be used to unconditionally generate samples similar to those in a dataset, it is also possible to use *conditional* generative models. This involves the use of conditioning information to generate a *conditional probability distribution*. Examples of using classes to generate particular types of image, or in the case of text-to-speech, treating input text or phoneme sequences as conditioning. It is often possible to substitute conditional generative models for typical regression tasks

instead of deep networks trained with a regression loss such as MSE or L1. The use of probabilistic generative modelling is particularly effective in *one-to-many* problems, where there are many different but correct outputs that can be associated with a particular input. A typical regression approach using an MSE loss might yield an *average* of the possible outputs. A good example of this is the task of generating speech intonations from phoneme sequences, as explored by Hodari (2022). There may be two good speech intonations for a particular sentence, but the average of the two will sound unnatural. Hodari makes use of *variational autoencoders* to overcome this problem.

3.2.1 Generative Adversarial Nets

Generative Adversarial Networks (GANs), introduced by Goodfellow et al. (2014), are used throughout much of this thesis. GAN optimisation can be thought of as a game between two players. We create two deep networks, a Generator, G and Discriminator D. The discriminator is classifier between real and fake samples. The networks are trained in turn. D is trained to *discriminate* real samples from the dataset and artificial samples from the generator, whereas G is trained to maximise the realness, as estimated by D, for generated samples. The training process can the described as adversarial because each network's parameters are optimised to minimise some cost that is at least partially defined by the negative optimisation target of the opposing network.

In more detail, the discriminator cost $J^{(D)}$, is given by

$$J^{(D)} = -\frac{1}{2}\log D(\mathbf{x}) - \frac{1}{2}\log(1 - D(G(\mathbf{z}))). \tag{3.1}$$

The losses are very close (or the same) to being opposites of each other which we call a *zero-sum minimax game*

$$J^{(G)} = -J^{(D)}. \tag{3.2}$$

Goodfellow et al. (2014) introduces the following approximation,

$$J^{(D)} = -\log(D(G(\mathbf{z}))), \tag{3.3}$$

called a "non-saturating heuristic", which has a better behaved gradient.

The generator network typically takes as input a random vector, denoted $\mathbf{z}$, which it maps to an output. It is also possible to create a *conditional* GAN, which takes conditioning information in addition to $\mathbf{z}$. GANs do not generate an explicit distribution

which is sampled from, and instead the G network directly generates samples from an implicit distribution.

There are a very large number of possible training setups and losses that have been used, including a novel approach with similarities to GANs that is introduced in Chapter 5 of this thesis. These setups and losses have been introduced in many cases to overcome some of the challenges in training GANs. Training GANs is difficult because, due to the adversarial nature, training can easily become unstable, with one network overcoming the other. When training GANs, small changes to the potency of one network can result in drastic shifts to the learning rates required for stable training. It is also very difficult to debug GAN training, as losses are not a direct measure of how well the model is performing. It is often possible to use *auxilliary* losses, for example the MSE, in addition to a discriminator, to improve stability. This approach is taken by Kong et al. (2020) with HiFiGAN, for example. It is also possible to draw a distinction between a GAN and a training setup that makes use of an *adversarial loss*. In the latter case, there is no random vector $\mathbf{z}$ given as input to the generator. In this case the adversarial loss can be used to alleviate some of the averaging effects of regression losses. These averaging effects can be described as *oversmoothing artefacts*. In the case where $\mathbf{z}$ is not provided, the network will always provide the same output for a given input. A lack of repeatability (without controlling the pseudo-random number generator through fixed seeding) is a key defining feature of generative models.

GANs are the class of models that are used most heavily throughout this thesis. As well as the quality-enhancing anti-oversmoothing impact that they have, they also make it possible to encourage models to behave in specific ways by training a model to achieve particular outcomes as measured by a constantly updating adversary. The constant updating makes this approach more robust to trivial solutions than more typical supervised learning approaches. This is descussed in great detail throughout part II.

3.2.2 Autoregressive models

Autoregressive models dominate in text processing and are key to Large Language Models (LLMs). They are also highly important for speech synthesis. Both WaveNet (Shen et al., 2018) and Tacotron (Wang et al., 2017), are two key models that rely on autoregression. Autoregressive models typically make use of a sequence processing network, and use the previous output as an input. From this they can predict an ex-

plicit probability distribution for the current time step. The autoregressive modeling approach is an intuitive way to deal with the curse of dimensionality, in that at the time each sample is generated a good understanding of the present structure of the output is known to the model. A particular downside of autoregressive models is that sample generation is inherently sequential. High performance computers (and, in particular GPUs) deliver the most benefit when computation can be parallelised across very large numbers of processes. However, there are ways to mitigate these issues and WaveRNN (Kalchbrenner et al., 2018) and LPCNet (Valin and Skoglund, 2019) showed that waveform-level autoregressive models can still be reasonably computationally performant.

3.2.3 Diffusion and flow matching

Diffusion models, sometimes called diffusion probabilistic models, as introduced by Sohl-Dickstein et al. (2015), have taken the world of image synthesis by storm. I take the liberty of eliding a detailed description of of diffusion models, which are described elsewhere. Instead I will describe a similar class of models called *flow matching* models, which are used in section 7.3.3. Both these classes of model map from noise (sampled from a straightforward distribution) to desirable samples of, e.g., speech. To quote the memorable explanation given by Song et al. (2021), "creating noise from data is easy; creating data from noise is generative modeling." Flow matching models have been shown to be effective for speech synthesis, and the terminology and notation below matches that introduced by Mehta et al. (2024) for MatchaTTS.

Let x be an observation in a dataset, which is assumed to be sampled from a complicated unknown distribution $q(x)$. We define a *probability density path*, p_t. Note that this time variable, t, does not correspond to time in terms of position in output sequence, rather a parameterisation of a distance between a target distribution and an initial distribution. We set t to be bound to the range $[0,1]$ and set $p_0(x) = \mathcal{N}(x;0,I)$ with the aim that $p_1(x)$ will approximate $q(x)$. To do this we construct an Ordinary Differential Equation (ODE) to model the *flow* between samples drawn from our explicit distribution at $t = 0$ and our estimated distribution at $t = 1$.

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)); \qquad \phi_0(x) = x. \tag{3.4}$$

By numerically integrating through this ODE it is possible to draw samples from

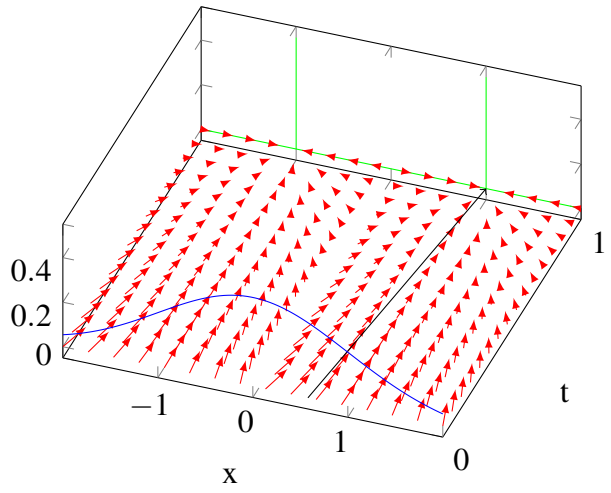


Figure 3.1: Plot showing a flow matching vector field (red) between a Gaussian probability density function (blue) with variance 1 and that of an implicit distribution (green) representing a coin toss, given by $p_1 = \frac{1}{2}(\delta(x-1)+\delta(x+1))$ with δ representing the Dirac delta function. A black arrow shows an example vector v_t which goes from a sample drawn from a Gaussian distribution (in this case 0.58) and an observed coin toss. Figure is inspired by a hand-drawn version by Gustav Henter and is reproduced here with his kind permission

our estimated distribution. Hovever, in order to do this it is necessary to have a vector field, v_t. We estimate this vector field using a deep network:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p_t(x|x_1)} \|u_t(x|x_1) - v_t(x;\theta)\|^2. \tag{3.5}$$

At each training step, the vector u_t is given by a line from a sample from the Gaussian prior to the observed sample at that training step.

To illustrate with a very basic example, we take the case of a fair coin, with heads denoted by 1, and tails -1. Figure 3.1 shows the implicit prior in blue, along with an implicit distribution (unknown at training time) in green. An example flow field, representing the function u which we aim to learn is shown in red. An example v_t vector is given in black. At training time, a random value of t is selected, and the MSE difference between a vector predicted by a deep network, u_t and v_t is minimised. It is possible to numerically integrate over t in the range $[0,1]$ at inference time to generate samples. This numerical integration can involve a number of steps through t of our choosing. By doing this, we iteratively arrive at a sample from our implicit distribution. While the coin toss example can be thought of as zero dimensional, the

flow matching technique generalises well to high dimensional data. A case where a similar but high dimensional distribution is estimated is explored in Section 7.3.3. Note that due to the nature of ODEs, paths generated during this integration process cannot overlap. It is important that $p_0(x)$ has the same dimensionality as the desired output of the generative model, in the case of MatchaTTS this is 80 mel spectrogram features. Importantly, the values sampled from the Gaussian at p_0 are independent, whereas the output samples are highly self-correlated (due to the nature of mel spectrograms). This means that the structure, and these correlations, emerge during the iterative process, where at each timestep of t, and for each sample value, the deep network has a full view of *all* samples at the previous timestep. This allows diffusion and flow matching models generate complicated outputs.

3.3 A note on computational performance

Computational performance is a key part of this thesis. There is a trend followed by some machine learning papers where performance is expressed using the number of parameters in a model. This is not appropriate as the computational performance is highly dependent on *how* these parameters are used – both how parallelisable the system is and how many times each parameter is used. For this reason, the only measure of performance used in this thesis is the time taken to run on specific named hardware. This ties the analysis of system performance to the real world.

Chapter 4

Speech Synthesis

4.1 Introduction

Speech synthesis, as introduced in this chapter, refers to the task of generating artificial speech signals. Although the most prevalent subdomain of this is synthesising speech from a text source: text-to-speech (TTS), this is not the only application of synthesised speech. Other tasks include *speech enhancement*, where the aim is to take a speech recording and enhance it, either to remove noise or artefacts (Valin, 2017), or to make it more intelligible to people with hearing loss (Shifas et al., 2019). In *Dialogue Speech Synthesis* (DSS) speech might be synthesised in response to natural speech in order to take part in a dialogue (Nishimura et al., 2022). It is common to use TTS as a constituent component for other types of system, often alongside ASR. For example, voice conversion can be performed by applying ASR to retrieve a text representation of source speech, which can then be resynthesised. For DSS, ASR and TTS can be used either side of a text-based dialogue system. However, such approaches are problematic in that text alone omits much of the required context to generate natural-sounding speech. The challenge of synthesising speech with insufficient context is explored by Hodari (2022) and many others.

Because of the prevalence of TTS within the many applications of speech synthesis, the two terms are occasionally used synonymously. In this thesis I use the term *speech generation* to refer to any processes that synthesise speech from any representation. In this chapter, I introduce background to state-of-the-art techniques for speech generation.

4.2 A note on computational performance

As the following sections of this chapter, and the following chapters of the thesis, emphasise computational performance, it is worth defining how this is interpreted in this work. The measure of computational performance used exclusively in this thesis is how *fast* a second of audio is generated on a specific piece of computer hardware (inference) or how long a model takes to train (training). While some papers in machine learning describe performance in terms of number of parameters in a model, this is largely irrelevant. For example, convolutional net will use many more computations for each paramater than a feed-forward one. The number of floating point operations is a better metric, but still less appropriate than time. This is because some models are more amenable to parallelisation than others, meaning that modern hardware can run them faster, even if they require more operations. There are many other metrics that could be used, including memory and electrical power consumption, but these typically correlate with time performance and go beyond the scope of this thesis.

4.3 Text-to-speech

A range of techniques were applied to TTS prior to the emergence of deep learning, including unit selection (Taylor et al., 1998) and Hidden Markov Models (HMMs) (Zen et al., 2009). This background introduces TTS starting with the application of deep sequence models to TTS. TTS systems often take as input a *linguistic specification*, which may not be in the typical written form of a language, which is sequences of *graphemes*. The conversion of sequences of graphemes into a more appropriate linguistic specification (typically converting graphemes to *phonemes*) is not covered in this thesis, but was studied in detail by Fong (2024).

WaveNet, from van den Oord et al. (2016), is regarded as the first successful use of deep networks to directly synthesise waveforms. WaveNet operates by treating each waveform sample as a conditional probability distribution. This means that WaveNet predicts a discrete probability distribution conditioned not only on externally provided conditioning information, $\mathbf{h}$, but the result of all prior output steps, using autoregression as introduced in Section 3.2.2. In the case of WaveNet as applied to TTS, $\mathbf{h}$ is a linguistic specification (passed through a convolutional conditioning network) and the probability at each output timestep, t, is given by

$$p(\mathbf{x} \mid \mathbf{h}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}, \mathbf{h}). \tag{4.1}$$

WaveNet makes use of *dilated causal convolutions* as shown in Figure 4.1, in order to process a sequence of input. The input to these causal convolutions requires some work to attain in the case of TTS, as at each input timestep WaveNet receives not only a linguistic representation but also F_0. External models are therefore required to predict, from a given text input, the duration of each phoneme and the value of F_0. Phoneme sequences are then upsampled by repetition to the required duration.

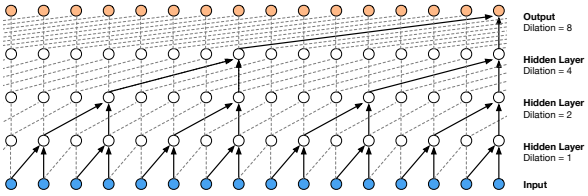


Figure 4.1: Figure showing dilated causal convolutions from van den Oord et al. (2016).

WaveNet showed much improved performance over prior state-of-the-art models. It came with the downside of being extremely computationally expensive. This was somewhat improved by the introduction by van den Oord et al. (2018) of Parallel WaveNet. WaveRNN from Kalchbrenner et al. (2018) improved things yet further, with the replacement of dilated causal convolutions with an RNN and a number of computational tricks applied to increase performance. However, the direct autoregressive modelling of waveforms has proven inherently expensive computationally.

To overcome this, and for many other reasons, many state-of-the-art models apply a multipart approach. The typical separation involves using an *acoustic model* and a *vocoder*. The acoustic model converts a sequence of linguistic representations (which can be graphemes *or* phonemes) into a sequence of *acoustic representations*. Where phonemes are used, this requires the use of a *front end* system to generate a sequence of phonemes. The acoustic representation is most often a mel spectrogram, although some vocoders receive other representations.

A key work in acoustic modelling was the Tacotron model from Wang et al. (2017). In this paper, character embeddings are used as input. These are converted into a sequence of acoustic representations using a sequence-to-sequence model that uses RNNs and attention. The output of Tacotron is a linear spectrogram, which is converted into a waveform using Griffin-Lim, an algorithm which is described in detail in Section

4.4.1.1. A diagram of Tacotron is given in Figure 4.2.

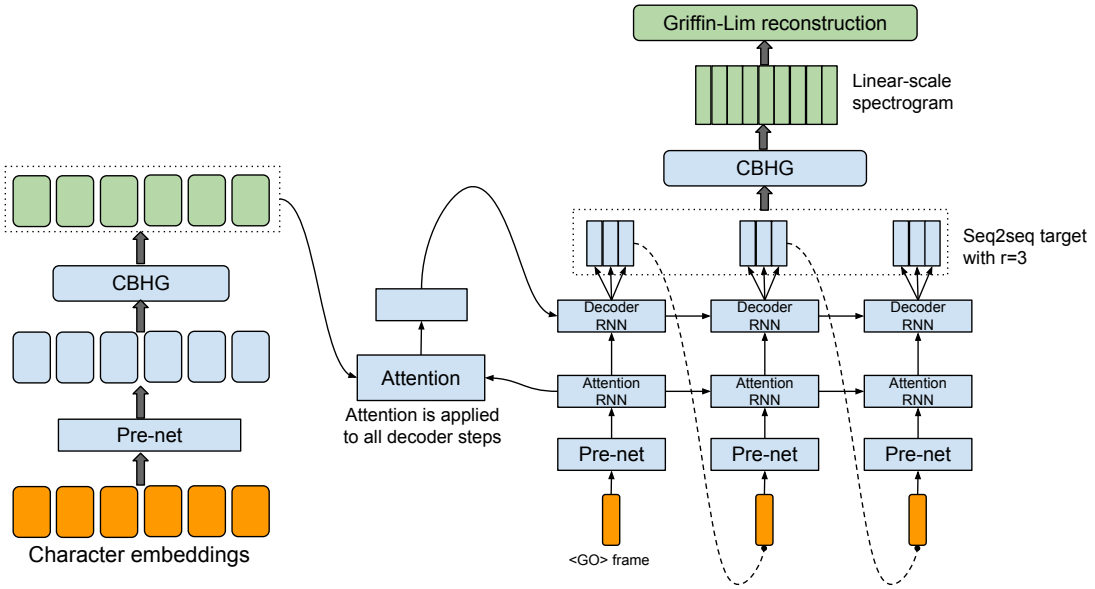


Figure 4.2: The Tacotron model architecture repreduced from Wang et al. (2017). The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesise speech.

Tacotron 2 was introduced by Shen et al. (2018) in order to improve output quality by removing the dependency on Griffin-Lim. Instead, a version of WaveNet is used that is conditioned on mel spectrograms. This *WaveNet Vocoder* is discussed in detail in Section 4.4.2.

The change from autoregressively generating waveform samples to autoregressively generating spectrogram timesteps improved performance, but only up to a certain point. More recently there has been a push towards non-autoregressive acoustic models. FastSpeech, from Ren et al. (2019), is an early example, and operates by using a duration prediction model to assign each phoneme in a sequence a duration. This is used to upsample phonemes by repetition to a spectrogram hop frequency. This results in a sequence that is the same length as the output, and is broadly time-aligned with an ideal target output. FastSpeech uses an encoder-only transformer, which performs sequence processing as opposed to sequence-to-sequence modelling, according to the distinction introduced in Section 3.1. This transformer converts time-aligned phonemes into mel spectrogram frames.

Two concurrently developed models, by Łańcucki (2021) and Ren et al. (2021), introduced similar improvements to FastSpeech: introducing FastPitch and FastSpeech 2 respectively. These models improve the way the duration model is trained, and also

introduced separate models that predict F_0 and energy. These predictions are also passed into the sequence processing network.

There are also approaches that apply non-autoregressive generative modelling to generate waveforms directly from a linguistic specification, such as the VITS TTS system from Kim et al. (2021). Such end-to-end systems are not used in this thesis. The reason for this is mainly practical and economic. The computational resources available in a university setting did not allow for repeated prototyping of new systems that are as expensive to train as end-to-end systems, which are necessarily more complicated than individual components. Although Kim et al. (2021) made significant progress towards more computationally efficient end-to-end systems, these systems are still much slower to train on equivalent hardware than those proposed in this thesis.

4.4 Vocoding and waveform synthesis

The separation of speech synthesis into the tasks of acoustic modelling and vocoding has some advantages. The use of an acoustic representation allows the development of two components to be tackled separately, where the acoustic representation is used analagously to intermediate representations in compilers. A new improved vocoder model can be effortlessly switched in without the expensive process of training a full system. Dividing systems can also make them easier to debug. However, the use of a fixed DSP-derived representation, typically a spectrogram, can cause issues. This is because DSP-derived representations are usually lossy, omitting information from the source signal. For example, magnitude spectrograms omit phase information that can be difficult to reconstruct. This problem, and a suggested new framework for vocoding that avoids these issues, are introduced in Chapter 8. The following subsections outline a taxonomy of vocoder approaches.

4.4.1 DSP-based waveform generation

Many TTS systems, from HMM synthesis until recent models, have made use of purely DSP-based vocoding modules. Early versions included STRAIGHT from Kawahara et al. (1999) and WORLD from Morise et al. (2016), which use a *smoothed spectrogram* alongside energy, F_0 and apperiodicity as input features. Although WORLD will be used in Chapter 5, these systems are no longer in widespread use for speech synthesis.

4.4.1.1 Griffin-Lim algorithm

The Griffin-Lim algorithm (GLA), from Griffin and Lim (1984), is a commonly-used DSP-based approach to convert linear magnitude spectrograms into waveforms. Because it depends only on iterative application of the discrete Fourier transform, which can be accelerated using the FFT, Griffin-Lim can be performed very efficiently and is highly parallelisable.

GLA takes a magnitude only spectrogram and estimates a phase for it, making use of redundancy in the overlaps of the STFT windows. To start with, the algorithm initialises a choice of phase, θ_0. This can be done using random values. If the magnitude spectrogram is based on a source spectrogram that has been modified, it may be beneficial to use the source phase as an initial phase. GLA aims to derive a 1d waveform signal, $\mathbf{x}$, by obtaining its complex spectrogram, $\mathbf{X}$ from its magnitude-only spectrogram, $\mathbf{r}$. The magnitude spectrum is combined with an initial phase, and the iSTFT transform is applied to get an initial time-domain signal, as follows

$$\mathbf{x_0} = \Re(\mathcal{F}^{-1}\{\mathbf{r}e^{i\theta_0}\}) \tag{4.2}$$

where $\Re$ denotes taking the real part of the signal and $\mathcal{F}^{-1}$ denotes the iSTFT. The next phase iteration is calculated by taking the phase angle of the STFT of the initial time domain estimate

$$\theta_1 = \angle\mathcal{F}\{\mathbf{x_0}\}. \tag{4.3}$$

This process is repeated iteratively according to the algorithm given in Algorithm 1. Griffin-Lim thus achieves waveform synthesis through *phase prediction*, meaning that the acoustic model does not need to model phase. Although some approaches, such as that of Espic et al. (2017), have used acoustic models to directly predict phase, this a challenging task and is not common.

input : $\mathbf{r}$, θ_0
$\mathbf{X}_0 \leftarrow \mathbf{r}e^{i\theta_0}$;
for $n \leftarrow 1$ **to** N **do**
  $\theta_n \leftarrow \angle\mathcal{F}\{\Re(\mathcal{F}^{-1}\{\mathbf{r}e^{i\theta_{n-1}}\})$;
end
return $\mathcal{F}^{-1}\{\mathbf{r}e^{i\theta_N}\}$;

Algorithm 1: Griffin-Lim algorithm

Griffin-Lim was extended by Perraudin et al. (2013) to converge to better results

faster. This made use of *momentum*, which is the process of using the difference between previous iteration results to guide future progress of the iterative algorithm. A more detailed analysis of momentum is given in Chapter 7.

4.4.2 Autoregressive vocoders

Although Griffin-Lim offers excellent computational performance, its output is exceeded in quality by the output of modern *neural vocoders*. These leverage deep learning to convert acoustic representations into waveforms. The first of these to be truly successful was the use of WaveNet as a vocoder, which was introduced alongside Tacotron 2 by Shen et al. (2018), and is discussed in Section 4.3. The WaveNet vocoder differs from the initial WaveNet introduced by van den Oord et al. (2016) in that it is conditioned on mel spectrograms rather than text.

Mel spectrograms are ubiquitous in modern speech synthesisers as acoustic representations, in contrast to GLA, which performs waveform synthesis from linear magnitude spectrograms. GLA does this through explicit phase prediction. The use of mel spectrograms also necessitates the use of upsampling in frequency. This can be thought of as the process of reversing the mel scaling by upsampling to a linear spectrogram, or as upsampling along the time axis to waveform sample rate along with reducing the frequency dimension. For many neural vocoders, the phase prediction task is *implicit*, where the output is simply a waveform and phase is not directly predicted. Nevertheless, phase is present in the output waveform but not in the input mel spectrogram. As discussed in Section 4.3, WaveRNN was also applied by Kalchbrenner et al. (2018) to the vocoding task, using mel spectrograms as input, yielding some computational improvements over WaveNet.

Other classes of model, such as LPCNet from Valin and Skoglund (2019), have accelerated the autoregressive vocoding process by incorporating *differentiable DSP* techniques. This reduces the number of computational operations required at each autoregressive step. The application of such strategies will be discussed in Section 4.4.4. Nevertheless, autoregressive models are inherently limited by the lack of parallelisability. This lack of parallelisability becomes increasingly limiting as hardware becomes more parallel, both in terms of the powerful GPUs used for training, and the *neural processing units* (NPUs) that are becoming available to consumers. An efficient autoregressive model, even one that uses few computational operations per time step, becomes slower relative to a more powerful model that can be fully parallelised as this

hardware develops.

4.4.3 Non-autoregressive generative vocoders

A large number of neural vocoder models overcome the performance issues associated with autoregression by using classes of generative model that are not autoregressive. Generative models are useful for "filling the gaps" between a mel spectrogram and a waveform. Generative models, which were introduced more thoroughly in Section 3.2, are practical for *one-to-many* problems, such as those where a waveform is incompletely specified by a mel spectrogram, which does not contain phase or full linear magnitude spectrogram information.

Although some approaches, such as from Yamamoto et al. (2019), have made use of *knowledge distillation* (Hinton et al., 2015) in order to convert the training of WaveNet into a smaller non-autoregressive model, the current state-of-the-art seems to be direct application of non-autoregressive modelling to the task of converting mel spectrograms to waveforms. WaveGlow, from Prenger et al. (2019), omits distillation and directly trains a Glow-based model. Another particularly important model of this genre is HiFiGAN, introduced by Kong et al. (2020). This achieved significant computational performance improvements compared with WaveNet or WaveGlow. HiFiGAN overcomes some of the instabilities in training GANs by using auxiliary losses. These include a mel spectrogram loss, where the mel spectrogram of the output waveform is differentiably calculated and compared with ground truth. HiFiGAN is discussed in more detail, then compared to a model introduced in this thesis, in Chapter 8.

4.4.4 Neural DSP

The previous section introduced the idea of using *differentiable DSP* (the calculation of a mel spectrogram in HiFiGAN) in order to calculate a training loss. It is also possible to use such techniques within models that are otherwise trained using deep learning. Many neural vocoders have taken this approach, creating what I will refer to as *neural DSP vocoders*.

A first key work in this field is from Airaksinen et al. (2016), who introduced GlotDNN. GlotDNN was published contemporaneously with WaveNet and uses similar features to earlier STRAIGHT and WORLD vocoders. It is perhaps best thought of as using deep learning to enhance these traditional vocoders rather than using differentiable DSP to improve performance of a neural vocoder.

LPCNet, from Valin and Skoglund (2019), extends WaveRNN by using an RNN to predict LPC coefficients, achieving significant speedup. Kons et al. (2019) successfully applied this to TTS, and LPCNet was further extended by Valin et al. (2022). This latter extension introduces an important distinction into our taxonomy of neural DSP vocoders, by using a fully differentiable implementation of LPC, whereas prior versions used deep networks to predict features that are then used by a DSP system. Vocoders in which differentiable DSP components are integrated fully into deep networks can be thought of as true neural DSP vocoders.

Another true neural vocoder is GELP from Juvela et al. (2019), which, unlike LPCNet which uses some pitch-derived features, predicts waveforms from only a mel spectrogram. This model uses a GAN to predict the synthesis filter coefficients. Interestingly, GELP applies filtering in the STFT domain, using the convolution theorem. This allows the application of the digital filter to be parallelised.

Another suite of models that have used DSP and deep learning together for vocoding are the Neural Source-Filter (NSF) models from Wang et al. (2020). As with LPCNet, these use pitch-derived features as input and have been updated more recently (Wang and Yamagishi, 2020) to become true neural DSP vocoders.

