Prosody Generation For Text-To-Speech Synthesis

Srikanth Ronanki

Doctor of Philosophy
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2019

Abstract

The absence of convincing intonation makes current parametric speech synthesis systems sound dull and lifeless, even when trained on expressive speech data. Typically, these systems use regression techniques to predict the fundamental frequency (F0) frame-by-frame. This approach leads to overly-smooth pitch contours and fails to construct an appropriate prosodic structure across the full utterance. In order to capture and reproduce larger-scale pitch patterns, we propose a template-based approach for automatic F0 generation, where per-syllable pitch-contour templates (from a small, automatically learned set) are predicted by a recurrent neural network (RNN). The use of syllable templates mitigates the over-smoothing problem and is able to reproduce pitch patterns observed in the data. The use of an RNN, paired with connectionist temporal classification (CTC), enables the prediction of structure in the pitch contour spanning the entire utterance. This novel F0 prediction system is used alongside separate LSTMs for predicting phone durations and the other acoustic features, to construct a complete text-to-speech system. Later, we investigate the benefits of including long-range dependencies in duration prediction at frame-level using uni-directional recurrent neural networks.

Since prosody is a supra-segmental property, we consider an alternate approach to intonation generation which exploits long-term dependencies of F0 by effective modelling of linguistic features using recurrent neural networks. For this purpose, we propose a hierarchical encoder-decoder and multi-resolution parallel encoder where the encoder takes word and higher level linguistic features at the input and upsamples them to phone-level through a series of hidden layers and is integrated into a Hybrid system which is then submitted to Blizzard challenge workshop. We then highlight some of the issues in current approaches and a plan for future directions of investigation is outlined along with on-going work.

Lay Summary

The ultimate goal of Text-To-Speech (TTS) systems is to create a more realistic sounding speech that is indistinguishable from humans. The absence of human-like natural prosody makes current speech synthesis systems sound dull and lifeless, even when trained on expressive speech data. An efficient way of representing and modelling of pitch and duration is necessary for generating natural-sounding expressive speech (e.g., children's storybooks). In this thesis, we formulate ways to identify different patterns in intonation and demonstrate the use of pitch templates for audiobook speech synthesis. We study various models for representation of duration and to help us understand the generation of melody in the context of audiobooks. Finally, we investigate the use of various neural network architectures for processing linguistic features at different timescales to synthesise natural-sounding speech.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Simon King for his guidance, continuous support, encouragement and supervision throughout my studies. He also played a great role in helping me choose research directions, and providing support to work on open source tools besides my PhD work. I should also thank him for giving all the detailed feedback on this thesis and the drafts of my papers. His feedback has helped me improve the content and presentation of this thesis.

I would like to thank Rob Clark, for having accepted me as his student and supervising my work during the first year of PhD. Thank you for believing in my skills during the internship and providing me a great opportunity to work at CSTR. I'm very thankful to Junichi Yamagishi, who has been the source of many innovative ideas. He has provided great insights of others work related to my research. I would also like to thank him for giving me an amazing opportunity to intern at National Institute of Informatics, Japan. I have met many wonderful researchers and colleagues.

I would like to thank my thesis committee members Antonio Bonafonte and Korin Richmond for taking time to review my thesis, giving valuable feedback and improving various sections of the thesis. I would like to thank Steve Renals and Hiroshi Shimodaira for their valuable feedback during my annual reviews. It was great fun and learning experience collaborating with Gustav Eje Henter, Oliver Watts and Zhizheng Wu. They were absolutely amazing. I would like to thank them for answering all my questions and teaching me all the great things.

I consider myself lucky to be part of CSTR and ILCC. The Edinburgh group meetings, reading groups have taught me how to read papers, appreciate research ideas, reviews ideas critically and most importantly how to present my work. I would like to thank all the group members who provided feedback on my PhD projects, presentations and research papers. I would like to thank all the faculty in Edinburgh ILCC group for providing the fun and interactive environment.

I want to thank my colleagues and fellow PhD students for making my time in Edinburgh enjoyable . Special thanks to Tom, Cassia, Felipe, Peter, Siva Reddy, Oliver, Catherine, Sam, Rasmus, Carol, Avashna, Joachim, Zack, Ondrej, Joanna, Mirjam, Herman, Gustav, Zhizheng for all the fun times and

lunch conversations. Special thanks to Siva, Spandana, Bharat, Gangi, Praveen and Rupa for making Edinburgh another home, all the fun weekend getaways, food experiments, long board game/movie nights. Thanks to Anusha, Tushar, Sruthilaya, Prudhvi, Subbu, Suneetha, Baji and Manvisha - I still relish all the wonderful moments we had during our outings to several European cities, restaurants and fun places. I would like to thank all my friends in the bay area and India for helping me maintain a certain degree of sanity throughout the whole PhD. Special thanks to Sruti, Phani, Sindhu, Abhinav, Abilash, Bharat, Ravali, Vignesh and Chaitanya for hosting me every time I go their place. Also, I am grateful to my masters supervisor Kishore Prahallad for encouraging me to apply for a PhD program.

I was very fortunate to work with brilliant collaborators during Amazon internship. I would like to specially thank Thomas Drugman for allowing me to work part-time at Amazon, while writing up my thesis. It has been an amazing experience to work with Amazon Alexa TTS group. Special thanks to Slava, Bajibabu, Jaime, Goeric, Tom, Nishant, Abdel, Jonas, Alexis, Vatsal, Poorvi, Roberto, Daniel, Andrew, Trevor, Bartosz, Bartek, Kayoko for all the fun times, long lunch conversations at Amazon.

I am grateful to my parents, and extended family in India and US for being a constant source of support and encouragement throughout my studies.

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.

[illegible]

(*Srikanth Ronanki*)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With more than 7000 languages[1] spoken by the world's population, the scope and impact of language technology research is enormous (Hill, 2002). Technologies such as Text-To-Speech (TTS), Automatic Speech Recognition (ASR) and language translation systems have transformed the way we seek knowledge and interact with computers. Yet, these technologies are far from achieving human language cognitive abilities.

With recent advances in speech recognition, natural language interaction with machines will likely be the most preferred interface. Also, with the growing number of intelligent and virtual assistant devices that offer handsfree interaction, speech synthesis has become one of the key facets of personal assistants where it underpins several successful applications: for example, Siri (Capes et al., 2017) is a deep learning-based personal assistant and interacts with millions of Apple customers; Google recently launched generative model-based speech synthesis, Wavenet (van den Oord et al., 2016) in its personal assistant to improve the naturalness; and Amazon has been developing a plethora of devices that enable handsfree interaction through its Alexa language technologies (Klimkov et al., 2017; Fan et al., 2017). These applications are currently limited in their expressiveness but are a good starting point for more research in this area (López et al., 2017).

Speech synthesis deals with many aspects of both voice and text, and our goal is to investigate the key elements of prosody (Nooteboom, 1997) that add richness and expressivity to the synthesized speech. Specifically, we evaluate statistical methods for modelling intonation and speech durations (perceived

[1] https://www.ethnologue.com/guides/how-many-languages

as speech melody and rhythm), the two important properties of the voice that are associated with prosody (Nooteboom, 1997). Based on an empirical analysis of the statistical methods, we later explore encoder-decoder based neural network paradigms which can perform the sequence-to-sequence regression in a way that takes the input linguistic features extracted from the text to predict acoustic features (generally derived from a vocoder (Hu et al., 2013)), including pitch and duration.

The work we present here is mainly developed in the framework of statistical parametric speech synthesis (Zen et al., 2009, 2013) and our proposed methods are evaluated on prosodically interesting data from children's storybooks provided by Usborne Publishing (see Section 3.1 for details).

1.1 Motivation

Statistical parametric speech synthesis (SPSS) is one of the main paradigms of speech synthesis where speech parameters are generated using regression techniques and then a vocoder is used to synthesize speech (Black et al., 2007; Zen et al., 2009; King, 2011). Some of these techniques include generation based on Hidden Markov Models (HMMs) (Yoshimura et al., 1999), Random Forests (Black and Muthukumar, 2015), Deep Neural Networks (DNNs) and their variants (Zen et al., 2013; Wu et al., 2015b; Zen and Sak, 2015).

Despite steady improvements over many decades of research, speech generated from these parametric techniques is still not convincingly natural (King and Karaiskos, 2013; King, 2014; King et al., 2017). A particular Achilles' heel is the *prosody* of the artificial speech – encompassing signal aspects such as fundamental frequency, signal energy, and speech-sound durations – which makes machine speech come across as unnatural, inappropriate, and unappealing. This failure makes synthetic speech unsuitable for many attractive applications, for example generating expressive and conversational speech for audiobooks.

Prosody is an integral part of human communication and is characterized by the use of intonation, loudness and rhythm to structure the information of an utterance into meaningful elements (e.g., syllable, word or phrase). Unlike other acoustic features (e.g., mel-cepstral features), prosody varies slowly and at multiple temporal resolutions. It is typically concerned with larger units of speech and reflects various features of the speaker or the utterance. It is often used to express intense emotions by placing prominence on certain syllables or emphasizing certain words of a sentence. Therefore, the realization of linguistically-motivated prosody is quite essential to emulate human-like speech across different domains.

Fundamental frequency (F0) and duration are two important factors of prosody, and a significant amount of work has been done to model them in statistical parametric speech synthesis (Zen et al., 2013; Fernandez et al., 2014; Zen and Sak, 2015; Wu et al., 2015b). Due to the ineffectiveness of conventional approaches, synthesized speech is still generally neutral in terms of prosody and cannot compete with good unit selection systems (King et al., 2017, 2018).

One likely reason for this is that pitch variation continues to be modelled locally at the frame level by models unable to capture long-term behaviour in the pitch contour. Although intonation is a supra-segmental property, conventional approaches, such as HMMs and regression tress (Zen et al., 2007) or DNNs (Zen et al., 2013), predict F0 frame-by-frame, based on limited linguistic contextual information (quinphone, part-of-speech and positional information). Predictions within an HMM state are assumed conditionally independent of *surrounding F0 predictions*, given the linguistic information. Surrounding F0 values only affect the local smoothing of the final contour via maximum likelihood parameter generation (Tokuda et al., 2000). Such a myopic approach may not be the best way to produce utterance-level supra-segmental structure. Some approaches attempt to leverage the suprasegmental characteristics of prosody and work quite well in estimating overall mean F0 when compared to frame-level models (Latorre and Akamine, 2008; Teutenberg et al., 2008; Qian et al., 2011). However, they are still far from generating those variations which appear in natural speech (for example, audiobooks).

Neural networks have rapidly grown to dominate SPSS (Zen et al., 2013; Wu et al., 2015b; Henter et al., 2016), for both duration and acoustic modelling (including F0). Typical approaches to SPSS using Neural Networks generally require input at the same temporal resolution as the output, typically a frame every 5ms, or in some cases at waveform sampling rate. It is therefore necessary to fabricate highly-redundant frame-level (or sample-level) linguistic features at the input. This makes the network inefficient to learn and capture long-term context from frame-level linguistic features. There are some approaches that have attempted to explore the hierarchical nature and the long-term dependencies of prosody (Ribeiro et al., 2016; Yin et al., 2016). However, these approaches use multiple networks, incurring additional computational cost compared to standard approaches.

In regards to the above mentioned problems, the work in this thesis is about the feasibility of using prosodically-enhanced models in Text-To-Speech.

1.2 Thesis Outline

Our main focus is intonation (pitch contour) generation for audiobooks in the SPSS framework. As a preliminary step, we propose to generate more variable and naturalistic intonation through the use of syllable-level pitch contour templates learned from the data in Chapter 4. The hope is that treating intonation prediction as a *classification* rather than a *regression problem* will mitigate the over-smoothing seen in conventional techniques. By using a long short-term memory recurrent neural network (LSTM-RNN) to predict a pitch-template sequence, long-range information about linguistic context and surrounding predictor state can influence the output, enabling high-level prosodic structure to be expressed. The approach for creating the inventory of syllable-level templates will be discussed in more detail in the Section 4.2.1 from Chapter 4, which is also an extended version of the work presented in Ronanki et al. (2016a).

On the other hand, we consider duration modelling as it is a challenging and important problem in generating natural-sounding synthetic speech, particularly in expressive or conversational scenarios. In Chapter 5, we propose a new approach to duration modelling for statistical parametric speech synthesis in which a recurrent statistical model is trained to output a phone transition probability at each timestep (acoustic frame). Unlike conventional approaches to duration modelling – which assume that duration distributions have a particular form (e.g., a Gaussian) and use the mean of that distribution for synthesis – our approach can in principle model any distribution supported on the non-negative integers. Generation from this model can be performed in many ways; here we consider output generation based on the median predicted duration (see Section 5.3). The median is more typical (more probable) than the conventional mean duration, is robust to training-data irregularities, and enables incremental generation. Furthermore, a frame-level approach to duration prediction is consistent with a longer-term goal of modelling durations and acoustic features together.

Alternatively, in the work by Henter et al. (2016), we focus on statistical techniques for improved duration modelling. We describe a new application of robust statistical methods using mixture density networks (MDN) to duration prediction in speech synthesis (see Section 5.6.1). Later, we extend this work to perform duration modelling with LSTM-MDNs and is integrated with a Hybrid system (parametric guided unit-selection synthesis system) which is then submitted to the Blizzard Challenge 2016[2] Merritt et al. (2016b). Hybrid synthesis is used as it allows for the benefits of extremely natural-sounding unit selection, and is unaffected by the degradations introduced by vocoding, whilst also exploiting the flexibility of SPSS. Also, the data used for the 2016 Blizzard Challenge comes from children's audio books and therefore is much more prosodically rich making it ideal for testing techniques for globally controlling prosody.

In Chapters 4 and 5, we focus on intonation and duration modelling and investigate independent and separate neural network based approaches for generation of these prosodic features. Other acoustic features such as spectral features are generated through conventional approaches and used together with duration and intonation to generate speech. There isn't much attention paid to the interactions of acoustic and prosodic features and we will address this to some extent in the following chapters.

Since prosody is a supra-segmental property, we consider an alternative approach to intonation generation in Chapter 7, which exploits long-term dependencies of F0 by effective modelling of linguistic features using recurrent neural networks. For this purpose, we propose a hierarchical encoder-decoder (Ronanki et al., 2017b) where the encoder takes word and higher level linguistic features at the input and upsamples them to phone-level through a series of hidden layers. The other phone-level and syllable-level linguistic features such as phone identity, syllable stress, accent etc., are appended as additional features at their respective hidden layers. This phone-level hidden output from the encoder is further upsampled to frame-level using force-aligned durations. Finally, the recurrent decoder predicts all vocoder features (including F0) con-

[2]http://www.synsig.org/index.php/Blizzard_Challenge_2016

sidering the previously predicted frames. Such a hierarchical-encoder enables the network to memorize the input text over a long span and predicts frame-level F0 alongside other acoustic features in a single network. We further extend this work with a multi-scale parallel encoder-decoder for SPSS in Chapter 8 where linguistic features and other suprasegmental features of different scale are processed independently through multiple encoders in parallel.

The emphasis in this thesis is on using linguistic information for prosody modelling with deep neural networks. For several decades, unit selection synthesis has become one of the main driving force in most commercial speech synthesis systems owing to its high quality synthetic speech. However, it requires a huge amount of effort to build a new voice and any modification to the selected units for synthesis may induce unwanted glitches. Considering audiobook synthesis, the generated prosody is expressive but at times can be so unnatural that it may influence the intelligibility of the speech. On the other hand, parametric systems offer much flexibility and consistency but the absence of convincing prosody makes them sound dull and lifeless, even when trained on expressive speech data. A question that remains unanswered is whether parametric approaches can generate prosody better than unit selection synthesis systems in the context of audiobooks. If parametric systems are to be used for modelling prosody, are we paying enough attention to the interaction between F0 and duration? We hypothesize that speech synthesis using recurrent neural networks has so much to offer in this direction and is a good testbed to answer these questions.

The guiding hypothesis in this thesis is that recurrent neural networks can generate improved prosody when trained efficiently making use of better representation of linguistic and acoustic features. Therefore, in Chapter 4, we use syllable templates learned from hierarchical clustering of F0 contours for intonation representation. And in Chapter 5, duration is encoded as transitional probability at frame-level thus allowing generation to be performed in many ways. Chapter 7 and Chapter 8 consider using natural linguistic representation as input and perform model-level upsampling to achieve sequence-to-sequence regression. Finally, Chapter 9 investigates the benefit of combining the parametric approach with unit-selection.

1.2.1 Contributions

The contributions of this thesis are:

1. We evaluate a template-based approach for modelling speech intonation. In this approach, we define intonation modelling as a classification problem as opposed to the traditional convention of treating it as a frame-level regression problem in order to mitigate over-smoothing. We show that unsupervised clustering can be used to discover different shapes of F0 contour present in the data.

2. We propose a new approach to duration modelling for parametric synthesis in which a recurrent model is trained to output a phone transition probability at each timestep (acoustic frame). For the first time, we consider frame-level duration modelling and show that the output generation based on median predicted duration can in principle model any distribution supported on non-negative integers.

3. We propose the use of a hierarchical encoder-decoder to improve intonation modelling in an effective and computationally-efficient way. Unlike the usual approach – which requires linguistic feature flattening and upsampling – our proposed encoder accepts linguistic features at their natural timescales, and performs upsampling within the model architecture enabling us to model long-term dependency efficiently.

4. Because of the sequential nature of text-dependent features, we propose a novel multi-scale parallel encoder-decoder to process the linguistic features and other suprasegmental features at different time-scales separately, making use of multiple parallel encoders. We hypothesize that such an encoder may enable the control of output speech generation, although not proven in this thesis.

5. The code for duration modelling and encoder-decoder paradigms is made open-source within the Merlin toolkit. A DNN-guided unit selection system is designed making use of Multisyn and Merlin toolkits and is submitted for Blizzard Challenge 2017. The recipes for the system along with benchmarks used in the challenge are made available through

the Merlin toolkit, thus enabling various academic and research groups to replicate the results and benefit from it.

1.3 Chapter Structure

Chapter 2 presents the background material for this thesis and **Chapter** 3 describes the procedure for preparation of state-aligned labels and details of the corpus used for the experiments presented in this thesis.

The rest of the thesis is divided into two parts: Chapters from 4 to 6 form part-I which investigates the two important acoustic properties of prosody; In **Chapter** 4, we describe a classification-based approach to intonation prediction with syllable F0 templates replacing frame-level regression. In **Chapter** 5, we propose an approach to joint modelling of duration and acoustic parameters in statistical parametric speech synthesis by appending a transition probability to the frame-level acoustic model outputs. **Chapter** 6 then summarizes the findings from Part-I of this thesis.

Part-II consisting of Chapters 7-10 presents new paradigms for SPSS in order to achieve improved naturalness in synthesized speech. **Chapter** 7 presents a hierarchical encoder-decoder for SPSS. **Chapter** 8 presents a multi-scale parallel encoder-decoder, an approach similar to the hierarchical model but that processes different levels of linguistic features with multiple encoders in parallel. **Chapter** 9 describes a DNN-guided unit selection system, which was submitted for the Blizzard Challenge 2017. Finally, **Chapter** 10 summarizes the thesis, findings, contributions, and discusses the scope for future work.

1.4 Published Work

- The template-based approach to intonation generation for SPSS presented in Chapter 4 is published in Ronanki et al. (2016a).

  - Ronanki, S., Henter, G. E., Wu, Z., and King, S. (2016a). "A template-based approach for speech synthesis intonation generation using LSTMs." In *Proc. Interspeech*, San Francisco, USA.

- Median-based generation of synthetic speech durations introduced in Chapter 5 is published in Ronanki et al. (2016c).
    - Ronanki, S., Watts, O., King, S., and Henter, G. E. (2016c). "Median-Based Generation of Synthetic Speech Durations using a Non-Parametric Approach." In *Proc. SLT Workshop*, Puerto Rico, USA.
- The hierarchical encoder-decoder of Chapter 7 is published in Ronanki et al. (2017b).
    - Ronanki, S., Watts, O., and King, S. (2017b). "A Hierarchical Encoder-Decoder Model for Statistical Parametric Speech Synthesis." In *Proc. Interspeech*, Stockholm, Sweden.
- Experiments and results from subjective listening tests presented in Chapter 9 are based on Ronanki et al. (2017a).
    - Ronanki, S., Ribeiro, S., Espic, F., and Watts, O.(2017a). "The CSTR entry to the Blizzard Challenge 2017." In *Proc. Blizzard Challenge Workshop*, Stockholm, Sweden.
- The work on duration modelling for English and three other Indian languages is published in Ronanki et al. (2016d,b).
    - Ronanki, S., Wu, Z., Watts, O., and King, S. (2016d). "A Demonstration of the Merlin Open Source Neural Network Speech Synthesis System." In *Proc. SSW9*, Sunnyvale, USA.
    - Ronanki, S., Reddy, S., Bollepalli, B., and King, S. (2016b). "DNN-based speech synthesis for Indian languages from ASCII text." In *Proc. SSW9*, Sunnyvale, USA.

1.4.1 Other publications

Some of the other publications I have contributed to during my PhD (last two are from my Internship at Amazon Alexa, Cambridge):

- Watts, O., Ronanki, S., Wu, Z., Raitio, T., and Suni, A. (2015). "The NST–GlottHMM entry to the Blizzard Challenge 2015." In *Proc. Blizzard Challenge Workshop*, Berlin, Germany.

- Henter, G. E., Ronanki, S., Watts, O., Wester, M., Wu, Z., and King, S. (2016). “Robust TTS duration modelling using DNNs.” In *Proc. ICASSP*, volume 41, pages 5130–5134, Shanghai, China.

- Merritt, T., Ronanki, S., Watts, O., , Wu, Z., and Clark, R. A. J. (2016b). “The CSTR entry to the Blizzard Challenge 2016.” In *Proc. Blizzard Challenge Workshop*, Cupertino, USA.

- Hodari, Z., Watts, O., Ronanki, S., King, S. (2018). “Learning interpretable control dimensions for speech synthesis by using external data.” In *Proc. Interspeech*, pages 32–36, Hyderabad, India.