Whereas the models discussed so far use DSP in the time domain (i.e. filtering), another class of model uses the iSTFT, i.e. DSP in the frequency domain, to directly synthesise waveforms. An example of this is iSTFTNet from Kaneko et al. (2022). This enhances the computational performance of HiFiGAN by replacing some of the generator layers with an iSTFT. However, this iSTFT uses a very small window size of just 8 samples, which limits the computational upside. iSTFTNet2 improves this with the use of 2d convolutions, which increases output quality. The iSTFTNet models were developed contemporaneously with and independently from *autovocoder* (Webber et al., 2023), which is introduced in Chapter 8 and also makes use of the iSTFT.

4.5 Evaluation

Evaluation is a key part of speech synthesis research. Unlike ASR, which heavily uses Word Error Rate (WER) as a key performance metric, there are no immediately obvious objective measures for evaluating the quality of synthesised speech.

4.5.1 Subjective evaluation

The basis for subjective evaluation of speech synthesis is simple: get people to listen to it and say if it is good. A number of refinements to this approach have been developed. For TTS, it is typical to recruit paid participants who are instructed to score samples based on how *natural* the speech sounds. It is good practice to require participants to be L1 speakers of the language that the test is being conducted in, unless the intelligibility to other speakers is being deliberately tested. A common approach is to use Mean Opinion Scores (MOS), where participants are instructed to score each sample on a 1-5 scale. This has the drawback of being quite variable depending on the participant, as one person's view of what constitutes a '5' may differ from another's. MUSHRA (ITU-R, 2015) is a more sophisticated approach that makes use of anchor and reference samples to scale scores between 0 and 100. Listening tests will be described in detail at the points where they are used throughout this thesis. Participants in listening tests are paid a reasonable amount for their time, and experiments in this thesis involving paid human participants were cleared using the University of Edinburgh Informatics ethics approval process.

While the previous paragraph describes what I will refer to as *formal listening tests* there is another, equally important type of subjective evaluation. I will refer to this as the *informal listening test*, and it consists of an experimenter listening to samples as they emerge from models during training. Many approaches can be instantly disregarded by listening to samples that very clearly sound bad, and where formal evaluations would be wasteful. There are many such cases described in this thesis.

4.5.2 Objective evaluation

Many attempts have been made to automate the formal listening test through the use of automatic metrics. These are referred to in this thesis as *objective* analysis. An easily used metric is the WER after synthesised speech is passed through an ASR system. This is not a particularly useful metric for TTS speech, as intelligibility is a very low bar for modern TTS systems. Measures of naturalness are more difficult, but recent attempts include using deep networks to predict MOS based on a labelled training set. MOSNet, from Lo et al. (2019), is a typical example of this approach. Another approach is the Fréchet Audio Distance (FAD), introduced by Kilgour et al. (2019), which is more typically applied to non-speech audio tasks (Pascual et al., 2022). FAD works by extracting intermediate representations from an audio classifier, and calculat-

ing the difference between the average and variance of the representations at this layer from ground truth samples to the desired category of audio. Other objective metrics are available, such as emotion detection (Tomashenko et al., 2022a), and these will be introduced at the points where they are used.

4.5.2.1 PESQ

PESQ is a signal-processing derived scoring system for audio quality that is used in this thesis, particularly in Chapter 7. It is a sample-based metric that performs time alignment and aims to predict a score that corresponds to the MOS 1-5 scale (Rix et al., 2001).

4.6 Summary

This chapter introduces background on speech synthesis techniques. Future chapters develop on this work. In particular, Autovocoder (Chapter 8) aims to improve on neural DSP methods introduced in this chapter by reversing the conventional arrangement of DSP and machine learning, where fast signal processing is used for the one-off task of encoding waveforms into acoustic representations (mel spectrograms), forcing deployed vocoders to use slow, difficult to parallelise, and computationally expensive models to generate from these features every time a waveform is synthesised.

Part II

Controllable speech synthesis

Chapter 5

Hider-Finder-Combiner

At the time the work in this chapter was started, 2019, it was typical for speech synthesisers to be separated into two main components: a sequence-to-sequence model converts sequences of graphemes or phonemes into sequences of frames in the frequency domain, and a neural vocoder generates a waveform (Kons et al., 2019). These systems yield high quality results, but are limited to synthesis of voices for which audio exists. This is true even when using *adaptation*, by injecting high-level information (speaker identity (Doddipatla et al., 2017), speaking style (Wang et al., 2018), expressivity, etc.) into the model. Whilst these systems had impressive *adaptation* ability (Jia et al., 2018), they are driven by data and generally offer no explicit *control* of speech parameters such as pitch or timbre. Some neural vocoders (Wu et al., 2019; Valin and Skoglund, 2019) accepted an explicit pitch parameter as input but these did not achieve the highest levels of speech quality (Govalkar et al., 2019).

Using signal processing, manipulation of arbitrary speech parameters is challenging, not least because speech parameters co-vary. However, using signal processing simply to extract parameters – to *annotate* waveforms – is far less challenging.

The limitations of current neural approaches, and the relative ease of annotating speech compared to signal processing manipulation, together motivate the approach presented in this chapter.

The goal in this work is true *controllability* of speech. This is performed in the mel spectrogram domain because of its ability to represent most aspects of speech and its widespread use as the interface between sequence-to-sequence models for speech synthesis neural vocoders (Lorenzo-Trueba et al., 2019; Kalchbrenner et al., 2018).

The experiments in this chapter will be conducted by first using F_0 as the parameter under control. F_0 is a good candidate for *control parameter* because there are well

established techniques for the analysis and modification of this parameter that can be used as benchmarks. Later, the generalisability of the proposed technique will tested by the application of the system to a suite of new parameters.

A novel machine learning approach will be taken which, in theory, requires only a training dataset *annotated* with this control parameter. The method aims to modify precisely *one control parameter* whilst leaving all other aspects of the speech signal unchanged.

The techniques described in this chapter were developed in the hope that they would generalise to arbitrary control parameters. Although this chapter introduces the first attempts at generalising to new paramters, further developments and methods that yielded better results with a new parameter are described in Chapter 6.

This chapter first describes an ideal controllability, before offering a high level description of a novel adversarial deep learning system that aims to achieve it. The three component networks of this system were first trained in their respective tasks separately, using features calculated using the WORLD vocoder from Morise et al. (2016).

The work described in this chapter resulted in a publication (Webber et al., 2020). The initial section, especially the part described in Section 5.3 was written up in a more basic form (Webber, 2019) for the MScRes that formed the first year of a combined four-year Centre for Doctoral Training program. The work was developed into publishable form during my PhD work and is described more thoroughly in this thesis.

5.1 Defining controllability

What does it really mean to control certain aspects of a speech signal?

We can annotate many parameters on a speech signal; e.g., F_0 (perceived as pitch), spectral tilt (voice quality), or formant frequencies (articulation) can be estimated automatically from waveforms. We could imagine annotation of other parameters using external information (e.g., physical articulator positions) or human perception (manual labelling). In concert, certain combinations of parameters can uniquely and completely represent a speech signal.

Some combinations of parameters, for example mel spectral features, aim to summarise only the *perceptually important* features of speech, while leaving other ambiguities. There is a precedent in DSP vocoding, specifically in the WORLD and STRAIGHT vocoders (Morise et al., 2016; Kawahara et al., 1999), to generate *smoothed*

spectrograms, which contain only a spectral envelope, with harmonics removed. This also motivates the choice of F_0 as the control parameter in this chapter, as the work described here replicates this removal of information in a fully learned manner, with no hand-crafted representations used as intermediate representations.

Ideal controllability implies being able to modify one parameter while changing as few other parameters as possible. However, many parameters are deeply intertwined with others. For example, mel spectrograms depend on (at least) F_0, speaker characteristics and, orthographic content. Ideal control of F_0 means changing only that part of the mel spectrogram which is dependent on F_0. This has a large cascade of implications as many features are correlated with F_0.

5.2 Motivation and approach

The proposed system uses machine learning techniques to modify a speech signal by an arbitrary *control parameter*. Conventional deep networks can be thought of as learned *universal function approximators*. Consider the function f that maps input mel spectrogram x to output mel spectrogram z and approximate f with a network F trained to minimise error ε, where the error is a difference between a target output and an achieved output

$$\varepsilon = |f(x) - F(x)| \tag{5.1}$$

averaged over all (x,z) in some training set.

One way to attempt to control a parameter during synthesis is as an additional input to the model. Parameters that relate more to speech style could be offered as additional inputs alongside text to a TTS model. Parameters that instead relate to the speech quality could be given to the vocoder as an additional input. However, this approach has a critical flaw. If we add another control parameter y as input to the network, so that

$$z \approx F(x,y) \tag{5.2}$$

F will simply learn to ignore y if x contains sufficient information to predict z. Even if x contains only partial information, when we attempt to control the speech z by varying control parameter y we risk contradicting information contained in x. An example of this would be providing F_0 to a neural vocoder alongside a mel spectrogram. Mel spectrogram features are highly dependent on F_0, so the F_0 value can be ignored without penalising the model during training.

To solve these related problems, we employ a hidden representation h which contains as little information as possible about y, but all *other* information from x that, when combined with y, will predict z. We use a *hider* (H) and *combiner* (C) network for this, with a *finder* (D) adversary.

$$h = H(x), \tag{5.3}$$

$$z \approx C(h,y), \tag{5.4}$$

$$y \approx D(h). \tag{5.5}$$

This is similar to a *Generative Adversarial Network* (GAN), introduced by Goodfellow et al. (2014), in which a generator network aims to fool a discriminator (adversary) network into misclassifying its output, e.g., as being genuine. Generator and discriminator are trained in turn. In the proposed system, the *finder* is the discriminator. During training, the output of z of equation 5.4 is equivalent to x, where the ground truth values of y are used. This removes the need for paired data with different values of y.

5.3 Training the combiner on WORLD features

Prior to the rest of the work described in this chapter, the combiner model was trained separately using features from the WORLD vocoder (Morise et al., 2016). Please note that, although significant improvements were later made to the network architecture, this process is described in an earlier MSc thesis (Webber, 2019). This separate training of the combiner allowed this component to be tested, developed and debugged without the instabilities of GAN training causing additional difficulties. While these very early experiments made use of features that were generated by the WORLD vocoder, the training scheme described in the rest of this chapter had no dependency on the WORLD vocoder, other than labelling the dataset with pitch tracking information. The later results did not use weights that were pretrained using WORLD features.

First, instead of adversarial training, a signal processing approach was used to derive features that have information about F_0 removed. The *smoothed spectrogram* generated by the cheaptrick algorithm, introduced by Morise (2015), fulfilled this role. This smoothed spectrogram, sometimes called a spectral envelope, defines the structure of a speech signal spectrogram with STRAIGHT harmonics removed. This means it should contain no information about F_0, although some of this information may leak through.

The *combiner* network takes as input the smoothed spectrogram and a new value of the control parameter and produces as output a mel spectrogram. For these initial experiments, the mel spectrogram features were inverted into audio using the Amazon Universal Vocoder (AUV) from Lorenzo-Trueba et al. (2019), with a pretrained model kindly provided by Amazon. However, due to computation performance issues, the vocoder used for the results in the rest of this chapter made use of WaveRNN, by Kalchbrenner et al. (2018).

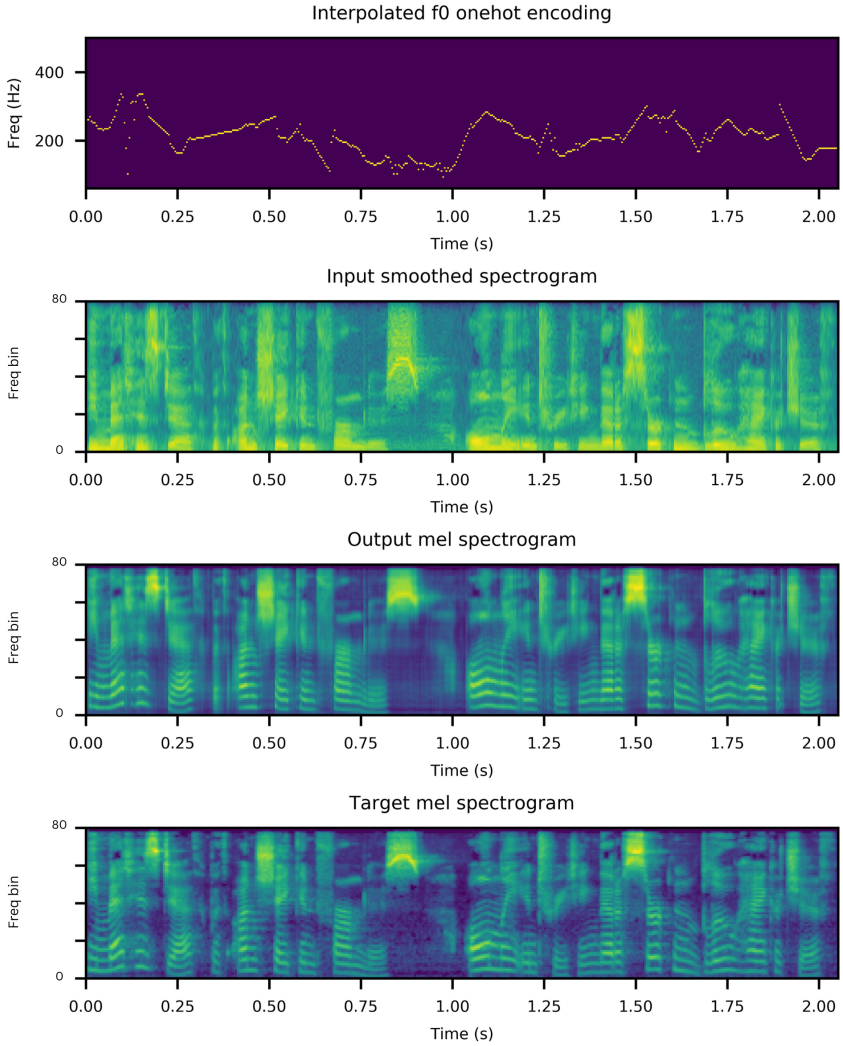


Figure 5.1: Spectrograms for the copy synthesis task - recreating a mel spectrogram from pitch and a smoothed spectrogram.

The initial combiner approach used was an RNN, specifically a gated recurrent unit (GRU) (Cho et al., 2014), that takes a smoothed spectrogram sequence, concatenated at each frame with F_0 values and aperiodicities, where aperiodicities are unpitched ele-

ments of the source signal extracted by WORLD. All of these were initially generated by the WORLD vocoder, using an open source Python interface. The aperiodicity indicates how much non-periodic noise is present in the initial audio signal. The WORLD vocoder is capable of going from audio to this trio of features and back again. However, the cycle of audio $\rightarrow$ features $\rightarrow$ audio using a signal processing based vocoding system such as WORLD will significantly degrade audio quality. Neural vocoders that were SOTA when this work was done, such as AUV or WaveRNN, are better at generating natural sounding speech audio from spectral features than WORLD (Kalchbrenner et al., 2018). The RNN was used to combine these three features, generating a mel spectrogram that the Amazon Universal Vocoder can convert into a high quality signal.

Figure 5.1 shows how similar results from the combining network can look to the reference spectrograms when performing copy synthesis, i.e. using the combiner with original pitching information. Figures 5.2 and 5.3 show that varying pitch input to the network results in credible modified outputs that on initial visual inspection clearly seem to be following input control values. To generate monotonic speech the F_0 value was set to the mean estimated F_0 value for the whole sample.

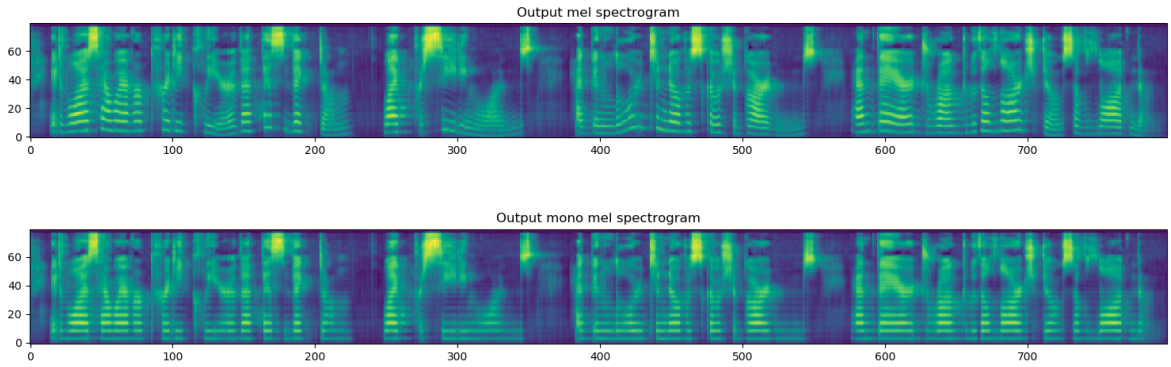


Figure 5.2: Comparing spectrograms with original estimated pitch and fixed pitch set to the mean of the original. The flat F_0 and harmonic contours in the monotonic output show that the combiner network properly obeyed input pitch instruction. The x-axis refers to time step, the y is mel scaled frequency bin.

To check the pitch of the resulting audio, the `REAPER` pitch tracker was used. This showed that the output signal was largely of fixed pitch and that unvoiced sections had been preserved. The results of this are shown in Figure 5.3.

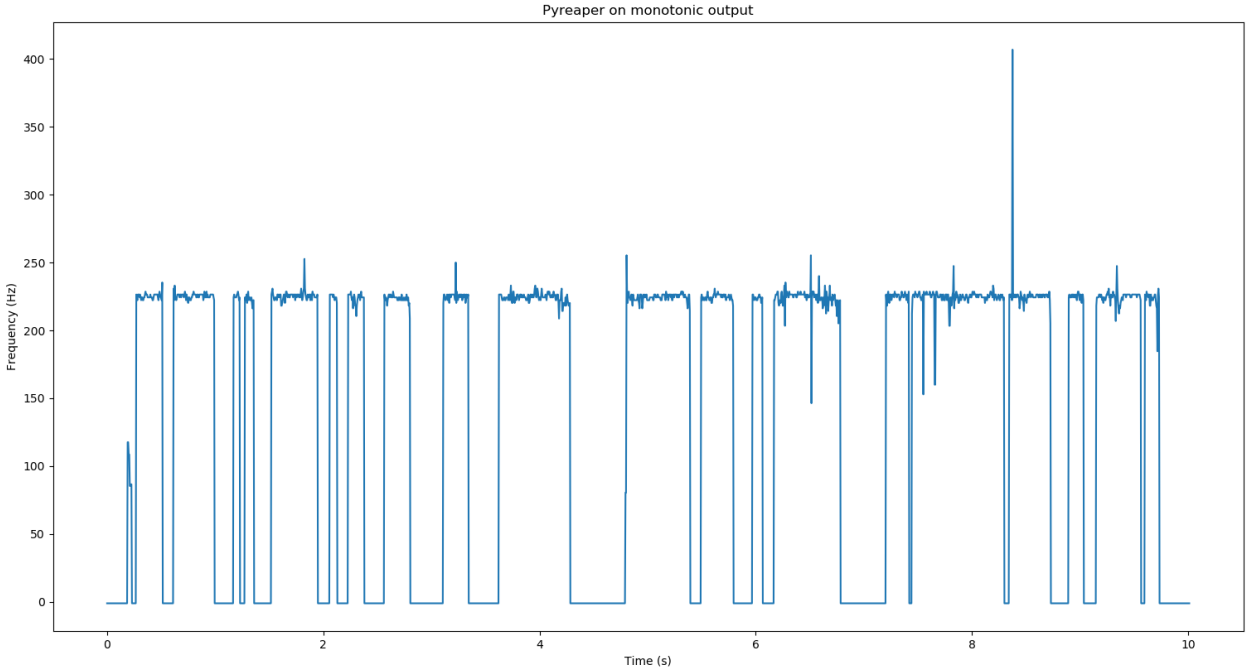


Figure 5.3: Results from a pitch tracking tool on output monotonic audio from my network and Amazon's Universal Vocoder where smoothed spectrum and monotonic F_0 values are used as input. A zero value for F_0 is used to encode unvoiced sections.

The results of the early combiner model prototypes were highly encouraging. This led to the development of an adversarial system of training that did not depend on highly engineered features such as those provided by the WORLD vocoder. A start was made on an adversarial training setup as part of the MSc thesis, but the bulk of the work in making it perform well for F_0 was done as part of this PhD.

5.4 The Hider-Finder-Combiner System

Figure 5.4 shows the architecture, and the MSc thesis (Webber, 2019) describes the earlier experiments from which this architecture emerged.

1. The *hider* takes as input a mel spectrogram and produces as output a 2-dimensional hidden embedding (one axis is time, as in the spectrogram) which preserves sufficient information for reconstructing the spectrogram but as little information about the control parameter as possible.

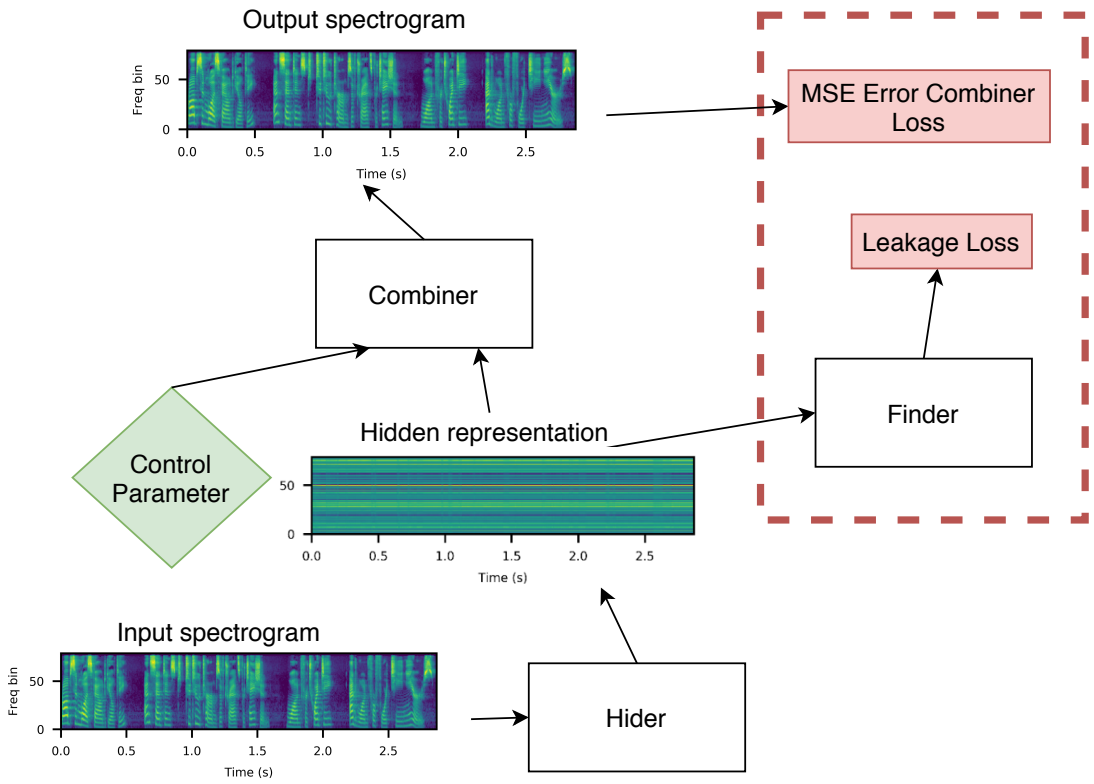


Figure 5.4: The complete architecture. Components in the dashed box are used only during training.

2. The *finder* network is an adversary that attempts to detect the value of the control parameter from the output of the *hider*. It is only used to train the system and is not required during generation.

3. The *combiner* network takes as input the hidden embedding and a new value of the control parameter and produces as output a mel spectrogram.

In the following, we use F_0 as an example control parameter. The model is trained from scratch without any supervision of the individual networks, using only ground-truth mel spectrograms paired with corresponding control parameter values; input and output spectrograms are identical in training. The goals of training are (1) when given the original value of the control parameter, the *combiner* output is the same as the *hider* input, and (2) the *finder* cannot detect the value of the control parameter from the *hider* output.

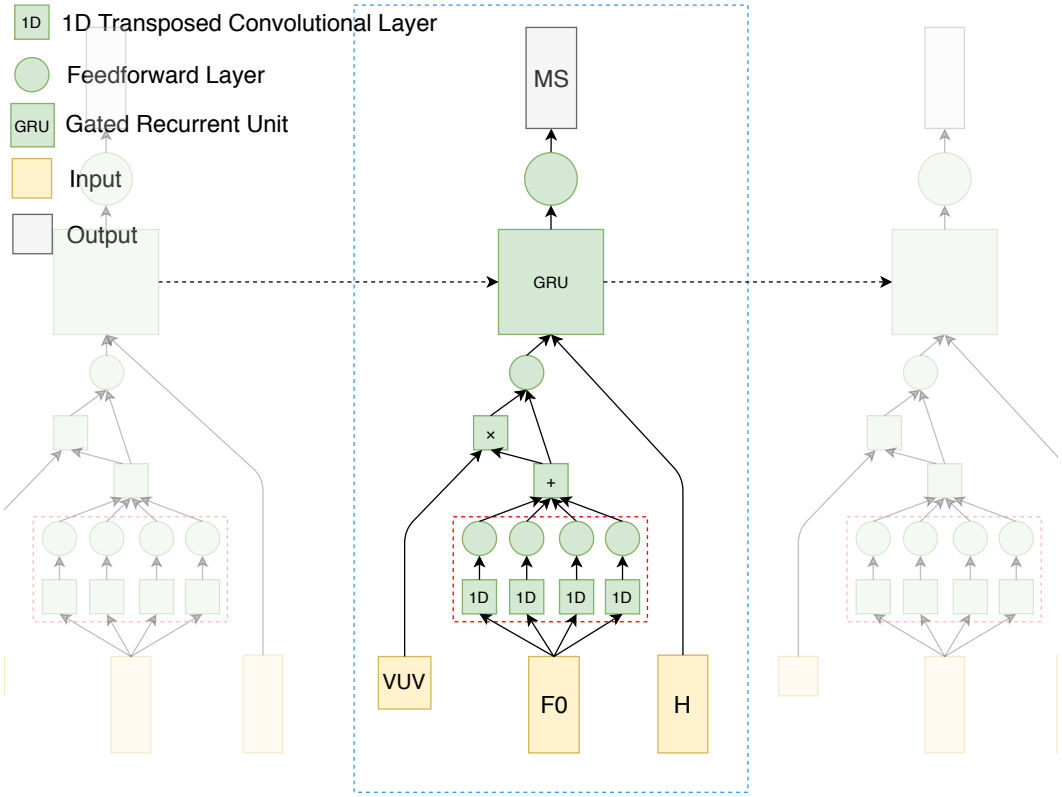


Figure 5.5: The combiner network architecture across three time frames. The blue dotted box shows one frame. The red dotted box shows a parallel bank of transposed convolutions (PBTC). H is the output of the Hider network, VUV is a voiced/unvoiced flag, F_0 is the control parameter.

5.4.1 Hider

Input ground-truth mel spectrograms are passed through a fully-connected layer, into a convolutional layer of kernel size 10, then a 3-layer 800-node GRU with tanh activation. Finally, features are resized to the desired dimensionality (a hyperparameter) using a fully-connected layer, yielding h.

5.4.2 Finder

The architecture of this network is extremely simple: the h vector is passed directly into a 2-layer, 300-node stacked GRU with tanh activation, then one linear layer, and finally a softmax layer which produces a probability distribution across quantised F_0.

5.4.3 Combiner

The architecture of the combiner is shown in Figure 5.5. Inputs are a voicing flag, the control parameter, and the output of the hider network. The combiner first passes a 1-hot embedding of the control parameter through a parallel bank of transposed convolutions (PBTC). A PBTC consists of an array of 1D transposed convolutional layers, with each of these convolutions using a different dilation and the results being summed together. The aim of this PBTC is to generate the harmonic structure in the spectrogram characteristic of F_0. To do this the aim was to make use of *checkerboarding*, an artefact of transposed convolutions that are normally considered harmful (Odena et al., 2016), but in the one-dimensional case somewhat resemble harmonics. Because convolutional networks are powerful, it was initially hoped that this approach would later generalise to other control parameters where a single control parameter results in structured spectral energy. The output of the PBTC is duplicated, with one of the two copies being masked by voicing (zeroed out in unvoiced regions). This duplication allows the system to learn whether to take into account voicing information, and was found to reduce artefacts in unvoiced sections of speech. For the PBTC, 10 dilated transposed convolutional layers were used with kernel size 50 and dilations in the range 2-20. The result of the PBTC is concatenated with the hidden embeddings coming from the hider, and passed to a 3-layer 1200-node stacked GRU with tanh activation. Finally, a fully-connected layer outputs a mel spectrogram.

5.4.4 End-to-end adversarial training

Similar to GANs, a two stage process is used to update models at each training step. First, the *finder* (adversary) is trained. For each utterance, the input mel spectrogram is passed through the *hider*. It outputs a hidden embedding that is used to train the *finder*, with ground truth F_0 as target, to minimise cross-entropy loss. Second, the hider and combiner networks are jointly trained by backpropagating through both networks to minimise an adversarial loss defined as the weighted sum of the *combiner loss*, $\mathcal{L}_{\text{combiner}}$ and the *leakage loss*, $\mathcal{L}_{\text{leakage}}$,

$$\mathcal{L}_{\text{adversarial}} = \mathcal{L}_{\text{combiner}} + \beta \cdot \mathcal{L}_{\text{leakage}}, \tag{5.6}$$

with these components formally defined in the following section. The MSE *combiner loss* is measured at the output of the combiner, whose target is the same as the input to the hider: a ground-truth mel spectrogram. The combiner also takes ground-truth F_0

as an input.

The leakage loss measures how much information related to the control parameter is 'leaked' by the hider into the hidden embedding. Learning a representation that contains *no* information pertaining to a specific variable is non-trivial. *Mutual information* quantifies the amount of information that one random variable contains about another and this would be applicable to neural networks, which map from one random variable to another. However, calculating mutual information is difficult and can require a large number of training samples (Belghazi et al., 2018).

A simpler solution was used: the hider network should produce a hidden embedding from which the finder outputs a uniform probability distribution over the quantised control parameter (CP): it should be *maximally uncertain* about the CP. The MSE error between finder output and a uniform distribution is equivalent to the variance of the distribution and thus the leakage loss was defined. A high value implies the finder network has certainty about the CP. Zero loss implies that all CP values are equally probable, because the hider has completely removed all information about the CP from the hidden embedding. Maximal uncertainty at finder output is preferable to a confident-but-incorrect finder. In the latter case, the hider could learn to fool the finder network with a solution that is easier to learn than actually hiding the CP – perhaps, where F_0 is the CP, by doubling F_0 via only removing odd harmonics – but that the combiner could use instead of its control parameter input value.

We define h as the hidden representation and y as the correct pitch (or more generally, control parameter) classification. The use of such a loss function for minimising the leakage of information pertaining to the y variable, which in this case refers to pitch classification, is believed to be novel as applied to adversarial speech modifying networks. Its definition is given by

$$\mathcal{L}_{\text{leakage}} = \text{Var}(\text{softmax}(F(H(x)))). \tag{5.7}$$

Despite being conceptually simple, the leakage loss is shown in Figure 5.7 to be an effective measure of information leakage.

We can simplify the above equation by acknowledging the output of the softmax function is the predicted probability distribution of y. Where we notate the probability distribution across all 100 permissable values as Y,

$$P(Y|x) = \text{softmax}(F(H(x))). \tag{5.8}$$

Our loss is therefore

$$\mathcal{L}_{\text{leakage}} = \text{Var}(Y|x). \tag{5.9}$$

We define this to be our leakage loss, $\mathcal{L}_{leakage}$, as used in Equation 5.6. Investigations into other functions that may serve as good leakage losses are given in the next chapter.

Data: For each speech sample in the dataset: mel spectrogram and F_0 values
Result: Trained network parameters
foreach *speech sample* **do**
  `/* First train finder network */`
  Generate hidden state by passing mel spectrogram into hider network;
  Use finder network on hidden state to get estimated F_0;
  Calculate cross-entropy loss between estimated F_0 and F_0;
  Backprop through finder parameters and update using Adam optimiser;
  `/* Train hider and combiner */`
  Pass F_0 and hidden state into combiner;
  Calculate MSE combiner loss and leakage loss, sum;
  Backprop loss through *both* hider *and* combiner parameters;
  Update combiner and hider parameters using Adam optimiser;
end

Algorithm 2: Training algorithm for adversarial training

Algorithm 2 shows how training was conducted using the leakage loss as part of an adversarial training scheme.

5.4.5 Experimental Setup

The LJSpeech corpus from Ito and Johnson (2017) was used for training, comprising recordings of 50 chapters of non-fiction books by a female US English speaker. The last chapter was held out for the evaluation. Batch size was 1 utterance with 12693 training steps per epoch. Parameters were updated using the Adam optimiser after every training step. A validation set of 128 randomly-selected utterances was used for early stopping and all systems reported below were trained for 6 epochs. For the input and output a frame size of 50 ms and a frame shift of 12.5 ms are used to extract an

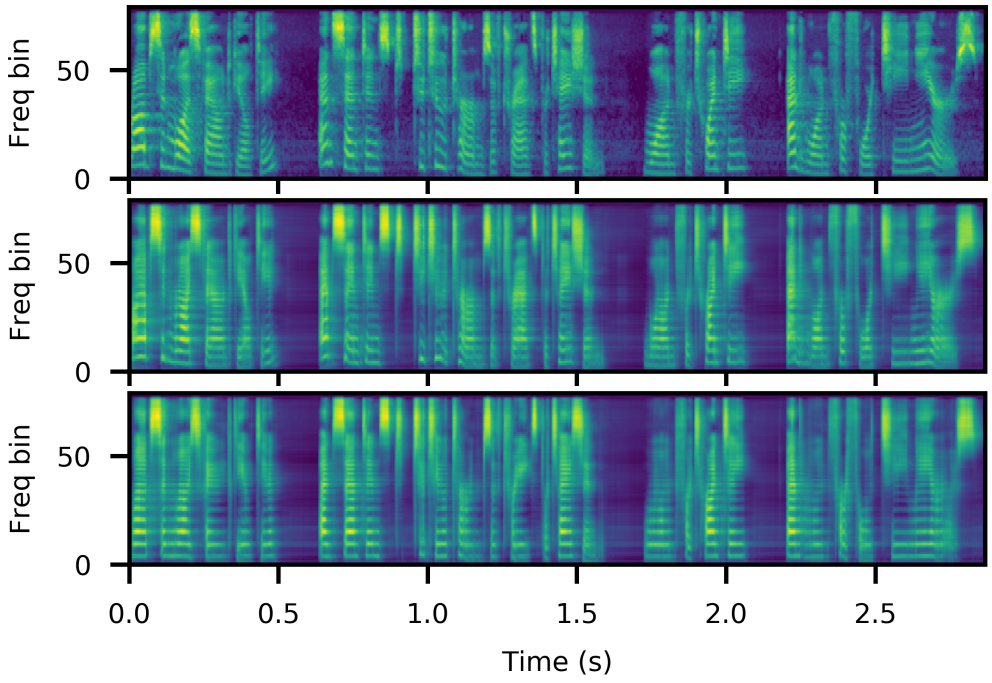


Figure 5.6: Example of changing F_0 to a constant value of 214 Hz. Top: ground truth mel spectrogram. Middle: output with $\beta = 200$. Bottom: output with $\beta = 800$.

80-band mel spectrogram with LibROSA (McFee et al., 2015). A feature width of 80 was also selected for the hidden embedding. The audio sampling rate was 22.05 kHz.

The value of β in Equation 5.6 determines how much the leakage of control parameter value into the hidden embedding contributes to training. At $\beta = 0$ there is no adversarial loss, the hidden embedding will contain control parameter information, and the combiner is free to ignore its input control parameter. Small non-zero values of β lead to partially-modified speech, as shown in Figure 5.6. Increasing β brings the output speech closer to the desired control parameter value, (bottom of Figure 5.6) but excessively large values of β cause output quality to degrade because too little importance is given to the combiner MSE loss. Overall, the MSE combiner loss is positively correlated with β and the leakage loss is negatively correlated. Pilot experiments showed that $\beta = 560$ is a satisfactory trade-off between output quality and control accuracy, and this value is used in the following evaluation. The value of β is dependent on the normalisation of the leakage loss, and a more sophisticated analysis of this is given in the following chapter.

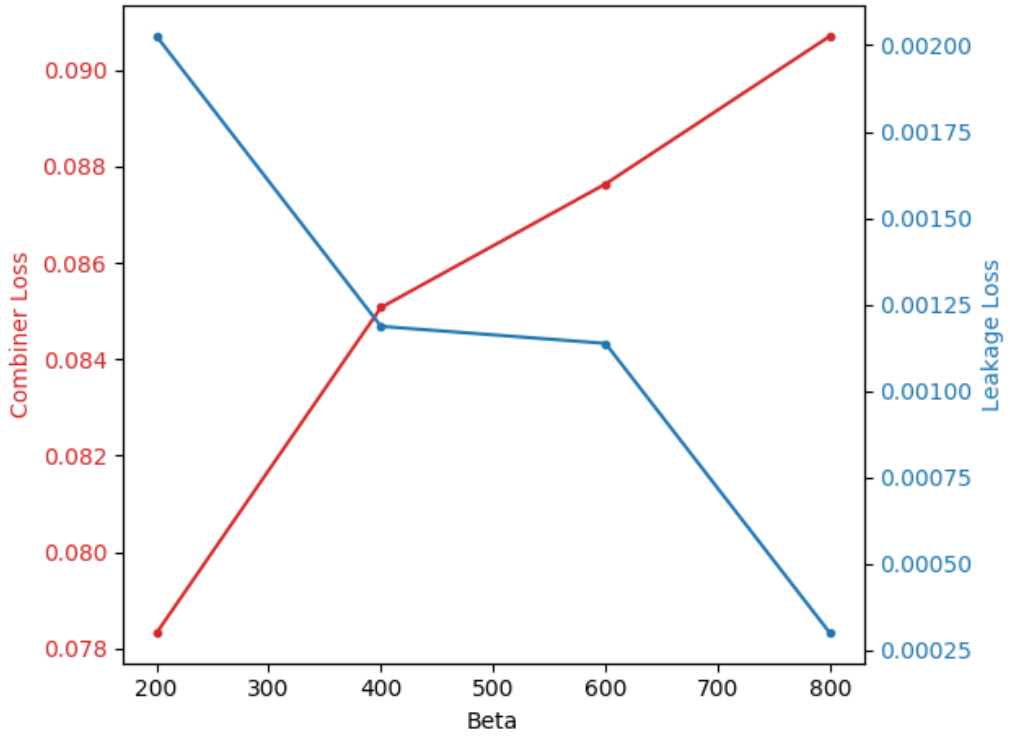


Figure 5.7: Combiner and leakage losses at various values of β for F_0. Results after 1 epoch of training.

5.5 Evaluation

30 utterances were selected at random from the test chapter, and four control types were defined: the original F_0 trajectory (copy); scaled F_0 by -50% to +50% of its original value in steps of 10; entirely new, but plausible, F_0 contours generated through human performance (‘drawn’), as also used by Evrard et al. (2015), with average value set to match the original F_0. As a baseline, we used a digital signal processing (DSP) approach that also operates on mel spectrograms. As a second baseline, we used WORLD, with speech waveforms as input and output. All audio was generated from the baseline and the proposed Hider-Finder-Combiner approach (HFC) mel spectrograms by an open-source implementation of a state-of-the-art neural vocoder, WaveRNN. We provide an objective evaluation of how closely the output of our system follows the specified control parameter value, and a subjective evaluation of the audio quality.

5.5.1 The Baseline DSP System

To disentangle the harmonics of F_0 and vocal tract (VT) resonances (formants), in order to allow F_0 modification without changing formants, source-filter separation is performed with GFM-IAIF, introduced by Perrotin and McLoughlin (2019b), on mel spectrograms. This decomposes the spectral envelope into two sets of linear predictive (LP) coefficients (Makhoul, 1975), modeling the glottis and the VT respectively. F_0 scaling is applied to the excitation signal obtained by inverse filtering the speech frame with the glottis and VT filters. This modification follows the principle of the phase vocoder as introduced by Flanagan and Golden (1966), but modified slightly to use mel spectrograms: since phase reconstruction is done by the neural vocoder, the DSP system only performs frequency-stretching or compressing on the amplitude spectrum by a rescaling factor. Also, each mel spectrogram frame is interpolated to a linear frequency scale before F_0 modification, then mel-scaled again afterwards. Finally, the speech frame is reconstructed by filtering the F_0-scaled source signal with both glottis filter and VT filters before going back to the mel-frequency scale to feed the neural vocoder.

5.5.2 Objective Evaluation

To assess the fidelity of the output to the control parameter, F_0 was extracted from the synthesised speech waveforms with WORLD, and the Root Mean Square Error (RMSE) used to compare this with their F_0 control values on a logarithmic scale. Table 5.1 (top) shows the median values of the $\log_2(F_0)$ RMSE distributions for each system and F_0 modification type. A Kruskal-Wallis rank-sum test using a χ^2 distribution shows that the RMSE differences between methods are not significant for copy (χ^2 = 1.2, $p = 0.55$), are significant for scale ($\chi^2 = 49.3$, $p < 1e^{-10}$) and marginally significant for drawn F_0 (χ^2 = 7.2, $p < 0.03$). A post-hoc Dunn test for pairwise comparison between methods assesses that all measures are different for scaled and drawn F_0 ($p < 0.01$), except between HFC and WORLD for drawn F_0 ($p = 0.5$). The proposed system performs as well as the others for copying the F_0 trajectory. For F_0 scaling, there is significant but small difference, with an RMSE of 0.02 octave higher for HFC than the DSP method. For drawn F_0, HFC performs significantly better than DSP, and as well as WORLD. From these results, it appears that HFC is better for arbitrary modification of F_0 (drawn F_0) than the DSP method. The superiority of the DSP method for the narrow case of F_0 *scaling* is not surprising, given the simplicity of stretching mel spectrograms

	WORLD	DSP	HFC
F_0 copy	0.16	0.16	0.16
F_0 scale	0.18	0.20	0.23
Drawn F_0	0.15	0.18	0.14

	GFM-Voc	DSP	HFC
F_1	0.33	0.26	0.36
F_2	0.32	0.34	0.41

Table 5.1: RMSE (in octave) of $\log_2(F_0)$, $\log_2(F_1)$ and $\log_2(F_2)$ between control and synthesis for the models in consideration.

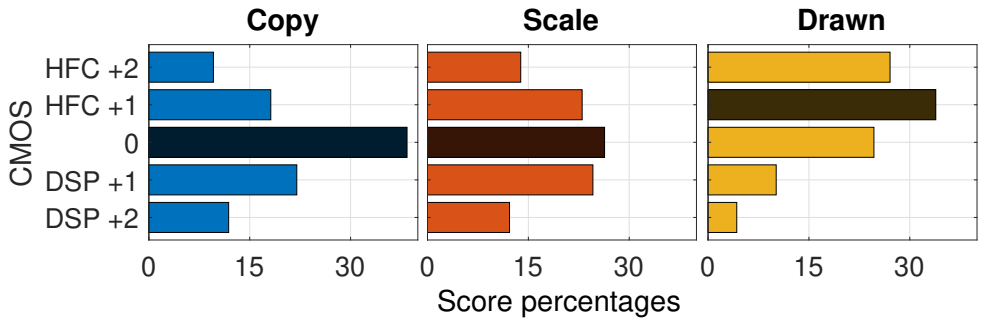


Figure 5.8: CMOS between DSP and HFC for different F_0 modifications. The dark bars indicate the median score for each condition.

of the glottal source signal. A more sophisticated statistical analysis which corrects for multiple testing is left as future work.

5.5.3 Subjective Evaluation

To assess naturalness, a comparative mean opinion score (CMOS) evaluation was performed. Listeners were asked to compare utterances in pairs on a five point scale, corresponding to whether they thought either option sounded much more natural, slightly more natural, or whether both options sounded similarly natural. Each pair comprised the same utterance generated with DSP and HFC. From our speech material, synthesis with original F_0 (copy); scalings of $\pm 20\%$ from the original F_0, and the drawn F_0, were selected, leading to a total of 120 pairs that were randomly split into two blocks. 50 native English speaking US residents were recruited and paid using the Prolific Academic platform. 25 listeners judged block 1, and 25 others judged block 2. An equal number of pairs was presented in each order (DSP-HFC, HFC-DSP) and the ordering of pairs was randomised per listener.

Figure 5.8 shows the CMOS for the three different conditions. A Kruskal-Wallis rank-sum test followed by a Dunn test for pairwise comparison shows that the *drawn* distribution is significantly different from the two others ($p < 1e^{-15}$), and that the *copy* and *scale* distributions are marginally significantly different ($p < 0.03$). This evaluation shows that HFC is as good as DSP for *copy* and *scale*, and better for the generation of new F_0 trajectories. These results are consistent with the objective measures, and confirm that HFC is better for the most practically-useful case of arbitrary modification of F_0.

5.6 Controlling arbitrary parameters

This section describes a number of experiments that were conducted to show the generality of the HFC approach. The results of the work described in this section motivated a change of emphasis in research that is described in Section 5.7. The implementation of this changed approach will be described in Chapter 6.

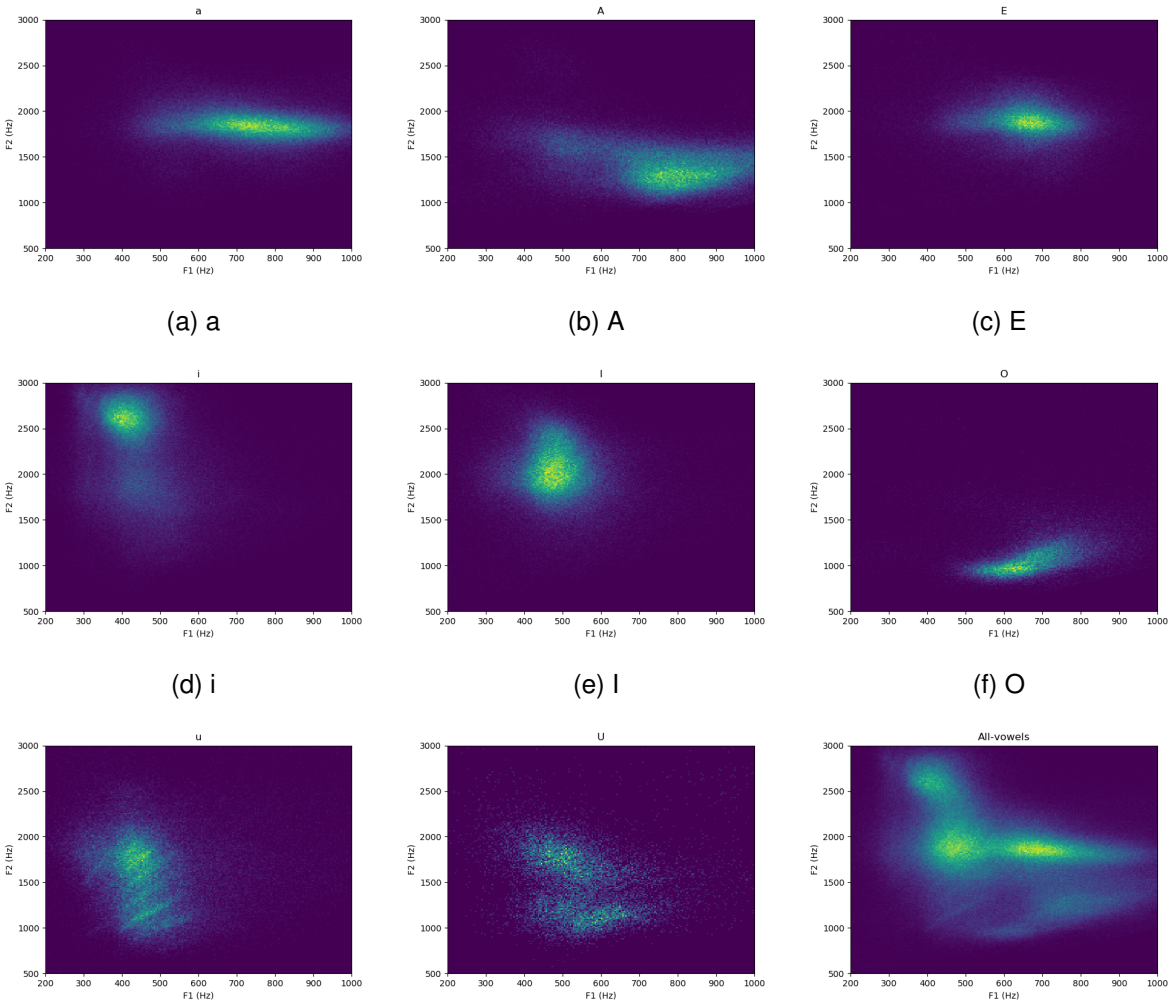
5.6.1 Formant control

5.6.1.1 Motivation

The hider-finder-combiner approach was intended to generalise to the modification for any control parameter which can be annotated on speech waveforms. A first attempt to demonstrate this generality was made by modifying the first or second formant (F_1/F_2), which was annotated on training data using the GFM-IAIF VT filter from Perrotin and McLoughlin (2019b). F_1 and F_2 are significantly more challenging to manipulate than F_0. This is discussed in Section 2.4, and is caused at least in part by the noisiness of extracted F_1 and F_2 values. The results described in this section were achieved using exactly the same model architectures as described in Section 5.4.5. As with F_0, the control parameter was quantised into 80 linearly spaced bins: F_1 between 0.2 and 1 kHz, F_2 between 0.5 and 3 kHz.

Firstly, F_1 and F_2 labels were calculated for the LJSpeech dataset. A pre-calculated phone alignment with the Combilex phoneset was acquired[1], using the Montreal Forced Aligner. Figure 5.9 shows the distribution of F_1 and F_2 distribution for all the vowels in this phone set.

[1]Thanks are extended to Jason Taylor, who extracted this alignment using Montreal Forced Aligner and kindly shared this with me.



(a) a (b) A (c) E

(d) i (e) I (f) O

(g) u (h) U (i) All Vowels

Figure 5.9: Histograms of extracted F1/F2 values during sections of utterances labelled with specific Combilex vowels. Utterances are from the LJ Speech corpus

5.6.1.2 Experiment

50 utterances of median duration from the test chapter were synthesised. For these sentences, either F_1 or F_2 was scaled by between -40% and $+40\%$ of its original value in two steps of 20 in each direction. Both HFC and DSP methods were compared against GFM-Voc (Perrotin and McLoughlin, 2019a), which extracts the VT filter with GFM-IAIF, changes the formant position through pole modification, then filters the unchanged source signal to reconstruct the speech waveform. This is the same process that is applied in DSP, except that in the former, the waveform is obtained by inverse Fourier transform of the spectrogram, using the original phase, while in the latter the mel spectrogram is fed to the neural vocoder. Since GFM-IAIF is used for both data annotation and parameter modification in DSP, we used Praat, by Boersma and Weenink (2001), for independent extraction of F_1 and F_2 values from all the syntheses, and the RMSE to their respective control trajectories was computed and is shown in Table 5.1 (bottom). A Kruskal-Wallis rank-sum test paired with a post-hoc Dunn test shows that distributions for all methods are significantly different for both F_1 and F_2 ($p < 1e^{-3}$). Although formant modification accuracy is lower for HFC than the other methods, the RMSE difference remains below 0.1 octave in all cases, demonstrating that HFC can modify formants using precisely the same architecture as for modifying F_0.

5.6.1.3 Analysis

However, despite this encouraging early result, it was not possible to use the mel spectrogram outputs of formant controlled signals to synthesis high quality audio using the WaveRNN vocoder. Informal listening tests showed output quality to be very poor and unintelligible, meaning a thorough evaluation was not possible. Despite extensive efforts, sweeping through hyperparameters such as the number of layers in each component and various learning rates, it was not possible to improve the quality of output through further hyperparameter experimentation.

5.6.2 Controlling DSP-extracted vocal effort correlates

As it had not been possible to get good synthesis results with F_1 and F_2, a new set of features was identified. The goal was to use features that had been defined as part of the *Glottal Flow Model-based Iterative Adaptive Inverse Filtering* (GFM-IAIF) project by Perrotin and McLoughlin (2019b). This was implemented by the authors as a set of

Matlab routines which use digital signal processing methods to extract features, many of which are correlates of vocal effort.

A dataset from Valentini-Botinhao et al. (2019) that contains both Lombard and quiet speech, with each text prompt recorded in both speech modes, was used. The Lombard speech was recorded by exposing the speaker to white noise through headphones while he performed the utterances. This allowed statistical analyses to be performed on the features with respect to both speech modes.

The chosen parameters are all, in the original GFM-IAIF code, derived from full linear magnitude spectrograms. Three features were initially chosen. These were centre of gravity (CoG), spectral tilt (ST) and intensity (I). These parameters were all thought to be correlated to vocal effort. These parameters were measured in both the Lombard and quiet speech parts of the dataset. By measuring the correlations between these parameters in the ground truth dataset and after modifying parameters individually using the HFC system it was possible to gain further insight into modeling and controlling vocal effort.

The choice of new parameters enabled the development of a new proposed loss. This was named the *fidelity* loss, and was designed to measure the control parameter in the output mel spectrogram and penalise output where the control parameter had not been modified appropriately. Therefore, one of the main tasks in this work was deriving these (non-F_0) parameters in a differentiable way. Differentiable F_0 extraction was not widely available at the time this work was done, but has since been used by Watts et al. (2023). Differentiable extraction was implemented in the hope that a new loss could be used to train the HFC system. This meant implementing the above GFM-IAIF feature extraction using PyTorch. Some of this code has now been submitted to the torchaudio project. This and other codes are made public alongside this thesis.

The parameters were originally extracted from full linear spectrograms by GFM-IAIF. A differentiable method for converting mel spectrograms to full magnitude spectrograms was implemented. This allowed ground truth *and* output values for the features to be derived from mel spectrograms, allowing a fair comparison between input and output values. The torchaudio library was used for spectrogram extraction and for mel-scaling. Unfortunately, at the time that this work was done, torchaudio did not have an effective method for mel-scaling inversion, which was therefore implemented by me as follows.

Mel-scaling works by constructing a *mel basis* matrix, F that reduces the dimensionaltiy of the spectrogram, s so that

$$\mathbf{F} \cdot \mathbf{s} = \mathbf{s}_{\text{mel}} \tag{5.10}$$

It is possible to use a Penrose-Moore pseudo-inverse (denoted by a superscript $+$) to find a least squares solution for s when the mel is known

$$\mathbf{F}^{+} \cdot \mathbf{s}_{\text{mel}} \approx \mathbf{s}. \tag{5.11}$$

While optimally stable methods for finding least squares solutions of this type, using an algorithm by Lawson and Hanson (1995), typically avoid forming an explicit pseudo-inverse, in this case no significant difference (with my own analyses using MSE) was found between using a per-utterance least squares solution and finding the pseudo-inverse once.