- Latorre, J., Lachowicz, J., Lorenzo-Trueba, J., Merritt, T., Drugman, T., Ronanki, S., Klimkov, V. (2019). “Effect of data reduction on sequence-to-sequence Neural TTS.” In *Proc. ICASSP*, Brighton, UK.

- Prateek, N., Łajszczak, M., Barra-Chicote, R., Drugman, T., Lorenzo-Trueba, J., Merritt, T., Ronanki, S., Wood, T. (2019). “In Other News: A Bi-style Text-to-speech Model for Synthesizing Newscaster Voice with Limited Data.” In *Proc. NAACL-HLT*, Minneapolis, USA.

Chapter 2

Background: Prosody modeling for speech synthesis

This chapter presents background material for this thesis. A brief overview of various approaches to text-to-speech is given with more detail provided for speech synthesis based on deep neural networks (DNNs). We also touch upon some of the recent work in neural speech synthesis. A detailed overview of systems with particular emphasis on prosody modeling is also given, providing the theoretical foundations motivating the main claim of this thesis. Finally, the chapter ends with a detailed description of objective and subjective evaluation methodologies used in this thesis.

2.1 Text-to-Speech systems

This section presents a brief overview of conventional approaches to TTS and then details recent work using deep neural network based approaches, including advanced generative models.

2.1.1 Conventional approaches

Conventional TTS systems are broadly divided into two parts: a front-end and a back-end. The front-end analyzes the form of text and converts the given input text into an intermediate *linguistic representation* and the back-end uses this representation to produce a waveform (see Figure 2.1). Due to the nature of this pipeline, the front end is heavily language specific while the waveform production is relatively language independent (since it typically involves gen-

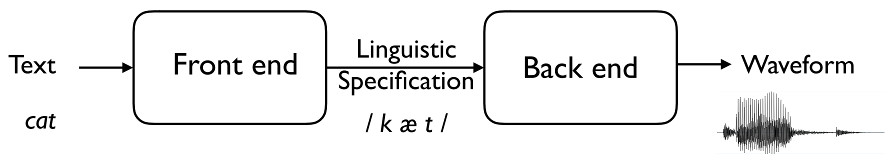


Figure 2.1: The conventional modules used in the common form of speech synthesis.

erating waveform from an intermediate acoustic representation e.g., source-filter vocoders such as STRAIGHT (Kawahara, 2006), WORLD (Morise et al., 2016), and neural vocoder (Lorenzo-Trueba et al., 2018) which generates waveform from mel-spectrograms).

2.1.1.1 Front-End

The front-end analyses the text and converts it into a linguistically-meaningful representation often termed the *linguistic specification* – in its most simple form it contains the phones of an utterance, but it generally contains many other features describing the linguistic context of each phone and suprasegmental features of the whole utterance. The main task of a front-end is to derive the features that can reduce the gap between text and speech.

In a conventional front-end pipeline (Festival[1] (Black et al., 2001), for example), the input text is first tokenised based on whitespace, and surrounding punctuation is added as features to the tokens. After tokenisation, a disambiguation step identifying token parts of speech (POS) is performed to disambiguate between different possible interpretations of surface tokens. This applies to identifying cardinals, ordinals, nominals and numerals. For example, 4-digit numbers are pronounced differently when they specify a year and also certain group of alphabets like Roman numerals are pronounced differently. After token disambiguation, text normalisation is then performed to convert these non-standard words and abbreviations into words. This is generally achieved with weighted finite-state transducers as proposed by Sproat (1996). Each word is then assigned a POS tag using a HMM tagger, which assigns the most likely sequence of POS tags given a sequence of words (Jurafsky and Martin (2014) pp. 218–220). The POS tags, if assigned correctly, further help to disambiguate between various possible pronunciations of words (e.g., record, live).

[1] http://www.cstr.ed.ac.uk/projects/festival/

Given a sequence of words, a carefully prepared lexicon (also known as the dictionary) is generally used to derive each word's pronunciation. If a word is not found in the lexicon, its pronunciation is predicted using letter-to-sound (LTS) rules, often termed as Grapheme-to-phoneme (*G2P*) and it is then syllabified. In Festival, the LTS "rules" are a classification which can be automatically learned from the lexicon (Black and Taylor, 1997b; Black et al., 2001). G2P is commonly achieved using joint-sequence models (Bisani and Ney, 2008) or neural networks (Rao et al., 2015) or a combination of both.

Phrase breaks in the utterance are predicted through a combination of rules and using a classification tree (Black et al., 2001). Once the pronunciation of all words is defined, pause markers (often denoted by "*pau*" in Festival) are inserted where phrase breaks are predicted. Finally, post-lexical rules are applied to the phoneme sequence.

Festival (used as the front-end in this thesis) in general extracts several other suprasegmental features based on the linguistic context. These suprasegmental features can include for example syllable features such as onset, coda, syllable stress and positional features such as position of syllable in word and position of stressed syllable in word and position of word in phrase. The linguistic features derived through this pipeline are processed and passed to the back-end to generate waveform. The features derived from the Festival front-end, which are used in the synthesis back-end in this thesis are listed in Table 3.3 of Chapter 3.

2.1.1.2 Back-End

The task of the back-end is to convert the intermediate linguistic specification into a speech waveform. In most parametric based approaches, the back-end involves generation of a certain intermediate acoustic representation. For this reason, the back-end is relatively language independent and often termed the *waveform generator*. Various techniques have been proposed to address waveform generation, given a linguistic specification from the front-end.

In the early days of TTS research (Klatt, 1987), researchers primarily focused on rule-based synthesis techniques (Carlson and Granstrom, 1976), where the rules are designed by experts. **Formant synthesis** is one such technique which uses formants, the resonance frequencies of the vocal tract, to synthesize the speech (Klatt, 1980). **Articulatory synthesis** is also one of the earliest fundamental paradigms, which tries to model the human articulatory system, for example, vocal cords and the vocal tract (Scully, 1990; Greenwood, 1997). The shape of the vocal tract can be controlled in a number of ways by modifying the speech articulators, such as the tongue, jaw and lips. Both these techniques rely on a set of well-defined rules designed by expert linguists. Although speech synthesized from these techniques is quite intelligible, it usually sounds artificial and robotic (Taylor, 2009, §13.2.7).

Concatenative synthesis is perhaps the most common approach for waveform generation in commercial systems. This approach is based on concatenation of pre-recorded speech units to generate the utterance. This approach is commonly used to concatenate words from a limited domain vocabulary, for example, telecommunication services, announcements at public places such as train stations and airports. For large vocabulary generic synthesis, smaller speech units (phonemes or diphones or syllables) are used for concatenation. Some of the early systems used a fixed-size unit inventory for synthesis, and this is commonly known as *diphone synthesis* (Sagisaka et al., 1992; Donovan and Eide, 1998).

Unit selection synthesis emerged as an extension of diphone synthesis, making use of a large speech database which contains many occurrences of each unit across different contexts (Hunt and Black, 1996; Black and Taylor, 1997a). The task of the waveform generator is to select the sequence of units that best matches the target utterance and then use signal processing algorithms, such as overlap and add (OLA) to concatenate the units (Black et al., 2001). Signal modifications are to be minimized to preserve the naturalness. Since unit selection synthesis uses the original recordings in the database, it is more natural than the other two methods i.e., articulatory and formant synthesis.

Parametric synthesis, also known as model-based synthesis, uses a parametric representation of speech waveforms, which are modeled via statistical frameworks. For this reason, these systems are often grouped under the term *statistical parametric speech synthesis* (SPSS, Black et al. (2007); Zen et al. (2009)).

There are two common types of paradigm that are in use today: *Hidden Markov Model (HMM) based synthesis* (Yoshimura et al., 1999) and *Deep Neural Network (DNN) based synthesis* (Zen et al., 2013). Both these paradigms involve two stages: (1) Training stage, and (2) Synthesis stage, and often use a *vocoder* for analysis and synthesis. In the context of speech synthesis, a *vocoder* (or voice encoder) is used to transform the speech waveform into a sequence of acoustic parameters and reconstruct back with a minimal error. During the training stage, the vocoder is used to extract acoustic parameters from a database of speech waveforms. The statistical models are then trained to optimize a mapping function between linguistic features derived from the front-end and acoustic features extracted from the vocoder. At synthesis time, the input linguistic features are mapped to acoustic parameters using a pre-trained model and then a vocoder is used to generate the waveform. The most common forms of vocoder are based on the source-filter model of speech production, for example WORLD and STRAIGHT. In this thesis, STRAIGHT is used for both analysis and synthesis. We will provide further details about acoustic feature extraction in Section 3.3.2.

Parametric systems offer flexibility in terms of manipulation and control of acoustic parameters and are well suited for various tasks such as voice conversion and speaker adaptation. Traditional approaches to HMM-based speech synthesis (Yoshimura et al., 1999; Zen et al., 2004, 2007) use decision tree-clustered context-dependent hidden Markov models to represent probability densities of speech parameters given linguistic specification. The Decision tree can be viewed as a mapping function from the linguistic specification to the acoustic space. Within each HMM state, the spectrum is modeled by a continuous multivariate Gaussian while F0 is modeled by a multi-space probability distribution (Yamagishi, 2006). At synthesis time, speech parameters are generated from the probability densities to maximize their output probabilities. HMM-based synthesis is quite effective and robust but it has some limitations: 1) Gaussian Mixture Models (GMMs) cannot model highly correlated features, which restricts its capacity; 2) The Markov property makes the HMM less capable (compared to neural networks) of capturing temporal dependency in the sequence; 3) Building decision trees relies on a partition of the input space and hence results in reduction of data per region and poor generalisation (Zen et al., 2013). Therefore, decision trees can be replaced by a more powerful regression model, for example, a random forest (Black and Muthukumar, 2015) or deep neural network (Zen et al., 2013). Since, the work in this thesis is concerned with the DNN-based approach for waveform generation, Section 2.1.2 will provide further details on this class of approach.

Hybrid synthesis is a generic term covering various techniques that combine unit selection and parametric synthesis. The most common hybrid systems in general use parametric based models to inform selection of units (Yan et al., 2010; Qian et al., 2013; Merritt et al., 2016a). There are two common types of current hybrid systems: (1) Target cost function as basis; (2) Multiform synthesis. The first type of techniques employ statistical models (HMMs or DNNs) to predict the acoustic properties of speech and then use distance in embedding space or parametric space as a target cost. While some approaches replace the target cost as proposed in Hunt and Black (1996), others combine linguistic-based target cost and parametric-based target cost. The second type of approach based on multiform synthesis allow some units to be generated via vocoding, whilst others are retrieved from the speech database (Pollet and Breen, 2008; Sorin et al., 2011; Fernandez et al., 2015). Both these approaches may also modify some of the speech units as per the predicted target. Chapter 9 in this thesis presents a hybrid synthesis system which drives a unit selection synthesiser using the output from a neural network acoustic and duration model.

While the above approaches still form the basis of most speech synthesis systems, new paradigms such as *WaveNet* (van den Oord et al., 2016) and *Tacotron* (Wang et al., 2017c) have recently been developed[2], and improvements have been made in parametric systems using deep neural networks. Recent progress has been largely motivated by three factors: (1) Use of GPUs to train the models more rapidly, (2) a substantial increase in the amount of widely available text and speech databases, and (3) improvements made in other machine learning areas including speech recognition and computer vision, due to which, there have been significant improvements in ways to do speech synthesis. These have led to considerably more naturalness and variations in synthesized speech, thus allowing for better modeling of prosody.

2.1.2 Deep neural network-based speech synthesis

A neural network is a collection of large number of processing units called nodes, often arranged in a series of layers. Neural networks are considered powerful regression models that can capture the complex interactions between input and output features. Neural networks for speech synthesis flourished

[2]Concurrently with the work reported in this thesis.

in the early 1990s (for example, Sejnowski and Rosenberg (1988); Weijters and Thole (1993); Tuerk and Robinson (1993)). The techniques investigated throughout this thesis rely on two broad distinctive types of network: feed forward neural networks and recurrent neural networks.

Feed-forward neural network

A feed-forward neural network is the first and simplest type of neural network wherein connections between nodes do not form a cycle. The models are called feedforward as the information from input to output always flows in the forward direction, through the hidden nodes. A very common type of feedforward neural network is a multi-layer perceptron, where nodes are arranged in a series of hidden layers, with connections being made between each layer in the network. Each node receives information from all the nodes in the previous layer and passes it through an activation function before sending to the next layer. The goal of a feedforward network is to approximate a function $f(x)$ whose output is y, given the input x. The parameters of the function are determined by weights associated with the connections between nodes. Figure 2.2 illustrates the process of forward propagation at time t whose input is x_t.

If $f(x_t)$ denotes the forward mapping of x_t at time t, we compute activation of the first hidden layer (h_1) as:

$$h_1 = a(W_{in}x_t + b_1) \tag{2.1}$$

where $a(.)$ is an activation function, W_{in} represents the weight matrix between input and hidden layer and b_1 represents the bias for the first hidden layer. The hidden activation h_n for the remaining $n-1$ layers can be generalized as:

$$h_n = a(W_n h_{n-1} + b_n) \tag{2.2}$$

where W_n represents the weight matrix and b_n represents the bias for corresponding hidden layer. The final output ($\widetilde{y_t}$) is approximated as below:

$$\widetilde{y_t} = a(W_{out}h_n + b_{out}) \tag{2.3}$$

where W_{out} represents the weight matrix between final hidden layer and output layer and b_out represents the bias for output layer.

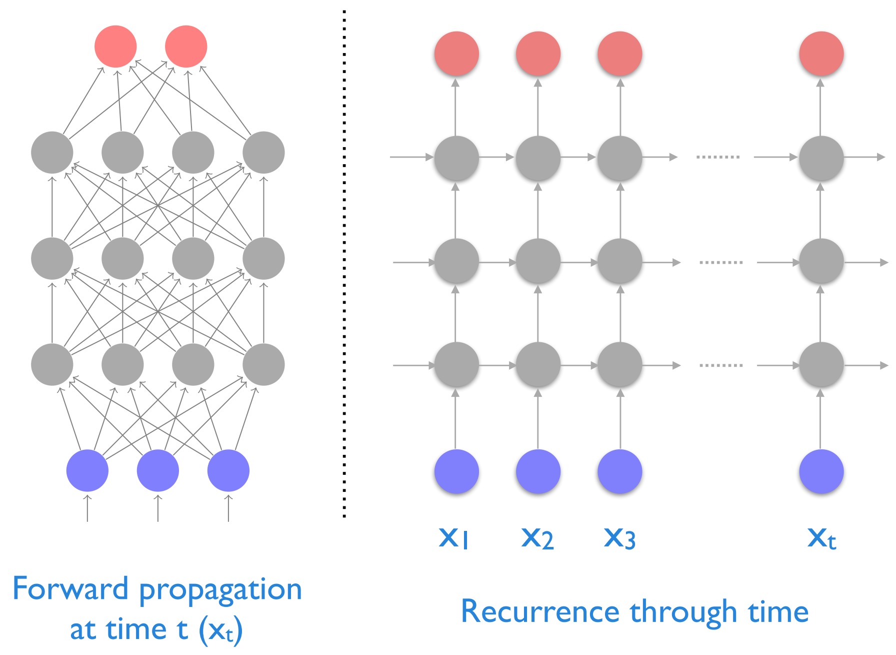


Figure 2.2: Schematic diagram of feedforward and recurrent neural networks

Recurrent neural network

A feedforward neural network assumes that all inputs across time are independent of each other. But most of the real-world problems including speech synthesis and recognition depends on the context. The idea behind a Recurrent Neural Network (RNN) is to make use of sequential information by allowing the network to capture the temporal dependency between inputs and outputs. RNNs are called recurrent as they perform the same task at each time step with output being dependent on previous computations. The following equations describe the basic vanilla RNN:

$$h_t = g(W_h x_t + U_h h_{t-1} + b_h) \tag{2.4}$$

where h_t is the hidden output at time instance t, W_h, and U_h denote the weight connections from input and past hidden state, and b_h denote the bias.

Figure 2.2 illustrates the process of recurrence through time, whose output is calculated through a series of recurrent layers. In theory, RNNs can make use of information over a long sequence but in practice they are limited to a few timesteps as the gradient (*Backpropagation* is used to compute the gradient) tends to get smaller as we move backward through the hidden layers. This phenomenon is commonly known as the *vanishing gradient problem*. There are several variants of RNN, for example, Long short-term memory (LSTM), Gated recurrent unit (GRU) which are much better at capturing long-term dependencies than simple RNNs. Hochreiter and Schmidhuber (1997) proposed the basic idea of LSTM and the most commonly used architecture was described in Graves and Schmidhuber (2005) and is formulated as:

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + p^i \odot c_{t-1} + b^i) \tag{2.5}$$
$$f_t = \sigma(W^f x_t + U^f h_{t-1} + p^f \odot c_{t-1} + b^f) \tag{2.6}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W^c x_t + U^c h_{t-1} + b^c) \tag{2.7}$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + p^o \odot c_t + b^o) \tag{2.8}$$
$$h_t = o_t \odot g(c_t) \tag{2.9}$$

where i_t, f_t, c_t, o_t and h_t are input gate, forget gate, cell state, output gate and block output at time instance t, respectively. $\sigma()$ and $g()$ are the sigmoid and tangent activation functions, respectively; x_t is the input at time t; W*, and U* are the weight matrices applied on input and recurrent hidden units, respectively; p* and b* are the peep-hole connections and biases, respectively; and $\odot$ means element-wise product.

Gated Recurrent Unit (GRU) architecture proposed by Cho et al. (2014) is one of the most popular variants of LSTM. Unlike LSTM, the GRU does not use any peep-hole connections and output activation functions. It also combines the input and forget gate to form an update gate. Wu and King (2016) presented an even simpler variant for LSTM where only the forget gate is retained, removing the output gate and peep-hole connections. The input gate is also replaced by the forget date in the form of $1 - f_t$. The simplified LSTM (SLSTM) architecture is similar to the GRU, except that it uses a memory cell state and is formulated as:

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \tag{2.10}$$
$$c_t = f_t \odot c_{t-1} + (1 - f_t) \odot g(W^c x_t + U^c h_{t-1} + b^c) \tag{2.11}$$
$$h_t = g(c_t) \tag{2.12}$$

The experiments conducted by Wu and King (2016) on a read-speech corpus demonstrate that the simplified LSTM with significantly fewer parameters than the vanilla LSTM achieves similar performance. For this reason, SLSTMs are used for most of the work in this thesis.

Cost function

Neural networks rely on a cost function for training and optimizing a model. The cost function is used to optimize the DNN predictions. In this thesis, we use mean square error (MSE) loss as cost function for approximating acoustic features as shown in the below equation:

$$C(w,b) = \frac{1}{2T} \sum_{x} ||y - f(x)||^2 \tag{2.13}$$

Here, w denotes the collection of all weights in the network, b denotes the biases and T is the total number of training inputs. For classification, loss functions such as Hinge and cross-entropy are commonly used.

In Chapter 4 of this thesis, we consider using a Connectionist Temporal Classification (CTC) based cost function. Given an input sequence (X) and output sequence (Y), the CTC conditional probability marginalizes over the set of valid alignments by computing the probability for a single alignmnet step-by-step as shown below:

$$P(Y|X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X) \tag{2.14}$$

where A denote the set of valid alignments, a_t denote a single alignment and p_t denote the probability of single alignment given input sequence X.

The CTC alignments give us a natural way to go from probabilities at each time-step to the probability of an output sequence. This type of loss function is commonly used when the network needs an alignment between input and output sequence whose lengths may vary.

Training mechanism

While designing a neural network, in the beginning, we initialize weights and biases. In this thesis, we followed the implementation of the Merlin Neural Network Toolkit (Wu et al., 2016), where weight matrices (w) between layer A and layer B are initialized by sampling independently from a Gaussian distribution with zero mean and a standard deviation of $\frac{1}{\sqrt{n_A}}$ where n_A is the number of nodes in the layer A.

Given the initialization of network parameters, the cost function is used to compute the error between the target and predicted output. Neural networks are then trained using *backpropagation* and the parameters (weights and biases) are updated iteratively so as to minimise the error between actual output and predictions. This process begins by finding the partial derivative of the cost function with respect to network parameters. Such a process is known as *gradient descent* which is an iterative optimization algorithm to find the local minimum of a function. *Stochastic Gradient Descent* (SGD) is similar to gradient descent, but the gradient is approximated on a subset of training samples and updates the parameters accordingly. However, the updates are usually done over each mini-batch (subset) in order to avoid large variances in estimated gradients. When these updates are done for the full data, it is considered as one training epoch.

For recurrent neural networks, the frame-level error is computed and accumulated for each utterance using L2 loss. This loss is used to compute the gradients and make updates during back-propagation. Unless mentioned otherwise, the recurrent neural networks used in this thesis are trained with a batch size of 1 (except in Chapter 8, where batch size of 3 is used). SGD is used as optimizer throughout this thesis.

The optimization process continues iteratively over several epochs until a stopping criterion is met. This criterion often follows a pre-defined total number of training epochs or when the error on the validation set doesn't improve for a certain number of epochs. If the training is stopped based on validation error, such a criterion is often referred to as *early stopping*. We follow this stopping criterion based on the implementation of the Merlin Neural Network toolkit (Wu et al., 2016).

Conventional DNN-based SPSS

Deep neural networks (DNNs) use multiple layers of hidden representations to learn complex interaction between input and output features. In DNN-based speech synthesis, DNNs are used to learn the mapping from text-based linguistic features to acoustic features, in particular vocoder parameters extracted from speech.