The inverting of the mel spectrogram was done in PyTorch. Using `Tensor.pinverse` over `torch.lstsq` has the advantages of

1. only finding an inverse matrix once, meaning all future inversions are simply matrix multiplications, and

2. being trivially differentiable, meaning that the control parameter measured on output mel spectrograms can be used to calculate losses

Extracting GFM-IAIF features from mel spectrograms in this way was useful in allowing the features to be consistently measured between the input *and output* of the HFC system, without the use of a neural vocoder to bring the output into the time domain.

Training on derived losses from the CPs measured on output mel spectrograms (the so-called *fidelity loss*) was implemented. It is believed this is a new contribution in this work. The fidelity loss is defined as being the MSE between the output CP measured using DSP techniques on the output of the combiner network and ground truth CP values measured using the same DSP techniques on ground truth mel spectrograms. The fidelity loss was not studied in detail, as preliminary investigations did not indicate GFM-IAIF features were likely to be successful choices of control parameters.

5.6.3 Control parameter specifics

The features are calculated as follows (see also the open-source Matlab script[2]): CoG is a weighted average of the magnitude spectrogram in the region between 80Hz and

[2] `https://github.com/operrotin/GFM-IAIF`

5000Hz for each time step in the spectrogram. Intensity is the sum of each time step in the spectrogram. To calculate the ST, a linear fit was done to the spectrogram in the 80-5000Hz region. The gradient of this was used as ST.

5.6.4 Output from GFM-IAIF features

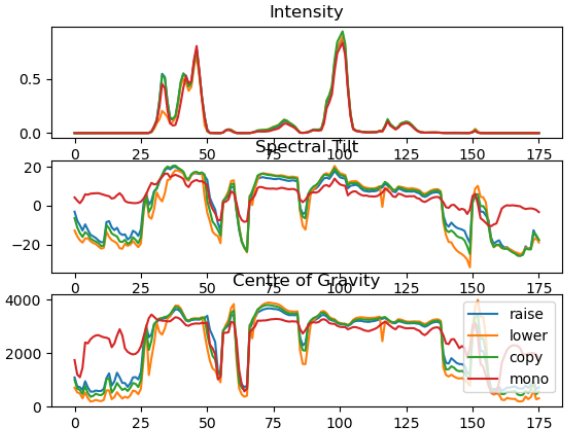
Figure 5.10 shows the three measured control parameters where *each* of the control parameters are in turn modified. There was relatively little effect from controlling parameters. The normal course of action in this case would be to increase the β value, which would weight the leakage loss higher. However, Figure 5.11 shows that output mel spectrograms are already very low quality, to the extent that they did not yield anything resembling speech when converted to waveform with a vocoder. Previous experience shows that increasing beta degrades output spectrogram quality in exchange for improvements to accuracy of output control parameter.

It was noted in early experiments that harmonics were not well-reconstructed by HFC when features other than F_0 were used. It was therefore decided to add F_0 as an additional input, alongside the control parameter, both during training and inference. This was done by using an additional PBTC and concatenating this block's output with a PBTC for the new control parameter values. The mel spectrograms shown in Figure 5.11 show the output from this approach, although the results were largely similar to those obtained without this additional F_0 input. It was a source of considerable frustration that it was not possible to achieve good results with DSP-derived features other than F_0. Despite much experimentation it was not clear why the systems did not generate credible spectrograms. The training losses did not provide an indication as to why this would be as they were largely similar to with F_0. A lot of time was spent on these features before a decision was taken to change approach in the way described in the following chapter.

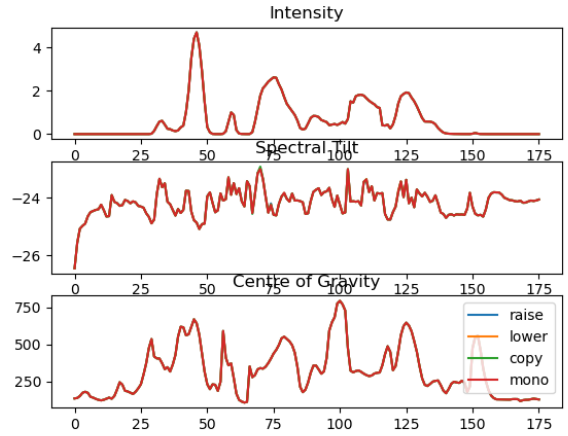
5.6.5 Analysis of GFM-IAIF features for Lombard and quiet speech

Although it became apparent that HFC did not work well with these variables in its current form, it is still beneficial to examine how these variables vary between Lombard and quiet parts of the dataset.

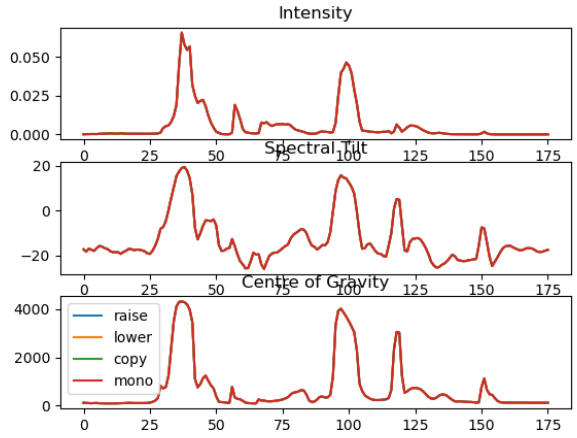
By measuring the chosen variables across the quiet and Lombard sections of the dataset it was possible to show that they were indeed good analogues for vocal effort. Histograms for the variables were generated across both parts of the dataset are shown



(a) CoG

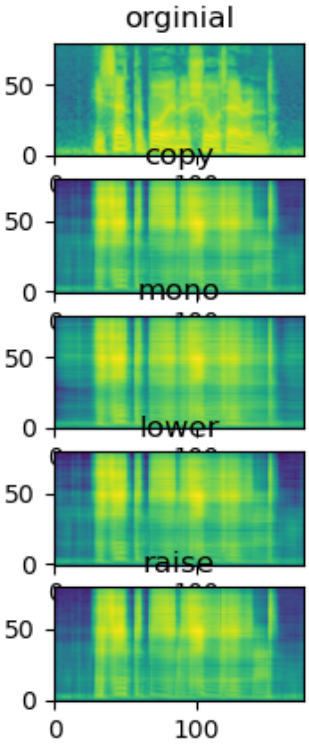


(b) ST

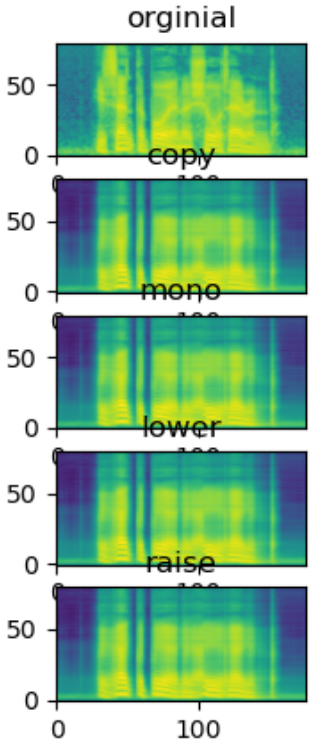


(c) I

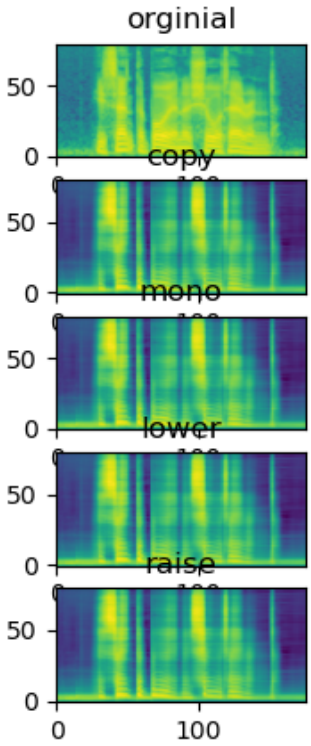
Figure 5.10: Effect of controlling each control parameter, Centre of Gravity (CoG), Spectral Tilt (ST) and Intesity (I) on all possible CP features



(a) CoG

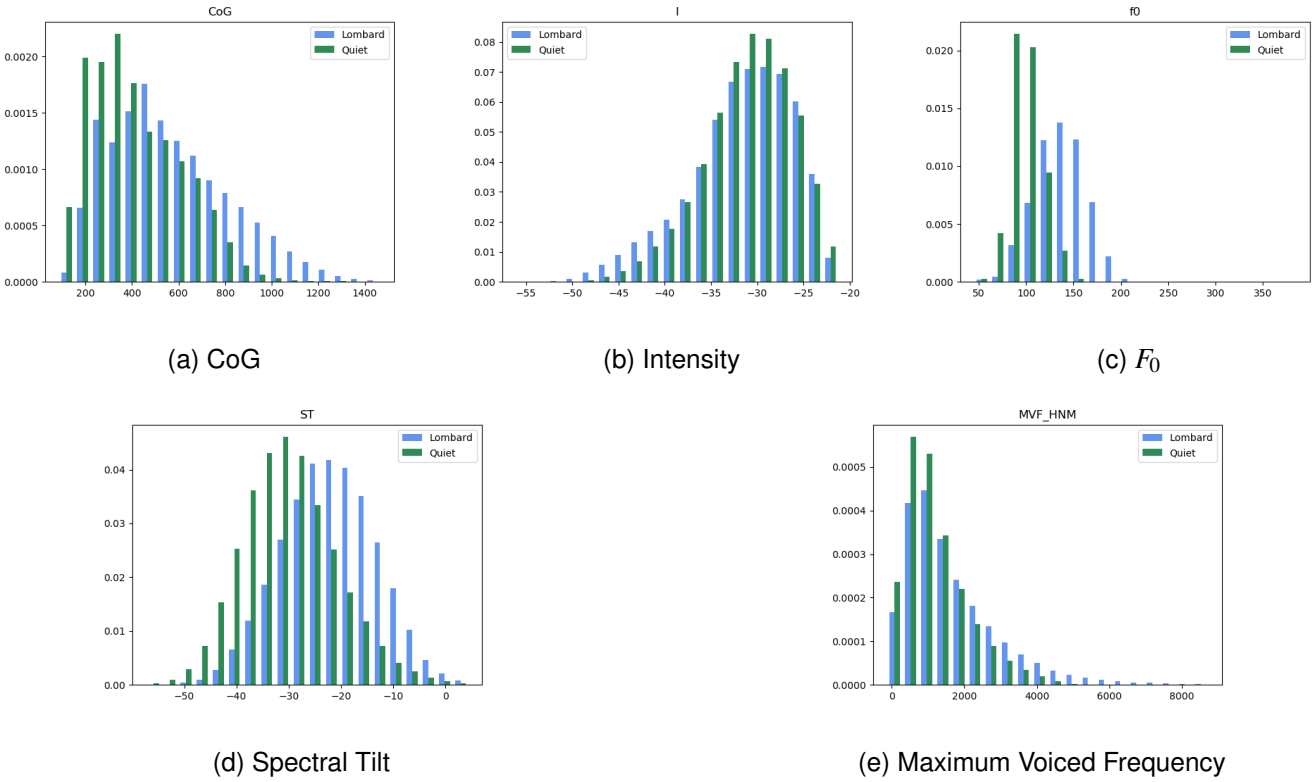


(b) ST



(c) I

Figure 5.11: Output mel spectrograms showing effect of controlling each control parameter on a range of features



(a) CoG (b) Intensity (c) F_0

(d) Spectral Tilt (e) Maximum Voiced Frequency

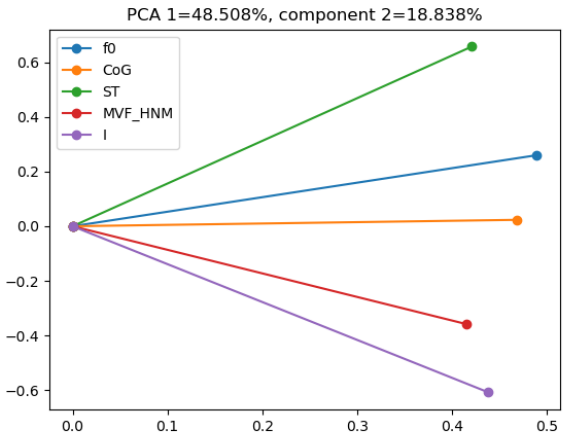
Figure 5.12: Histograms of extracted CP parameters on the Lombard dataset

in Figure 5.12. In addition to the three variables targeted by HFC in this section, F_0 and Maximum Voiced Frequency (MVF) were also analysed.

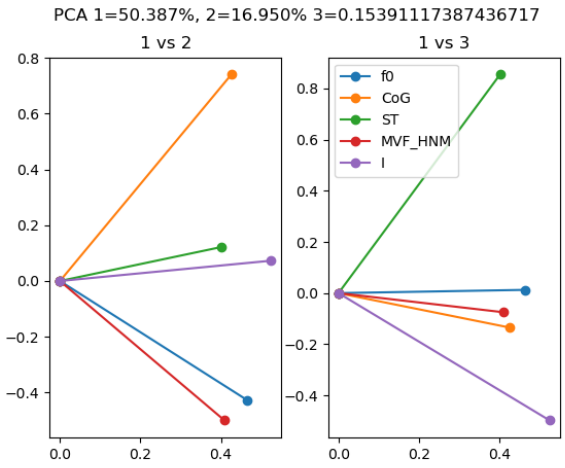
Using all these variables it was possible to do a PCA analysis as shown in Figure 5.13. This provides an interesting analysis of which of these features correspond best to vocal effort, at least with respect to this dataset. In future, these PCA values themselves could yield vocal effort features when combined.

5.7 Chapter summary and motivating a change of emphasis

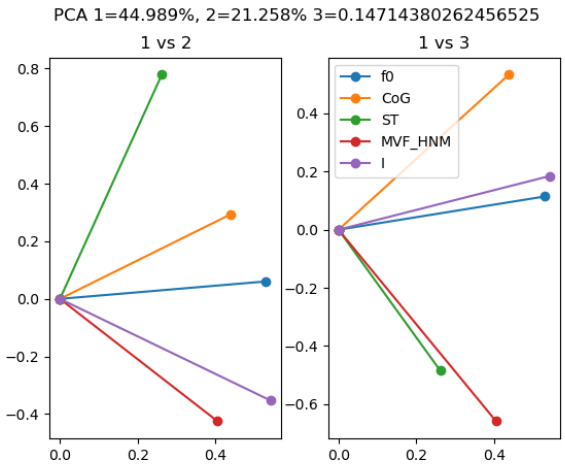
This chapter introduced a novel neural architecture for controlling specific aspects of speech signals. This system makes use of adversarial training to learn to control parameters in a way that at the system level should be able to control arbitrary parameters. However, while this system worked well for F_0, it was not found to generalise well. The system could modify F_1 and F_2, although yielded unintelligible output. This was not the case for other signal-processing derived features, where the quality of system



(a) Full dataset



(b) Loud PCA



(c) Quiet PCA

Figure 5.13: First Two PCA components for params on dataset plus 1, 2, 3 PCA components for parameters on quiet and Lombard

output was not high even enough to be worth thoroughly evaluating.

There are some hypotheses for this lack of generality that are put forward here. The first is that, while the system of training was not parameter specific, it seems reasonable to hypothesise that the component networks, in particular the combiner network, *were* parameter specific. The use of PBTC (see Section 5.4.3 within the combiner was designed quite specifically to be capable of generating the kind of striations that harmonics form within a mel spectrogram. The PBTC architecture which was created during this PhD has since been used to add new harmonics in speech-to-singing conversion models, such as by Zhou and Lu (2022). A systematic analysis by Jayashankar et al. (2023), based on the work in this thesis, showed that the PBTC is more effective than other state-of-the-art techniques at this task. However, as the PBTC is a relatively small learned system its mere presence should not degrade performance too badly. There may be benefits to introducing a more powerful, modern sequence processing architecture to replace the PBTC when a feature other than F_0 is controlled.

Another potential reason for the problems discovered in this chapter is that the features chosen other than F_0 are difficult to measure and could be ambiguous. They also vary greatly *within* utterances, and are likely to behave very differently in voiced, unvoiced and silent regions. This is likely to add significant challenge when forming representations and synthesising.

It is also hypothesised that problems were caused by the dependence on mel spectrograms as intermediate features. These mel spectrograms can be interdependent with control parameters in complex, highly structured ways (e.g. periodic harmonics when the control parameter is F_0). This makes hiding and combining highly specialised as different features affect the signal in varying ways. Note that deep learning architectures are highly specialised by field. Image processing depends largely on localised, translation independent CNNs, whereas language models and machine translation systems are designed to model long term temporal dependencies. When different control parameters can have such different systematic effects on a mel spectrogram, it is challenging to derive a system that generalises well without modification.

In addition, the dependency on mel spectrograms introduces a further dependency on neural vocoding. This encumbered my early work in this area, as these vocoders were slow and difficult to fine-tune to new datasets on the compute available to me. In addition, because mel spectrograms omit phase completely, modification of phase is outside the control of the HFC system. Although phase is not thought to encode linguistic information, it was shown by Loweimi et al. (2021) to encode information

relating to voice quality or speaker ID. The neural vocoder fulfills two functions in speech synthesis. First, it upsamples mel spectrograms to the same dimensionality as sample rate. Second, it generates phase, with modern approaches integrating generative models for this purpose Kong et al. (2020).

There is also yet another an element of HFC that may have caused problems when generalising to arbitrary parameters. For the leakage loss, the hider network in all the work described in this chapter aimed to fool the finder into outputting a *uniform* distribution. This does not reflect a reasonable prior for these signals, where the underlying data is likely to follow a different distribution. The different distributions that different control parameters take may have interfered with the portability of the HFC system, given that all experiments targeted the same prior. Future work with HFC could instead target a prior that is formed by taking a normalised histogram of the control parameter across the training set. A detailed analysis of this would be useful future work applied to F_0, where even the uniform distribution *was* successful in achieving high-quality output signals and good control of the control parameter.

The work in this chapter shows the potential of using learned representations with information hiding to control speech signals. The limitations just described motivate two streams of further work. Firstly, a more stable choice of control parameter, and more sophisticated component networks were used for further experiments with HFC. These experiments are discussed in the following chapter, alongside a more rigourous analysis of the leakage loss. Secondly, a decision was made to attempt to generate audio directly from a learned representation, which avoids the use of a neural vocoder. The details of a step towards this are described in Chapter 8, which explores the possibility of synthesising audio directly from representations of audio that are learned by an auto-encoder. Extending this autoencoder approach to become a HFC-type model is left as future work by this thesis.

Chapter 6

HFC — New architectures, a global parameter and Voice Conversion-based Privacy

The purpose of the work in this chapter was to find another working use-case for the Hider-Finder-Combiner (HFC) system beyond F_0, where high quality audio output can be achieved along with results that are faithful to the required change of control parameter.

Previous work in Chapter 5 hypothesised on some reasons why HFC did not generalise well to parameters that are not F_0. These were that

1. the prior subcomponent architectures were too specialised to the task of synthesising harmonics,

2. chosen control parameter features were too ambiguous and difficult to extract, resulting in noisy signals that are difficult to extract, and that

3. the *prior* targeted by the optimisation of the hider is a flat probability distribution, rather than a prior that accurately reflects the distribution of the control parameter.

We now address these likely causes, while applying HFC to a useful task: speaker identity. The task of converting utterances to sound as if they were made by another speaker is well-known and has important applications in ensuring speaker privacy (Tomashenko et al., 2020). It is through the lens of voice privacy that the model introduced by this chapter will be evaluated.

6.1 Introduction

As the ability of state-of-the-art systems to extract sensitive information from speech signals has increased, so has the importance of speech privacy. Speech privacy has become a prominent research area since the VoicePrivacy Initiative was introduced by Tomashenko et al. (2020). This initiative was followed by the establishment of biennial VoicePrivacy Challenges[1](Tomashenko et al., 2022b). A speaker's identity is deeply entangled among all other acoustic and semantic characteristics of speech (Williams and King, 2019). It is therefore challenging to remove speaker-identifying information without destroying other components of the speech signal. While the VoicePrivacy Challenge systems and evaluation campaigns have been exploring the trade-offs between retaining source speech attributes while simultaneously preserving privacy, in this chapter I apply HFC as a principled attempt to achieve *variable privacy* – which I define as the ability to select or control how much information is hidden or concealed. The approach is principled as it applies a machine-learning system in such a way that it is trained to remove information about a speaker and the weighting of this training signal is controllable alongside the training signal that delivers good signal reconstruction, corresponding to output quality. However, evaluations showed that varying the weights of these training losses did not affect output as expected. As more national and international bodies consider regulating both privacy and the use of Artificial Intelligence (Mourby et al., 2018; Voigt and Von dem Bussche, 2017), privacy solutions must keep pace with these changing needs.

Considering how rich the speech signal is with information, there are may different ways to view voice privacy beyond traditional definitions. These include neutralising prosody or other speaking patterns that could identify a person, removing aspects of speech that convey a health condition, transforming the age or gender of the speaker (Noé et al., 2021; Benaroya et al., 2023), and concealing or masking targeted words (Williams et al., 2023). Voice privacy can also be viewed within a paradigm of how speech data is transmitted between humans or machines, and how that information may be intercepted (Williams et al., 2022). Controllability is key to developing solutions that can address multiple types of privacy. I introduce a new mechanism for controlling leakage of identity-bearing information using adversarial information hiding that deliberately balances the trade-offs between maintaining source-speech characteristics and modifications of speaker identity. I draw from lessons learned in voice con-

[1] https://www.voiceprivacychallenge.org/

version techniques like CycleGAN and StarGAN, both from Kaneko et al. (2019a,b), which were not designed specifically with privacy in mind. This means that converted speech from such models may leak personal information in unpredictable ways.

The contribution outlined in this chapter, and in an accompanying paper accepted for publication, addresses the idea of controllable privacy in a new way. I extend the work from the previous chapter, specifically the F_0-modifying model which was most effective, which I will now refer to as HFC-F_0. This model has some similarities to Fader Networks developed by Lample et al. (2017) independently from the work of the previous chapter. The HFC-F_0 model was devised to modify F_0 contours of source speech, a control parameter which, like speaker identity, is deeply entangled with many other aspects of a speech signal (Williams and King, 2019).

In this chapter, I introduce HFC-VP as a system that creates a disentangled hidden representation which can then be recombined with an arbitrary *speaker embedding*. To the best of my knowledge, the techniques that introduced as part of this thesis represent the first effort to achieve anonymisation solely through minimising the mutual information between a predicted probability distribution of speaker identity and a prior distribution of speaker identity, from a large dataset of 904 speakers. This principled and "unguided" approach means that a single system could be capable of modifying different aspects of speech, such as prosody or lexical content, exactly where it is necessary to preserve privacy.

This chapter outlines substantial changes to the original HFC-F_0 system that was presented in the previous chapter and in Webber et al. (2020). The training losses have been significantly adapted (Section 6.3.1) in a manner that allows the underlying architecture to be repurposed for the task of controllable speaker privacy. The proposed system also replaces the component networks, which now use powerful transformer encoders and residual convolutional networks in place of RNNs. The resulting system outperforms a strong neural source-filter baseline in evaluations. It also provides the possibility of sharing an anonymised speech representation that can be resynthesised at a later date. Audio examples are available online[2].

6.2 Background

There is increasing interest in using speech signal disentanglement and privacy control with speaker anonymisation (Champion et al., 2022). Systems, such as those of Qian

[2] https://hfcvp.github.io/

et al. (2020); Mosiński et al. (2023); Peyser et al. (2022); Noé et al. (2021); Kovela et al. (2023), have applied *inductive biasing* by using models that only have the power to learn specific attributes. For example, bottlenecks have been used to guide synthesis by accounting for features that are desirable to preserve. An example is preserving the lexical content of utterances alongside such bottleneck features. Other privacy-preserving approaches, such as from Noé et al. (2021), have applied Fader Networks (Lample et al., 2017) to modify specific scalar attributes of the speech signal.

The Hider-Finder-Combiner (HFC) architecture was first introduced by Webber et al. (2020) for the purpose of performing general modifications to the speech signal. The original paper showed that the system worked well for F_0 modification, although the previous chapter revealed limitations when extending to other parameters. Here, I significantly extend this architecture (Section 6.3) and also show how this extension adapts the architecture, enabling speaker privacy.

The HFC system equates the problem of speech signal modification with *disentanglement* (Williams and King, 2019; Qian et al., 2020). Parameterising certain aspects of speech signals (which are called *control parameters*) is challenging because the input to machine learning-based speech modification systems (e.g., mel spectrograms) contains information relating to other aspects of the speech signal that may be desirable to change as well as aspects that may be desirable to preserve. The HFC system overcomes this problem by using adversarial information hiding.

As in HFC-F_0, HFC-VP employs a *Hider* network to generate a latent representation, h, which contains as little information as possible about the control parameter, but retains all *other* information from its input so that, when combined with the control parameter, in this case another network, the *Combiner* can reconstruct the original signal. HFC also uses a *Finder* adversary to ensure that the Hider network does not leak information relating to the control parameter. This Finder predicts the conditional probability distribution given h:

$$h = H(x) \tag{6.1}$$

$$x \approx C(h, c) \tag{6.2}$$

$$F(h) \approx p(c|h) \tag{6.3}$$

where c is the class assigned to the control parameter. In the previous chapter the control parameter is a time sequence of binned F_0 values. The HFC architecture is shown in Figure 6.1.

The way that HFC is trained is similar to a typical GAN, in which a generator net-

work aims to fool a discriminator (adversary) network into mis-classifying its output (e.g., to treat output as genuine (Goodfellow et al., 2014)). In HFC, as in GANs, the generators (Hider and Combiner together) and the discriminator (Finder) are trained in turn. This chapter introduces a more sophisticated analysis and description of the original training losses, as well as how these losses are now modified in order to make them more well-motivated. This analysis is given in Section 6.3.3. Fader networks are somewhat similar to HFC, although simpler in that they only attempt to disentangle a single scalar value.

Chou et al. (2018) used a similar three-component network to HFC and Fader Networks, applying these to the task of voice conversion. However, this work differs from that presented here in that it takes a simpler approach compared with our leakage losses discussed in Section 6.3.1, uses RNN-based component networks rather than a transformer as described in section 6.3.4, is not evaluated on a privacy-related task and does not generalise to unseen target speakers.

6.3 Method

6.3.1 Losses

In the previous chapter, three losses were used to train HFC. A finder loss trains the adversary, while the hider and combiner, trained alternately with the finder, use a combiner loss and a leakage loss, which measure reconstruction quality and information leakage respectively. This leakage loss is defined as $\mathrm{Var}[F(c|H(x))]$, i.e. the variance of the conditional probability predicted by the finder.

While these proposed losses were useful for the goal of F0 modification, they are further analysed here and made more consistent. The original HFC used mean squared error (MSE) to measure the leakage, but an entropy-based approach to train the finder. In this chapter, I separate these approaches, presenting two pairs of loss functions that can be used, adopting an information theoretic approach by describing the leakage loss as a measure of information leakage between a predicted distribution and that of a *prior* distribution, where this prior is the general distribution of the underlying data.