DNN-based speech synthesis consists of two stages: offline training and run-time synthesis. During offline training, linguistic features derived from text are aligned with corresponding acoustic features extracted from speech

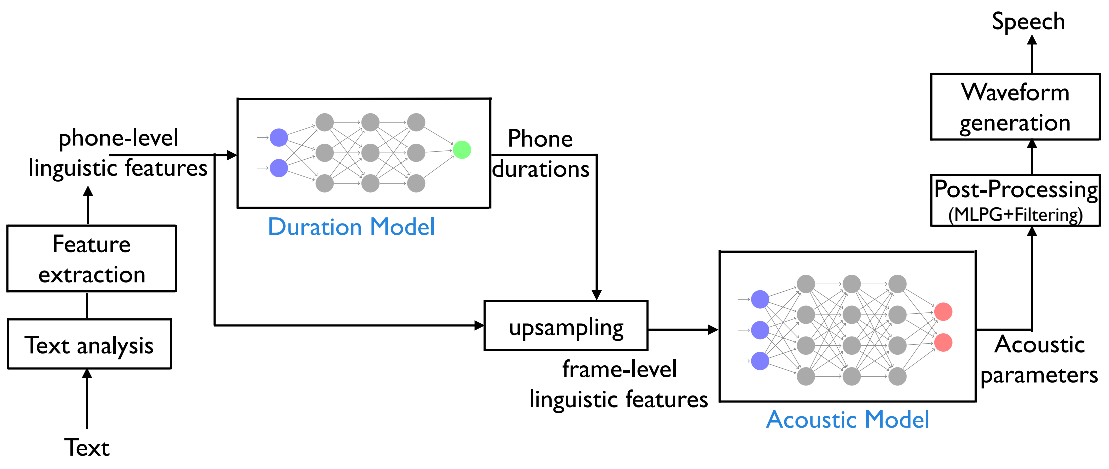


Figure 2.3: Schematic diagram of the conventional SPSS system

signals. The alignment may be obtained by performing HMM-based forced-alignment at phoneme or HMM-state level. The acoustic features in general include vocoder parameters, such as Mel-Cepstral Coeffcients (MCCs), Band aperiodicity coefficients (BAP), fundamental frequency (F0) and their dynamic features (delta and delta-delta coefficients). The input to a DNN is frame-by-frame, and so it cannot handle the dependency between frames very well. To solve this, Zen et al. (2013) included dynamic features in the acoustic parameters.

Given the alignment between linguistic features and acoustic parameters, a feedforward neural network called the acoustic model is trained using any of the popular deep learning toolkits, namely Theano, Tensorflow, PyTorch. In this thesis, the Merlin toolkit (Wu et al., 2016; Ronanki et al., 2016d) is used which relies on Theano for training deep neural networks. The later versions of Merlin support Tensorflow and Keras as well but they are not used for experiments in this thesis. At run-time synthesis, phone-level linguistic features are used to predict phone/state durations. The predicted durations are used to upsample phone-level linguistic features to frame-level to create a one-to-one mapping. Then the upsampled frame-level linguistic features are mapped to acoustic parameters by a trained DNN using forward propagation. Maximum likelihood parameter generation (MLPG) (Tokuda et al., 2000) using variances computed from the training data is applied to output features; spectral enhancement post-filtering or Global Variance (GV) is applied to the resulting MCC trajectories (Wu et al., 2015b) for smoothing. The vocoder (for example, STRAIGHT) is used to synthesize the final waveform.

Application of DNNs for SPSS

Most recent approaches with DNNs for SPSS aim to overcome the limitations given by the conventional modeling of speech parameters with decision trees and Gaussian distributions. However, recent improvements in deep learning and data availability have caused a resurgence of these methods. Zen et al. (2013) re-introduced neural networks for speech synthesis where a DNN was used as a fairly straightforward drop-in replacement for the usual regression tree of HMM-based SPSS, performing the complex mapping from linguistic features to speech parameters on a frame-by-frame basis. Later, others (Qian et al., 2014; Wu et al., 2015b, for example) explored several variants on this basic idea.

In Qian et al. (2014), a DNN is trained to predict acoustic features (LSP, F0 and VUV) with minimal data (5 hours), with each layer in the network pre-trained by Restricted Boltzmann Machines (RBMs). Wu et al. (2015b) introduced multi-task learning in DNN-based speech synthesis to predict a perceptual representation of the target speech as a secondary task. Wu et al. (2015b) also used stacked bottleneck features to produce a wide context around the current frame. Both these variations have shown to improve the overall naturalness over a baseline HMM significantly.

Given that speech is a time series, and SPSS is a sequence-to-sequence regression problem, it is then natural to apply various recurrent architectures. Fan et al. (2014) applied Bi-directional Long Short-Term Memory (BLSTM) neural networks to better capture temporal information for sequence transformation. The work also showed that deep bi-directional LSTMs with two layers of recurrence work better than shallow ones with one layer of recurrence. Zen and Sak (2015) proposed a low-latency, streaming speech synthesis architecture using unidirectional LSTM-RNNs with a recurrent output layer. The recurrent output layer makes smooth transitions between acoustic features at consecutive frames without the need of MLPG.

Wu and King (2016) investigated the effects of LSTM for SPSS. The results show that a simplified LSTM with only forget gates is as good as a conventional LSTM (with all gates and peepholes). Zen et al. (2016) proposed further optimizations such as weight quantization, multi-frame inference, and robust inference using an ϵ-contaminated Gaussian loss function for deployment on mobile devices.

Due to the unimodal nature of the objective function used in DNNs and the lack of ability to predict variances, Zen and Senior (2014) investigated the use of a mixture density output layer that can estimate full probability density functions over real-valued acoustic features conditioned on the corresponding linguistic features. Wang et al. (2017a) proposed a recurrent mixture density network (RMDN) that incorporates a trainable autoregressive model. An advantage of incorporating an autoregressive model is that the time dependency within acoustic feature trajectories can be modeled without using the conventional dynamic features.

Some approaches aimed at extracting spectral features from raw spectral representations for acoustic modeling use deep learning techniques. Hu and Ling (2016) presented a method for deriving spectral features using a deep belief network (DBN) for hidden Markov model (HMM)-based parametric speech synthesis. Takaki and Yamagishi (2016) investigated the use of deep autoencoder-based, non-linear, data-driven and unsupervised low-dimensional feature extraction using spectral envelopes for statistical parametric speech synthesis. Wang et al. (2015) used lexical word embeddings that are pre-trained using neural network language model (NNLM) for RNN-based SPSS. The results indicate that such input features are useful in the absence of manually labelled POS and TOBI features.

DNNs are also used in post-filtering for SPSS to deal with the over-smoothing effect of parameter generation. These approaches aim to map generated spectral features towards natural ones using DNNs or DBNs. Chen et al. (2015) trained a DNN in a generative way without fine-tuning to map generated spectral features towards natural ones. Muthukumar and Black (2016) investigated the use of RNNs as a potential postfilter for synthesis. Kaneko et al. (2017) proposed a postfilter based on a generative adversarial network (GAN) to compensate for the differences between natural speech and predicted speech focusing on the differences caused by over-smoothing. However, all these post-filters still depend on parameter generation.

All the above studies have demonstrated that LSTMs can achieve significantly better performance on SPSS than deep feed-forward neural networks. However, all these systems rely on a pipeline of steps: 1) text analysis and input feature extraction to derive linguistic features; 2) a duration model; 3) an acoustic model to generate parameters; 4) post-processing and 5) a vocoder to generate the final waveform. The following section on novel paradigms aims at joint optimization of some of these steps.

2.1.3 Novel paradigms: End-to-End systems

While parametric speech synthesis systems are vastly improved with the re-introduction of deep neural networks, the vocoder has become a bottleneck in achieving higher naturalness. For this reason, most commercial systems rely on hybrid unit selection which uses SPSS to inform selection of units. Even though they perform well by combining the benefits of both unit selection and SPSS, there is little flexibility in terms of prosody modeling and the method doesn't scale for multi-speaker and multi-language modeling. Any modification to the selected units introduces artefacts and degrades the overall synthesis quality.

The drawbacks with such hybrid systems can be summarised as:

- They are too complex and require multiple components or systems to perform waveform synthesis.
- They are difficult to fine-tune without affecting the naturalness of synthesized speech.
- It requires lot of engineering to select desired units and the prosody of final output often sounds inappropriate for a given context.

Hybrid speech synthesis involves lot of engineering and lacks underlying theory. Therefore, researchers have been working on end-to-end systems which can generate natural waveforms and as well as allowing for flexibility.

Advances in neural networks such as convolutional neural networks and sequence-to-sequence (seq2seq) recurrent networks have led to new paradigms in neural speech synthesis. These techniques, however, require substantial computational power and large datasets to realise their full potential. Fortunately, there has also been a revolution in the availability of computational resources and data – now virtually limitless amounts of speech, in an endless variety of voices, can be obtained from many different sources but may not be of sufficient quality.

van den Oord et al. (2016) introduced WaveNet, a deep neural network for generating quantized audio waveforms. It was first of a kind towards direct modeling of waveform samples and used a fully probabilistic and autoregressive model to predict each audio sample conditioned on all previous ones. Similarly, Mehri et al. (2017) proposed sampleRNN for unconditional audio generation using stateful recurrent neural networks in a hierarchical structure to capture underlying sources of variations in the temporal sequences over very long time spans. Extending the work by sampleRNN, Char2Wav implemented an attention-based phoneme duration model and an equivalent F0 prediction model, effectively providing local conditioning information to a SampleRNN-based vocoder. In a similar direction, Tamamori et al. (2017) proposed a speaker-dependent WaveNet vocoder for synthesizing speech waveforms, by utilizing acoustic features from an existing vocoder as auxiliary features of WaveNet.

Despite its superior quality over existing hybrid approaches, the original Wavenet was too slow because of its sequential generation of one audio sample at a time. Therefore, the focus quickly turned to optimizing Wavenet-like systems for real-time synthesis. The work by Arik et al. (2017) presented Deep Voice, a production-quality text-to-speech system constructed entirely from deep neural networks making use of optimized WaveNet inference kernels on both CPU and GPU. Later, Deep Voice 2 (Arık et al., 2017) introduced multi-speaker neural Text-to-Speech which can learn hundreds of unique voices from less than half an hour of data per speaker while achieving high quality audio synthesis. Similarly, van den Oord et al. (2017) introduced Probability Density Distillation, a new method for training a parallel feed-forward network from a trained WaveNet with no significant difference in quality.

Although, WaveNet-like paradigms have shown that they can achieve better quality than hybrid systems, they still rely on other statistical parametric systems to provide F0 and duration along with upsampled linguistic features thus limiting the scope of WaveNet to a vocoder with high naturalness. Wang et al. (2017c) have proposed an end-to-end speech synthesis system named *Tacotron*, in order to capture the full sentence context, leveraging sequence-to-sequence models. Although this can be trained from characters and thus allows systems to train on multiple languages and domains without any expert knowledge, the naturalness was quite limited due to the reconstruction of speech waveforms from linear-spectrograms using the Griffin-Lim algorithm. Therefore, Shen et al. (2017) proposed mel-spectrogram prediction with seq2seq networks followed by a modified WaveNet model acting as a vocoder to synthesize time-domain waveforms from those spectrograms (subsequently named *Tacotron2*). This approach combined the benefits of both Tacotron and WaveNet thus allowing for better prosody modeling with high naturalness. Similarly, Ping et al. (2018) proposed *Deep Voice 3*, a fully-convolutional attention-based neural text-to-speech (TTS) system, and compared several different waveform synthesis methods including Griffin-Lim, WORLD and WaveNet. A few other recent attempts (Wang et al., 2018; Skerry-Ryan et al., 2018) have also shown the capabilities of Tacotron2 for prosody modeling by extending the work towards expressive speech synthesis using global style tokens. All these methodologies have shown that neural networks can now directly generate synthetic speech waveforms, without the limited quality of a vocoder.

The work in this thesis doesn't consider any of these new paradigms, as they are still at very early stages at the time of writing and require substantial improvements to work with audiobooks.

2.2 Speech prosody

Despite decades of research, synthetic speech is still not convincingly natural (King, 2014). A particular shortcoming is prosody: durations, energy, and intonation tend to be dull, inappropriate, and unattractive. These issues prevent the use of synthesised speech in otherwise appealing applications, such as audiobooks. This section presents a brief overview of speech prosody, providing the theoretical foundations motivating the main claim of this thesis.

2.2.1 Linguistic theory on Prosody

Although prosody is generally explained from different aspects of the speech signal (such as emotions and sentence types), it is also concerned with the linguistic domain. The realization of speech signal is typically associated with the structure of linguistic units such as syllables, words, phrases or utterances and prosody is often used to indicate the prominence of such linguistic units.

Traditionally, there are two perspectives on prosodic phenomena: a phonological approach and a phonetic approach (Shattuck-Hufnagel and Turk, 1996; Nooteboom, 1997). A phonological approach focuses on higher-level abstract constituents and the associated prominence. The study of prosodic structure is believed to be a linguistic universal (Turk and Shattuck-Hufnagel, 2014). The universals of these structures are extracted from prosodic phonology and the prosodic constituents represent different types of linguistic information. Shattuck-Hufnagel and Turk (1996) provides an overview of hierarchical prosodic constituency and their interactions. Similar to prosodic constituency, prominence is also realized hierarchically (Turk and Shattuck-Hufnagel, 2014), with each level associated with a linguistic unit (such as syllable prominence and word prominence). Wagner and Watson (2010) reviews prosodic prominence to be typically influenced by factors such as discourse and information structure. In linguistics, intonation is realized as a phonological combination with tonal events describing pitch events (such as High (H), Low (L) and their permutations). Silverman et al. (1992) introduced Tones and Break Indices (ToBI) as a common framework for the annotation of tonal events and constituent boundaries. However, it requires a lot of human effort to manually transcribe the F0 contour with respect to multiple acoustic cues.

On the other hand, a phonetic approach is more focused on the acoustic signal and the variations that cannot be associated with individual segments. This thesis focuses on the phonetic approach and attempts to model both segmental and suprasegmental variations of the prosody using neural networks. We'll discuss the theory and background on intonation and duration modelling using this type of approach in the following sections.

2.2.2 Intonation modeling

In this section, we give a brief overview of how intonation has been generated in synthetic speech, with a particular emphasis on the use of statistical methods for this task. This provides context for the proposed method in Section 4.2.

Conventional intonation modelling

Conventional HMM+regression tree speech synthesis predicts frame-wise

F0 from contextual linguistic features either using a multi-space probability distribution (MSD) (Yoshimura et al., 1999) or by continuous F0 modelling (Yu and Young, 2011). Recently the regression tree has been replaced by a deep neural network (DNN) (Zen et al., 2013), significantly improving subjective naturalness (Watts et al., 2016).

Long memory in prosody modelling

Recurrent neural networks (RNNs), in principle, enable long-range dependencies to affect prediction output. They provide superior output compared to feedforward DNNs for acoustic modelling (Zen and Sak, 2015; Wu and King, 2016), at least when natural speech durations are used for synthesis. Our investigation considers the more practical scenario of F0 prediction with RNNs in a full synthesis system, without 'oracle' cues from natural speech durations. Fernandez et al. (2014) proposed predicting state-level F0 statistics and durations using so-called long short-term memory (LSTM) units. However, predictions were made at the state level (3-state HMM), while we predict syllable-level templates that do not assume within-state stationarity. We use simplified LSTMs (Wu and King, 2016), and parametrise our F0 contours using templates. Yin et al. (2016) proposed two different model structures: cascade and parallel DNN, to embody hierarchical and additive properties of F0. Similarly, part of our approach considers using a hierarchy of DNNs for predicting F0.

Parametric decomposition for F0 modelling

Several parametric decomposition approaches have been proposed for the stylization of prosody. The *Tilt* model (Taylor, 1998) was designed to represent intonational events such as pitch accents and boundary tones. The analysis algorithm uses a set of transformations and parametrizes intonational events to produce *Tilt* parameters (Taylor, 1998, 2000). The Fujisaki model of intonation (Fujisaki and Hirose, 1982; Fujisaki et al., 1998) provides a parsimonious representation of F0 and identifies two components: the phrase (or global component), and the accent (or local component), whose amplitudes and timings are given by underlying phrase and accent commands. In the last couple of decades, several approaches described the methods for automatic extraction of Fujisaki model parameters and its applications for speech synthesis (Agüero et al., 2004; Mixdorff, 2015, for example). Several time-varying functions were also used for parametric decomposition of F0 contour: Bezier polynomials (Escudero et al., 2002), Legendre polynomials (Hsia et al., 2010) or cubic spline functions (Wu et al., 2008).

Time-frequency decomposition for F0 modelling

Since prosody varies slowly and constitutes multiple temporal resolutions, there has been considerable amount of research towards efficient representation of intonation. In 2008, discrete cosine transform (DCT) coefficients were proposed for representing F0 patterns (Latorre and Akamine, 2008; Teutenberg et al., 2008). The use of DCTs to represent F0 at, e.g., syllable and word levels, enables a compact representation of complex contours of varying durations (Qian et al., 2011). The continuous wavelet transform (CWT) has also been found to improve F0 modelling in HMM SPSS (Suni et al., 2013). Ribeiro and Clark (2015) explored a multi-level representation of F0 by combining both DCT and CWT representation, at different wavelet scales representing F0-contour variations from phone up to phrase and utterance.

In Yin et al. (2014), a regression tree was used to model DCT-parametrised phrase-level F0 trajectories, to generate smoothly varying contours. In Chapter 4 of this thesis, we similarly consider using the DCT to parameterise each syllable-level contour. However, the reconstruction will be done with template class predicted, and joins smoothed, using deep learning techniques.

Template-based F0 generation

While the vast majority of F0 prediction approaches are based on regression rather than classification, the idea of clustering and/or discretising F0 contours is not new (cf. Obin et al. (2014); Dall and Gonzalvo (2016)). In Anumanchipalli et al. (2011), *k*-means clustering identified representative accent shapes over syllables, with phrase-level structure predicted by a regression tree. Modelling local and global components of F0 separately was shown to improve upon conventional F0 models.

We combine key elements of a template-based approach similar to Anumanchipalli et al. (2011) with much stronger, long-range predictors (LSTMs). The use of templates is intended to capture and reproduce salient, syllable-level features of the F0 contour without over-smoothing. The choice of template per syllable should *not* be made independently of surrounding choices, so we experimented with the connectionist temporal classification (CTC) framework (Graves et al., 2006) to predict *sequences* of templates.

2.2.3 Duration modeling

In this section, we give a brief overview of how speech-sound durations have been generated in synthetic speech, with a particular emphasis on the use of statistical methods for this task. This also provides context for the non-parametric frame-based duration models proposed in Section 5.3.

A Brief Review of Modelling and Generating Durations

Duration modelling has a long history in speech synthesis. In early, formant-based synthesis systems, phone durations were generated by rule (Klatt, 1987). Rules were commonly hand-crafted, rather than learned from data, meaning that no statistical modelling was used. The concatenative synthesis approaches, especially the single-instance concatenative diphone systems required duration modelling, but subsequent unit selection synthesis systems (Hunt and Black, 1996) did not necessarily require modelling or generating durations, since the units themselves (typically diphones) possess intrinsic durations. Having said that, some approaches allowed predicted durations to be incorporated into the target cost (Bulyko and Ostendorf, 2001) while some other modified the duration of selected units where there is a large difference with target.

The rise of statistical parametric speech synthesis (SPSS) (Zen et al., 2009; King, 2011) has introduced a new methodology for duration generation, in which a statistical model (probability distribution) is created to describe speech-sound durations. This distribution is supplemented by two things: a machine learning method to predict the properties of the statistical model from the input text (i.e, the linguistic features), and a principle for generating durations from the model distribution, such as random sampling or taking the mean of the distribution.

The first SPSS systems were based on simple hidden Markov models (HMMs), which implies that state durations (commonly five states per phone) are assumed follow a memoryless geometric distribution. Decision trees (DTs) were used to predict the distribution parameter based on training data, with

Table 2.1: Methods for generating speech-sound durations (last column) in different types of TTS. Statistical methods incorporate a distribution, a method of predicting its properties, in addition to a method of generating output from the predicted distribution.

TTS type	Distribution	Level	Predictor	Generation
Formant	-	Phone	-	Rule
Concat.	-	Phone	-	Exemplar
HMM	Geometric	State	DT	Mean
HSMM	Parametric	State	DT	Mean
NN	Gaussian	State	DNN/RNN	Mean
Proposed (Sec. 5.3)	Non-parametric	Frame	DNN/RNN	Median

the mean of the predicted distribution used for generation.

In reality, a geometric distribution is quite far removed from how natural speech durations are distributed. Zen et al. (2004) introduced the idea of using hidden semi-Markov models (HSMMs) to describe durations in the context of speech synthesis. These models track the amount of time (acoustic frames) spent in the current HMM state, and allow the frame counter to influence the probability of transitioning, making it possible to model a much wider class of duration distributions. Typically, HSMM durations would be assumed to follow a parametric distribution of some sort; the widely-used *HMM-based speech synthesis system* (HTS) (Zen et al., 2007) defaults to Gaussian duration distributions, for instance, although the skewness and non-negativity of speech durations mean that other choices of distribution (log-normal, gamma) might be more suitable (Campbell, 1989; Huber, 1990). As with HMMs, decision trees were used to predict distribution parameters per state, and the mean of the predicted distribution was used for generation.

Recently, SPSS has seen stronger machine-learning techniques such as deep and/or recurrent neural networks replace decision trees for predicting the properties of duration distributions from the text, e.g., Zen and Sak (2015); Zen et al. (2016). Most often, these systems train DNNs or RNNs to minimise the mean squared prediction error, which is theoretically equivalent to using a Gaussian duration model (with a globally tied standard deviation) in conjunction with mean-based duration generation. The evolution of approaches to duration generation for TTS is summarised in Table 2.1.

Looking at Table 2.1, we see that all canonical methods predict only a few numbers per phone; at most 5 means and 5 standard deviations for Gaussian state-level predictions, of which only the means affect output generation. This renders the methods incapable of predicting general duration distributions with more degrees of freedom. In contrast to these prior methods, we propose a fundamentally non-parametric approach in which a model is trained to predict the phone transition probability at each timestep, i.e., acoustic frame, as outlined on the final line of Table 2.1. This set-up can in theory describe any probability mass function on the positive integers. (In practice, performance may be limited by biases in the learning algorithm used.) It thus becomes possible to meaningfully represent any and all important properties of the duration distribution, for instance its skewness, which most models hitherto have ignored.

Median-Based Generation

All methods in Table 2.1 that predict a distribution of durations automatically support a wide variety of methods for generating output durations from this distribution. While in principle, natural speech is a random sample drawn from the true duration distribution, the output naturalness of sampling methods in speech synthesis has been found to perform poorly unless highly accurate models are used (Henter et al., 2014a). As a consequence, most SPSS systems use deterministic generation methods, which in practice is synonymous with generating the mean duration.

In Chapter 5, we consider a scheme where durations generated at synthesis time are based on the median – rather than the conventional mean – of the predicted duration distribution. Since we are no longer assuming that the duration distribution is symmetric, the median will typically differ from the distribution mean.

Frame-Level Duration Prediction

Typically, duration is modelled at either state level or phone level when using neural networks. However, Watts et al. (2015a) described a joint model of duration and speech parameters, where a deep neural network was trained to simultaneously output acoustic parameters and a 5-dimensional (phone) state-duration vector for each frame. Synthesis with this type of network results in a chicken-and-egg problem, where state durations are both an input and an output of the sequence prediction network. This was solved by iteratively refining state-duration predictions over multiple passes, which is slow. In our proposed method, a single pass suffices for duration generation. Furthermore, despite the frame-level granularity of the approach in (Watts et al., 2015a), it is not straightforward to identify a probabilistic interpretation of their scheme.

A general advantage of duration predictions made at the frame level is that they can be unified with the (traditionally distinct) prediction of acoustic features, such as pitch and vocal tract filter MCCs, that takes place for each frame. Such joint modelling of durations with acoustic properties of speech was a major factor in motivating the set-up in Watts et al. (2015a). It is suspected that generating durations and fundamental frequency contours that are jointly appropriate may be of importance for synthetic speech prosody; cf. (Fernandez et al., 2014; Ronanki et al., 2016a).

2.3 Evaluation of speech synthesis

Speech synthesis evaluation is in itself another research topic and is quite essential when benchmarking the performance of a system. A brief overview of the objective metrics that are used in this thesis work and subjective methodologies that have been used for evaluating listening tests is given below.

2.3.1 Objective evaluation

In conventional SPSS systems, a vocoder (e.g., STRAIGHT or WORLD) is generally used to reconstruct the final waveform from the predicted acoustic parameters. Therefore, objective evaluations are used to compare the predicted parameters with reference parameters extracted from natural speech. Most of the metrics in general use are distance measures. The underlying assumption is that a model with relatively smaller distance performs the best, leading to a better speech synthesis system.

However, it is not always the case that the best-performing system in terms of objective measure leads to the best perceived quality. Considering audiobook level prosody, the objective measures remain suboptimal and may not always reflect the perceived quality of synthesized system. Even though objective evaluations may not always correlate with the perceived naturalness and prosody, they can still be used to optimize the modelling parameters.

Unless mentioned otherwise, STRAIGHT was used as the vocoder in this thesis and below we present the objective measures to measure the performance of predicted parameters.

Acoustic measures:

Mel-Cepstral Distortion (MCD) (Kubichek, 1993a) measures the distortion between predicted and extracted (from natural speech) Mel-Cepstral Coefficients and is defined as:

$$MCD = \frac{\alpha}{T}\sum_{t=1}^{T}\sqrt{\sum_{d=1}^{D-1}(x_d(t)-\hat{x}_d(t))^2} \tag{2.15}$$

$$\alpha = \frac{10\sqrt{2}}{ln10} \tag{2.16}$$

where T is the total number of frames in the test data and D is the dimensionality of the mel-cepstral coefficients extracted at each frame. In this thesis, we use 60 coefficients per speech frame and the zeroth coefficient which is similar to energy is excluded from MCD computation, as shown in Equation 2.15. The constant α is included for historical reasons (Kominek et al., 2008).

Band Aperiodicity distortion (BAP) (Fonseca De Sam Bento Ribeiro, 2018) measures the distortion of Band Aperiodicity coefficients and is defined as:

$$BAP = \frac{1}{10T}\sum_{t=1}^{T}\sqrt{\sum_{d=1}^{D}(x_d(t)-\hat{x}_d(t))^2} \tag{2.17}$$

BAP follows the same intuition as MCD and the distortion for BAP is computed using 25 band aperiodicities.

Root Mean Square Error (RMSE) is a frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed and is defined as

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(x_t-\hat{x}_t)^2}{T}} \tag{2.18}$$

where x_t and $\hat{x}_t$ denote reference and predicted values at time t, respectively. RMSE is quite often used in this thesis to measure the deviation between estimated and predicated values for duration and F0 parameters. When calculating RMSE for F0, the error value was calculated on a linear scale instead of the log-scale which was used to model the F0 values.

Voicing error (V/UV) is often measured along with F0 RMSE. This is because F0 is interpolated through unvoiced regions for modeling purposes and the voicing decision is stored and later predicted for F0 reconstruction. Therefore, voiced/unvoiced (V/UV) error is calculated as percentage of frames that were assigned the incorrect voicing label.

Linear Correlation Coefficient (CORR) measures the strength and the direction of a linear relationship between two variables.

$$CORR = \frac{T\sum(x_t\hat{x}_t) - (\sum x_t)(\sum \hat{x}_t)}{\sqrt{T(\sum x_t^2) - (\sum x_t)^2}\sqrt{T(\sum \hat{x}_t{}^2) - (\sum \hat{x}_t)^2}} \tag{2.19}$$

where x_t and $\hat{x}_t$ denote reference and predicted values respectively. If x_t and $\hat{x}_t$ have a strong positive linear correlation, CORR is close to +1. A CORR value of exactly +1 indicates a perfect positive fit. If x_t and $\hat{x}_t$ have a strong negative linear correlation, CORR is close to -1. A CORR value of exactly -1 indicates a perfect negative fit. CORR is often used in this thesis to also measure the performance of F0 and duration.

Except for CORR, in any of these metrics, a lower value indicates better performance. While the above objective metrics do not map directly to perceptual quality, they are often useful for system tuning.

Classification measures:

Chapter 4 of this thesis involves classifying syllable templates. Therefore, we present some of the objective measures that are typically considered for evaluating the performance of a classification system.

Accuracy is the most conventional performance measure and it is a ratio of correctly predicted data points to the total number of data points. The model with highest accuracy generally indicates a better performance. However, accuracy is not a great measure when measuring the performance of a model on a dataset with imbalanced classes.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.20}$$

where TP denote number of true positives, TN denote number of true negatives, FP denote number of false positives and FN denote number of false negatives.

Recall is defined as the number of true positives divided by the number of true positives plus the number of false negatives. True positives are the data points that are actually positive and classified as positive where as false negatives are the data points that are actually positive but instead identified as negative by the model. Recall is generally expressed as the ability of a model to find all the relevant examples within a dataset.

$$Recall = \frac{TP}{TP + FN} \tag{2.21}$$

Precision is the ratio of the number of true positives to the total number of true positives identified by a model. False positives are cases where model incorrectly identifies as positive when they are actually negative.

$$Precision = \frac{TP}{TP + FP} \tag{2.22}$$

F1 score is the harmonic mean of precision and recall taking both measures into account in the following equation:

$$F1_{score} = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{2.23}$$

2.3.2 Subjective evaluation

Objective measures are not only used to optimize hyper parameters during training but also often used as an indication of the quality of synthetic speech. However, subjective listening tests still remain the standard method for evaluation of synthetic speech. Since, the primary focus of this thesis is to improve the overall naturalness of the SPSS by enhancing the prosody model, subjective listening tests will be of paramount importance in evaluating our systems. They tend to be costly and at times require huge investment in terms of both time and resources, as they require human listeners. Even though the work in this thesis is carried out on children's storybooks, we haven't used any children to evaluate our systems as that requires considerable amount of effort – which includes obtaining consent from their parents. We instead employed normal-hearing native English speakers (University of Edinburgh students, remunerated for their time) for listening tests, conducted in sound-insulated booths over high-end Beyerdynamic headphones. Below here, we present some of the subjective evaluation methods that are used in this thesis.

Mean Opinion Score (MOS):

One of the most common and conventional ways to evaluate speech synthesis is via a mean opinion score (MOS) test. MOS is commonly used in the Blizzard Challenge listening tests (Clark et al., 2007a; King and Karaiskos, 2013, 2016) and has become standard practice when evaluating speech synthesis systems. Listeners are asked to listen to one speech sample at a time and rate it on a scale of 1 to 5 in terms of quality or naturalness, where 1 indicates *bad* and 5 indicates *excellent*. When evaluating MOS, listening tests are usually balanced to remove potential effects of repeated listening of the same utterance under different conditions by the same participant. There are also variations of the MOS test such as differential MOS (DMOS) and comparison MOS (CMOS) where listeners are asked to evaluate each speech sample in comparison with a reference sample but these are not used in this thesis. We present results based on MOS in Chapter 9 of this thesis, which were conducted as part of the Blizzard Challenge (Merritt et al., 2016b; Ronanki et al., 2017a).

MUSHRA:

MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) (ITU Recommendation ITU-R BS.1534-3, 2015) is an established testing paradigm in the field of speech coding and is just beginning to be used within speech synthesis (Henter et al., 2014a, 2016). MUSHRA allows listeners to make their judgements by listening to all conditions of a single sentence side-by-side on a single screen. MUSHRA was used in Chapters 4 and 5 of this thesis, where listeners ranked several parallel, unlabelled stimuli from the same text but different systems in terms of preference. The scale ranged from 0 (least preferred) to 100 (most preferred). The test also includes a hidden reference (vocoded speech in this thesis), which acts as an upper-bound from which listeners can make their judgements. Listeners are informed about this upper anchor condition and are asked to identify and rate it as 100. This hidden reference not only helps listeners to make better judgements but also helps us to filter any outliers (bad judgements). At times, a bottom line system can also be used as lower anchor so as to stabilise listeners ratings.

Preference test:

The most simple methodology is a preference test also known as an AB test. In this paradigm, listeners are provided with two speech samples of a single sentence and are asked to choose one of them based on naturalness. The testing conditions are anonymised and the stimuli are ordered randomly in each screen. Depending on the experimental design, listeners may also be given a third option to indicate *no preference* if they can't choose between A and B. If this option is not given, then the test is called as forced choice preference test as the listeners are forced to choose between A and B. There is another variation of this preference test commonly known as ABX test. Similar to CMOS and MUSHRA, this includes a reference, where listeners give their judgements based on this reference.

Chapter 3

Data and feature extraction

This chapter presents the data we used in this thesis and the feature extraction procedure for neural network training, which includes preparation of input linguistic features and output acoustic features. The Hidden Markov Model Toolkit (HTK) was used for building HMMs and the scripts for forced-alignment of speech with text using HMMs was provided by Zhizheng Wu and is now part of the Merlin toolkit.

3.1 Data

Since the goal of this thesis is to investigate the benefits of intonation and duration modelling, we considered the data from audiobooks that are prosodically interesting. For this purpose, we chose the speech database created for the Blizzard Challenge 2016 (King and Karaiskos, 2016) and 2017 (King et al., 2017), as they include narrations of children's storybooks.

Description: The database - provided to the Blizzard Challenge 2016 by Usborne Publishing Ltd. consists of the speech and text of 50 childrens audiobooks spoken by a British female speaker. The total duration of the audio is approximately 4.5 hours. The speech database released for the 2017 Blizzard Challenge (King et al., 2017) included six additional audiobooks, extending the total duration to 6 hours.

The chapters in part-I used the speech database released for the 2016 Blizzard Challenge whereas the chapters in part-II made use of six additional audiobooks released as part of Blizzard Challenge 2017.

Audiobook segmentation: The audiobooks consisted of an excessive amounts of silence between paragraphs and a frequent chime sound, denot-

Table 3.1: Details of the speech database after audiobook segmentation

Segmented data	**Blizzard 2016 (Part-I)**	**Blizzard 2017 (part-II)**
No. of utterances in Train set	5500	6866
No. of utterances in Validation set	134	134
No. of utterances in Test set	253	253
Total utterances	5887	7253
Avg. duration of each utterance	2.65 sec.	2.84 sec.

ing to turn the page. For the work in this thesis, we made use of a manual segmentation of the audiobooks carried out by two other Challenge participants[1][2] and kindly made available to all other participants. The total duration of the database is reduced to 5 hours 43 minutes after segmentation (long silences were trimmed and chime sounds were removed).

The details of the training, validation and test split after segmentation are presented in Table 3.1. The training data for the chapters in Part-II is increased by considering the additional 6 audiobooks released for Blizzard Challenge 2017. For the experiments presented in both parts of this thesis, 3% of this data was set aside for testing. The test set consists of three entire stories: *Goldilocks and the Three Bears*, *The Boy Who Cried Wolf* and *The Enormous Turnip*, having a total combined duration of approximately 10 minutes.

3.2 Data exploration

This section aims at understanding the prosodic variability in our chosen audiobook corpus (children's storybooks from Blizzard 2016-17). For comparative analysis, we also selected recordings by three other female speakers from publicly available datasets: CMU Arctic corpus (SLT) (Kominek and Black, 2004), LJSpeech non-fiction audiobook corpus (read by Linda Johnson) (Ito, 2017) and Boston radio news corpus (f3a) (Ostendorf et al., 1995). From each of the four datasets, we randomly selected 1000 audio files and extracted F0 for every 5ms using WORLD (Morise et al., 2016) vocoder. For Boston radio corpus (f3a), we only have 391 utterances and used all of them.

[1]Innoetics: https://www.innoetics.com
[2]IIIT-H: http://speech.iiit.ac.in

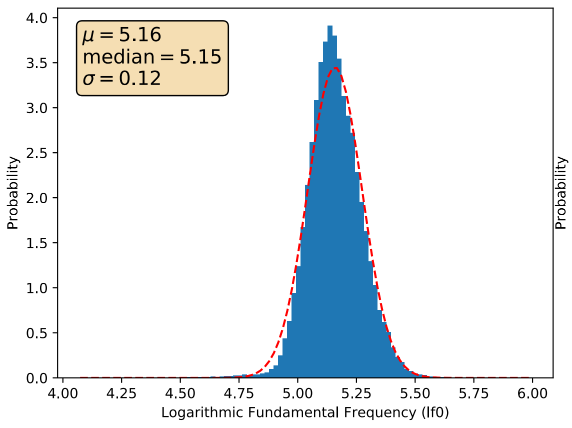
Table 3.2: Analysis of global statistics of lf0 and mean prosodic variations based on lf0 per utterance

Corpus	**Global statistics**			**Average per utterance**	
	Min	**Max**	**Mean**	**Variance**	**Range**
CMU Arctic (SLT)	4.073	5.991	5.158	0.105	0.729
Boston Radio (f3a)	4.095	6.189	5.163	0.321	1.902
LJSpeech (Linda)	4.134	6.346	5.276	0.235	1.237
Blizzard 2016-17 (Lesley)	4.081	6.275	5.148	0.267	1.268

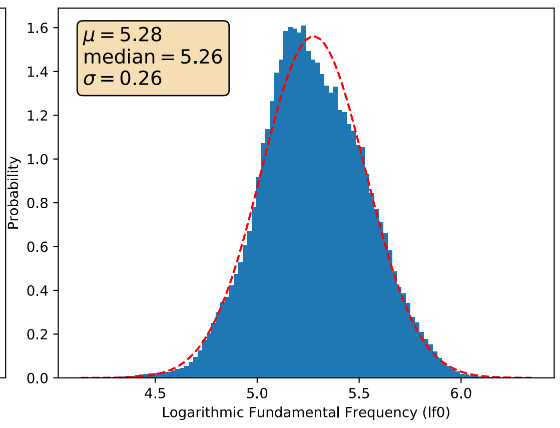
First, we compute the global statistics of fundamental frequency from each corpus by considering only voiced F0 frames. The descriptive statistics such as minimum (Min), maximum (Max) and mean (Mean) are computed in the natural logarithmic scale for each corpus and the values are reported in Table 3.2. We observe that these global descriptive statistics are more or less similar for all four datasets. CMU Arctic corpus has the lowest maximum lf0 value (5.991) as the recording style is more neutral and read in a consistent manner. Non-fiction audiobooks from the LJSpeech corpus have the highest maximum lf0 value (6.346) and the children's storybooks corpus (6.275) is also in the similar range. Audiobooks typically report higher maximum lf0 value as the speakers often mimic characters voices from the book.

We also study the average variance and range in the natural logarithm of fundamental frequency for each utterance in all four datasets. The values are reported in Table 3.2. We notice that among the four corpora, the neutral-style utterances from CMU Arctic have the lowest mean variance and range per utterance, making it more tractable and easier to model than the other three corpora. Non-fiction audiobook corpus (LJSpeech) has a slightly higher mean variance and range given greater expressiveness, and the Boston radio news corpus has the highest mean variance and range. Even, the children's storybooks from Blizzard corpus also has higher variance and range compared to CMU Arctic and LJSpeech corpora. This indicates that both news corpus and children's storybooks may pose more challenging problems when modelling.

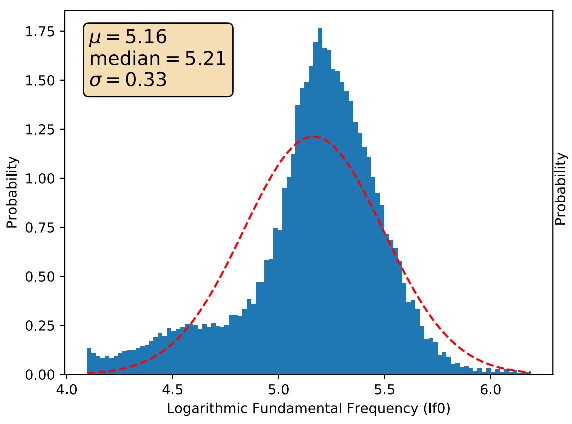
For further analysis, we also present the histogram distribution of F0 from all four datasets in Figure 3.1. As seen from the figure, the CMU Arctic corpus and non-fiction audiobook corpus have relatively lower variance and the



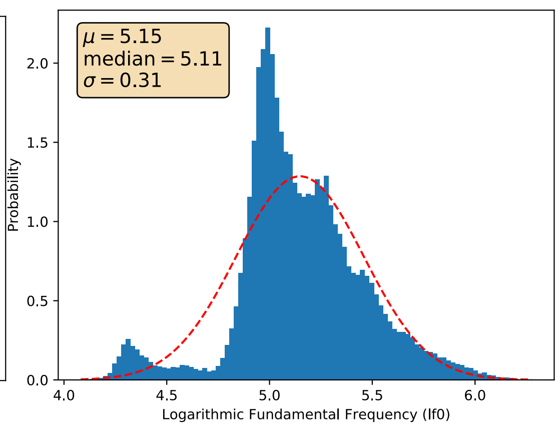
(a) CMU SLT Read Speech Corpus



(b) LJSpeech non-fiction Audiobooks



(c) Boston Radio News Corpus (f3a)



(d) Blizzard 2016-17 Children's storybooks

Figure 3.1: Illustration of F0 histogram from various datasets

distribution of lf0 nearly fits a Gaussian. The Boston radio news corpus has highest variance but the distribution is more uniform with only one peak at the center. The center of Gaussian is also towards the peak distribution of lf0.

Similar to the news corpus, the children's storybooks corpus we explore in this thesis has also higher variance and range. But, unlike the news corpus, the distribution of lf0 is quite random with several local peaks. Even the center of Gaussian doesn't align with the peak distribution of lf0, indicating a skewed distribution. This type of corpus poses more serious problems for those machine learning algorithms which assume a pre-determined distribution (e.g., Gaussian). Hence, we believe that the prosodic variance of this database makes it more challenging and interesting for modelling prosody and therefore it was chosen for the work presented in this thesis.

3.3 Feature extraction

This section details the preparation of linguistic features and acoustic features for neural network training. Section 3.3.1 describes the procedure for extraction of linguistic features using Festival and HTK. Section 3.3.2 describes the procedure for extraction of acoustic features using STRAIGHT vocoder.

3.3.1 Linguistic features

Label preparation: For preparation of phoneme labels, the lexicon used was Combilex-RPX (Richmond et al., 2009) whose pronunciations are made of a *combilex phoneset*. Combilex is an advanced multi-accent lexicon for English, developed by CSTR specifically for speech technology purposes - primarily text-to-speech synthesis (TTS). The final phone-label sequence was derived using the Festival toolkit (Black et al., 2001) as specified in Section 2.1.1.1. Festvox's `ehmm` (Prahallad et al., 2006) was used to insert pauses into the phone-label sequences based on the acoustics. The Hidden Markov Model Toolkit (HTK) (Young et al., 2002) was used to extract MFCCs and train monophone hidden Markov models. The procedure for building HMMs followed an initial estimation of alignment and several rounds of re-estimation was performed making use of *Baum-welch* re-estimation algorithm (Eddy et al., 1995; Yoshimura et al., 1999). After model training finished, the data was force-aligned at state level using pre-trained monophone HMMs.

The resulting label sequences were coupled with text-derived linguistic features encoding a subset of the questions used by the decision-tree clustering in HTS (Zen et al., 2007), for instance linguistic context such as quinphone identity and part-of-speech; positional information within the syllable, word, and phrase; and so on. This produced a vector of 481 input features per phoneme, to which 9 numerical features at frame-level were appended as in (Wu et al., 2015b) and all features normalised to the range $[0.01, 0.99]$.

The complete list of 481 input features which includes binary and continuous-valued numerical linguistic features are presented in Table 3.3 and Table 3.4. See Appendix A for a full description of binary and numerical linguistic features.

Table 3.3: Binary linguistic features

Unit size	Feature description	Feature dimension
Phoneme	Identity of current phoneme	49
	Identity of previous phoneme	50
	Identity of next phoneme	50
	Identity of previous-to-previous phoneme	50
	Identity of next-to-next phoneme	50
	Position and manner of articulation of current phoneme	99
Syllable	Identity of the nucleus of current syllable	20
	Articulatory features of the nucleus of current syllable	37
Word	Guessed parts-of-speech of current word	11
	Guessed parts-of-speech of previous word	11
	Guessed parts-of-speech of next word	11

3.3.2 Acoustic features

In this thesis, STRAIGHT (Kawahara, 2006) was used to extract three feature streams: spectral envelope, F0 and aperiodicities. Pitch synchronous analysis is a common procedure to extract features from a periodic signal. However, due to intrinsic fluctuations in speech periodicity, STRAIGHT features are extracted using a pitch-adaptive window assuming the speech signal over this window is statistically stationary. A Gaussian weighting in the frequency domain was introduced to reduce the levels of the side lobes of the time window and a complementary time window was introduced to further reduce the temporal periodicity. A spectrogram with reduced temporal variation is computed using this complementary time window. This spectrogram undergoes smoothing using the basis function of the 2nd order B-spline function to make it less sensitive to F0 estimation errors. A smoothing kernel (like a triangular lobe) to recover smeared values at harmonic frequencies is used to compute the final spectral envelope. The spectral features were Mel-warped by transforming the full resolution spectra into Mel generalized cepstral features (MGCs) using SPTK[3] and transforming back to the frequency domain.