By changing the loss of both leakage and finder to the MSE between two discrete probability distributions, I then measure the divergence between the two probability distributions with a theoretical basis that was introduced by Pearson (1900), with the understanding that this approach was shown to be useful for GAN training by Mao

et al. (2017).

This MSE-based approach is defined by the following two equations:

$$\mathcal{L}_{\text{leakage}} = \frac{1}{C}\sum_{c}(F(c|H(x)) - p(c))^2 \tag{6.4}$$

$$\mathcal{L}_{\text{finder}} = \frac{1}{C}\sum_{c}(F(c|H(x)) - I(c|x))^2 \tag{6.5}$$

where I is the "onehot" Indicator function.

It is *alternatively* possible to use a more information-theoretic approach by using the KL divergence between the prior distribution and the posterior distribution predicted by the finder. Ignoring constant terms this yields the following two loss functions:

$$\mathcal{L}_{\text{finder}} = D_{KL}(p(c|x)||F(c|H(x))) \tag{6.6}$$

$$= -\log(F(c_i|H(x_i)) \tag{6.7}$$

$$\mathcal{L}_{\text{leakage}} = D_{KL}(p(c)||F(c|H(x))) \tag{6.8}$$

$$= -\sum_{c} p(c)\log F(c|H(x)) \tag{6.9}$$

where c is the class (speaker identity in our case) and x is the input mel spectrogram. I optimise H to minimise $\mathcal{L}_{\text{leakage}}$ and F to minimise $\mathcal{L}_{\text{finder}}$. Both H and C are optimised to minimise a MSE reconstruction loss, $\mathcal{L}_{\text{combiner}}$. I weight these terms by deriving a total loss with which to optimise both H and C

$$\mathcal{L}_G = \mathcal{L}_{\text{combiner}} + \beta\mathcal{L}_{\text{leakage}}$$

For the experiments in this chapter, I exclusively use these new losses defined in Equations 6.4 and 6.5. The KL-based losses are given for information purposes and an implementation of both pairs of losses is provided in the accompanying code release. A thorough experimental comparative evaluation of the two approaches is left to future work. I note that $\mathcal{L}_{\text{leakage}}$ defined in Equation 6.4 is equivalent to the original HFC variance loss when the prior distribution is uniform. This prior is calculated empirically by normalising the histogram of the dataset across classes to sum to 1.

One final improvement made to the calculation of $\mathcal{L}_{\text{leakage}}$ is the introduction of a scaling factor. The maximum value possible for the leakage loss (assuming a uniform prior distribution) is a function of the number of classes used for c, N, and is given by

$$\max(\mathcal{L}_{\text{leakage}}) = \frac{(N-1)}{N^2} \tag{6.10}$$

which arises, intuitively, from the case where one class is assigned probability 1, and all others zero. It is possible to prove this more rigorously using Popoviciu's inequality on variances (Bhatia and Davis, 2000). I therefore add this as a scaling factor as follows

$$\mathcal{L}_{\text{leakage}} = \frac{N^2}{N-1}\text{Var}[F(c|H(x))]. \tag{6.11}$$

As the leakage loss is already scaled by an arbitrary hyperparameter, the primary benefit of applying this normalisation factor is to make the β value independent of N. This should also explain why the values of β are several orders of magnitude smaller in this chapter than the previous.

6.3.2 Data

The train-clean-360 split of the LibriTTS dataset from Zen et al. (2019) was used for training. This dataset contains 360 hours of read speech audio at 22.05 kHz from 904 speakers. HFC-VP synthesises mel spectrograms which are converted into the audio domain using a version of HiFiGAN implemented and trained by SpeechBrain (Ravanelli et al., 2021). At training time the system uses both speaker ID and speaker embeddings extracted with SpeechBrain's ECAPA-TDNN, pretrained on the Voxceleb dataset (Nagrani et al., 2017). Because ECAPA-TDNN embeddings are calculated from mel spectrograms, it is possible to calculate these embeddings directly from HFC-VP's output without vocoding, which could be useful for future experiments, enabling a *fidelity loss* like the one described in the previous chapter to be used. However, implementing this is left as future work.

6.3.3 Model and training

The training of the system is mostly unmodified from the previous chapter. The control parameter is changed to be speaker identity. Two representations of speaker identity are used. For the finder loss speaker identity is represented using the index of the speaker in the training set, which is used as a one-hot embedding. For the combiner, the input control parameter takes the form of a pretrained speaker embedding, specifically the ECAPA mel spectrogram embeddings from SpeechBrain[3]. Although the use of mel

[3] `https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb-mel-spec`

spectrogram-derived embeddings means it should be possible to calculate embeddings directly on the output of HFC, without using a vocoder, such experiments are left to future work. The training setup is shown in Figure 6.1. As the finder is not used during inference, and only this part makes use of a speaker index from a known training set, HFC can generalise to unseen speakers.

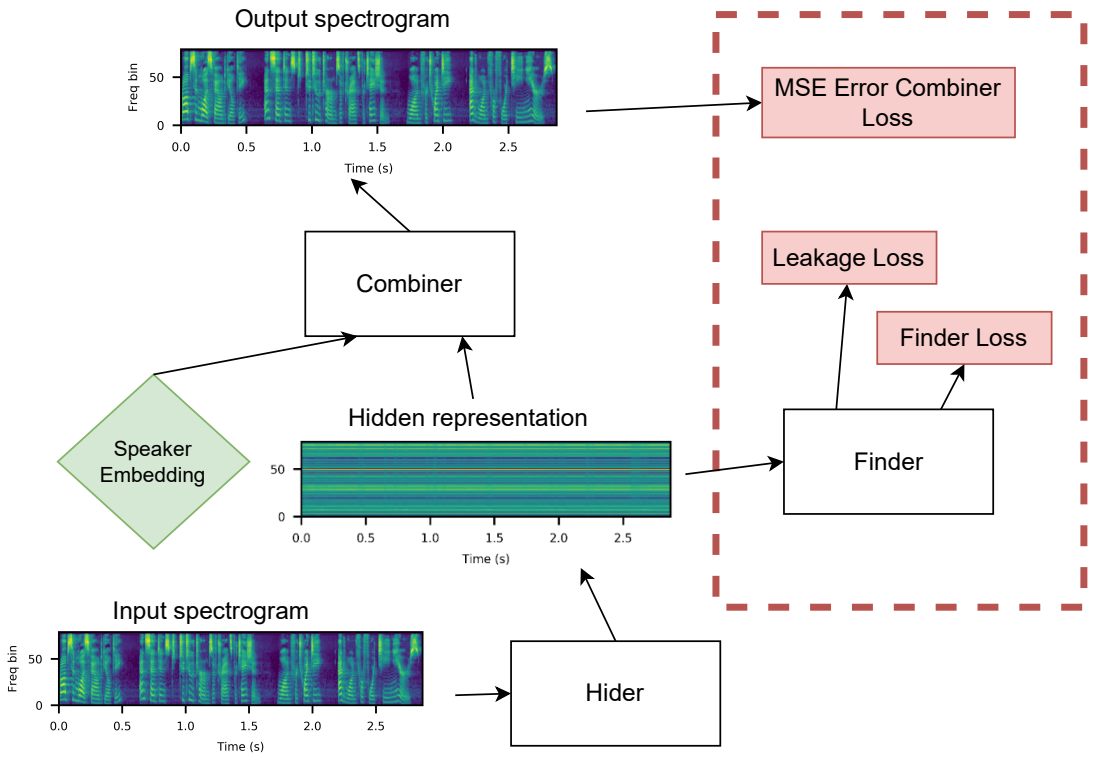


Figure 6.1: Adversarial training setup modified from the previous chapter to indicate that speaker identity is now the control parameter.

6.3.4 Component networks

Hider: Unlike the hider network from HFC-F_0, which uses an RNN, the proposed hider network is based on the generator residual blocks from HiFiGAN V2 Kong et al. (2020) without the upsampling blocks. The 80 input mel spectrogram features per time step are treated as input channels to a 1D convolution of width 7 which increases the number of channels to 128. This is passed into 3 residual blocks, each containing 3 1D convolutions of kernel sizes 3, 7, 11 and dilations 1, 3 and 5 respectively. The residual blocks use a leaky-ReLU discontinuity and weight-norm. The output of these residual blocks is reduced to 80 channels (the dimensionality of our hidden representation) using another 1D convolution with kernel width 7. Our hidden representation is a

time-sequence with the same duration as the input mel spectrogram. A true seq-to-seq approach with a fixed size hidden representation is an interesting avenue to explore in future work as it would allow for concealment of speaker-specific duration patterns.

Finder: The finder network is based on that used in the previous chapter. The hidden representation is passed into a 1D convolution with kernel size 9. The convolution is applied along the width dimension rather than the time dimension. This is passed into a 3 layer GRU of size 200. A final linear layer maps the output to the number of speakers in the dataset. Hyper-parameters (model sizes and finder learning-rate) were found by sweeping, training the finder on ground truth mel spectrograms as a speaker classification model. On the LibriTTS train-clean-100 dataset the model achieved 97.3 % speaker classification accuracy on a held-out validation set. However, the finder was trained from scratch for the HFC training.

Combiner: The combiner network is completely changed from HFC-F_0. HFC-VP uses a model based on FastSpeech 2 with the token encoder transformer removed. Instead a linear layer maps the 80 hidden features to 384 features. The speaker embedding is projected from 192 features to 384 using a linear layer and then summed to the hidden projection. Unlike FastSpeech 2, F_0 and energy projections are not used; incorporating these is left as future work. A four-layer transformer encoder-only network is used followed by a convolutional postnet. During training, the output of both the transformer and the postnet are trained using the combiner loss. The hyper-parameters for the postnet and transformer are as in FastSpeech 2 (Ren et al., 2021), specifically the SpeechBrain implementation.

6.3.5 Hyper-parameters

Initial learning-rates were selected using PyTorch Lightning's automatic learning-rate selection. The hider learning rate was found by training the finder as a speaker identity classifier on ground truth mel spectrograms, and using the automatic learning-rate selection tool for this task. The generator loss learning-rate was found by performing a copy synthesis task using just the combiner on mel spectrograms. Exponential learning-rate decay was used after the first 100 epochs with $\gamma = 0.999$. An initial range for β was chosen using informal experimentation. For $\beta < 5 \times 10^{-2}$ the control did not significantly affect speaker identity. For $\beta > 7 \times 10^{-2}$ training did not converge on acceptably high quality audio output. The hider, finder and combiner networks contain 2.2, 0.87 and 23 million trainable parameters respectively. Samples, source code and

Table 6.1: Losses for Combiner and Leakage modules change based on different values of β. Both decrease as β increases.

Loss	$\beta = .050$	$\beta = .060$	$\beta = .065$	$\beta = .070$
Combiner	0.02962	0.02499	0.02677	0.02475
Leakage	0.005686	0.004225	0.001497	0.001379

trained model files are available online[4]

6.3.6 Anonymisation

The HFC-VP system allows the specification of a target speaker through a pre-trained speaker embedding. The choice of which speaker's embedding to use is non-trivial. Some users requiring anonymisation may want to speak with a consistent voice between utterances, whereas others may not want to be easily associated with speech that they have previously shared. For our experiments I take the *utterance* level approach, such that each utterance is anonymised to a random external speaker using a pool of speaker embeddings from VCTK (Veaux et al., 2013). VCTK was selected due to its high quality and to verify that HFC performs well with target speakers that are unseen in training. Other approaches aim to maximise the difference between target and source speech embeddings (Tomashenko et al., 2020), which could yield improved results in objective analysis, as cases where the target speaker and source speaker happen to be similar could be incorrectly identified as being poorly anonymised. This approach is not used here as it leaks information about the speaker, something that is to be avoided for the anonymisation task: speech anonymised in this way reveals characteristics that are likely to be *opposite* to the source speaker, which could yield sensitive information through triangulation. To evaluate the system in terms of *utility* and *privacy* I use the baseline and benchmarks provided as part of the VoicePrivacy Challenge (VPC) 2024, which are freely available[5].

6.4 Results

A key hyper-parameter is the choice of the value β used for balancing the importance of the adversarial information-hiding loss with the reconstruction loss. Results of the

[4] https://github.com/jacobjwebber/hider_finder_combiner

[5] https://github.com/Voice-Privacy-Challenge/Voice-Privacy-Challenge-2024

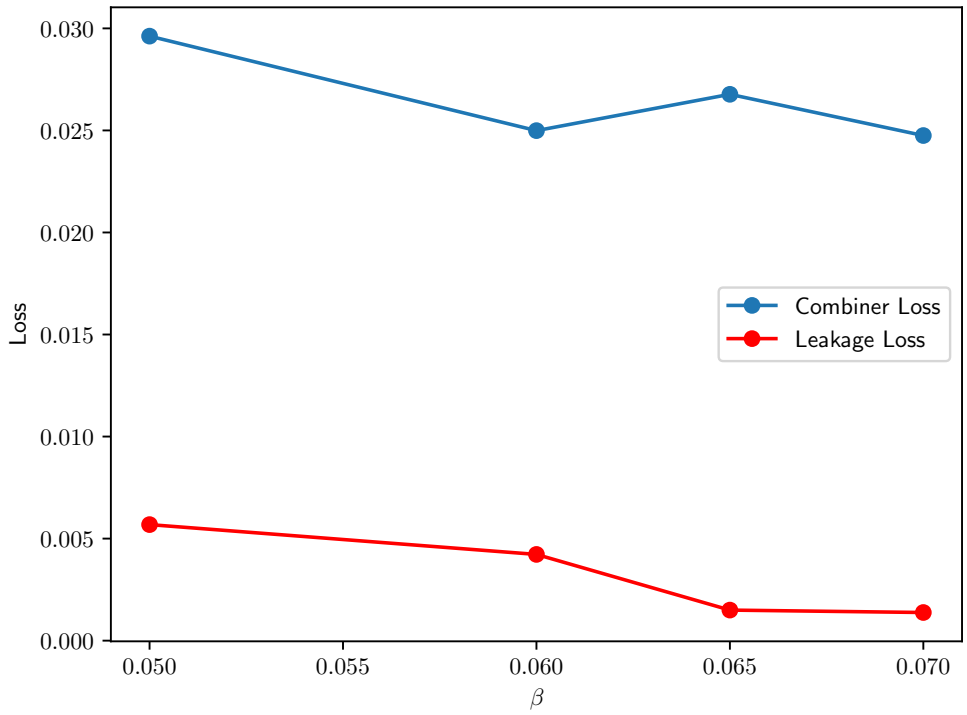


Figure 6.2: Plot showing how losses are affected by values of beta

losses with varying values of this hyper-parameter are shown in Table 6.1 and Figure 6.2. Interestingly these show that *both* losses reduce with β. This differs from an expectation that β parameterises a trade-off between these two variables. It was not possible to achieve stable training with $\beta > 0.07$.

The objective evaluation approaches are taken from the VPC 2024 evaluation plan from Tomashenko et al. (2024), using their freely available implementation. I evaluate 4 HFC systems with varying values of beta and baselines B1 and B2 from the VPC 2024 evaluation plan. I also include HiFiGAN copy synthesis from ground truth mel spectrograms, with no attempt at anonymisation. Baseline B1 uses a neural source-filter approach. Baseline B2 uses signal processing to hide speaker identity, specifically the McAdams coefficient, as described in Patino et al. (2021). The McAdams method works by using the McAdams coefficient to manipulate the spectral envelope of the output signal. The spectral envelope is likely to contain physiological information about the speaker. Due to compute and time constraints, I was unable to recreate results with baseline B1, and therefore unable to verify or fill in missing results from Table 6.2. We use Unweighted Average Recall (UAW) to evaluate the Speech Emotion Recognition (SER) performance of our systems, measuring whether the emotional content of the source is preserved. An ASR-derived word error rate (WER) measures the

Table 6.2: Objective results comparing system performance across tasks for retained lexical content (WER), emotion recognition (UAR), and speaker anonymisation (EER). Down arrows (↓) and up arrows (↑) indicate whether lower or higher values represent better or worse performance, respectively. For anonymisation, higher EER is preferred.

Task	No Anonymisation		VPC Baselines		HFC-VP			
	Original	HiFiGAN	B1	B2	$\beta = .050$	$\beta = .060$	$\beta = .065$	$\beta = .070$
WER Libri Test ↓	1.839	2.024	6.253	10.381	30.976	3.572	6.024	**3.343**
UAR IEMOCAP Test ↑	71.062	56.656	42.314	**53.491**	46.785	49.416	47.104	48.444
EER OA m Libri Test ↑	0.418	2.229	N/A	**28.508**	18.485	14.053	17.817	12.697
EER OA f Libri Test ↑	8.761	8.941	N/A	**33.210**	19.195	12.998	18.428	11.496
EER AA m Libri Test ↑	0.418	0.667	6.682	**26.988**	17.775	9.134	15.366	7.127
EER AA f Libri Test ↑	8.761	8.761	10.584	**32.118**	16.104	11.694	14.597	10.218

retained lexical content. A speaker verification model measures an Equal Error Rate (EER), which is a measure of how well anonymised our output is. As with Tomashenko et al. (2024), the systems are tested with both anonymised (AA) and original (OA) enrollment data. I do not finetune EER models on anonymised speech. The datasets used for evaluation are the IEMOCAP emotion recognition dataset and the LibriSpeech test set. I show these results in Table 6.2.

The WER of all HFC systems, with the exception of $\beta = 0.05$ is better than B2, to the extent that a formal listening test for speech utility preservation is not necessary. In addition to the very strong WER score, I demonstrate the output quality of our system by making samples available online[6]. This is in contrast to baseline system B2 which – although it obtains very good scores for privacy – imposes a high cognitive load on listeners due to its poor signal quality. Our proposed system outperforms the benchmark neural system B1 both in terms of privacy and utility.

A detailed analysis of computational performance is left as future work. However, as the largest component of HFC is a much reduced version of FastSpeech 2 Ren et al. (2021), our system is inevitably faster. The overwhelming majority of compute time used during inference is taken by the HiFiGAN vocoder.

[6] https://hfcvp.github.io/

6.5 Conclusion

I have presented a system that anonymises speech in a principled way, demonstrating strong performance that captures the trade-offs between speech utility and anonymisation performance with objective measures. As our finder is trained to classify between the 904 speakers of LibriTTS, the information that HFC-VP obscures is likely only that which is useful for categorising these speakers. Other attributes (e.g., accent or prosody) may be maintained, particularly as the dataset does not seem to be particularly diverse in terms of speaker demographics. The study of which attributes are affected and which remain untouched opens interesting avenues of future research. It is also possible to modify features other than or in addition to speaker identity by adding classification heads to the finder, or by adding additional finders, as well as by adding extra feature embeddings to the internal representation of the FastSpeech2-style combiner. The ability to redact specific characteristics, with the only requirement being that the characteristic is labelled on some dataset, is an exciting feature of HFC-VP-type systems.

The work in this chapter clearly shows that the HFC approach from the previous chapter can be applied to global parameters that are constant for a whole utterance. It also shows the importance of using a more powerful deep network as a combiner in order to generalise to parameters other than F_0. There are many more experiments that could be done to further verify and improve the results in this chapter. An obvious verification would be to conduct subjective evaluations of the systems presented, which time and budget did not allow.

Part III

Efficient audio synthesis

Chapter 7

Phase and the sign spectrogram

In Section 4.4 I highlighted the use of either linear magnitude spectrograms or mel spectrograms as intermediate representations between modules in automatic audio synthesis systems that are not *end-to-end*. The use of these representations has become established because the are relatively simple to visualise and synthesise using acoustic models. Given that full linear complex spectrograms are a complete representation of an audio signal, the conversion of mel spectrograms into audio signals (vocoding) can be thought of as requiring both upsampling and interpolation in the frequency direction, as well as phase estimation. Phase estimation is required as the magnitude spectrogram does not contain enough information on its own to *coherently* construct a time-domain signal (see Chapter 2).

The work described in this chapter, and in Chapter 8, were started due to problems caused by the poor level of performance achieved by early neural vocoder systems. The work done was largely completed in 2020, although the development of SignLM and SignCFM (introduced in Section 7.3.3) are more recent additions. At that time WaveRNN (Kalchbrenner et al., 2018) had yielded much better computational performance than WaveNet (Shen et al., 2018), with some sacrifice to audio quality. Models like LPCNet (Valin and Skoglund, 2019) had yielded even further computational performance improvements, at the expense of further quality degradation, as well as the loss of some generality.

In Section 4.4 I introduced a taxonomy of vocoders: methods such as LPCNet use explicit periodic sources which are then filtered, which cannot easily generalise to synthesising audio samples with more than one pitched source. Methods such as WaveNet and WaveRNN perform *implicit* phase prediction, by using probabilistic techniques to directly synthesise waveforms.

I was interested in examining the task of vocoding from a more fundamental principled approach, and to do so I aimed to separate the phase prediction task from the upsampling aspect of vocoding, so that these tasks could be analysed and improved separately. After some initial attempts to predict phase angles directly, including the supervision of an (unpublished) MSc project that applied the pix2pix (Isola et al., 2017) system to generate phase from a linear magnitude spectgrogram, it seemed possible that the rotational aspect of phase may be difficult to model. In this chapter I propose using *explicit* phase prediction for the task of vocoding, building on work such as that of Espic et al. (2017) by targeting a representation of phase that I hypothesised would be easier to predict using machine learning methods. This representation is the *sign spectrum*, which fully represents phase angles using just one bit of information.

In addition to wanting to take a principled approach, research into neural vocoding in 2020 were highly practically motivated. At that time, the computational overheads associated with neural vocoding were high. It was therefore common to use techniques that were based on pure signal processing to generate waveforms. The most obvious of which is Griffin-Lim (Griffin and Lim, 1984) which performs magnitude spectrogram inversion. Griffin-Lim uses an iterative process to reconstruct phase information that corresponds to a magnitude spectrogram. This approach was used in systems such as DCTTS (Tachibana et al., 2018) which targeted fast synthesis.

While Griffin-Lim dates back to the 1970s, an important update was made more recently in the discovery of the *Fast* Griffin-Lim technique Perraudin et al. (2013). This makes use of a version of momentum to improve convergence.

The aim of work described in this chapter, and the one that follows, was to combine the benefits of neural vocoding alongside the performance benefits and interpretability of digital signal processing.

7.1 The sign spectrum

The sign spectrum was introduced in Van Hove et al. (1983). In this section I introduce the background: the sign spectrum, and the algorithm for sign spectrum inversion. The sign spectrum inversion algorithm has significant similarities to the Griffin-Lim algorithm. The two algorithms were published months apart, and the papers shared a coauthor.

The sign spectrum fully represents both the magnitude spectrum and original phase of a signal. This phase information is recoverable from only 1 additional bit of infor-

mation to the magnitude spectrum. It was analysed in terms of quality of reproduction and for the ASR task by Loweimi et al. (2020).

7.1.1 Definition

Let $x(n)$ be a 1-D *real*, *causal* sequence of *finite extent*, such that $x(n)$ is zero outside of $0 \leq n \leq L-1$.

Let $X(z)$ and $X(\omega)$ respectively be the z transform and Fourier transform of $x(n)$, which are calculated according to

$$X(z) = \sum_{n=0}^{L-1} x(n) z^{-n} \tag{7.1}$$

$$X(\omega) = \sum_{n=0}^{L-1} x(n) e^{-i\omega} \tag{7.2}$$

The Fourier transform of a real signal will be complex, and can therefore be defined using either a Cartesian or Polar representation.

$$X(\omega) = X_R(\omega) + X_I(\omega) = |X(\omega)| e^{i\theta_X(\omega)} \tag{7.3}$$

I add the condition that $|X(\omega)| > 0$ for all values of ω to ensure that $\theta_X(\omega)$ is always well-defined. In practice, the phase of frequencies of very low amplitude is immaterial. The phase angle is given by

$$\theta_X(\omega) = \text{atan}_2(X_I(\omega), X_R(\omega)) \tag{7.4}$$

and falls in the range $-\pi < \theta_X(\omega) \leq \pi$. The sign spectrogram reduces this phase information down to 1 bit, denoted as $S_x^\alpha(\omega)$, by applying the following conditional

$$S_x^\alpha(\omega) = \begin{cases} +1 & \alpha - \pi \leq \theta_X(\omega) \leq \alpha \\ -1 & \text{otherwise} \end{cases} \tag{7.5}$$

to the original phase $\theta_X(\omega)$, where α is an arbitrary constant between zero and π. This process of converting the angle to assign therefore divides the phase circle equally into two halves, along a line the angle of which is given by α.Van Hove et al. (1983) then go on to introduce the function G_X^α, defined as

$$G_X^\alpha = S_X^\alpha |X(\omega)| \tag{7.6}$$

which is named the *signed Fourier transform*. It is then further proven by Van Hove et al. (1983) that the signed Fourier transform *uniquely specifies* a real signal. This is in contrast to magnitude only spectrum, which can define multiple signals. For example, a signal and its negation will both have the same magnitude spectrum. It is also shown that the sign and magnitude of a sign spectrum are *orthogonal*.

7.1.2 Recovery algorithm

This section introduces the original Sign Spectrum Inversion Algorithm (SSIA) from Van Hove et al. (1983).

The algorithm is similar to Griffin-Lim in that it iteratively converts between time and frequency domains, taking the real part in the time domain and modifying the spectrum to ensure that it remains correct, where correctness is defined as equivalence to the input sign spectrum. The effect of corrupted sign spectra on the process, i.e. situations where the input itself is incorrect, are studied in Section 7.2.2.

Let $X_p(\omega)$ be the estimate of $X(\omega)$ after iteration p. The inverse Fourier transform is applied to $X_p(\omega)$ to yield $x'_p(n)$

$$x'_p(n) = F^{-1}(X_p(\omega)) \tag{7.7}$$

Then, x''_p is derived by taking the real part of x'_p.

$$x''_p(n) = \mathrm{Re}(x'_p(n)) \tag{7.8}$$

This is then converted back into the frequency domain

$$X''_p(\omega) = F(x''_p(n)) \tag{7.9}$$

To derive $X_{p+1}(\omega)$, we transform $X''_p(\omega)$ to ensure that $G^{\alpha}_{X_{p+1}}(\omega) = G^{\alpha}_{X}(\omega)$. This transformation has two parts. Firstly, we replace the magnitude of $X''_p(\omega)$ with the correct magnitude. Secondly, we reflect the phase angle of $X''_p(\omega)$ about the line made through the origin of the complex plane at angle α where necessary to ensure the sign, according to Equation 7.5 is correct.

$$X_{p+1} = \begin{cases} |X(\omega)|e^{i\theta_{X''_p}(\omega)} & \text{if } S^{\alpha}_{x''_p}(\omega) = S^{\alpha}_{x}(\omega) \\ |X(\omega)|e^{i(2\alpha-\theta_{X''_p}(\omega))} & \text{if } S^{\alpha}_{x''_p}(\omega) = -S^{\alpha}_{x}(\omega) \end{cases} \tag{7.10}$$

While this describes the iteration process, we initialise the phase at the zeroth iteration as follows

$$\theta_{x_0}(\omega) = \begin{cases} \alpha - \frac{\pi}{2} & \text{if } S_x^{\alpha}(\omega) = 1 \\ \alpha + \frac{\pi}{2} & \text{if } S_x^{\alpha}(\omega) = -1 \end{cases} \tag{7.11}$$

so that

$$X_0(\omega) = |X(\omega)| e^{i\theta_{x_0}(\omega)}. \tag{7.12}$$

Note that owing to Equation 7.11, initialising the phase with $\theta_0(\omega)$ has no effect when $\alpha = \frac{\pi}{2}$.