[3]http://sp-tk.sourceforge.net

Table 3.4: Continous-valued numerical linguistic features

Unit size	**Feature description**	**Feature dimension**
Phoneme	Position of phoneme in current syllable	2
Syllable	Stress, Accent and other positional features	21
Word	Number of content words and other positional features	9
Phrase	Number of syllables, words and position of phrase in utterance	8
Utterance	Number of syllables, words, phrases in utterance	3

Table 3.5: Acoustic features and their dimension

Feature	Dimension	Dimension (incl. dynamic features)
MGC	60	180
BAP	25	75
LF0	1	3
VUV	1	-

SPTK was used to compute 60-dimensional mel-cepstrum coefficients as well as 25-dimensional band aperiodicities and logarithmic fundamental frequency ($\log F0$) every 5ms. However, we found that F0 extracted from STRAIGHT is not robust on this audiobook data and therefore replaced it with F0 extracted using the GlottalHMM toolkit (Raitio et al., 2010).

Continuous F0 modelling was used by many researchers (Fons-ant and Naessentialr, 1969; Taylor, 2000; Escudero et al., 2002) but primarily applied in concatenative systems. It was then performed in HMM-based speech synthesis (Yu and Young, 2011) and later continued for DNN-based speech synthesis (Zen et al., 2013; Wu et al., 2015b). It has proven to be better at modelling than discontinuous F0. Therefore, we interpolate F0 in linear domain through unvoiced regions using linear interpolation and store the voicing decision in the form of a binary variable VUV, termed as *voiced-unvoiced*. A log-scale is applied to continuous F0, as in most conventional systems (Wu et al., 2016).

To generate smooth trajectories, delta and delta-delta features (also known as *dynamic features*) are computed from all the vocoder features (MGC, BAP and LF0) and are appended to the output features. This brings the total output dimension to 259. The details of the output features used are presented in Table 3.5. Dynamic features and MLPG algorithm at synthesis time have been naturally inherited from HMM-based speech synthesis and were also used in deep feed-forward neural networks to provide smooth transition between acoustic frames Zen et al. (2013); Wu et al. (2015b). Theoretically, RNNs can make use of sequential information and no longer need dynamic features and MLPG to smooth the parameters trajectory. Klimkov et al. (2018) investigated the use of parameter generation and variance restoration for RNN-based speech synthesis. The results indicate that the use of L1 loss with proper regularization outperforms conventional L2 loss and doesn't require to apply MLPG. However, they are still shown to be effective with L2 loss even when using LSTMs for acoustic modelling (Zen and Sak, 2015; Klimkov et al., 2018). The experiments presented in this thesis are optimized using L2 loss and hence MLPG was used even when trained with LSTM-based neural networks.

Finally, per-component mean and variance normalisation is performed to bring different features in the same scale and reduce assymmetry in the cost computation. The normalisation also speeds up the training by avoiding many extra iterations that are usually required when some features take on much larger values than the rest.

Part I

Investigating the elements of prosody

Chapter 4

A template-based approach for speech synthesis intonation generation

This chapter is an expanded version of the work in Ronanki et al. (2016a) and therefore the text is closely related to that.

This work was completed in collaboration with others. Discussion of ideas was done between myself, Zhizheng Wu, Gustav Eje Henter and Simon King. The code for the LSTM system used was provided by Zhizheng Wu. I expanded this to incorporate a CTC output layer and later made it open-source in the Merlin speech synthesis toolkit (Wu et al., 2016). The code for hierarchical clustering of syllable-level DCT vectors was implemented by myself making use of numpy and scipy libraries in python. The code for running and analysing the MUSHRA test was provided by Gustav Eje Henter. Fine-tuning of the systems was then done by myself with the help of Zhizheng Wu and Gustav Eje Henter.

4.1 Motivation

The absence of convincing intonation makes current parametric speech synthesis systems sound dull and lifeless, even when trained on expressive speech data. Most of current DNN systems perform frame-level modelling of $F0$, and cannot handle either semantic prosody or its hierarchical representation which requires modelling at a higher level. Therefore, in this chapter, we focus only on intonation (pitch contour) generation and the final waveform generation is done through the pipeline of statistical parametric speech synthesis (SPSS). In chapter 5, we focus on modelling and generation of speech synthesis durations, another dimension of prosody.

Although intonation is a supra segmental property, conventional approaches, such as HTS (Zen et al., 2007), predict F0 frame-by-frame, based on limited linguistic contextual information (quinphone, part-of-speech and positional information). Predictions within an HMM state are assumed conditionally independent of *surrounding F0 predictions*, given the linguistic information. Surrounding F0 values only affect the local smoothing of the final contour (via MLPG (Tokuda et al., 2000)). Such a myopic approach may not be the best way to produce utterance-level supra segmental structure.

Typically, these systems use regression techniques to predict the fundamental frequency (F0) frame-by-frame. This approach leads to overly-smooth pitch contours and fails to construct an appropriate prosodic structure across the full utterance. In order to capture and reproduce larger-scale pitch patterns, this chapter proposes a template-based approach for automatic F0 generation, where per-syllable pitch-contour templates (from a small, automatically learned set) are predicted by a recurrent neural network (RNN). We propose to generate more variable and naturalistic intonation through the use of syllable-level pitch-contour templates learned from data. The hope is that treating intonation prediction as a *classification* rather than a *regression problem* will mitigate the over-smoothing seen in conventional techniques. By using a recurrent neural network to predict the pitch-template sequence, long-range information about linguistic context and surrounding predictor state can influence the output, enabling high-level prosodic structure to be expressed. This draws on recent successes of artificial neural networks in automatic speech recognition (ASR) (Hinton et al., 2012), and the use of deep (Zen et al., 2013; Ling et al., 2013) and recurrent (Zen and Sak, 2015; Wu and King, 2016) neural network techniques in SPSS, as previously discussed in Chapter 2 (cf. Section 2.1.2).

We also consider improvements to this basic idea, including learned smoothing of template joins and hierarchical prediction of the templates, and evaluate against a state-of-the-art SPSS baseline.

For modelling intonation in speech synthesis, clustering has been used in several papers in some way or the other. Fujisaki model is one of the popular one where (Mixdorff, 2015) introduced clustering of F0 contours.

The remainder of this chapter is laid out as follows: Our approach to template-based intonation generation is detailed in section 4.2. Sections 4.3

and 4.4 describe the experimental set-up and results of an evaluation on expressive speech data from children's audiobooks and section 4.5 concludes.

4.2 Proposed template-based approach

Our proposed method has three parts:

1. An inventory of syllable-level templates derived from training data
2. A neural network classifier to predict template class from input text features
3. F0 contour reconstruction from this sequence of templates

4.2.1 Creating the inventory of syllable F0 templates

The F0 contour of each utterance is interpolated through unvoiced regions, including pauses then segmented into syllables using force-aligned phone labels. The unvoiced regions are interpolated linearly using the *interpolate* function from *scipy* in linear domain. An example F0 contour with syllable-level segmentation is shown in Figure 4.1.

Each syllable F0 contour is converted to logarithmic scale based on the equivalent rectangular bandwidth (ERB[1]) scale (Smith III and Abel, 1999) and is given in Equation 4.1.

$$F0_{ERB} = \log_{10}(0.00437 * F0 + 1) \tag{4.1}$$

We then (approximately) decompose each syllable contour into a sum of N weighted zero-phase cosine functions. If x is a contour of length N (measured in frames), then

$$c_k = 2w[k] \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi(2n+1)k}{2N}\right) \tag{4.2}$$

for $0 \leq k < N$, where

$$w[k] = \begin{cases} \frac{1}{\sqrt{4N}} & \text{if } k = 0 \\ \frac{1}{\sqrt{2N}} & \text{if } 1 \leq k \leq N. \end{cases} \tag{4.3}$$

[1]The ERB scale is often used in intonation research where a perceptual scaling is needed to weight the perceptual importance of differences in fundamental frequency. We later found the simple logarithmic scale to perform similarly to the ERB scale in our experiments.

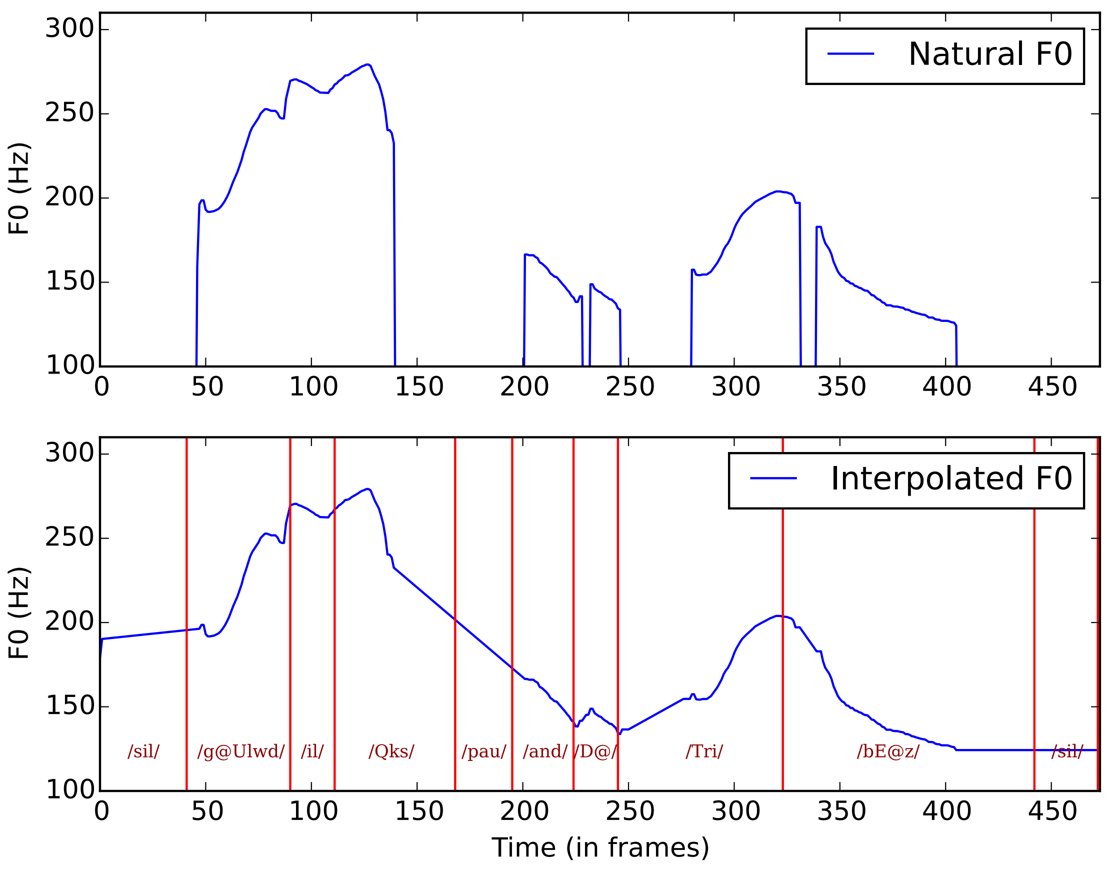


Figure 4.1: Natural F0 and interpolated F0 contours for the sentence "*Goldilocks and the three bears*" at 5 ms frame rate. Red lines indicate syllable-level segmentation.

This yields outputs: c_0, representing the mean F0 over the syllable, and $\boldsymbol{c} = [c_1, \ldots, c_{N-1}]^\top$, representing the shape of the contour. Since, the length of each contour varies for each syllable, we consider only the first K coefficients for clustering. This representation is clustered hierarchically (Johnson, 1967) to group similar intonation pattern vectors $\boldsymbol{c}$ together, as follows:

1. Assign each syllable F0 contour to a separate cluster. With M such contours there will be M clusters, each containing a single contour.

2. Find the pair of clusters with the smallest Euclidean distance between their mean contours and merge them into a single cluster.

3. Repeat previous step until a stopping criterion is met.

Figure 4.2 provides a schematic diagram of the procedure for computing clustered templates. The procedure for clustering F0 contours includes three stages and involves normalization of two features: duration and mean pitch.

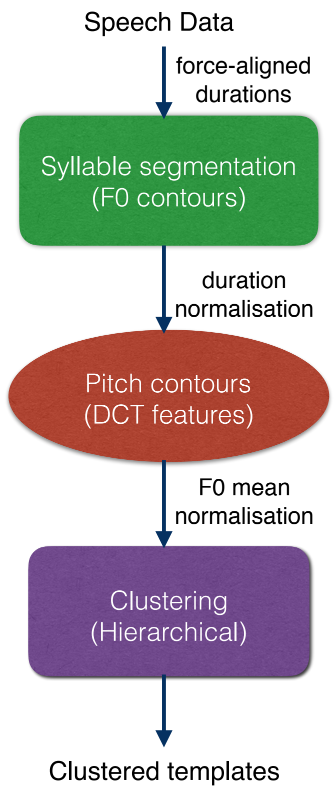


Figure 4.2: Overview of formation of clustered templates.

The first stage of the process performs syllable segmentation of interpolated F0 contours using force-aligned durations as shown in Figure 4.1. The second stage (represented in brown color, in Figure 4.2) performs a discrete cosine transformation of syllable level pitch contours, where syllables of varying durations are normalized to a fixed number of coefficients. The motivation behind using DCT for clustering is that it stores most of the energy in initial coefficients and the reconstruction loss can be minimized by using fewer and fixed number of coefficients for clustering. Also, the process for duration normalization is inherently done by DCT. As shown in Equation 4.2, the DCT transforms a syllable F0 contour of length N into N-discrete cosine coefficients. However, for clustering purpose, we chose to use fixed number and only first few coefficients. Following previous work on intonation modelling using the DCT (Teutenberg et al., 2008; Qian et al., 2011), we used $K = 9$ coefficients (including c_0) for our model, in order to minimise the loss in reconstruction. If the length of syllable contour is less than 9 coefficients, we interpolate the extra points between $K - 2$ and $K - 1$ on the transformed vector. The third stage involves mean pitch normalization, where syllable contours that are of same

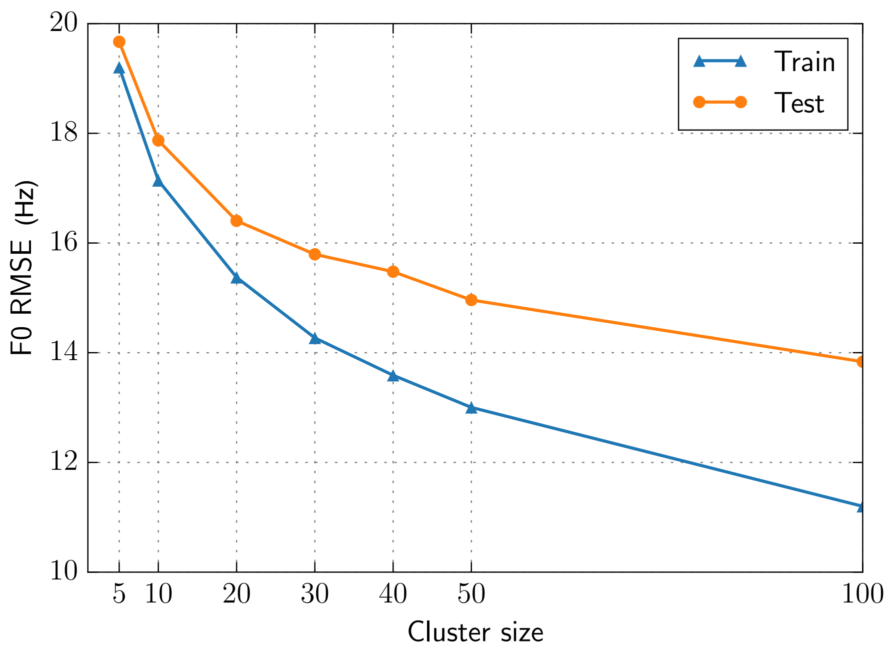


Figure 4.3: RMSE between natural F0 contour and the F0 contour reconstructed from clustered templates.

shape but having different F0 mean are clustered together. This is achieved by excluding c_0 (mean pitch over each syllable) from hierarchical clustering. This means, only coefficients from c_1 to c_{K-1} are considered for clustering. For reconstruction, both duration and mean pitch are predicted independently of pitch contour templates and the procedure for F0 reconstruction is further explained in Section 4.2.3.

Most hierarchical clustering algorithms use a bottom-up (or agglomerative) approach where each data point is treated as a singleton cluster at the outset, and then successively merge cluster pairs until all data have been merged into a single cluster. Hence, the stopping criteria for hierarchical clustering can be a little tricky: we can either stop when the RMSE between clusters falls above a threshold, or until we arrive at a specific number of clusters. However, clustering the data with too many centroids may end up with fewer examples to be learned from while training the neural-net classifier. At the same time, very few clustered templates may lead to high loss in reconstruction of the original F0 contour. Therefore, we followed an experimental procedure to determine the number of clusters.

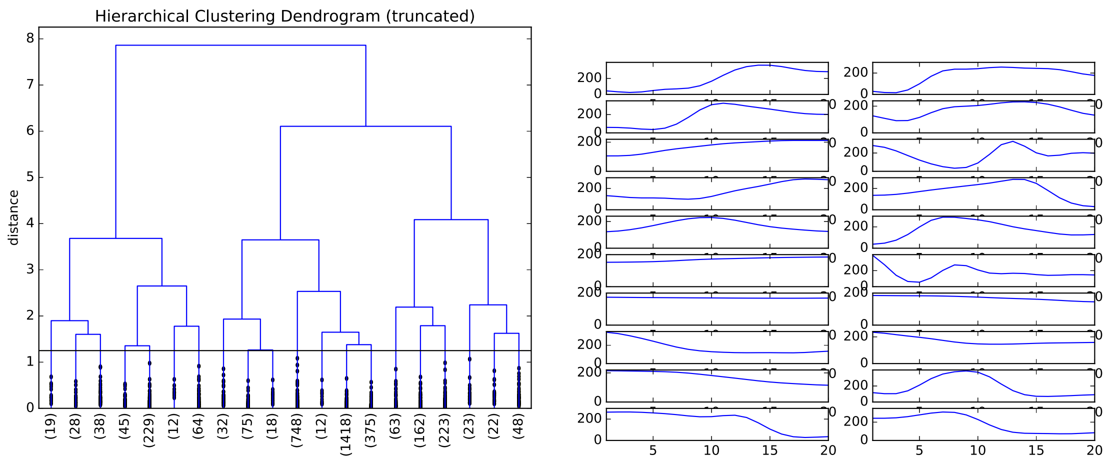


Figure 4.4: Dendogram derived from hierarchical clustering for a set of 20 templates.

Procedure to determine the number of clusters:

We used a decomposition-plus-reconstruction approach to determine the number of clusters, as shown below:

- Compute Eucledian distance to each of the clustered templates from syllable F0 contours over the patterns $\boldsymbol{C}$ (Equation 4.2).
- For reconstruction, replace each of the syllable F0 contour with its template (cluster with least distance) and apply inverse discrete cosine transform (IDCT) but using natural mean F0 and duration.
- Compute RMSE between natural F0 contour and the reconstructed contour from the templates.

The data from Blizzard Challenge 2016 is used for clustering and the complete details of the data are given in Section 3.1. Figure 4.3 shows the plot of RMSE for number of clusters from 5 to 100 computed over training and test data. It can be seen that the RMSE decreases in both train and test sets with increase in number of clusters. This is due to the fact that the reconstruction loss can be reduced by retaining contours that are of different shape whilst considering the location of pitch accents from nucleus as well.

For clustering, only 10% of the training data is considered and rest of the data is annotated using the clustered model. Figure 4.4 shows a set of twenty syllable F0 templates found by clustering of the data (along with dendrogram), plotted at the average F0 (180 Hz) and syllable duration (20 frames) of the

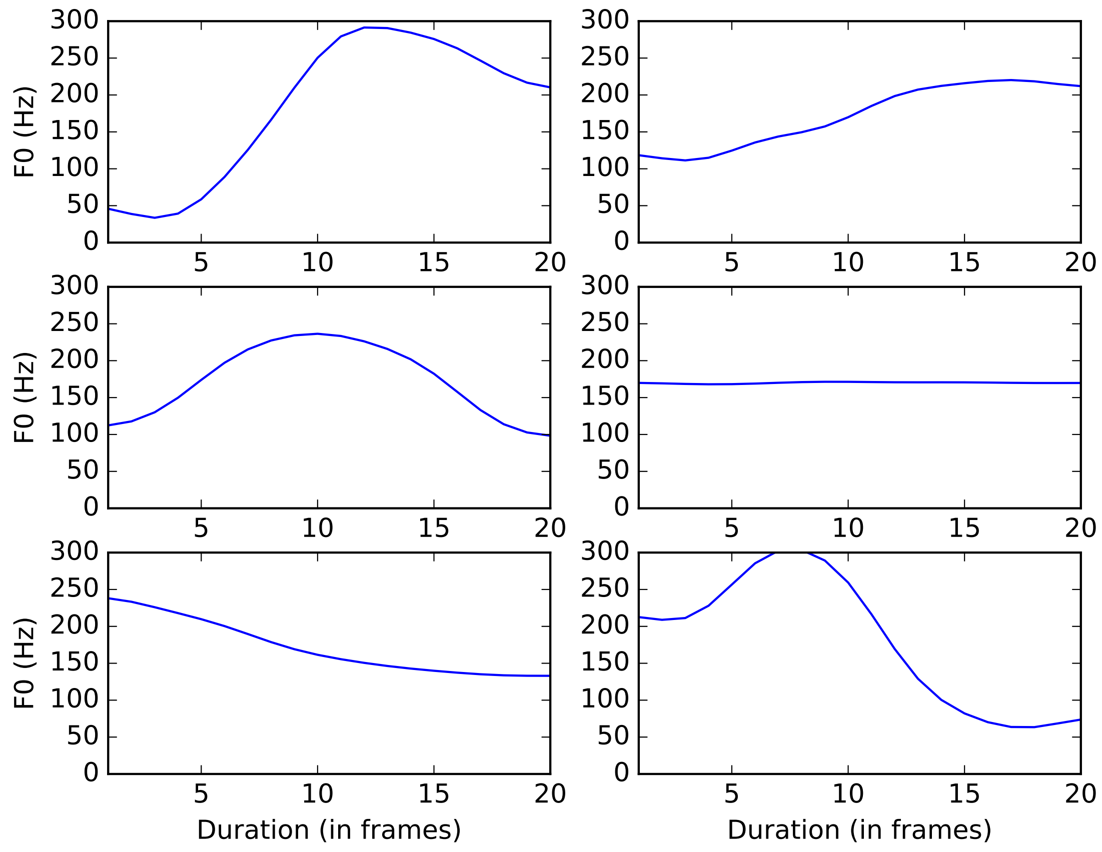


Figure 4.5: A set of six syllable F0 templates found by clustering the data described in Section 3.1, plotted at the mean F0 (180 Hz) and mean syllable duration (20 frames) of the speaker. Template 1 and 2 are in top row; Template 3 and 4 are in middle row; Template 5 and 6 are in last row.

speaker. The dendrogram shows that some of the clusters are formed from very few examples. The highest number of examples for a cluster is 1418 while the cluster with fewest examples has 12. Considering the amount of training data, such a clustering may later lead to a class-imbalance problem for the neural-net classifier. Also, some of the clusters are quite redundant as they closely resemble other clusters.

Therefore, the number of clusters are reduced from 20 to 6 constrained by these conditions:

- To retain a more balanced set of clusters with each cluster having sufficient examples.

- To have more diverse contour shapes that can be represented with fewer number of templates.

Figure 4.5 illustrates a 6-template model found from unsupervised clustering of the data. All 6 templates are formed by computing an average over a very high number of syllable F0 contours and slightly resembles ToBI shapes (Silverman et al., 1992). By choosing a minimal number of templates, the clustering led to a very diverse set of templates – templates 1 and 2 denote a rising intonation, which we typically observe in the utterances ending with question mark, whereas templates 5 and 6 denote falling intonation, as when reading surprised statements ending with exclamation mark. Also, the range is quite large for these templates plotted in Figure 4.5 as they were reconstructed with mean syllable duration. Each one of these templates can form different shapes based on the target duration we set for each template during reconstruction. The location at which pitch accents are placed for each syllable also differs as the target duration changes. These templates are parametrized for different duration and different mean F0 so that they can be utilized for generating different F0 contours. We provide an example utterance reconstruction using these 6 templates in the Section 4.2.3.

4.2.2 Predicting templates using neural network classifiers

We now describe two different systems for predicting F0 templates using neural networks:

1. A hierarchical deep neural network classifier (HC).
2. A simplified LSTM with a connectionist temporal classification (CTC) output layer (SLSTM-CTC).

4.2.2.1 Hierarchical classifier (HC)

Table 4.1 reveals that the frequency distribution of templates is far from uniform. Template 4 (Figure 4.5), by far the flattest, accounts for 65% of the training data, presumably corresponding to unstressed syllables. Classifications made by a DNN trained to minimise either mean square error or cross-entropy on this data were even more biased towards predicting the single most frequent class (cf. "DNN" in Table 4.1), since it had such a large prior probability. This would produce flat and boring intonation. We could have used a balanced set of class numbers, but that would decrease the amount of training

Template	1	2	3	4 (flat)	5	6
Train	852	5216	1853	26725	5784	1013
Dev.	26	139	50	653	147	28
Test	65	362	106	1553	377	70
DNN	0	127	4	2247	155	0
HC	15	298	27	1676	499	18
SLSTM-CTC	16	200	61	1958	287	11

Table 4.1: Template counts in the data (see Section 3.1) and corresponding test-set predictions by the methods of Section 4.3.3.

data for classification of these templates. Since the amount of training data is considerably small, we didn't investigate much in this direction.

We propose to use a hierarchy of DNN classifiers to diversify F0 template generation. A first classifier predicts whether or not to use the most frequent (and, here, flattest) template for the current syllable. If a less common template is called for, a second classifier chooses among the remaining templates. This gave a more diverse template distribution (cf. "HC" in Table 4.1), which should produce more interesting synthetic speech.

In our implementation, the first classifier is a simplified LSTM (S-LSTM) as in Wu and King (2016), while the second is a feed-forward DNN (Since second classifier considers only non-flat templates (all other templates except template 4), an LSTM can't be used which processes information through time).

4.2.2.2 SLSTM-CTC

In another approach, we added a connectionist temporal classification (CTC) output layer (Graves et al., 2006) mapping from phone-level linguistic features to syllable-level templates, after the S-LSTM layer. The CTC output is a softmax output layer with $N + 1$ nodes, representing a probability distribution over the syllable templates plus an additional "blank" unit. The blank unit is the most common softmax output, particularly for phones in the beginning and middle of syllables; only when the network believes it is at the end of a syllable is the predicted template output node activated. Similar to HMMs, the CTC loss function also makes use of the forward-backward algorithm to make utterance-spanning decisions. This should enable a more appropriate distri-

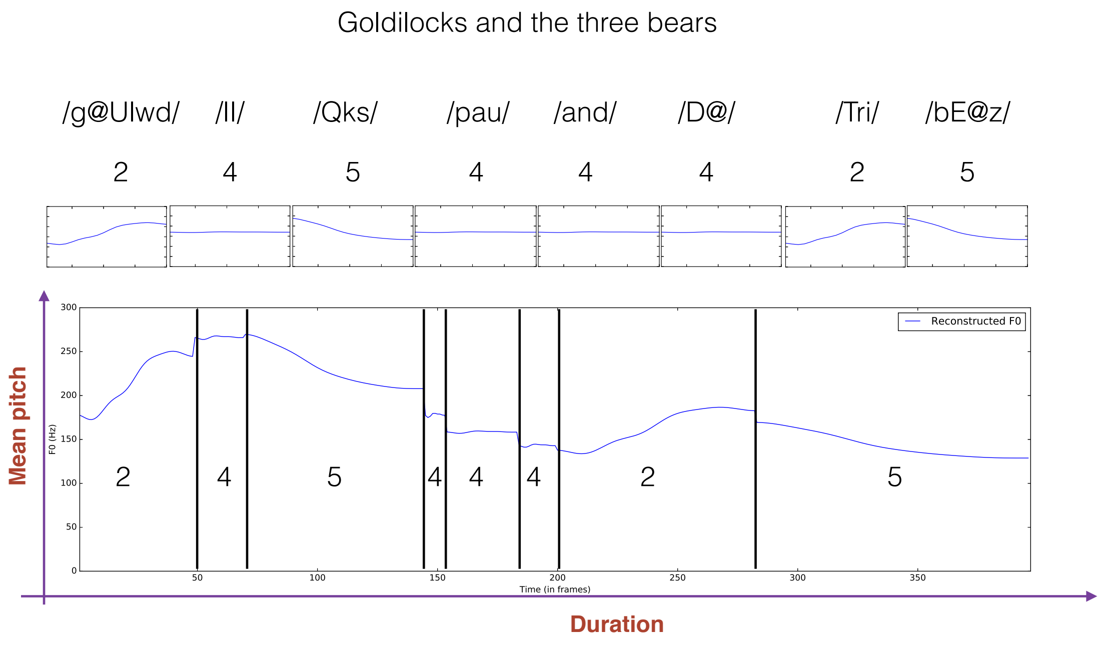


Figure 4.6: An example utterance showing how F0 contour is reconstructed from a six template set.

bution of predicted templates even for non-uniform data, without the need for two different predictors as in the proposed hierarchical DNN classifier.

4.2.3 F0 contour reconstruction

Figure 4.5 illustrates a 6-template model found from clustering of the data and Figure 4.6 shows an example utterance reconstructed from these six templates. Given the utterance, we segment into syllables (derived from Front-end) and use oracle templates (derived from clustering) for reconstruction of the F0 contour. The F0 contour for a given utterance is reconstructed using the inverse discrete cosine transform (IDCT):

$$x[n] = \frac{c_0}{\sqrt{N}} + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} c_k \cos\left(\frac{\pi(n+0.5)k}{N}\right). \tag{4.4}$$

where $x[n]$ is the reconstructed F0 contour. As seen in Figure 4.6, templates 2, 4 and 5 are used multiple times to produce a different contour each time. Since each template can produce a complete set of different F0 contours given a target duration and mean pitch of the syllable, the initial six templates are quite sufficient to reconstruct the final F0 contour. We zero-pad each template (DCT vector) to the target duration before applying inverse transformation.

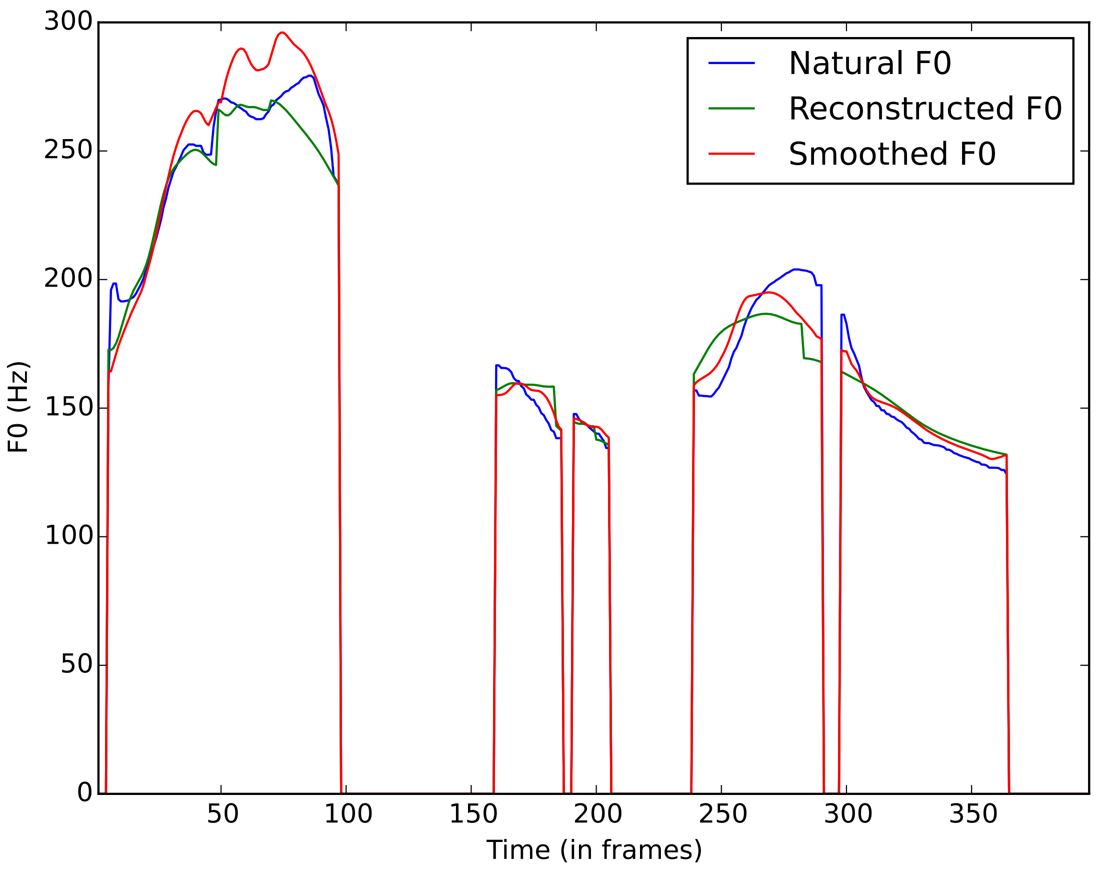


Figure 4.7: Natural F0 and reconstructed F0 contours for the sentence “Goldilocks and the three bears” at 5 ms frame rate.

For reconstruction of each template using IDCT, we use duration (N) and mean pitch (c_0) from oracle syllable F0 contour. Voiced-unvoiced decision is stored at the time of interpolating and is applied to the reconstructed F0 contour.

At synthesis time, the sequence of templates are predicted using either a Hierarchical classifier or CTC classifier. The mean F0 (c_0) is taken from the baseline system and c_1–c_{K-1} from the predicted template. We could have used another network to predict c_0 for each syllable but for the experiments presented in this thesis, we used the baseline system to compute c_0 by considering the mean over all the frames within each syllable. The reconstructed contour may be discontinuous at template joins, which was found to degrade perceived naturalness. To solve this, the F0 contour value predicted by the template system at each frame was appended to the frame-level linguistic features used as inputs to the acoustic model (see Figure 4.8), allowing the model to predict the final F0 (as statics and dynamics) and the voiced/unvoiced flag needed for

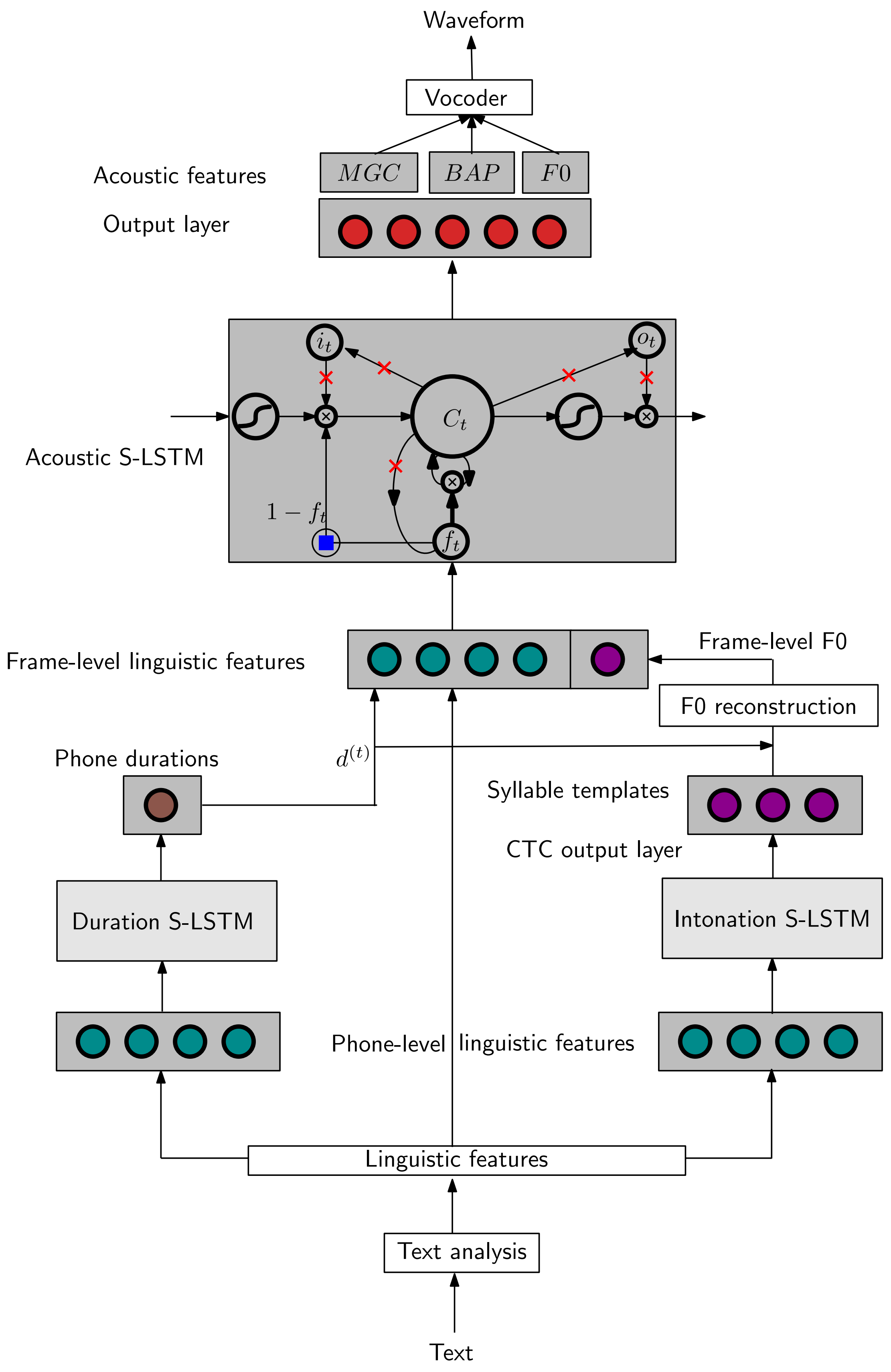


Figure 4.8: Schematic diagram of the proposed synthesis system SLSTM-CTC from Section 4.2.2.2 using syllable templates. A single LSTM unit is shown, although there is a full hidden layer of such units. Connections crossed out in red are present in standard LSTMs, but are omitted in the simplified LSTM units (Wu and King, 2016) used here.

waveform generation. During training, a template decomposition of the natural F0 was used to train the S-LSTM acoustic model to reconstruct smooth and natural F0 contours from discontinuous input. Figure 4.7 shows the preliminary (discontinuous) reconstruction along with the output after smoothing for the same example used in Figure 4.6. A pilot listening test (with five listeners from CSTR listening to 10 examples) indicated that this smoothing removed discontinuity artifacts in the F0 contours.

4.3 Experimental set-up

4.3.1 Data

We evaluated on prosodically interesting data from children's audiobooks, as provided by Usborne Publishing Ltd. for the 2016 Blizzard Challenge[2], containing text and speech of 50 children's audiobooks read by a British female speaker. The details of the data and split for train-validation-test was provided in Section 3.1.

4.3.2 Feature extraction

Phone-level linguistic features were derived following the procedure explained in Section 3.3.1. The proposed CTC system only utilised the phoneme-level binary input features to predict a distribution over the syllable templates from Section 4.2.1. While CTC can handle differing input sequence length compared to output sequence length, HC relies on one-to-one mapping for predicting syllable templates. Therefore, for HC, per-syllable input feature vectors were created by removing phoneme-level features for position and articulation, and replacing quinphone identities by a syllable-identity vector concatenating 7 50-dimensional one-hot phone vectors, allowing seven phones per syllable.

Duration prediction, identical for all systems, used the unmodified binary features as input to a DNN to predict a six-dimensional vector, comprising five-state durations and total phone duration (the last to regularise training, and not used in synthesis).

[2]http://www.synsig.org/index.php/Blizzard_Challenge_2016

STRAIGHT (Kawahara, 2006) was used to extract 60 MGC, 25 BAP and $\log F_0$, along with delta and delta-delta features every 5ms. Per-component mean and variance normalisation was performed, as previously discussed in Section 3.3.2.

4.3.3 Reference systems

Alongside the two proposed systems (Section 4.2.2) we created several reference SPSS systems which, except where noted, also predicted all acoustic outputs from linguistic features: MSE was a framewise-regression baseline predicting F0 using S-LSTMS. BMK used forced-aligned natural durations and natural F0 contours from the test-set recordings. VOC was a top line of vocoded speech. BOT was a bottom line using piecewise-constant F0 per syllable (the mean natural F0). Oracle used templates derived from the natural F0 contour of the test utterance. However, all systems except VOC used the same acoustic S-LSTM to generate frame-level acoustic features (MGC+BAP).

4.3.4 Training and synthesis

The SLSTM-CTC system used a six-layer network with the first four layers being feed-forward layers with 1024 nodes of tanh activation functions each, followed by an S-LSTM layer with 512 nodes and finally a softmax layer with seven output nodes. The HC system used a similar architecture with softmax output layers of two and five output nodes.

Each network was initialised using small random weights and subsequently optimised using stochastic gradient descent (no pre-training) with a fixed learning rate, manually tuned to yield close-to-optimal results on the development set in 30 epochs or less. Early stopping was used to avoid overfitting, and by selecting the model (epoch) with best dev-set performance.

One acoustic-model S-LSTM and one duration S-LSTM were trained, similar in structure to the intonation-prediction S-LSTM but with a linear output layer. Meta-parameters such as batch size and regularisation criteria followed Wu et al. (2015b).

For synthesis, `ehmm` phone sequences derived from the test data were used as inputs to duration and intonation prediction. This simply amounts to us-

	Classification measures		F0 measures	
Model	Accuracy	F1 score	RMSE (Hz)	Corr.
MSE	-	-	45.9	0.40
HC	61.1%	0.590	46.9	0.36
CTC	63.8%	0.593	46.1	0.40
Oracle	100%	1	40.8	0.58

Table 4.2: Classification correctness and F1 score of predicted syllable templates, along with RMSE and correlation of the predicted F0 contour, all measured w.r.t. held-out natural speech.

ing oracle pause insertion, with no other acoustically-derived information provided to the predictors.

After duration prediction, a sequence of frame-level linguistic features was generated using the predicted durations, and fed into the acoustic model to generate parameter trajectories as in Wu et al. (2015b). MLPG (Tokuda et al., 2000) using variances computed from the training data is applied to output features; Global variance from the development data was applied to the generated MGCs. Finally, waveforms were synthesised using the STRAIGHT vocoder (Kawahara, 2006) and normalised according to ITU P.56 (P.56, 2011).

4.4 Results

4.4.1 Objective measures

To evaluate the different systems, we calculated the root-mean-square error (RMSE) and Pearson correlation (Corr.) between natural and predicted F0 contours. The template-based approaches were additionally evaluated by classification accuracy and F1 score (Table 4.2). The objective measures and the metrics used for classification here were detailed clearly in section 2.3.1.

Considering accuracy and F1 score, both proposed systems (SLSTM-CTC and HC) performed reasonably well given the amount of data used for training and non-uniform distribution of template classes. Since the *SLSTM-CTC* loss function also makes use of the forward-backward algorithm to make utterance-spanning decisions, it performed slightly better than *HC*.