7.2 Fast sign spectrogram inversion

This section applies the momentum approach to spectrogram inversion introduced for Griffin-Lim in Section 4.4.1.1 and applies it to the signed spectrogram. In doing so I present the Fast Sign Spectrum Inversion Algorithm (FSSIA), which is an original contribution of this thesis. The fast algorithm developed by Perraudin et al. (2013) has become widely used various audio synthesis tasks, including TTS (Tachibana et al., 2018). It exceeds the base Griffin-Lim algorithm in both audio quality and the number of iterations required to converge. The FSSIA aims to apply these improvements to a similar task.

7.2.1 The improved algorithm

The improved algorithm functions mostly as the SSIA does, with the update in Equation 7.5 replaced by 7.14. In order to add momentum, the update yielding $X_{p+1}(\omega)$ is extended to yield $\tilde{X}_{p+1}(\omega)$, which is then used in place of the former.

$$\Delta(\omega) = X_{p+1}(\omega) - \tilde{X}_p(\omega) \tag{7.13}$$

$$\tilde{X}_{p+1}(\omega) = X_{p+1}(\omega) + \beta\Delta(\omega) \tag{7.14}$$

where β is a hyper-parameter.

Before turning to experimental results, I apply one further extension to the sign spectrum as discussed in Section 7.1. While the originally proposed sign spectrum was based on a single Fourier transform of a full signal, here I use the STFT, as discussed in Section 2.2, which applies the Fourier transform to a sequence of windowed sections

of the signal. As before, we distinguish between a *sign spectrum*, where the transform is applied to a full signal, and the *sign spectrogram*, where the transform is applied to a sequence of overlapping windows. These windows overlap because, typically, the STFT is configured so that the hop size is smaller than the window size, which results in redundancy in the STFT. The effect of this redundancy on the sign spectrogram algorithm is left as future work.

7.2.2 FSSIA Results

To verify the usefulness of the fast sign spectrogram inversion algorithm, experiments were conducted to measure the convergence of the algorithm when applied to a range of audio signals. This convergence is measured with respect to the PESQ score, specifically the PESQ-FR (Rix et al., 2001), as discussed in Section 4.5.2. This version of PESQ depends on a reference signal being available. In this case, that reference is the signal before the sign spectrogram transform is applied.

Our source samples are taken from the librosa (McFee et al., 2015) toolkit, which contains the 12 example audio files shown in Figure 7.1. The content of these files ranges from music (including classical, ragtime and pop) as well as 3 speech samples from LibriSpeech (Zen et al., 2019).

Samples were resampled to 16 kHz. A window size of 1024 and a hop size of 256 were selected. The FSSIA is applied with 100 iterations to recover the signal. Figure 7.1 shows that on all samples the FSSIA outperforms the original sign spectrum inversion algorithm. As with the FGLA when compared with the GLA, the PESQ for FSSIA score goes higher in fewer iterations than SSIA. Importantly for this thesis, the libri1, libri2 and libri3 (all speech samples) PESQ scores appear to converge at noticeably higher PESQ.

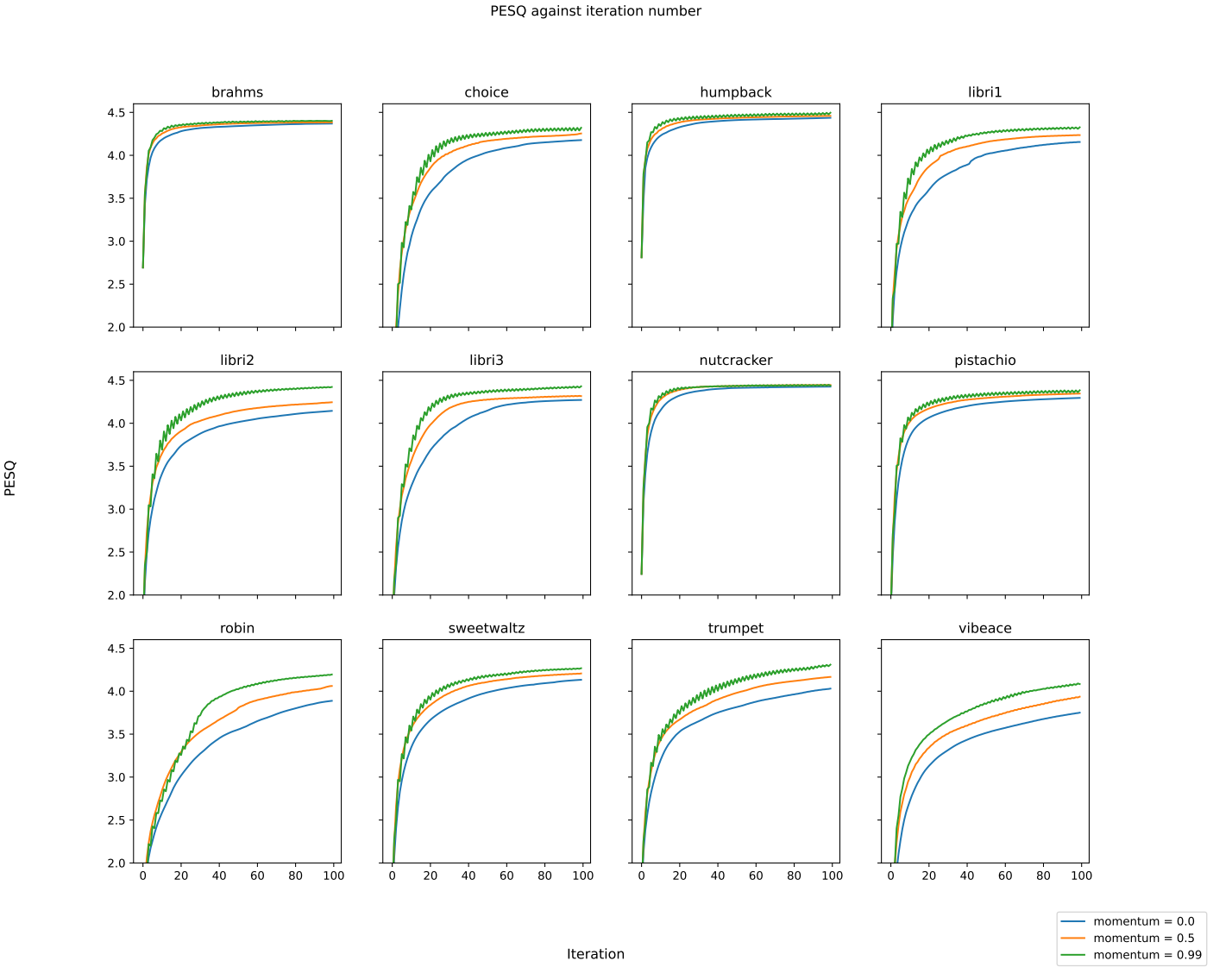


Figure 7.1: PESQ score against iteration number for iterative recovery of a real audio signal from a sign spectrogram. Audio samples are taken from the LibRosa examples, and are resampled to 16 kHz. Where momentum is equal to zero this is equivalent to the original algorithm for signal recovery from the sign spectrum.

7.2.3 Recovering from a noisy signal

As I am interested in applying sign spectrograms to the task of phase prediction, the aim of FSSIA presented here is to enable high quality synthesis from synthetic signs. As artificial signals may contain inconsistencies or inaccuracies, it is therefore necessary to consider the effect that noise in the input signs can have on the performance on the algorithm. In this section I measure the effect that corrupting the sign informa-

tion, i.e. flipping the sign with a given probability, has on signal reconstruction. To do this I make use of the same samples and FSSIA hyperparameters as in Section 7.2.2, applying the FSSIA algorithm with a momentum value of 0.99.

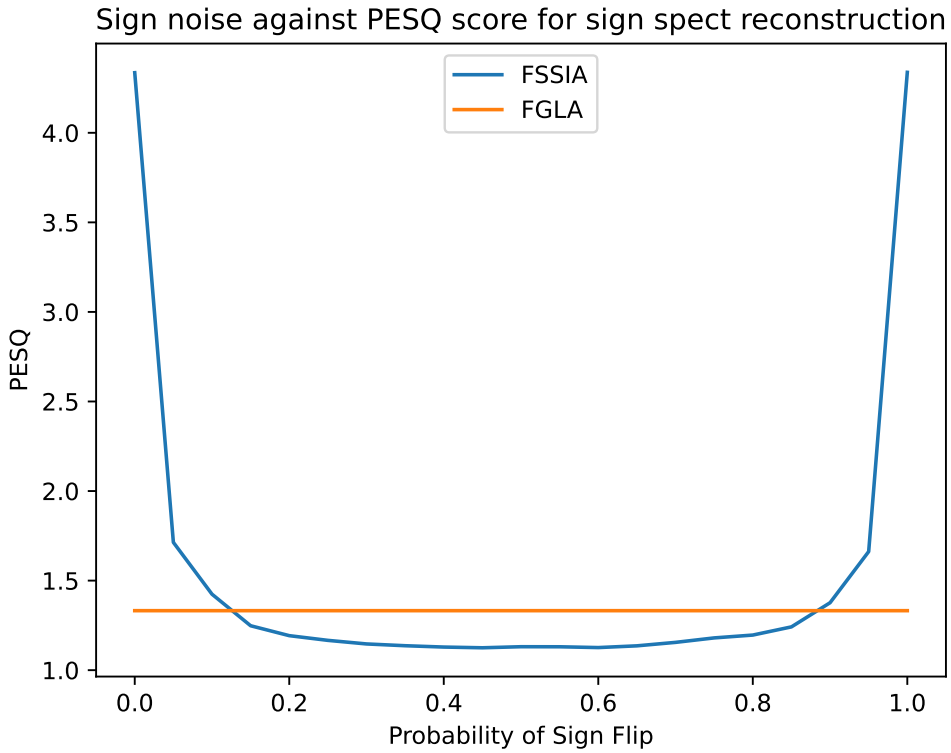


Figure 7.2: Showing PESQ score against probability of multiplying each bin by -1. This figure shows the effect of noise in the sign signal, as well as that inverting the sign in the sign spectrum has no perceptual effect on the reconstructed signal.

Figure 7.2 shows the effect of flipping signs in the sign spectrogram. In each bin and at each frame timestep, values are multiplied by -1 with a range of probabilities between 0 and 1. In the case where this probability is 1, we simply multiply the whole sign spectrogram by -1.

This figure illustrates two points. Firstly, when the sign information is significantly corrupted, the FSSIA yields lower quality signals than FGLA. This is likely due to the continual reintroduction of sign information during the iterations, as opposed to the FGLA, which starts with a random phase but is then unguided by this (see Section 4.4.1.1). In addition, we show that signals with opposite signs are perceptually equivalent, at least in terms of PESQ score.

An analysis of flipping the sign with probability 1 (i.e. muliplying the sign spec-

trogram by -1) can be performed analytically. By doing this it is possible to show an equivalence between multiplying a signal by -1 in the time domain, the complex spectrogram domain, and the sign spectrogram domain. From the linearity for the Fourier transform, the negative of a signal has the negative spectrum. Using Equation 7.3 and $e^{i\pi} = -1$ we can show

$$-X(\omega) = |X(\omega)|e^{i(\theta_x(\omega)+\pi)} \tag{7.15}$$

which means

$$\theta_{-X} = \theta_x + \pi. \tag{7.16}$$

From 7.5 we therefore see that

$$S^{\alpha}_{-x}(\omega) = -S^{\alpha}_{x}(\omega) \tag{7.17}$$

.

The PESQ score where $p(\text{flip}) = 1$ is the same as where $p(\text{flip}) = 0$ because the sign of a audio signal is not perceivable. We can therefore see that, although a sign spectrogram uniquely defines a signal, it does not uniquely define a *perceived* signal. For the tasks that the work in this chapter was intended to be applied (speech generation) to we are happy to output any of the set of signals that are perceptually equivalent in the audio domain.

7.2.4 A motivation for generative modelling

The following claims are made

- A sign spectrum uniquely specifies a time-domain signal (Van Hove et al., 1983)
- One magnitude spectrum can be rendered into many time-domain signals when rendered as audio using FSSIA with different signs. These time-domain signals are not perceptually equivalent according to PESQ or informal listening tests.
- Multiple time-domain signals are perceptually equivalent and acceptable, as there are some changes in the time domain that human ears cannot perceive.
- The magnitude spectrum of a signal and its associated signs contain *orthogonal* information (Van Hove et al., 1983).

These points suggest that generative modelling could be used to predict appropriate signs for a magnitude spectrum. Generative modelling, as introduced in Chapter 3.2, is particularly well suited to tasks of this nature, where there are many correct solutions that are identifiable from an input. These appropriate solutions can be learned.

7.3 Sign prediction

In this section I introduce three attempts to apply generative modelling to the task of predicting a sign spectrogram from a linear magnitude spectrogram. In turn I apply GANs, autoregressive modelling and Conditional Flow Matching (CFM) to the task. This approach is motivated by the unsuitability of regression modelling for the task. This is motivated by the claim that magnitude and sign information are orthogonal but *not* independent, at least not independent in specific domains, such as speech audio signals. This lack of independence is shown by the fact that PESQ, and the human ear (in informal listening tests), could easily identify inappropriate sign information after the application of the FSSIA. So, while the original signs are not recoverable from magnitude, there is a constrained subset of signs that sound good with any magnitude.

The alternative case is that any known good sign spectrum will work with any known good magnitude spectrum, which is not plausible. I argue that for each magnitude spectrogram that represents high quality speech audio, there is a subset of sign spectra that when paired with this magnitude spectrogram will produce audio that is perceived as sounding good. It was hoped that generative models could sample from the subset of good sign spectra.

For all the experiments below conditional generative models are used. The systems are trained using speech samples from VCTK, downsampled to 22.05 kHz. For the target output, positive signs are encoded as 1 and negative signs are encoded as 0 or -1 (depending on the model). The conditioning is given by the log-magnitude linear spectrogram. The spectrogram was constructed with a window size of 2048 samples, and a hop size of 512 samples. A validation set was held out to conduct PESQ score evaluation. Learning rates for SignLM and SignCFM are derived from PyTorch Lightning's learning rate optimisation tool.

7.3.1 SignGAN

The first attempt to generate sign spectra was using a conditional GAN. The *G* component of SignGAN consists of a convolutional component and a recurrent component. The convolutional deep residual net was constructed from 100 basic blocks, with each of these consisting of a two dimensional convolution followed by a batch norm and leaky ReLU activation function, with the addition of a residual. The output of this residual component was passed into an LSTM which was recurrent along the time dimension of the linear spectrogram. The *D* component was a basic patchGAN discriminator using the open-source implementation associated with the pix2pix paper from Isola et al. (2017). However, contrary to Isola et al. (2017), the GAN set up is a classic "vanilla" GAN as described in Section 3.2.1. In this case the disciminator is simply tasked with classifying samples as real or fake.

The GAN-based approach was not successful in generating good-sounding samples. Despite a number of configurations and ablations being tried, including varying learning rates of the two optimisers, varying the number of layers in the residual network, using one dimensional convolutions and treating the frequency axis as channels, and removing the recurrent component, the signs generated by SignGAN when used with the FSSIA led to PESQ scores never reaching more than 1.4, with these scores showing no improvement during training. As discussed in Section 3.2, GANs are inherently unstable and difficult to train. In the case of sign spectrograms it is not possible to use an auxiliary loss directly on output signs, as non-generative regression or classifier losses are not suitable due to the orthogonality of magnitude and sign information.

7.3.2 SignLM

After the lack of success with SignGAN, I attempted to apply an autoregressive model to the task of sign prediction. The model was constructed as a conditional autoregressive model and, as with SignGAN, the linear log magnitude spectrogram was used as conditioning. The linear log magnitude spectrogram is passed into a two dimensional convolutional network consisting of 5 of the basic blocks introduced in Section 7.3.1, each of which outputs 8 channels at each frequency and time coordinate. The motivation for using 2 dimensional convolutions is so that at each timestep and frequency bin, the autoregressive model can be conditioned on information from adjacent timesteps and frequencies (up to the limit of the convolutional receptive field). The

autoregressive model itself consists of a 10 layer LSTM. The LSTM is autoregressive along the frequency dimension, with timesteps modelled independently. The decision to model timesteps independently is motivated by the fact that, unlike Griffin-Lim, the FSSIA does not need STFT overlapping redundancy to operate, meaning a series of good signs should be possible to predict independently for each timestep. This approach could have led to overlap-add artefacts. The LSTM predicts a logit for each frequency bin in turn. This is converted into a probability of positive sign using a sigmoid function, before a sign is sampled from a Bernoulli distribution. The model was trained using teacher forcing and a log-likelihood maximisation-derived loss (binary cross-entropy). During inference the synthesis is iterative along the frequency axis, but can be parellelised along the time axis.

As with SignGAN, it was not possible to get high-quality output from SignLM. The PESQ scores from output did not improve consistently during training and did not exceed 1.4. In addition, the binary cross-entropy loss did not converge. Informal listening tests showed that output samples were of very poor quality. Further experiments involving more powerful conditioning and recurrent networks could be applied, but time and compute was not available to do this at the time of writing.

7.3.3 SignCFM

The third and final approach tested in this chapter is the application of Conditional Flow Matching (CFM) models to the task of sign prediction. As before, linear log-magnitude spectrograms are used as the condition. The model chosen to parameterise the *flow* vector was taken from Matcha-TTS from Mehta et al. (2024), using their open-source implementation. This uses a 1d convolutional U-Net model. The only modification required was to change the number of groups for the grouped convolutions to be a factor (5) of the number of frequency bins (1025). The number of timesteps of t (see Section 3.2.3) to integrate through for the flow matching algortihm was not highly important to experimental results, although an informal sweep showed 10 was a suitable value. Unlike SignGAN and SignLM, the PESQ score on the validation set did show some improvement during training.

Figure 7.3 shows that the PESQ score improves with training step. Training was stopped when this stopped improving. However, while the PESQ score did rise a little, informal listening tests showed severe phase artefacts in the output audio derived using the FSSIA. In informal listening tests, the samples output by SignCFM sound con-

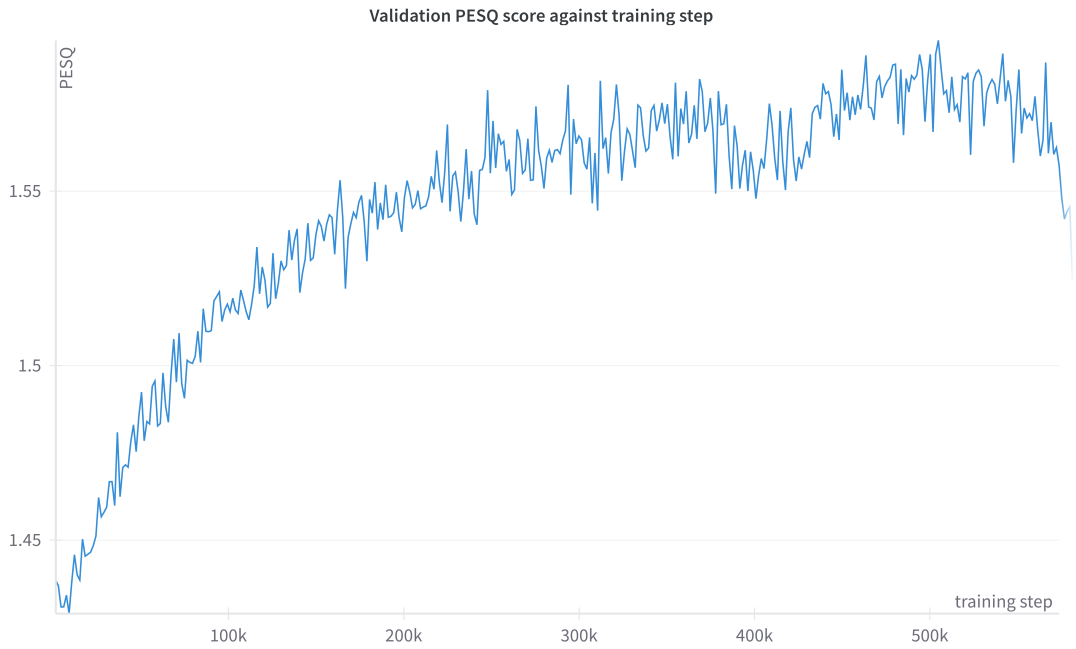


Figure 7.3: Plot showing PESQ against training step averaged over 200 validation set samples during training of SignCFM

siderably worse than Griffin-Lim samples, despite having similar PESQ scores. The analysis of an objective perceptual quality metrics in terms of how they are impacted by phase artefacts is left as future work.

7.4 Conclusions and summary

This chapter describes the novel idea of adding momentum, an optimisation technique commonly used in both machine learning and signal processing, to the sign spectrum inversion algorithm. The sign spectrogram is a fully lossless representation of a real signal that stores the magntitude spectrogram along with one sign bit per value. The code for all experiments in this chapter, along with code for sign spectrum and FSSIA calculation are available online[1]. This is implemented in Python using the PyTorch library, so it can run on GPU or CPU, and can also support automatic differentiation.

I have highlighted the potential appropriateness of generative modelling in predicting phase from magnitude spectrograms, and conducted some early experiments to attempt to do this. Although at this stage it was not possible to generate high quality audio using generative modelling and the FSSIA, it is hoped that the work in this

[1] `https://github.com/jacobjwebber/sign-spect`

chapter could be developed further in order to do so. It is possible that a reason that the reason that it is so hard to generate good signs using generative modelling is that the tolerance to improper signs, as shown in Figure 7.2 is too small. One approach that may improve results is the use of a suitable set of auxilliary losses. PESQ was chosen as a reasonable approximation of output quality, but was only used as a validation loss. There could be benefit in finding a set of losses that can encode the amount of phase artefacts and use these as training losses. This is made possible through the use of a differentiable implementaion of the FSSIA, but is left as future work. The difficulties in explicit phase synthesis from magnitude spectrograms motivates the next chapter, where learned representations are used that encode phase information alongside magnitude information.

Chapter 8

Autovocoder

8.1 Introduction and background

This chapter describes the development of the *autovocoder* system, covering two sets of experiments. The *autovocoder* system was born of a frustration with typical neural vocoders as used for speech synthesis, like those used in the Hider-Finder-Combiner experiments of Chapter 5. *Autovocoder* is designed to be an alternative to such systems as a final stage of waveform synthesis. As *autovocoder* is a high quality audio autoencoder, it may well have other uses. Recent work in generative modelling for audio synthesis (Pascual et al., 2022) shows the uses of waveform-based autoencoders in generative modelling using diffusion models. The Wav2vec 2.0 by Baevski et al. (2020) system also shows the value of learned audio representations for the ASR task.

Generating a natural-sounding speech waveform is a challenging task. For this reason, text-to-speech (TTS) systems are usually divided into *acoustic modelling* to map input text to an intermediate acoustic representation, which is then input to a *waveform generator*. The acoustic representation is, – in almost all state-of-the-art systems – a mel spectrogram. The frame rate of this spectrogram is much lower than the waveform sampling rate, which simplifies the task for the acoustic model but creates a challenge for the waveform generator. Other tasks, such as Voice Conversion (VC), also involve the generation of a waveform from a spectrogram.

Beginning with WaveNet (Shen et al., 2018), deep learning has achieved extremely natural-sounding waveform generation, but many such *neural vocoders* are autoregressive (to model the autocorrelated nature of speech waveforms) which is computationally expensive, not amenable to parallelisation, and precludes on-device synthesis. More recent neural vocoders reduce computational cost either by leveraging knowl-

Table 8.1: Comparing encoding and decoding of the proposed system with the current state-of-the-art (SOTA).

System	Acoustic feature extraction (encoder)	Waveform generation (decoder)
SOTA	fast DSP	slow neural network
Autovocoder	fast DDSP + simple neural network	simple neural network + fast DDSP

edge of speech production, for example with the use of traditional signal-processing ideas in Neural Source-Filter (NSF) models (Wang et al., 2020) and LPCNet (Valin and Skoglund, 2019), or by making use of generative modelling to remove the autoregression, as in HiFi-GAN (Kong et al., 2020).

The system described in this chapter, *autovocoder*, generates high-quality waveforms very efficiently from a frame-based representation of similar size to the mel spectrogram used in a typical TTS or VC system. The system uses machine learning to define that acoustic representation, then leverages fast, differentiable DSP (DDSP) operations for decoding. This reverses the conventional arrangement, where fast signal processing is used for the one-off task of encoding waveforms into acoustic representations (mel spectrograms), forcing deployed systems to use slow, difficult to parallelise, and computationally expensive models to generate from these features every time a waveform is synthesised: Table 8.1

Autovocoder combines the respective strengths of DSP and deep learning. DSP is fast and efficient, while machine learning can yield rich and maximally informative representations (for a given dimensionality). The mel spectrogram only represents spectral magnitudes and discards phase, even though this is known to be beneficial for both speech recognition (Loweimi et al., 2021) and speech synthesis (Espic et al., 2017). The learned representation of *autovocoder* is not required to discard phase. This avoids the need to recreate phase exclusively from magnitude content, which is challenging for machine learning-based systems (see Chapter 7).

Unlike WaveNet and other neural waveform synthesisers that are often called vocoders, *autovocoder* is a true vocoder because it both encodes from a waveform to an acoustic representation, and decodes that back to a waveform. *Autovocoder* has a simple and very fast decoder which relies on a sufficient representation of the signal. *Autovocoder* therefore learns the acoustic representation. Machine-learning is well-suited to learning representations (i.e., sufficient for reconstruction), whereas signal-processing-derived representations are generally lossy. Signal processing representa-

tions generally drop phase completely, because it is both tricky to estimate from waveforms and difficult for the acoustic model to predict from phones. However, it has been shown by Espic et al. (2017) that modelling *appropriately represented* phase information is possible and can improve TTS performance. Phase has also been shown by Loweimi et al. (2021) to contain important information for Automatic Speech Recognition. *Autovocoder* will learn to efficiently encode phase. Chapter 7 examines on a more theoretical level the minimal amount of information required to fully encode phase. For signal generation, we use the inverse discrete short-time Fourier transform (iSTFT) and overlap-add, which can be executed extremely efficiently on modern hardware.

This chapter describes the background and motivation for the development of *autovocoder* in detail. It then describes two main systems. The first of these, AV_{simple}, is an earlier prototype trained with a simple loss function. The latter model, AV_{GAN}, is architecturally similar but makes use of additional adversarial and mel spectrogram losses. The results of experiments with the latter system in particular show that the proposed approach offers high-quality speech waveform generation while also being extremely fast. This chapter was developed from work published by Webber et al. (2023).

8.1.1 Parallelism and autoregression

The rise of machine learning for image, speech, and language processing has been enabled by rapid advances in parallel computing performance. Most modern machine learning is highly parallelisable, but algorithms parallelise to different degrees.

Deep learning can largely be reduced to a sequence of matrix multiplications, which can be executed extremely efficiently on modern parallel hardware such as GPUs and TPUs. However, some kinds of machine learning algorithm are less good at taking advantage of this hardware.

Autoregressive sequence models require values from prior steps in the computation: output must be computed sequentially and not in parallel. Conversely, *embarrassingly* parallel algorithms require no communication between decomposable computations. Many recent TTS acoustic models are not autoregressive (Łańcucki, 2021; Ren et al., 2019); they are fast because a sequence can be processed in parallel without depending on prior steps. This allows all computations in a long sequence to be distributed across many processing units and processed in unison.