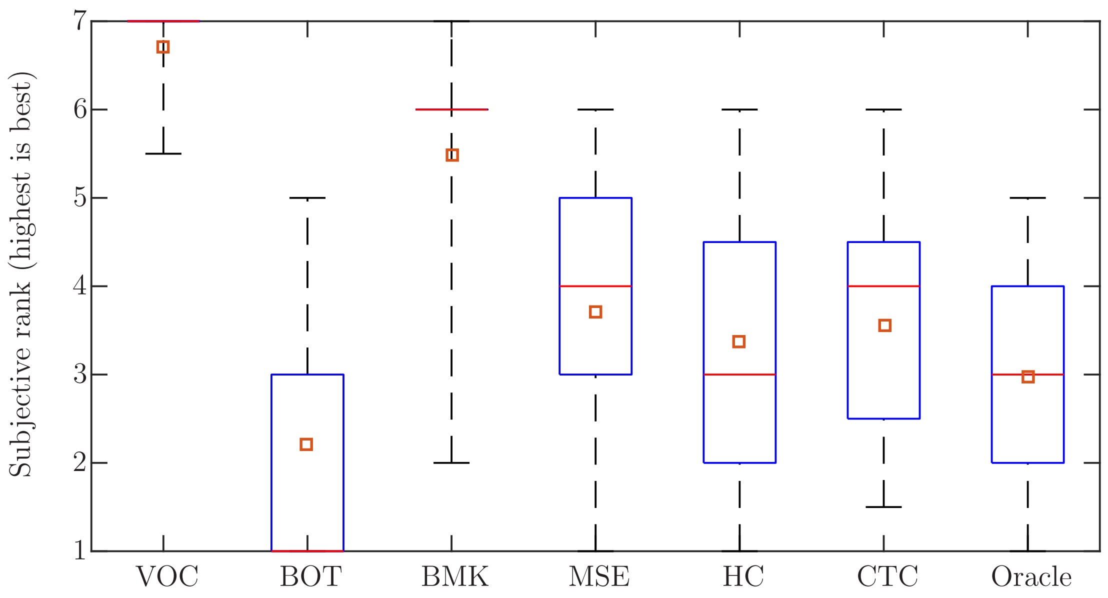


Figure 4.9: Box plot of aggregate ranks from listening test, comparing conventional and proposed systems against vocoded speech, bottom line and benchmark. Red lines are medians, orange squares means. Box edges (which may coincide) show quartiles. Whiskers cover 90% of the data.

The objective numbers for the *Oracle* system (correlation 0.58; lowest RMSE) indicate that natural F0 contours can be reasonably reconstructed from just six syllable templates. Note, however, that all the proposed systems (including *Oracle*) used c_0 values based on the mean F0 predicted by *MSE*. Instead using natural mean F0 for c_0 during F0 contour reconstruction increased the *Oracle* correlation to 0.89. The remaining systems, baseline and proposed, correlate less well with the held-out pitch contours and exhibit similar objective measures, though *HC*, which was designed to prioritize expressivity over accuracy, has the lowest correlation.

As shown in Table 4.2, the RMSE numbers are relatively high compared to any other datasets such as read text or news corpus. Since the data includes narrations of children's storybooks, the speaker is quite expressive while reading them. This has led to a very high RMSE (around 45 Hz, as shown in Table 4.2), as the audiobook data is more challenging to model.

However, objective measures can be misleading at times. For instance, there is no one correct way to speak, and F0 contours can differ notably among natural productions of the same text (cf. (Ribeiro et al., 2015, Sec. 4)), especially in expressive speech as considered here. Also, our proposed systems are

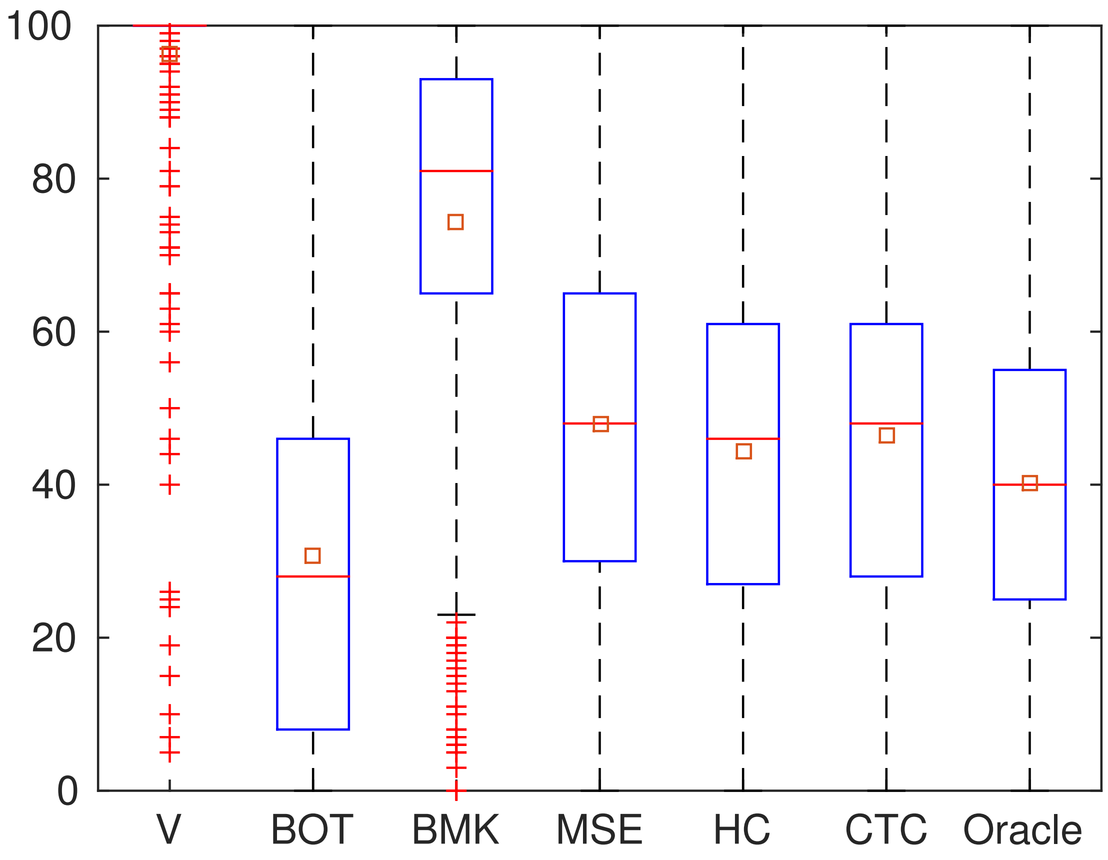


Figure 4.10: Box plot of absolute values from listening test.

designed to encourage expressivity instead of regressing to the mean.

4.4.2 Subjective evaluation

Next, the systems were evaluated subjectively using a hybrid between a MUSHRA (BS.1534-3, 2015) and a preference test. 20 native English speakers with no known hearing impairments, who were remunerated for their time, used GUI sliders to rank the relative naturalness of different systems producing the same sentence. Listeners were presented with screens containing one button per system (unlabelled and in random order), each speaking the same sentence. Listeners were asked to rank the relative naturalness of all stimuli using sliders before proceeding to the next screen. Each listener scored 20 out of 32 held-out sentences[3] in a randomised but balanced design; this was preceded by a two-sentence training phase and a tutorial screen. Tests took place in sound-insulated booths over high-quality headphones.

Figure 4.9 shows the box plot of aggregate ranks from the listening test,

[3]Samples are available at: `http://srikanthr.in/prosody-demo/intonation`

comparing conventional and proposed systems against vocoded speech, bottom line and benchmark. Vocoded speech was ranked the highest, followed by the natural-F0 (*BMK*) system; the bottom line was the least preferred. A Mann-Whitney U test and a Wilcoxon signed-rank test both found all pairs of systems to be significantly different ($p < 0.05$) except (*MSE*, *CTC*) and (*HC*, *Oracle*). Holm-Bonferroni (Holm, 1979) correction was applied because of the multiple comparisons. Figure 4.10 shows the box plot of aggregate values from the same listening test. The proposed *CTC* system performed as well as but unfortunately not (yet) better than the conventional baseline (*MSE*).

In stark contrast to the objective results (Table 4.2), the Oracle F0 contour was the least preferred among the main systems. This may point at an interesting interaction between duration and F0 that would affect many other results in the literature from systems that use oracle values for one, but synthesise the other. Perhaps a mismatch between the durations of the natural speech from which the templates were taken, and the durations of the synthetic speech onto which they were imposed, causes some inconsistency that listeners attend to. This explanation is consistent with Fernandez et al. (2014), which suggests that joint prediction of duration and F0 may be important in DNN-based acoustic models.

4.5 Conclusions

We have described a classification-based approach to intonation prediction with syllable F0 templates replacing frame-level regression. The listening test results suggest two things: that the proposed CTC approach matches the performance of the conventional approach, and has potential to exceed it once the issues with the Oracle template system are overcome; and, that there may be an important but under-explored interaction between duration and F0 that needs careful attention in the future.

Chapter 5

Median-Based Generation of Synthetic Speech Durations using a Non-Parametric Approach

This chapter is an expanded version of the work in Ronanki et al. (2016c) and therefore the text is closely related to that.

This work was completed in collaboration with others. Discussion of ideas was done between myself, Oliver Watts, Simon King and Gustav Eje Henter. The code for the LSTM system (with acoustic targets) used was provided by Zhizheng Wu. I expanded this to incorporate frame-level duration prediction. The code for frame-level duration feature extraction was written by myself and made open-source in the Merlin toolkit (Ronanki et al., 2016d). Fine-tuning of the systems was then done by myself with the help of Oliver Watts and Gustav Eje Henter.

5.1 Introduction

In Chapter 4, we focused on melody of speech through supra-segmental intonation generation using syllable-level F0 templates. This chapter focuses on generation of speech-segment durations, a characteristic of speech which contributes to the perception of its rhythm. For decades of text-to-speech research, duration modelling has been considered to be a challenging and important problem in generating natural-sounding synthetic speech, particularly in expressive or conversational scenarios. The absence of variation in the predicted durations often makes the synthetic speech dull. To overcome this problem,

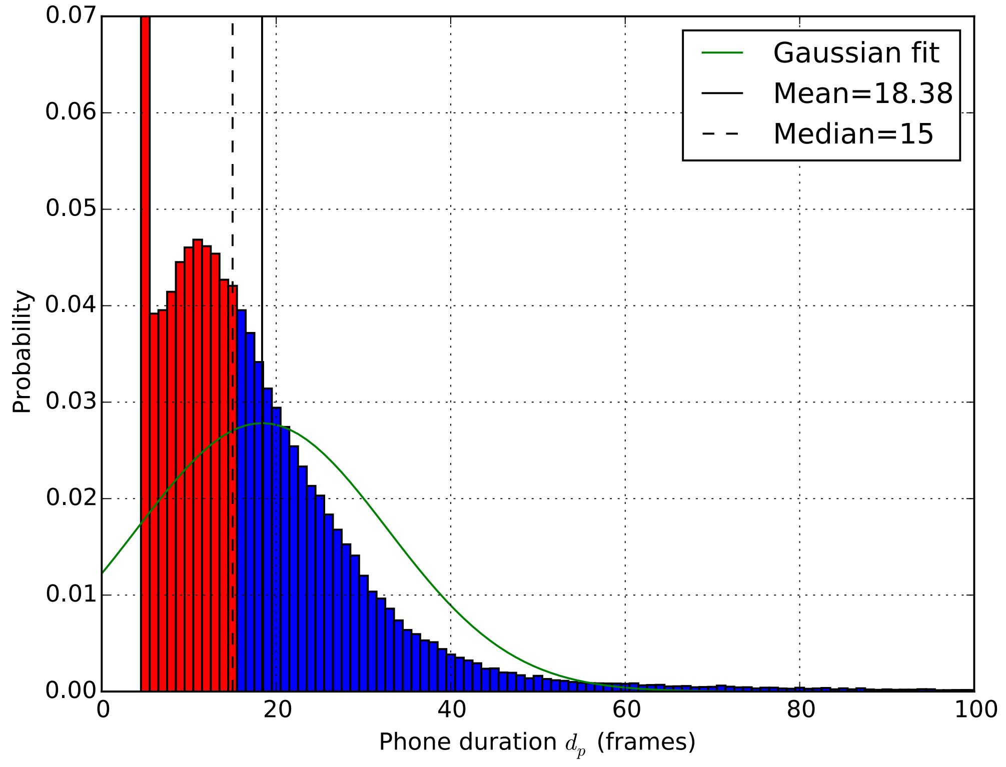


Figure 5.1: Duration distribution of training data, with median and mean quantiles.

we focus on statistical techniques for improved duration modelling, as a key step towards the overarching goal of more natural and appropriate synthetic speech.

As discussed in Section 2.1.2, steady improvements have been made in statistical parametric speech synthesis (SPSS), in particular through the adoption of deep and recurrent machine-learning techniques in recent years (Zen et al., 2013; Fan et al., 2014; Wu et al., 2015b). As we expect that high-level, long-range dependencies are of importance for the prosodic structure of speech, recurrent models such as LSTMs are well-suited to prosodic sequence modelling problems. For this reason, we made use of LSTMs for intonation prediction in Chapter 4 and similarly, the work by Zen and Sak (2015) used uni-directional LSTMs for duration prediction.

In spite of progress, however, the prosodic characteristics of synthetic speech still remain to be one of its major shortcomings, especially for audiobooks. One factor which we suppose contributes to this shortcoming is that current systems effectively model duration with a predetermined duration distribution (e.g., Gaussian distribution) and generate predictions from the mean of that distribution. This is problematic, as the distribution of speech-segment

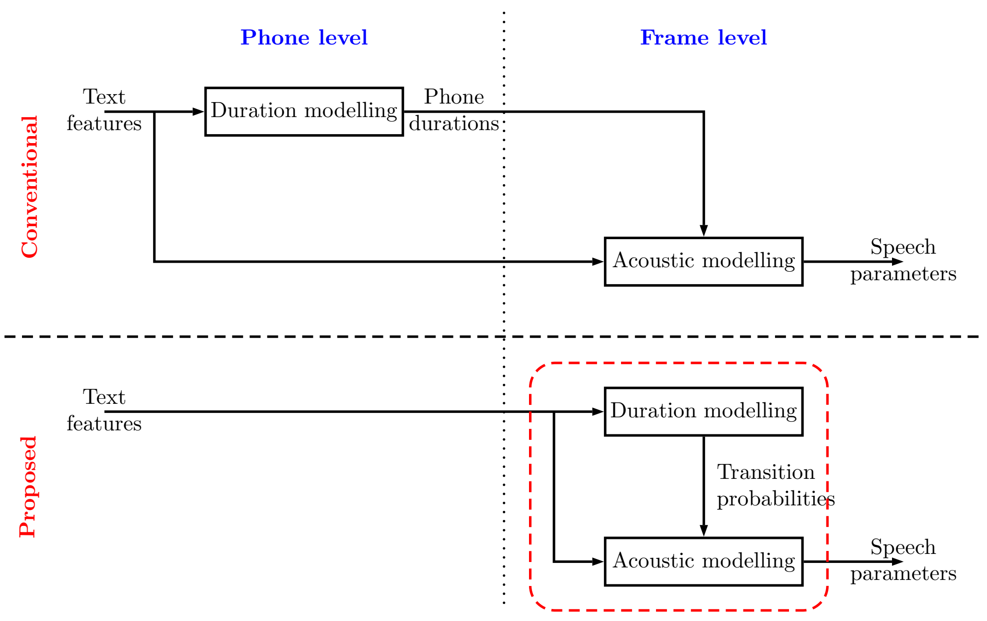


Figure 5.2: Schematic diagram of the proposed system

durations is well known to be skewed, and so using the mean of a normal distribution fit to such observations will not produce predictions that are most typical of the process (that is, have high probability). Also, the model is symmetric about the mean, and thus fails to account for the significant skewness in empirical duration distributions, with a heavy tail of long-duration realisations. On the 2016 Blizzard Challenge dataset described in Section 3.1, the mean duration is 18.38 frames per phone, while the median duration is 15 frames, indicating a global duration distribution skewed to the right (see Figure 5.1). However, we have to note that context-dependent duration distributions may not be as skewed as the global duration distribution and recurrent neural networks theoretically should be capable of modelling context-dependent distributions.

Another possible weakness of current approaches to duration generation is that they typically operate as an initial stage, separate from the generation of acoustic features (Zen and Sak, 2015). We consider it desirable to have a single model whose parameters are learned to simultaneously generate both segment durations and the frames of acoustic features within those segments (see figure 5.2). A major motivation for this is that such a joint model would allow the simultaneous adaptation (Wu et al., 2015a) and control (Watts et al., 2015b) of rhythmic, melodic and phonetic characteristics in a stable and consistent way.

However, one obvious difficulty in implementing such a model is that predictions of segment durations and of acoustic observations are conventionally made on two separate time-scales. Most recently, wavenet (van den Oord et al., 2016) has made waveform-level modelling possible in TTS, but still requires an external duration model and F0 prediction model. Our approach can in principle be integrated with WaveNet, for joint-prediction of F0 and duration or sample-level duration modelling.

The structure of this chapter is as follows: we present the motivation for using median-based duration prediction in Section 5.2. Section 5.3 presents the theory for conventional duration modelling and our proposed approach for duration modelling at acoustic frame level. Sections 5.4 and 5.5 describe the experimental setup and results respectively. We then present our related work on duration modelling using mixture density networks in Section 5.6. Finally, in Section 5.7 conclusion are drawn.

5.2 Motivation

In this chapter, we consider a scheme where durations generated at synthesis time are based on the median – rather than the conventional mean – of the predicted duration distribution. Conveniently, the median can be identified from the left tail of the distribution, whereas computing the mean requires evaluating the probability mass function over its entire, infinite support. This property enables median-duration based sequential output generation with no look-ahead or other overhead, which is attractive for incremental synthesis approaches. Unlike the mean, the median is nearly always an integer.

The advantages of generating median durations are more than computational: For skewed distributions, including the distribution of natural speech-sound durations as graphed in Figure 5.1, the median is frequently closer to the peak of the distribution than the mean is; cf. MacGillivray (1981); Basu and DasGupta (1997). This implies that, in the spirit of most likely output parameter generation (Tokuda et al., 2000), median-based duration prediction is likely closer to the peak density – the “most typical” outcome – than the mean is. (The mean duration is often atypically long.) Perhaps most importantly, the median is a *statistically robust* quantity, and is not affected by the tails of the real duration distribution. Statistical robustness is compelling for speech synthesis

(Henter et al., 2016), particularly for big and found datasets, as it reduces the sensitivity to errors and unexpected behaviour in the training corpus. Among other things, this could be of value with the highly expressive and variable training data used for the experiments presented in Section 5.4.

We here present an approach to duration modelling which in essence consists of predicting at each acoustic frame a probability that the model will subsequently advance to the next phone in the phonetic sequence. We show that – assuming a recurrent model is used – this approach may describe any duration distribution on the positive integers. Although it is feasible to model duration by explicitly choosing other distributions which may be more appropriate than a Gaussian (e.g., log-normal or gamma distribution), these still might not be optimal for a given conditional duration, and so the possibility of not having to commit to a predetermined distribution before model training is a powerful advantage of our approach. Furthermore, as our model moves from modelling duration at the level of the phonetic segment to the level of the acoustic frame, our approach is a necessary ingredient of our on-going work in unifying models of durations and acoustics, so that acoustic parameters and phone transition probabilities are predicted jointly for each frame.

5.3 Theory: Proposed approach

In this section, we describe how a non-parametric duration distribution can be modelled at acoustic frame level in parametric speech synthesis.

5.3.1 Preliminaries

Let $p \in \{1, \ldots, P\}$ be a phone index, and let $t \in \{1, \ldots, T\}$ be an index into frames. Let further D_p – a random variable – be the duration of phone p, and let $d_p \in \mathbb{Z} > 0$ be an outcome of D_p.[1]

Natural speech phones have different duration distributions that depend on the input text. In TTS, the properties of the distribution D_p are predicted from contextual linguistic features $\boldsymbol{l}_p$ extracted from the text by the synthesiser front-end as demonstrated in Section 3.3.1. Specifically, *duration modelling*

[1]In many TTS systems, D_p is a vector of per-phone state durations, but we restrict ourselves to phone-level predictions in this chapter; a state-based formulation is a straightforward extension.

is the task of mapping the sequence of linguistic features $(\boldsymbol{l}_1, \ldots, \boldsymbol{l}_P)$ to a sequence of predicted duration distributions $(D_1, \ldots, D_P)$. *Duration generation*, meanwhile, is the task of mapping $(\boldsymbol{l}_1, \ldots, \boldsymbol{l}_P)$ to a sequence of generated durations $(\widehat{d}_1, \ldots, \widehat{d}_P)$. In SPSS, one or the other of these mappings is learned from a corpus of parallel text and speech data using a machine learning method; in this chapter, deep and recurrent neural networks will be used. We will write $\mathcal{D}$ to denote a dataset of parallel input features $\boldsymbol{l}$ and random variable outcomes d used to train this predictor.

5.3.2 Conventional Duration Modelling

In statistical synthesisers that generate durations on the state or phone level, the conventional approach is to assume that the durations D_p follow some parametric family f_D with a parameter $\boldsymbol{\theta} \in \mathbb{R}^N$ (N being the number of degrees of freedom), i.e.,

$$\mathbb{P}(D_p = d) = f_D(d; \boldsymbol{\theta}_p). \tag{5.1}$$

Predicting the distribution D_p then reduces to the stochastic regression problem of predicting the distribution parameter $\boldsymbol{\theta}_p$ from the phone-level parallel input-output dataset

$$\mathcal{D}_p = ((\boldsymbol{l}_1, \ldots, \boldsymbol{l}_P), (d_1, \ldots, d_P)). \tag{5.2}$$

We will write $\boldsymbol{L}_p$ to denote the linguistic information available to the predictor at p, which is $\boldsymbol{l}_p$ for feedforward approaches and $(\boldsymbol{l}_1, \ldots, \boldsymbol{l}_p)$ for unidirectional RNNs.