These non-autoregressive TTS models still output a mel spectrogram and therefore

a vocoder is required. Each of the frames in these sequences is fairly self-contained, containing information specific to the time window that it represents.

Non-autoregressive waveform synthesis has proved more elusive. The human auditory system does not perceive the individual samples in a waveform, but rather groups of frequencies, similar to those represented by the mel spectrogram. These frequencies cannot be derived from individual samples, but only by large windows of samples, along with positional information as to the order of these samples.

Autoregression is a straightforward, simple approach for modelling between-sample dependencies, and was used by the first machine-learned waveform generator, WaveNet (van den Oord et al., 2016; Tamamori et al., 2017). WaveNet suffered very significant performance issues, which were to some extent addressed by Parallel WaveNet (van den Oord et al., 2018). More recent systems like WaveRNN (Kalchbrenner et al., 2018) have achieved better computational performance through more strategic use of autoregressive elements and careful engineering. Even more recently, sophisticated generative models such as GANs have enabled parallel waveform synthesis (Kong et al., 2020). However, these systems still rely on learned methods to generate highly correlated signal samples, which is inefficient with respect to both training data and compute.

8.1.2 Vocoding based on explicit harmonic synthesis

Some recent systems have considered generating waveforms with appropriate correlations by explicitly generating the periodic content. The Neural Source-Filter (NSF) model (Wang et al., 2020), employs a series of sinewave generators that generate harmonics. This requires a known input F_0. Artefacts for speech with less harmonicity in the source (e.g., breathy voice) were addressed in (Wang and Yamagishi, 2020) by using cyclic noise rather than sinewaves for the source. Another important example of DSP-informed waveform synthesis for TTS is LPCNet (Valin and Skoglund, 2019), which employs an RNN to drive a waveform generator based on Linear Predictive Coding. The speech signal is encoded into 20 DSP-derived parameters at a frame rate of 10 ms. This was subsequently used to generate speech alongside an LSTM-based speech syntheser in (Kons et al., 2019).

These models share the assumption that signals contain only one pitched source, which could limit flexibility, as this is built into the the model. These systems trace a lineage back to traditional DSP-based source-filter vocoders (Airaksinen et al., 2018),

notably STRAIGHT (Kawahara et al., 1999). Both NSF and LPCNet differ from the work in this chapter by continuing to depend on purely DSP-derived features.

While iSTFTNet (Kaneko et al., 2022), developed in parallel with this work, also uses the iSTFT for synthesis, it has key differences in that it employs a much smaller FFT and hop size (16 vs. 1024 and 4 vs. 256 respectively), which is likely to limit computational performance. These settings may be necessary because it still depends on the mel spectrogram. Puffin, by Watts et al. (2023), another vocoder developed in parallel to ours, relies on a pitch synchronous iSTFT operation to support low-complexity higher sampling-rate generation.

Recent work, such as wav2vec 2.0 (Baevski et al., 2020), has used self-supervised learned audio representations. These have been used in TTS (Du et al., 2022; Siuzdak et al., 2022), but they necessitate a *more* complex waveform decoder, plus F_0 input. The representations proposed in this chapter are designed to be efficiently invertible.

Autovocoder is trained as an autoencoder. Other systems have used autoencoders for unsupervised speech feature learning (Chorowski et al., 2019); however, they emphasise the use of the features for other tasks, and are not designed to perform very fast decoding back to a waveform. Unlike neural speech codecs such as SoundStream (Zeghidour et al., 2022), we do not target extremely low bit rates, but rather a frame-based representation that could directly replace the mel spectrogram in applications including TTS.

8.2 Baseline Autovocoder

This section describes the AV_{simple} system, which was later developed into the AV_{GAN} system described in Section 8.3. Each system was subjected to separate qualitative evaluation. A combined evaluation in terms of computational performance is given in Section 8.4.

8.2.1 Architecture

8.2.1.1 Encoder and decoder

The encoder first converts from time domain to frequency domain using a differentiable implementation of the STFT. The resulting complex spectrum is then used to derive four spectral components: magnitude, phase, real and imaginary. This use of redundant components is discussed in Section 8.2.1.2. These four spectral components

are stacked, with each treated as a channel, and fed into a purely convolutional residual network made up of what we term *basic blocks*. Each such block consists of two 2D convolutional layers of width 3, followed by a 2D batch norm and a ReLU nonlinearity. This basic block also applies a residual, by summing the input with the output, but only if the number of input channels is the same as the number of output channels.

The residual net used in the current *autovocoder* architecture consists of 11 basic blocks, the first five having 4 input and output channels, the middle one having 4 input channels but 1 output channel, with the remaining 5 blocks having 1 channel in and out. That single channel output is fed into a single linear layer that reduces the dimensionality per timestep from $(\text{windowsize})/2+1$ to our *representation size*, which we initially chose to be 128. This is the dimensionality of a single frame of the learned representation, which we chose to be similar to the frequency dimension of a typical mel spectrogram in the experiments.

Our decoder architecture is a mirror of the encoder, with the components applied in reverse order and with similar hyperparameters. The full architecture is shown in Figure 8.1. Note that the system has no autoregression at any level. All frames are processed at once by the network, and the subsequent overlap-add to assemble the complete waveform is performed by the differentiable iSTFT function of PyTorch (Paszke et al., 2019).

8.2.1.2 Redundant representations of complex numbers

Providing redundant representations of the complex spectrogram to the encoder allows the model to learn how best to represent magnitude and phase. For example, *autovocoder* could potentially learn that, given a magnitude spectrogram, phase can be very efficiently represented in terms of space, as in the sign spectrogram introduced in the previous chapter.

In the decoder of AV_{simple}, the residual network also outputs 4 channels, which are taken to represent real, imaginary, phase and magnitude parts of the spectrum. These are converted into two complex spectrograms, before an average of the two is taken in the Cartesian domain. Later experiments described in this chapter show that generating a Cartesian representation alone yields improved performance over polar or averaged representations. It is not immediately obvious why this is the case.

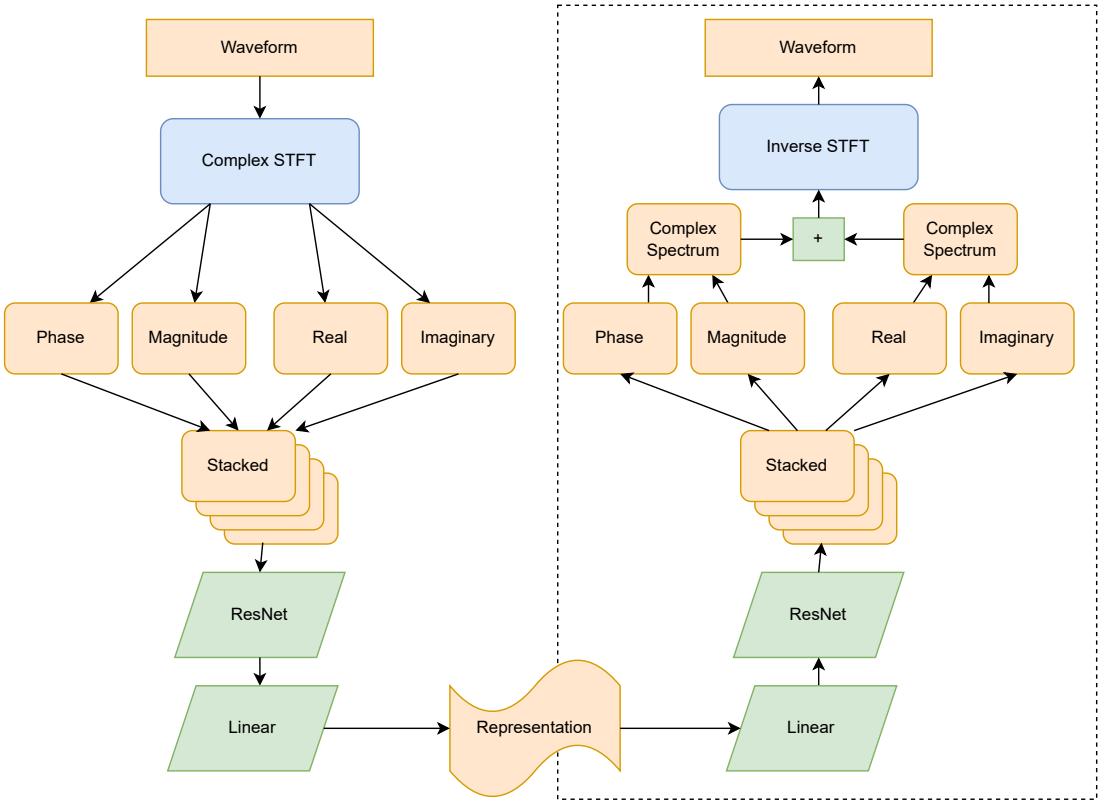


Figure 8.1: Autovocoder architecture. Dashed box shows decoder

8.2.2 Training setup

The system is trained with the use of only one loss: the mean squared error (MSE) of the output waveform in the time domain. This implies that the phase must be sufficiently well represented in the learned vocoder embedding. The model was trained as a *denoising autoencoder* (Vincent et al., 2008). A dropout factor of 10% was used on the embedding during training to increase robustness, with a view to a subsequent application where these features may be generated by a TTS acoustic model. The system was implemented using PyTorch (Paszke et al., 2019) and trained using the Adam optimiser.

8.2.3 Evaluation

AV_{simple} was evaluated against a range of comparable systems. Griffin-Lim (Griffin and Lim, 1984) is an iterative algorithm for phase recovery that converts magnitude spectrograms into phase-coherent time-domain signals. We used the librosa (McFee et al., 2015) implementation of Griffin-Lim, which uses 32 iterations. This was used with the same frame rate (i.e. hop size) as the STFT used by *autovocoder* and a window

size of 1024. Griffin-Lim uses full resolution ground truth magnitude spectrograms (513 bins), without mel-scale dimensionality reduction.

An open source implementation of WaveRNN[1] was used with a pretrained checkpoint that was trained on the same dataset as *autovocoder*.

The particular configuration of AV_{simple} evaluated here was determined using a combination of informal listening and the value of the MSE loss on a validation set and is as follows. The STFT and iSTFT hop size is 256 with a window size of 2048 samples. A representation dimensionality of 128 was chosen, slightly larger than the most common value of 80 bins for mel spectrogram bins (although 128 is also in use) because this value was found to perform better. 128 is still a substantial reduction of dimensionality with respect to the 1025 complex spectrum bins.

Autovocoder was trained on the single-speaker LJ Speech (Ito and Johnson, 2017) dataset with waveforms at a sample rate of 22.05 kHz. Evaluations used a test section of the dataset, which was not used during training. The system was trained for approximately 2 days on a single NVIDIA RTX 2080 Ti GPU.

In this chapter, we evaluate copy synthesis. In the case of *autovocoder* this means a complete pass through both the encoder and decoder parts of the system. For the comparison systems, it means synthesising from ground truth spectral features extracted from waveforms using DSP.

8.2.4 Listening tests

The AV_{simple} system was compared to relevant baselines using a MUSHRA (ITU-R, 2015) evaluation. Each MUSHRA screen presented 5 stimuli to the listener for evaluation. These were *autovocoder*, WaveRNN, Griffin-Lim, the original waveform and an anchor. The anchor was the reference, low-pass filtered at 3.5 kHz using an order 10 Butterworth filter implemented in SciPy (Virtanen et al., 2020).

25 listeners were recruited using Prolific[2]. A Qualtrics survey was generated using the *Qualtreats* tool[3] which comprise 10 MUSHRA screens. Each screen presented the reference, then asked the listener to rate the *quality* of the five samples. They were instructed to find the reference within those five samples and assign it a score of 100, whilst rating all samples. Responses from all MUSHRA screens where the hidden reference was not assigned a score of 100 were omitted from analyses. For any

[1] https://github.com/fatchord/WaveRNN

[2] https://www.prolific.co/

[3] https://github.com/CSTR-Edinburgh/qualtreats

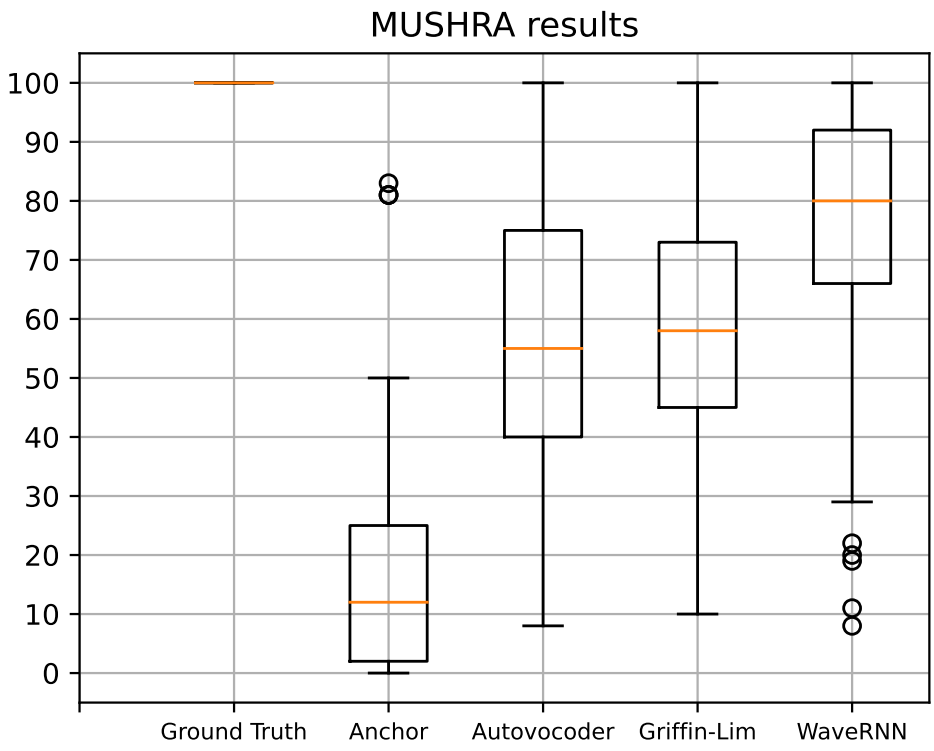


Figure 8.2: Results of the MUSHRA evaluation of AV_{simple}, labelled as *autovocoder*. All pairs apart from Griffin-Lim and *Autovocoder* are significantly different at $p > 0.01$

listener who failed to assign 100 to the ground truth sample less than 60% of the time (which was the mean across all listeners), all responses were omitted from subsequent analysis. This left 12 respondents and 105 MUSHRA screens for analysis. Figure 8.2 shows the results. A t-test used to evaluate statistical significance found that scores for all pairs except Griffin-Lim vs. *autovocoder* were significantly different at $p > 0.01$.

Griffin-Lim system was rated as no different in quality to *autovocoder*, despite the fact that is was synthesising from high-dimensionality representations (1025 features per time-step compared to the 128 of *autovocoder*).

8.3 Extending Autovocoder with an adversarial loss

Although the results of Section 8.2.2 show that *autovocoder* generates quality audio, it was necessary to improve performance in the copy synthesis task to be competitive with modern neural vocoders. This section outlines a number of changes that were made to improve the performance of AV_{simple}. The resulting system is referred to as

AV_{GAN}.

Like AV_{simple}, AV_{GAN} is trained as a denoising autoencoder. However, AV_{GAN} training also adopts the losses from HiFi-GAN (Kong et al., 2020): a mel spectrogram loss and two adversarial losses from multi-scale and multi-period discriminators. The multi-period discriminator works by using multiple sub-discriminators that each take every Nth signal, where each sub-discriminator uses a different value of N. The multi-scale discriminator uses a combination of different convolutional sub-networks that are each configured to have different receptive fields. In addition, a time-domain loss (per-sample squared error), as in AV_{simple}, is used, to improve phase reconstruction. There is a hyperparameter which weights this MSE loss, and if the weight of this term is too low, highly audible phase artefacts result. Informal experiments were performed with different weightings for the time-domain loss.

For the decoder, three representations of the complex spectrogram were considered. The first method that was considered was that used in AV_{simple}, where the network generates all four output channels, then the Cartesian mean of both complex forms is taken. As part of the development efforts that went into AV_{GAN}, two additional complex representations were used. These were purely Cartesian and purely polar, i.e. the network generates two output channels: real and imaginary, or magnitude and phase, respectively. The results of informal listening tests were very clear. Training with only polar output yielded poorer sound quality, as did the third method using all four output channels. Cartesian was clearly best.

In addition to the training scheme, subtle changes were made to other hyperparameters as a result of further experimentation, by tuning for both informal and formal listening tests, as well as tuning the new losses to achieve minimal values.

Unlike a neural speech codec such as (Zeghidour et al., 2022), there is no intrinsic need for a small representation size in *autovocoder*, since we are not aiming for a low bitrate. Therefor, as part of the experiments with AV_{GAN}, a range of representation dimensionalities were used. These were 128, 192 and 256, chosen to be comparable to the internal representations in current TTS models (Łańcucki, 2021; Kim et al., 2021), with a view to a future TTS application of *autovocoder*. Choosing an appropriate representation size is likely to involve a trade-off in the complexities of the acoustic model and the vocoder.

8.3.1 Evaluating the improved model

As with AV_{simple}, a MUSHRA test was performed to evaluate output quality. As time had passed after the development of AV_{simple}, which was compared to WaveRNN, a more modern comparitor, HiFi-GAN, was selected for AV_{GAN}. An open source implementation of HiFi-GAN[4] was used with a pretrained checkpoint that was trained on the same dataset as *autovocoder* using the the *V1* and *V3* generators. The *V1* generator is designed to achieve maximum output quality, and the *V3* generator is designed to run much faster, at the expense of some output quality.

AV_{GAN} and HiFi-GAN were trained on the single-speaker LJ Speech (Ito and Johnson, 2017) dataset with waveforms at a sample rate of 22.05 kHz. We used a pretrained model for HiFi-GAN that had been trained to 2.3 million steps. AV_{GAN} was trained with a dropout factor of 10% and the Adam optimiser for approximately 1 million steps (limited by available compute). Evaluations used a test section of the dataset, unseen during training, and identical for HiFi-GAN and *autovocoder*.

The systems were evaluated using a MUSHRA (ITU-R, 2015) style evaluation. Each MUSHRA screen presented 7 stimuli to the listener for evaluation. These were *autovocoder* (*AV 128*, *AV 192*, *AV 256*), HiFi-GAN (*HG V1*, *HG V3*), Griffin-Lim (*GL*) and the original ground-truth (*GT*) waveform. HiFi-GAN was substituted as a more relevant baseline as compared to the WaveRNN that was used in the previous MUSHRA test. This is because, like *autovocoder*, HiFi-GAN targets very high speed synthesis. To avoid each MUSHRA screen becoming too large, AV_{simple} and AV_{GAN} are not directly compared in this experiment. In addition, pretrained WaveRNN and HiFi-GAN models were used to avoid unecessary compute, and these systems have different held-out test sets. *Autovocoder* models were trained with test sets that matched their respective baselines.

45 listeners were recruited using Prolific[5]. A Qualtrics survey was generated using the *Qualtreats* tool[6] comprising 60 MUSHRA screens, split evenly into 3 tests, with each test assigned 15 listeners. Each screen presented the reference. Participants were then instructed to find the reference within those seven samples and assign it a score of 100, whilst rating all samples.

In line with the MUSHRA specification, participants who rated GT less than 90 in more than 15% screens were excluded. This left 17 respondents and 340 MUSHRA

[4] https://github.com/jik876/hifi-gan

[5] https://www.prolific.co/

[6] https://github.com/CSTR-Edinburgh/qualtreats

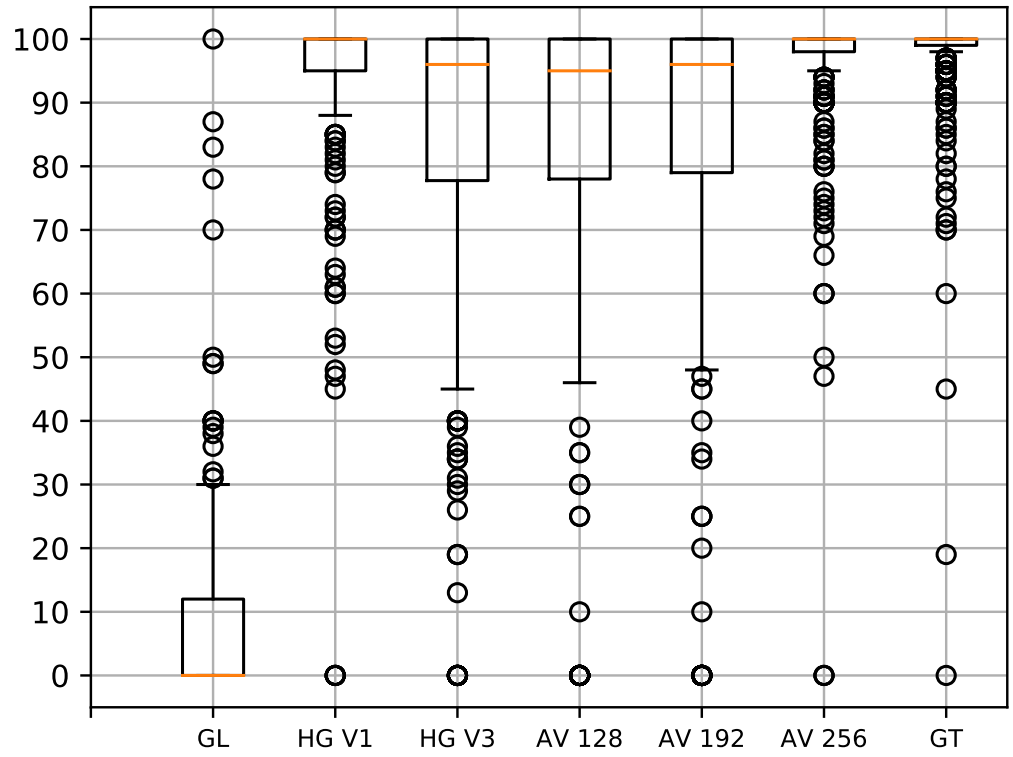


Figure 8.3: Results of the MUSHRA evaluation for AV_{GAN}. AV 128, AV 192 and AV 256 denote AV_{GAN} with learned embedding sizes for 128, 192 and 256 respectively

screens for analysis. This large number of exclusions was a result of the high difficulty of distinguishing between GT, AV 256 and HG V1. Analysis without these exclusions yielded broadly similar results. Figure 8.3 shows the results. In hindsight following the MUSHRA criteria in this way was probably excessive in the number of participants excluded. A pairwise t-test was used to evaluate statistical significance. Analysis yielded three sets of systems, {GL}, {HG V3, AV 128, AV 192} and {HG V1, AV 256, GT}. With $p > 0.01$ all systems were significantly different from those not in their own set, and there were no significant differences within sets.

8.4 Computational cost

The computational performace of both AV_{simple} and AV_{GAN} decoders were compared with several waveform generation systems. Each system was timed generating individual utterances (with no batching) on a CPU. Such utterance-by-utterance processing represents many typical use cases, such as on-device synthesis. The systems chosen were

Table 8.2: Speed of each system. Higher values indicate faster generation; a value below 1 indicates slower than real time.

System	Real-time factor
Griffin-Lim	18.43
AV_{simple}	81.68
AV_{GAN} 256,192,128	102.01, 101.34, 101.08
HiFi-GAN *V1, V3*	6.76, 42.18
WaveRNN	0.47
LPCNet	1.50

Griffin-Lim and HiFi-GAN as used in the listening test, plus LPCNet and WaveRNN[7]. A C implementation of LPCNet was used[8], which may give it a performance advantage over the Python/PyTorch implementation of *autovocoder*.

Timings were measured on a high-end 10 core desktop workstation processor (Intel i9-10850K). The value given in Table 8.2 was calculated by dividing the total duration of the generated audio by the total time taken to generate it. The measurement was for generating the test set of ten samples ten times. An average was taken of three runs. The test set contained a range of waveform durations from 2.0 s to 9.7 s.

LPCNet was not able to exploit the parallelism of the processor, and is therefore likely to perform relatively better on a system with fewer available cores. Time spent loading models from the file system was not found to be significant for any of the systems under test.

As shown in Table 8.2, *autovocoder* generates a waveform many times faster than the other waveform generators. Performance improvements of *autovocoder* over autoregressive systems should be even larger on hyper-parallel architectures such as GPUs.

The slight performance improvement from AV_{simple} to AV_{GAN} is likely explained by the reduction in window size for the iSTFT from 2048 samples to 1024.

[7] `https://github.com/fatchord/WaveRNN`
[8] `https://github.com/xiph/LPCNet`

8.5 Future application to TTS

There are two primary ways that *autovocoder* could be incorporated into a text-to-speech system. The first is by using the decoder architecture as the waveform generator within an end-to-end system. The second is by using *autovocoder* to extract representations from waveforms, and using a TTS acoustic model to generate such representations instead of mel spectrograms. The waveform could then be generated very quickly by the *autovocoder* decoder. In the former case the autoencoder task is purely for system design, as the autoencoder task is much faster to iterate with than a full end-to-end TTS system.

8.6 Conclusion

This chapter presents *autovocoder*, an alternative to typical neural vocoders. *Autovocoder* generates high-quality audio very quickly: it can reach the same quality as HiFi-GAN in fewer training updates, and is up to 16 times faster during generation. We intend that this system be used for the typical tasks of a neural vocoder, such as speech coding and speech synthesis. Verifying that speech synthesisers can accurately generate *autovocoder* features remains as important future work.

8.7 Frontground: Learned representations and fast waveform synthesis

This chapter has demonstrated the potential of learned representations of speech to enhance the computational performance of speech generation. In the time since *autovocoder* was developed, learned representations of speech, generally derived using self-supervised learning, have become an extremely fertile area of research. This section aims to provide a short summary of some of these developments and relate them to the work in this thesis.

Audio codecs are designed to store audio in as few bits as possible, by depending on structure in known forms of audio media and the limitations of human hearing. The use of *learned* representations in codecs goes back to a WaveNet-based codec from Kleijn et al. (2018) that was introduced with the goal of achieving very low bitrates when communicating speech signals. The LPCNet model used as a baseline earlier in this chapter has also been applied to the task of bitrate compression by Valin and Skoglund (2019). More recent neural codecs have achieved bitrate compression in part through *quantisation*. The first to take this approach was Gârbacea et al. (2019), which uses a vector-quantised VAE (VQ-VAE) (Du et al., 2022) to learn quantised representations for a codec, which are then synthesised to audio using a WaveNet. EnCodec, from Défossez et al. (2022), delivers stellar output quality when generating both speech and music from very low bitrate representations by using a combination of adversarial training, vector quantisation, and other training tricks. Importantly, EnCodec is *non-autoregressive*, and is sufficiently well-optimised that generates waveforms at a comparable speed to a conventional signal processing-based codec.

The emergence of fast neural codecs in the time since *autovocoder* was conceived has been an exciting development in and of itself, but it has also allowed the inception of a new class of model: the *codec language model*. Language modelling is the task of generating the next token in a sequence, and neural codec encodings, which are sequences of tokens, are well suited for be generation by language models. AudioLM, from Borsos et al. (2023), applies language modelling to the task of predicting neural codec encodings, for the SoundStream (Zeghidour et al., 2022) neural codec. AudioLM is demonstrated for the task of speech continuation for prompt utterances. With VALL-E, from Wang et al. (2023), the codec language modelling approach has been applied to the task of TTS by conditioning a large language model on text, with EnCodec used for waveform sythesis. VALL-E can additionally be conditioned on speech

prompts.

Although *autovocoder* is not used as part of a full TTS pipeline in this thesis, the work introduced in this section, some of it published some years after work on *autovocoder* began in 2020, shows the applicability of learned representations, especially *those from which generation is very fast*, to TTS and other speech generation tasks. The use of vector quantisation is a significant development that could have been applied to *autovocoder*.

Chapter 9

Conclusion and future work

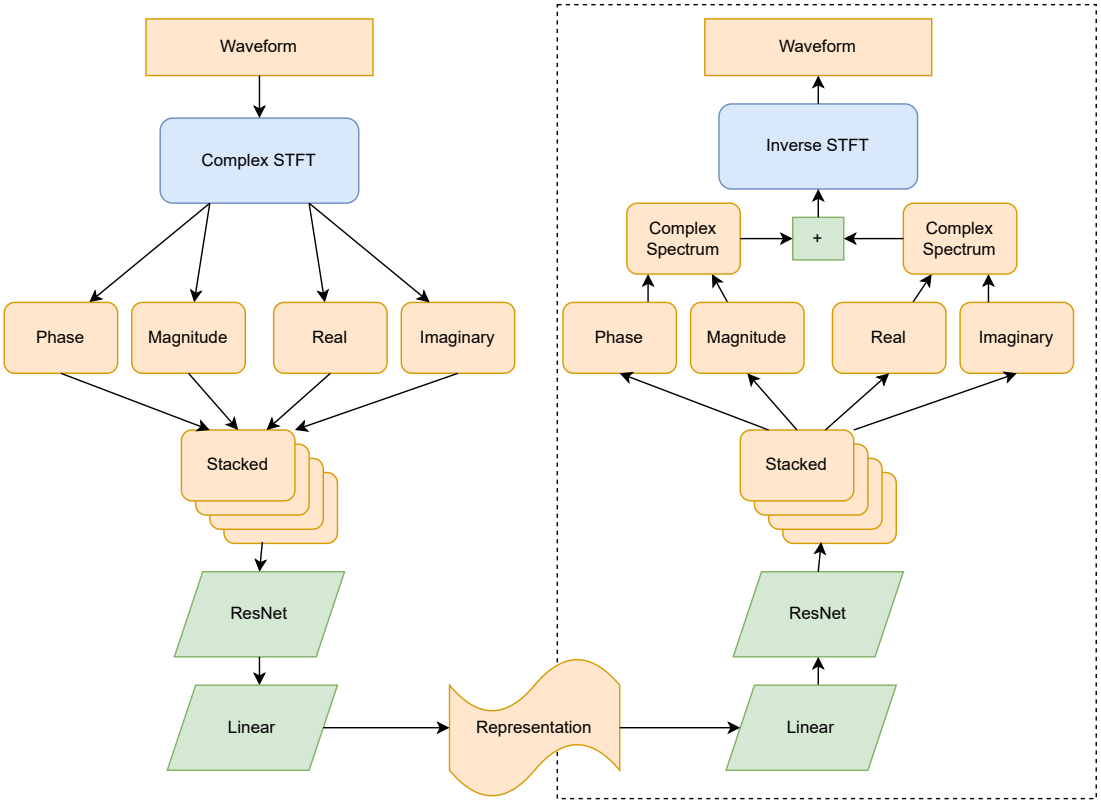
This thesis aimed to address two key challenges: fast and controllable speech generation. Through my research I have developed and proposed four novel methods that meet these goals separately. The combination of these streams into a unified approach that is both fast *and* controllable is left as future work.

The first two of these methods addressed *controllability* and made use of the Hider-Finder-Combiner architecture. These demonstrated the power of representation learning, showing that feature disentanglement can be used to achieve controllable signal generation. In Chapter 5, the HFC approach was successfully applied to the task of F_0 modification, and in Chapter 6 it was applied to voice privacy.

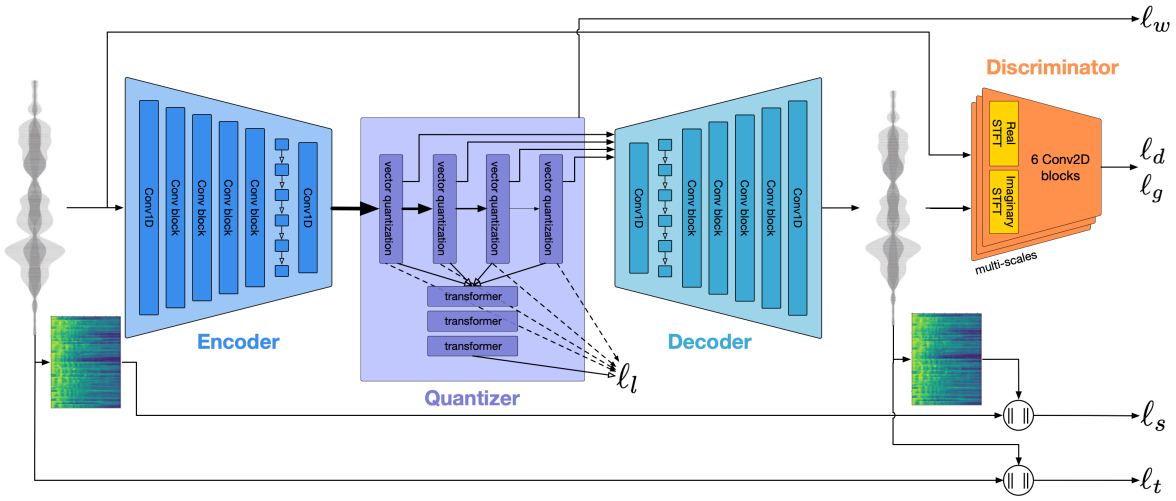
Part 3 introduced two methods that aimed to enable *fast* speech generation. The first of these aimed to generate a quantised representation of phase in order to speed up the vocoding process. The second of these, *autovocoder*, showed the computational performance benefits that can be achieved by using learned representations instead of traditional DSP-derived vocoder features.

The work in *autovocoder* appears particularly prescient given the recent popularity of neural codecs for speech generation tasks. Neural codecs have strong similarities with *autovocoder*, but generally target a low bitrate representation, rather than fast inversion. However, the most recent approaches, such as EnCodec (Défossez et al., 2022), achieve both of these goals: fast synthesis from low bitrate representations. EnCodec achieves low bitrates in part through *vector quantisation*. Figure 9.1 shows a comparison between the *autovocoder* architecture introduced in this thesis and EnCodec.

Key differences between these two methods are the use of the iSTFT to perform final waveform synthesis in *autovocoder*, and the use vector quantisation in EnCodec.



(a) *Autovocoder* architecture, repeated from figure 8.1



(b) EnCodec architecture, reproduced from Figure 1 of (Défossez et al., 2022).

Figure 9.1: Comparison of EnCodec and *autovocoder* architectures.

The performance of models such as AudioLM from Borsos et al. (2023) and VALL-E from Wang et al. (2023) indicate generating quantised representations may be useful for state-of-the-art audio generation models. A natural extension to the work in Chapter 8 would be to unify EnCodec and *autovocoder*, either by integrating an iSTFT into EnCodec with the goal of increasing computational performance, or by adding vector quantisation to *autovocoder* in order to allow the use of token-based language models as acoustic models. This is left as future work.

Chapter 7 introduced the use of a quantised representation of phase in order to achieve phase prediction from a range of state-of-the-art generative models. Contrary to the results of Borsos et al. (2023) and Wang et al. (2023), it was not possible to generate plausible phase using generative modelling and this quantised representation. Further work is needed to examine the difficulties of synthesising these representations, potentially through exploring alternate architectures for the deep networks that were used.

The most natural remaining future work that could extend this thesis would be to combine the approaches introduced in HFC and *autovocoder*. Both of these models operate by learning representations of speech. In the case of HFC this is to establish control through disentanglement, and in the case of *autovocoder*, this is done to find an alternative to mel spectrograms from which audio can be quickly synthesised. These could be combined by using an *autovocoder* style encoder and decoder for HFC, in place of the hider and finder. This would allow fast *and* controllable speech generation.

Bibliography

Airaksinen, M., Bollepalli, B., Juvela, L., Wu, Z., King, S., and Alku, P. (2016). GlottDNN — A Full-Band Glottal Vocoder for Statistical Parametric Speech Synthesis. In *Proc. Interspeech 2016*, pages 2473–2477.

Airaksinen, M., Juvela, L., Bollepalli, B., Yamagishi, J., and Alku, P. (2018). A comparison between straight, glottal, and sinusoidal vocoding in statistical parametric speech synthesis. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 26(9):1658–1670.

Allen, J. and Rabiner, L. (1977). A unified approach to short-time fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11):1558–1564.

Baevski, A., Zhou, H., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Proc. NeurIPS*.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. (2018). Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 531–540, Stockholmsmässan, Stockholm Sweden.

Benaroya, L., Obin, N., and Roebel, A. (2023). Manipulating voice attributes by adversarial learning of structured disentangled representations. *Entropy*, 25(2):375.

Bhatia, R. and Davis, C. (2000). A better bound on the variance. *The American Mathematical Monthly*, 107(4):353–357.

Boersma, P. and Weenink, D. (2001). Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–347.

Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., and Zeghidour, N. (2023). AudioLM: a language modeling approach to audio generation.

Champion, P., Jouvet, D., and Larcher, A. (2022). Are disentangled representations all you need to build speaker anonymization systems? In *Proc. Interspeech 2022*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Chorowski, J., Weiss, R. J., Bengio, S., and van den Oord, A. (2019). Unsupervised speech representation learning using WaveNet autoencoders. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 27(12):2041–2053.

Chou, J., chieh Yeh, C., yi Lee, H., and shan Lee, L. (2018). Multi-target Voice Conversion without Parallel Data by Adversarially Learning Disentangled Audio Representations. In *Proc. Interspeech 2018*, pages 501–505.

Doddipatla, R. S., Braunschweiler, N., and Maia, R. (2017). Speaker adaptation in DNN-based speech synthesis using d-vectors. In *Proc. Interspeech*, pages 3404–3408, Stockholm, Sweden.

Du, C., Guo, Y., Chen, X., and Yu, K. (2022). VQTTS: High-fidelity text-to-speech synthesis with self-supervised VQ acoustic feature. In *Proc. Interspeech*, pages 1596–1600.

Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. (2022). High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*.

Espic, F., Botinhao, C. V., and King, S. (2017). Direct modelling of magnitude and phase spectra for statistical parametric speech synthesis. In *Proc. Interspeech*, pages 1383–1387.

Evrard, M., Delalez, S., d'Alessandro, C., and Rilliard, A. (2015). Comparison of chironomic stylization versus statistical modeling of prosody for expressive speech synthesis. In *Proceedings Interspeech*, pages 3370–3374, Dresden, Germany.

Flanagan, J. L. and Golden, R. M. (1966). Phase vocoder. *The Bell System Technical Journal*, 45(9):1493–1509.

Fong, J. (2024). *Controlling text-to-speech pronunciation using limited linguistic resources*. PhD thesis, The University of Edinburgh.

Frigo, M. and Johnson, S. (2005). The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Govalkar, P., Fischer, J., Zalkow, F., and Dittmar, C. (2019). A comparison of recent neural vocoders for speech signal reconstruction. In *ISCA Speech Synthesis Workshop*, pages 7–12, Vienna, Austria.

Griffin, D. and Lim, J. (1984). Signal estimation from modified short-time Fourier transform. *IEEE Trans. Acoustics, Speech, and Sig. Proc.*, 32(2):236–243.

Gârbacea, C., den Oord, A. v., Li, Y., Lim, F. S. C., Luebs, A., Vinyals, O., and Walters, T. C. (2019). Low bit-rate speech coding with VQ-VAE and a wavenet decoder. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 735–739.

Henter, G. E., Merritt, T., Shannon, M., Mayo, C., and King, S. (2014). Measuring the perceptual effects of modelling assumptions in speech synthesis using stimuli constructed from repeated natural speech. In *Proc. Interspeech 2014*, pages 1504–1508.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hodari, Z. (2022). *Synthesising prosody with insufficient context*. PhD thesis, The University of Edinburgh.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976.

Ito, K. and Johnson, L. (2017). The LJ Speech dataset. `https://keithito.com/LJ-Speech-Dataset/`.

ITU-R (2015). Method for the subjective assessment of intermediate quality levels of coding systems. ITU Recommendation ITU-R BS.1534-3.

Jayashankar, T., Wu, J., Sari, L., Kant, D., Manohar, V., and He, Q. (2023). Self-supervised representations for singing voice conversion. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Moreno, I. L., Wu, Y., et al. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Advances in neural information processing systems*, pages 4480–4490.

Juvela, L., Bollepalli, B., Yamagishi, J., and Alku, P. (2019). GELP: GAN-excited linear prediction for speech synthesis from mel-spectrogram. In *Proc. Interspeech*, pages 694–698.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S., and Kavukcuoglu, K. (2018). Efficient neural audio synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2410–2419, Stockholmsmässan, Stockholm Sweden. PMLR.

Kaneko, T., Kameoka, H., Tanaka, K., and Hojo, N. (2019a). CycleGAN-VC2: Improved cyclegan-based non-parallel voice conversion. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6820–6824.

Kaneko, T., Kameoka, H., Tanaka, K., and Hojo, N. (2019b). StarGAN-VC2: Rethinking Conditional Methods for StarGAN-Based Voice Conversion. *Proc. Interspeech 2019*.

Kaneko, T., Tanaka, K., Kameoka, H., and Seki, S. (2022). iSTFTNet: Fast and lightweight mel-spectrogram vocoder incorporating inverse short-time Fourier transform. In *Proc. ICASSP*, pages 6207–6211.

Kawahara, H., Masuda-Katsuse, I., and de Cheveigné, A. (1999). Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds. *Speech Commun.*, 27(3–4):187–207.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. (2019). Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Proc. Interspeech 2019*.

Kim, J., Kong, J., and Son, J. (2021). Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *Proc. ICML*, pages 5530–5540.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kleijn, W. B., Lim, F. S. C., Luebs, A., Skoglund, J., Stimberg, F., Wang, Q., and Walters, T. C. (2018). Wavenet based low rate speech coding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 676–680.

Kong, J., Kim, J., and Bae, J. (2020). HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Proc. NeurIPS*, volume 33, pages 17022–17033.

Kons, Z., Shechtman, S., Sorin, A., Rabinovitz, C., and Hoory, R. (2019). High quality, lightweight and adaptable TTS using LPCNet. In *Proc. Interspeech*, pages 176–180, Graz, Austria.

Kovela, S., Valle, R., Dantrey, A., and Catanzaro, B. (2023). Any-to-Any Voice Conversion with F0 and Timbre Disentanglement and Novel Timbre Conditioning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., et al. (2017). Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5963–5972.

Lawson, C. L. and Hanson, R. J. (1995). *Solving least squares problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA. Revised reprint of the 1974 original.

Lo, C.-C., Fu, S.-W., Huang, W.-C., Wang, X., Yamagishi, J., Tsao, Y., and Wang, H.-m. (2019). MOSNet: Deep learning-based objective assessment for voice conversion. In *Proc. Interspeech 2019*, pages 1541–1545.

Lorenzo-Trueba, J., Drugman, T., Latorre, J., Merritt, T., Putrycz, B., Barra-Chicote, R., Moinet, A., and Aggarwal, V. (2019). Towards achieving robust universal neural vocoding. In *Proc. Interspeech*, pages 181–185, Graz, Austria.

Loweimi, E., Bell, P., and Renals, S. (2020). Raw Sign and Magnitude Spectra for Multi-Head Acoustic Modelling. In *Proc. Interspeech 2020*, pages 1644–1648.

Loweimi, E., Cvetkovic, Z., Bell, P., and Renals, S. (2021). Speech acoustic modelling from raw phase spectrum. In *Proc. ICASSP*, pages 6738–6742. cited By 0.

Lyth, D. and King, S. (2024). Natural language guidance of high-fidelity text-to-speech with synthetic annotations.

Makhoul, J. (1975). Linear predicion: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in Python. In *Proc. Python Sci. Conf.*, volume 8, pages 18–24.

Mehta, S., Tu, R., Beskow, J., Székely, É., and Henter, G. E. (2024). Matcha-TTS: A fast TTS architecture with conditional flow matching. In *Proc. ICASSP*.

Morise, M. (2015). Cheaptrick, a spectral envelope estimator for high-quality speech synthesis. *Speech Communication*, 67:1–7.

Morise, M., Yokomori, F., and Ozawa, K. (2016). WORLD: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D(7):1877–1884.

Mosiński, J., Biliński, P., Merritt, T., Ezzerg, A., and Korzekwa, D. (2023). AE-Flow: Autoencoder Normalizing Flow. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Mourby, M., Mackey, E., Elliot, M., Gowans, H., Wallace, S. E., Bell, J., Smith, H., Aidinlis, S., and Kaye, J. (2018). Are 'pseudonymised' data always personal data?

Implications of the GDPR for administrative data research in the UK. *Computer Law & Security Review*, 34(2):222–233.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press.

Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

Nagrani, A., Chung, J. S., and Zisserman, A. (2017). Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*.

Nishimura, Y., Saito, Y., Takamichi, S., Tachibana, K., and Saruwatari, H. (2022). Acoustic Modeling for End-to-End Empathetic Dialogue Speech Synthesis Using Linguistic and Prosodic Contexts of Dialogue History. In *Proc. Interspeech 2022*, pages 3373–3377.

Noé, P.-G., Emînî, A., Driss, M., Parcollet, T., Nautsch, A., and Bonastre, J.-F. (2021). Adversarial Disentanglement of Speaker Representation for Attribute-Driven Privacy Preservation. In *Proc. Interspeech 2021*.

Odena, A., Dumoulin, V., and Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*.

O'Shaughnessy, D. (1987). *Speech communication: human and machine*. Addison-Wesley.

Pascual, S., Bhattacharya, G., Yeh, C., Pons, J., and Serrà, J. (2022). Full-band general audio synthesis with score-based diffusion. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Proc. NeurIPS*, pages 8024–8035. Curran Associates, Inc.

Patino, J., Tomashenko, N., Todisco, M., Nautsch, A., and Evans, N. (2021). Speaker Anonymisation Using the McAdams Coefficient. In *Proc. Interspeech 2021*, pages 1099–1103.

Pearson, K. (1900). X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175.

Perraudin, N., Balazs, P., and Søndergaard, P. L. (2013). A fast griffin-lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4.

Perrotin, O. and McLoughlin, I. (2019a). GFM-Voc: A real-time voice quality modification system. In *Proc. Interspeech*, pages 3685–3686, Graz, Austria.

Perrotin, O. and McLoughlin, I. (2019b). A spectral glottal flow model for source-filter separation of speech. In *Proc. ICASSP*, pages 7160–7164.

Peyser, C., Huang, W. R., Rosenberg, A., Sainath, T., Picheny, M., and Cho, K. (2022). Towards Disentangled Speech Representations. In *Proc. Interspeech 2022*, pages 3603–3607.

Prenger, R., Valle, R., and Catanzaro, B. (2019). Waveglow: A flow-based generative network for speech synthesis. In *Proc. ICASSP 2019*, pages 3617–3621.

Qian, K., Zhang, Y., Chang, S., Hasegawa-Johnson, M., and Cox, D. (2020). Unsupervised speech decomposition via triple information bottleneck. In *International Conference on Machine Learning*, pages 7836–7846. PMLR.

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., et al. (2021). SpeechBrain: A general-purpose speech toolkit. arXiv:2106.04624.

Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2021). FastSpeech 2: Fast and High-Quality End-to-End Text to Speech. In *International Conference on Learning Representations*.

Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2019). FastSpeech: Fast, robust and controllable text to speech. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Proc. NeurIPS*, volume 32. Curran Associates, Inc.

Rix, A., Beerends, J., Hollier, M., and Hekstra, A. (2001). Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech,*

and Signal Processing. Proceedings (Cat. No.01CH37221), volume 2, pages 749–752 vol.2.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R. A., Agiomvrgiannakis, Y., and Wu, Y. (2018). Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. In *Proc. ICASSP*, pages 4779–4783.

Shifas, M. P., Chermaz, C., Chimona, T. S., Tsiaras, V., and Stylianou, Y. (2019). Benefits of the wavenet-based speech intelligibility enhancement for normal and hearing impaired listeners. In *ICA 2019 Proceedings*.

Siuzdak, H., Dura, P., van Rijn, P., and Jacoby, N. (2022). WavThruVec: Latent speech representation as intermediate features for neural speech synthesis. In *Proc. Interspeech*, pages 833–837.

Smith, J. (2007a). *Introduction to Digital Filters: With Audio Applications*. Music signal processing series. W3K.

Smith, J. (2007b). *Mathematics of the Discrete Fourier Transform (DFT): With Audio Applications*. W3K Publishing. BookSurge.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2256–2265. JMLR.org.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Tachibana, H., Uenoyama, K., and Aihara, S. (2018). Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4784–4788.

Tamamori, A., Hayashi, T., Kobayashi, K., Takeda, K., and Toda, T. (2017). Speaker-dependent WaveNet vocoder. In *Proc. Interspeech*, pages 1118–1122.

Taylor, P. A., Black, A. W., and Caley, R. (1998). The architecture of the festival speech synthesis system. In *Speech Synthesis Workshop*.

Tomashenko, N., Miao, X., Champion, P., Meyer, S., Wang, X., Vincent, E., Panariello, M., Evans, N., Yamagishi, J., and Todisco, M. (2024). The voiceprivacy 2024 challenge evaluation plan. *arXiv preprint arXiv:2404.02677*.

Tomashenko, N., Srivastava, B. M. L., Wang, X., Vincent, E., Nautsch, A., Yamagishi, J., et al. (2020). Introducing the VoicePrivacy Initiative. In *Proc. Interspeech 2020*, pages 1693–1697.

Tomashenko, N., Srivastava, B. M. L., Wang, X., Vincent, E., Nautsch, A., Yamagishi, J., Evans, N., Patino, J., Bonastre, J.-F., Noé, P.-G., and Todisco, M. (2022a). The VoicePrivacy 2020 challenge evaluation plan.

Tomashenko, N., Wang, X., Vincent, E., Patino, J., Srivastava, B. M. L., Noé, P.-G., et al. (2022b). The VoicePrivacy 2020 Challenge: Results and findings. *Computer Speech & Language*, 74:101362.

Valentini-Botinhao, C., Mayo, C., and Cooke, M. (2019). Hurricane natural speech corpus - higher quality version. DOI: `https://doi.org/10.7488/ds/2482`.

Valin, J.-M. (2017). A hybrid DSP/deep learning approach to real-time full-band speech enhancement. *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5.

Valin, J.-M., Isik, U., Smaragdis, P., and Krishnaswamy, A. (2022). Neural speech synthesis on a shoestring: Improving the efficiency of LPCNet.

Valin, J.-M. and Skoglund, J. (2019). LPCNet: Improving neural speech synthesis through linear prediction. In *Proc. ICASSP*, pages 5891–5895.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. *arXiv:1609.03499*.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., Casagrande, N., Grewe,

D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2018). Parallel WaveNet: Fast high-fidelity speech synthesis. In Dy, J. and Krause, A., editors, *Proc. ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926. PMLR.

Van Hove, P., Hayes, M., Jae Lim, and Oppenheim, A. (1983). Signal reconstruction from signed Fourier transform magnitude. *IEEE Trans. Acoustics, Speech, and Sig. Proc.*, 31(5):1286–1293.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Veaux, C., Yamagishi, J., and King, S. (2013). The voice bank corpus: Design, collection and data analysis of a large regional accent speech database. In *2013 International Conference Oriental COCOSDA held jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE)*, pages 1–4.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proc. ICML*, ICML '08, pages 1096–1103, New York, NY, USA. Association for Computing Machinery.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272.

Voigt, P. and Von dem Bussche, A. (2017). The EU General Data Protection Regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555.

Wang, C., Chen, S., Wu, Y., Zhang, Z., Zhou, L., Liu, S., Chen, Z., Liu, Y., Wang,

H., Li, J., He, L., Zhao, S., and Wei, F. (2023). Neural codec language models are zero-shot text to speech synthesizers.

Wang, X., Takaki, S., and Yamagishi, J. (2020). Neural source-filter waveform models for statistical parametric speech synthesis. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 28:402–415.

Wang, X. and Yamagishi, J. (2020). Using cyclic noise as the source signal for neural source-filter-based speech waveform model. In *Proc. Interspeech*, pages 1992–1996.

Wang, Y., Skerry-Ryan, R., Stanton, D., Weiss, R., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. (2017). Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech 2017*, pages 4006–4010.

Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R. J., Battenberg, E., Shor, J., Xiao, Y., Jia, Y., Ren, F., and Saurous, R. A. (2018). Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *Proceedings of the International Conference on Machine Learning*, volume 80, pages 5180–5189, Stockholmsmässan, Stockholm Sweden.

Watts, O. (2012). *Unsupervised Learning for Text-to-Speech Synthesis*. PhD thesis, University of Edinburgh.

Watts, O., Wihlborg, L., and Valentini-Botinhao, C. (2023). Puffin: pitch-synchronous neural waveform generation for fullband speech on modest devices. In *Proc. ICASSP*.

Webber, J. J. (2019). Controllable recurrent adversarial speech processing. Master's thesis, The University of Edinburgh, UK.

Webber, J. J., Perrotin, O., and King, S. (2020). Hider-finder-combiner: An adversarial architecture for general speech signal modification. In *Proc. Interspeech 2020*, pages 3206–3210.

Webber, J. J., Valentini-Botinhao, C., Williams, E., Henter, G. E., and King, S. (2023). Autovocoder: Fast waveform generation from a learned speech representation using differentiable digital signal processing. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Webber, J. J., Watts, O., Henter, G. E., Williams, J., and King, S. (2024). Voice conversion-based privacy through adversarial information hiding. In *Proc. 4th Symposium on Security and Privacy in Speech Communication*.

Williams, J. and King, S. (2019). Disentangling style factors from speaker representations. In *Proc. Interspeech 2019*, pages 3945–3949.

Williams, J., Pizzi, K., Das, S., and Noé, P.-G. (2022). New challenges for content privacy in speech and audio. In *Proc. 2nd Symposium on Security and Privacy in Speech Communication*, pages 1–6.

Williams, J., Pizzi, K., Noe, P.-G., and Das, S. (2023). Exploratory Evaluation of Speech Content Masking. In *Speech Communication; 15th ITG Conference*, pages 215–219.

Wu, Y., Hayashi, T., Tobing, P. L., Kobayashi, K., and Toda, T. (2019). Quasi-periodic wavenet vocoder: A pitch dependent dilated convolution model for parametric speech generation. In Kubin, G. and Kacic, Z., editors, *Proc. Interspeech*, pages 196–200.

Yamamoto, R., Song, E., and Kim, J.-M. (2019). Probability density distillation with generative adversarial networks for high-quality parallel waveform generation. *arXiv preprint arXiv:1904.04472*.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. (2022). SoundStream: An end-to-end neural audio codec. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 30:495–507.

Zen, H., Dang, V., Clark, R., Zhang, Y., Weiss, R. J., Jia, Y., Chen, Z., and Wu, Y. (2019). LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech. In *Proc. Interspeech 2019*, pages 1526–1530.

Zen, H., Tokuda, K., and Black, A. W. (2009). Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064.

Zhou, Y. and Lu, X. (2022). HiFi-SVC: Fast high fidelity cross-domain singing voice conversion. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6667–6671. IEEE.

Łańcucki, A. (2021). FastPitch: Parallel text-to-speech with pitch prediction. In *Proc. ICASSP*, pages 6588–6592.