In practice, most contemporary DNN-based synthesisers do not perform full distribution modelling, but map directly from $\boldsymbol{L}_p$ to the property of the distribution D_p that they wish to generate, such as its mean $\mathbb{E}(D_p)$. The dominant principle – and the only one to be considered for the baselines in this chapter – is to tune the weights $\boldsymbol{W}$ of a DNN or RNN $d(\boldsymbol{L}; \boldsymbol{W})$ to minimise the mean squared error (MSE) on the training dataset,

$$\widehat{\boldsymbol{W}}(\mathcal{D}_p) = \operatorname*{argmin}_{\boldsymbol{W}} \sum_{(\boldsymbol{L}_p, d_p) \in \mathcal{D}} (d_p - d(\boldsymbol{L}_p; \boldsymbol{W}))^2; \tag{5.3}$$

synthesis-time durations $\widehat{d}_p$ are then generated by

$$\widehat{d}(\boldsymbol{L}_p) = d(\boldsymbol{L}_p; \widehat{\boldsymbol{W}}(\mathcal{D}_p)). \tag{5.4}$$

The theoretically optimal predictor $\widehat{d}^{\star}$ that minimises the MSE is the conditional mean,

$$\widehat{d}_p^{\star}(\boldsymbol{L}_p) = \underset{\widehat{d}}{\operatorname{argmin}} \mathbb{E}\left((D_p - \widehat{d})^2 \mid \boldsymbol{L}_p\right) \tag{5.5}$$

$$= \mathbb{E}(D_p \mid \boldsymbol{L}_p), \tag{5.6}$$

so the end result is very similar to fitting a Gaussian duration model f_D and using the mean of that fitted Gaussian to generate durations.

5.3.3 Non-Parametric Duration Modelling

We will now describe a scheme that, unlike conventional, phone-level approaches with parametric families $f_D(d; \boldsymbol{\theta})$, is able to model and predict arbitrary duration distributions for D (restricted only by the biases of the machine learning method used). The key idea is to make predictions at the frame level about when phone transitions occur.

Assume phone durations are known up until the current frame t, and let $p(t')$ for $1 \leq t' \leq t$ be a function that maps from a given frame $t' \leq t$ to the phone it has been assigned to. We can then define a frame-level sequence $\boldsymbol{L}_t$ of linguistic features

$$\boldsymbol{L}_t = (\boldsymbol{l}_1, \ldots, \boldsymbol{l}_t) \tag{5.7}$$

$$= (\boldsymbol{l}_{p(1)}, \ldots, \boldsymbol{l}_{p(t)}) \tag{5.8}$$

up until t, which is constant for all frames within each phone.

For brevity, we shall write p for the current phone $p(t)$. Finally, we let t_0 be the final frame of the previous phone, so that we can define $n_t = t - t_0 \geq 1$, the duration of the current phone so far.

We now define the *transition probability* π_t for the phone p at time t, given the linguistic features up until this point – that is,

$$\pi_t = \mathbb{P}(D_p = n_t \mid D_p \geq n_t, \boldsymbol{L}_t). \tag{5.9}$$

As long as the transition probabilities satisfy $\pi_t \in [0, 1]$ and

$$\prod_{t'=t_0+1}^{\infty} (1 - \pi_{t'}) = 0 \tag{5.10}$$

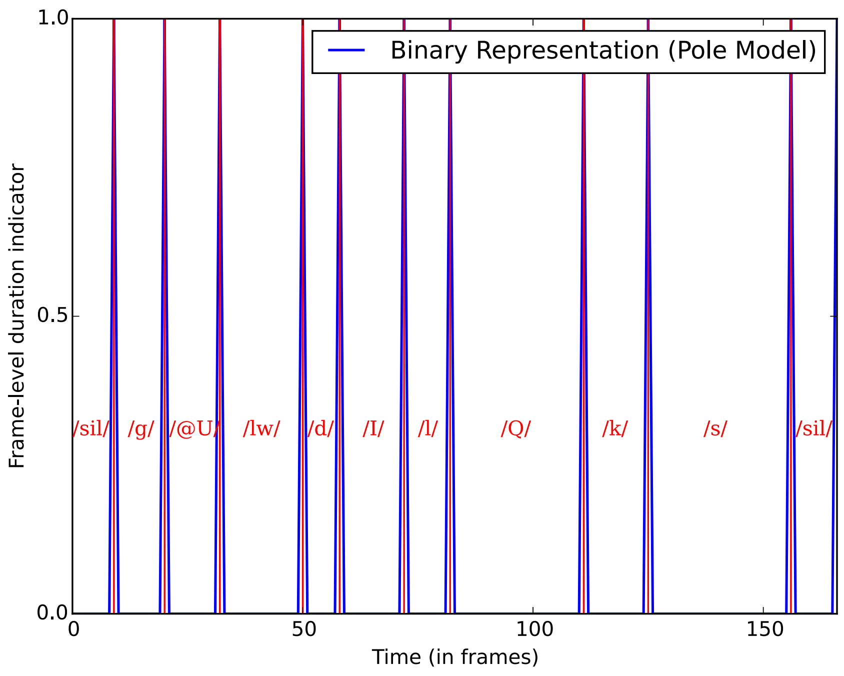


Figure 5.3: An example sentence showing the frame-level duration transition indicator (x_t) using binary representation. Red lines indicate phone level segmentation matching with frame-level transition.

they induce a unique duration distribution

$$\mathbb{P}(D_p = n_t \mid \boldsymbol{L}_t) = \pi_t \prod_{t'=t_0+1}^{t_0+n_t-1} (1 - \pi_{t'}) \tag{5.11}$$

on the positive integers.

We propose to build a predictor, based on training data, that estimates π_t from the linguistic input features. Specifically, we will train this predictor on the frame-level dataset

$$\mathcal{D}_t = (\boldsymbol{L}_T, (x_1, \ldots, x_T)), \tag{5.12}$$

where x_t is an indicator variable that equals one if and only if t is the final frame of the current phone, i.e.,

$$x_t = \begin{cases} 1 & \text{if } t = t_0 + d_p \\ 0 & \text{otherwise.} \end{cases} \tag{5.13}$$

An example sentence with time (in frames) on x-axis and frame-level duration indicator on y-axis is shown in Figure 5.3 denoting the binary representation given in equation 5.13, under the pseudonym of "*pole model*".

In this chapter, we consider a deep and unidirectional recurrent neural network $x(\boldsymbol{L}; \boldsymbol{W})$, with weights $\widehat{\boldsymbol{W}}$ trained to minimise the MSE in recursively predicting the indicator variable x_t – that is,

$$\widehat{\boldsymbol{W}}(\mathcal{D}_t) = \underset{\boldsymbol{W}}{\operatorname{argmin}} \sum_t (x_t - x(\boldsymbol{L}_t; \boldsymbol{W}))^2. \tag{5.14}$$

The hypothetical predictor $\widehat{x}^\star$ that minimises this MSE is the conditional mean,

$$\begin{aligned} \widehat{x}_t^\star(\boldsymbol{L}_t) &= \underset{\widehat{x}}{\operatorname{argmin}} \mathbb{E}\left((X_t - \widehat{x})^2 \mid \boldsymbol{L}_t\right) & (5.15)\\ &= \mathbb{E}(X_t \mid \boldsymbol{L}_t) & (5.16)\\ &= \sum_{x \in \{0,1\}} x \cdot \mathbb{P}(X_t = x \mid \boldsymbol{L}_t) & (5.17)\\ &= \mathbb{P}(X_t = 1 \mid \boldsymbol{L}_t), & (5.18) \end{aligned}$$

which is mathematically equivalent to the transition probability π_t. This means that, as long as the predictor of X_t is theoretically capable of generating arbitrary outputs for every frame, our approach can describe virtually any transition distribution – and thus any duration distribution – on the positive integers. LSTMs and other RNNs satisfy this requirement, due to their internal state (memory), here denoted $\boldsymbol{c}_t$, which evolves from frame to frame.

Unlike the maximum likelihood objective function, the MSE used here is not disproportionately sensitive to large errors in probability estimates of rare events (outliers). This makes the estimated $\widehat{\boldsymbol{W}}$ more statistically robust to errors in data and model assumptions.

5.3.4 From Transitions to Median Durations

Having defined a phone duration distribution in Equation 5.11 and trained a model to predict this distribution from data, we must consider how predicted durations $\widehat{d}$ are to be generated from this distribution. As discussed in Section 2.2.3, sampling typically yields poor naturalness, while mean-based generation is sensitive to the tails of the distribution and unsuitable for sequential generation. On the other hand, it is straightforward to derive the right tail probability of the phone duration distribution induced by the values of π_t seen thus far, as

$$\mathbb{P}(D_p > n_t \mid \boldsymbol{L}_t) = \prod_{t'=t_0+1}^{t_0+n_t} (1 - \pi_{t'}). \tag{5.19}$$

This relation straightforwardly enables synthesis based on the predicted *median* duration: by stepping from $n_t = 1$ and upwards, the (estimated) median duration $\widehat{d}_p$ of phone p is reached when $\mathrm{P}(D > d)$ first dips down to 0.5 or below,

$$\widehat{d}_p = \min_{n_t \in \mathbb{Z}} n_t \tag{5.20}$$

$$\text{such that } \mathbb{P}(D_p > n_t) \leq 0.5. \tag{5.21}$$

Equation 5.19 thus allows us to generate frames sequentially, advancing $p(t+1)$ to the next phone $p+1$ when the predicted median duration is reached, with no additional overhead at generation time.

Just as the mean squared error is minimised by the mean, the median is the theoretical minimiser of another error measure, namely the *mean absolute error* (MAE),

$$\mathrm{MAE}(\widehat{d}) = \sum_{p \in \mathcal{D}_p} \left| d_p - \widehat{d}_p \right|. \tag{5.22}$$

If a method improves the MAE, we would expect its prediction to be closer to the (conditional) median of the data. Interestingly, summing absolute rather than squared errors is a common component of the mel-cepstral distortion (MCD) (Kubichek, 1993b) often used to evaluate acoustic models in speech synthesis, but the same idea is less commonly seen for evaluating generated durations.

5.3.5 Adding External Memory

In the proposal described so far, progress through the current phone is tracked solely via the internal state of the predictor used ($\boldsymbol{c}_t$ for an LSTM). This is in contrast to the approach used by the hidden semi-Markov models in HTS, which achieved more general duration distributions than regular HMMs by maintaining an external counter of the number of frames spent in each state, and using it to compute the transition probability. However, nothing prevents us from adding our variable n_t, which counts the frame number within the current phone, as an input to the neural network $x(\cdot; \boldsymbol{W})$ that predicts frame-level transition probabilities, in addition to the regular, linguistic features $\boldsymbol{L}_t$. We call these augmented features $\boldsymbol{l}'_t$ and

$$\boldsymbol{L}'_t = ([\boldsymbol{l}_1^\intercal, n_1]^\intercal, \ldots, [\boldsymbol{l}_t^\intercal, n_t]^\intercal), \tag{5.23}$$

so that the augmented predictor is $x(\boldsymbol{L}'; \boldsymbol{W})$.

One may surmise that having n_t as an input feature could increase performance, as it provides highly relevant information to the model at all times, instead of hoping the predictor will discover the importance of duration tracking it on its own during training. To test this hypothesis, our experiments in Section 5.4 compare two systems that differ only in whether or not they incorporate an external counter n_t as an input to the central neural network.

As a side note, appending n_t to $\boldsymbol{l}_t$ makes the inputs to the frame-level neural network $x(\cdot; \boldsymbol{W})$ different with each frame. This makes it possible for the predicted transition distributions to differ from frame to frame as well, even without using a stateful predictor such as an RNN. This should, for example, enable the prediction of arbitrary transition distributions also when $x(\cdot; \boldsymbol{W})$ is a non-recurrent, feedforward neural network. We have, however, not explored this possibility.

5.4 Experimental set-up

5.4.1 Data

As a first inquiry into the viability of our frame-level duration-prediction approach, we conducted some experiments on a speech database created for the 2016 Blizzard Challenge(King and Karaiskos, 2016). For the purposes of the work presented here, 4% of the data was set aside as a test set. The test set consists of three whole short stories as described in Section 3.1.

5.4.2 Feature extraction

Phone-level linguistic features were derived by force-aligning the speech data with text – the procedure for extraction was detailed in Section 3.3.1. For acoustic or spectral modelling, 9 additional numerical features at frame-level are to be appended. Since, we're predicting durations at phone level, these numerical features were not used. All linguistic inputs were normalised to the range $[0.01, 0.99]$. The phone durations used to train conventional baselines were obtained by counting the number of frames in a phone. Contrary to the work presented in Henter et al. (2016), sub-phone states were not used in either DNN training or prediction. The target features for the conventional baseline is a single-dimensional normalized phone duration. This is consistent with our longer-term goal of freeing ourselves from depending on often arbitrary HMM sub-phone alignments.

5.4.3 System training

Four systems were trained: two phone-level predictors (which we identify as *Phone-DNN* and *Phone-LSTM*) to provide benchmarks and assess the impact of recurrent modelling on duration prediction, and two experimental systems implementing the new idea (which we identify as *Frame-LSTM-I* and *Frame-LSTM-E*). We'll further explain these systems in detail in the following subsections.

For training, all networks were initialised using small random weights, with no pre-training. Each duration prediction system was trained with a fixed learning rate, manually tuned to yield close-to-optimal results on the development set in 25 epochs or less. Early stopping was used to avoid overfitting, by aborting training once the objective function on the development set had failed to improve for five epochs. All systems were trained using the Merlin speech synthesis toolkit (Wu et al., 2016).

5.4.3.1 *Phone-DNN* and *Phone-LSTM*

The two baselines use phone-level linguistic features as input and are optimised to predict the (mean and variance normalised) duration of the phones in the training data. *Phone-DNN* was a feedforward DNN with six layers of 1024 nodes each. The hidden nodes used tanh activation and the output was linear. *Phone-LSTM* was configured with five feed-forward layers of 1024 nodes each and a final uni-directional SLSTM (Wu and King, 2016) hidden layer consisting of 512 nodes.

Algorithm 1 Switching criterion for *Frame-LSTM-I*

```
procedure SwitchProb($linguisticInput$)
    $nof \leftarrow$ length of linguisticInput
    for each $phoneNum$ in $nof$ do
        $remMass \leftarrow 1$
        for each frame $t$ in order do
            $ph \leftarrow linguisticInput[phoneNum]$
            $probSwitchAtT \leftarrow FrameLSTM(ph)$
            $remMass \leftarrow remMass * (1 - probSwitchAtT)$
            if $remMass <= 0.5$ then
                $predictedDur[phoneNum] \leftarrow t + 1$
                break;
            end if
        end for
    end for
    return $predictedDur$
end procedure
```

5.4.3.2 *Frame-LSTM-I* and *Frame-LSTM-E*

Both proposed systems used the same architecture as that of *Phone-LSTM* but – unlike the baseline systems – were trained with 1 datapoint per frame instead of per phone. The targets presented in training were 0.0 for non-phone-final frames, and 1.0 for phone-final frames. *Frame-LSTM-I* took only the vector representing the current phone as input; at run-time it was therefore required to rely on its *internal* memory (hidden state and LSTM cell state) to determine the phone transition probability at any given frame. The inputs to *Frame-LSTM-I* are identical for all frames in any given phone. *Frame-LSTM-E* implemented the idea described in Section 5.3.5: inputs encoding the current phone context are augmented with a frame counter indicating the number of frames that have passed since the start of the phone. *Frame-LSTM-E* can therefore also rely on this *external* memory of how much time has elapsed within the current phone when making phone transition predictions, in addition to internal memory.

Model	RMSE	MAE	Corr.
Phone-DNN	8.037	4.759	0.750
Phone-LSTM	7.789	**4.556**	0.765
Frame-LSTM-I	8.254	4.610	0.761
Frame-LSTM-E	8.294	**4.574**	0.754

Table 5.1: RMSE and MAE (both in frames per phone), along with Pearson correlation of the predicted durations, all measured w.r.t. force-aligned durations.

5.4.4 Synthesis

At synthesis time, `ehmm` phone sequences derived from the test data were used as input to each duration prediction model. This corresponds to using an oracle pausing strategy, but providing no other acoustically-derived information to the predictors. Generation from *Phone-DNN* and *Phone-LSTM* is straightforward: an (effectively mean) output is generated for each phone.

Median-based duration predictions were obtained from *Frame-LSTM-I* and *Frame-LSTM-E* by running the models recurrently over frame-level inputs and using the technique explained in Section 5.3.4 and summarised in pseudocode in Algorithm 1, to decide when the median had been reached and thus when to move to the next phone. By keeping track of the time spent in each phone, we obtain median-based predictions of phone duration either with external memory of time elapsed in current phone (*Frame-LSTM-E*) or without it (*Frame-LSTM-I*).

5.5 Results

Table 5.1 gives RMSE, mean absolute error (MAE) and Pearson correlation computed by comparing the predicted durations against the held-out reference. Silence segments were excluded from these calculations.

Firstly, results for the benchmark systems show that *Phone-LSTM* outperforms *Phone-DNN* in every respect, confirming the importance of the LSTM's recurrence for duration modelling. Turning to the experimental systems, it can be seen that while the mean-based predictions of the LSTM and the DNN baselines clearly outperform the proposed methods in terms of RMSE, the gap is much smaller when it comes to MAE. Median-based generation methods are

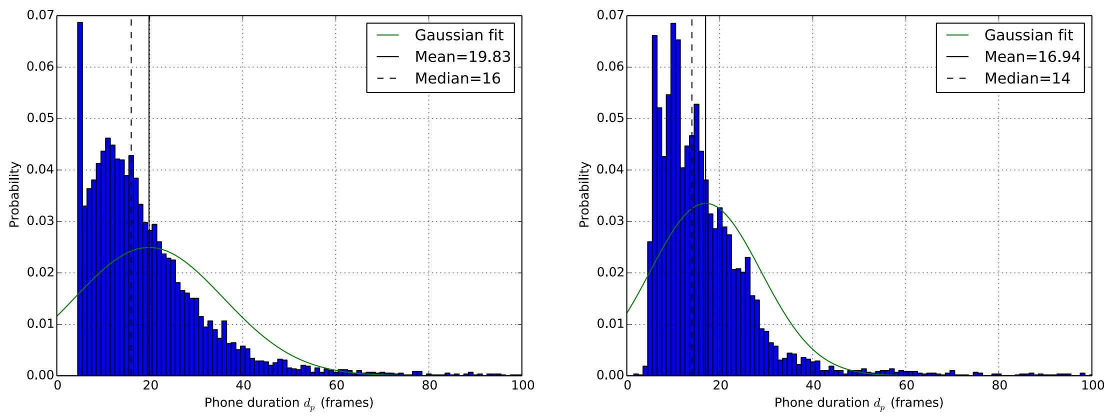


Figure 5.4: Duration distribution of test data, with Oracle (natural) duration distribution on the left and predicted (Frame-LSTM-E) duration distribution on the right.

expected to score better on MAE than RMSE, because – as discussed in Section 5.3.4 – the median is the theoretical minimiser of MAE, while the mean minimises the (R)MSE.

The breakdown of results by phonetic class given in Table 5.2 shows an interesting phenomenon: while the proposed system *Frame-LSTM-E*'s MAE is worse than the LSTM benchmark for vowels and slightly worse for consonants overall, for all classes of consonant except plosives the proposed method performs better.

Figure 5.4 shows the histogram distribution of natural and predicted duration distribution of the held-out test data. Also, from the figure, it can be interpreted that the median based prediction system (Frame-LSTM-E) is capable of learning a heavy-tail distribution although the predicted speech is little fast.

While the best-performing experimental system does not overall outperform the LSTM baseline even on MAE, the gap in performance is effectively closed (4.556 vs. 4.574 frames per phone). This means that we have devised a system which gives similar performance in the MAE sense as our existing one, but provides greater compatibility with our acoustic models, as it too operates at the frame level.

For further analysis, an example output for the frame-level duration indicator with *Frame-LSTM-E* is shown in figure 5.5. In the figure, output from *Frame-LSTM-E* is denoted in red, while tail probability expressed by $\mathbb{P}\left(D_p \leq n_t\right)$ is denoted in blue, and duration transition ($\mathbb{P}\left(D_p = n_t\right)$) in green. Since the

	Phone-LSTM			Frame-LSTM-E		
Phonetic class	**RMSE**	**MAE**	**Corr.**	**RMSE**	**MAE**	**Corr.**
Vowels	8.516	**4.848**	0.809	9.027	4.891	0.799
Consonants	7.313	**4.378**	0.709	7.815	4.382	0.694
Plosives	5.206	**3.608**	0.732	5.610	3.612	0.720
Fricatives	6.489	4.380	0.769	6.859	**4.246**	0.764
Nasals	6.833	4.459	0.568	7.376	**4.398**	0.550
Affricates	5.658	4.220	0.797	5.432	**3.746**	0.821
Glides + liquids	8.013	5.260	0.569	8.235	**5.075**	0.599

Table 5.2: Objective measures from Table 5.1 broken down by phonetic class.

duration values are mean-normalised at the time of training, we denormalise then to plot the predicted output (*Frame-LSTM-E* in this example). Ideally, we may expect the output to be close to zero for most of the frames and the final transition frame to be one. However, the training has only learnt to reach a local maxima and therefore falls after that. Hence, we compute $(1 - \pi_{t'})$ which also denotes the probability to remain in the same state (as in current phone) and then we determine its duration by further computing the tail probability of the phone duration distribution as presented in Section 5.3.4.

We consider this approach to be also effective when considering flexibility in duration modification. We can set the generated quantile $q \neq 1/2$ to alter the speaking rate, as in equation 5.21. The same idea can also be used to enforce a specific utterance duration by choosing $\widehat{q}$ such that actual and generated mean phone duration match on $\mathcal{D}_p$. Considering the benefits of non-parametric duration modelling at acoustic frame-level, the next logical step, therefore, is to apply this idea to joint modelling of duration and acoustic features. Similar to the work presented here, (Shen et al., 2017) effectively used sentence-level transition probability as a stop token for the end of sequence prediction. Also, as discussed in Section 2.1.3, the recent work in neural speech synthesis (Wang et al., 2017c; Shen et al., 2017) has shown that attention based approaches can implicitly model duration by predicting a sequence of acoustic parameters given the sequence of phonemes.

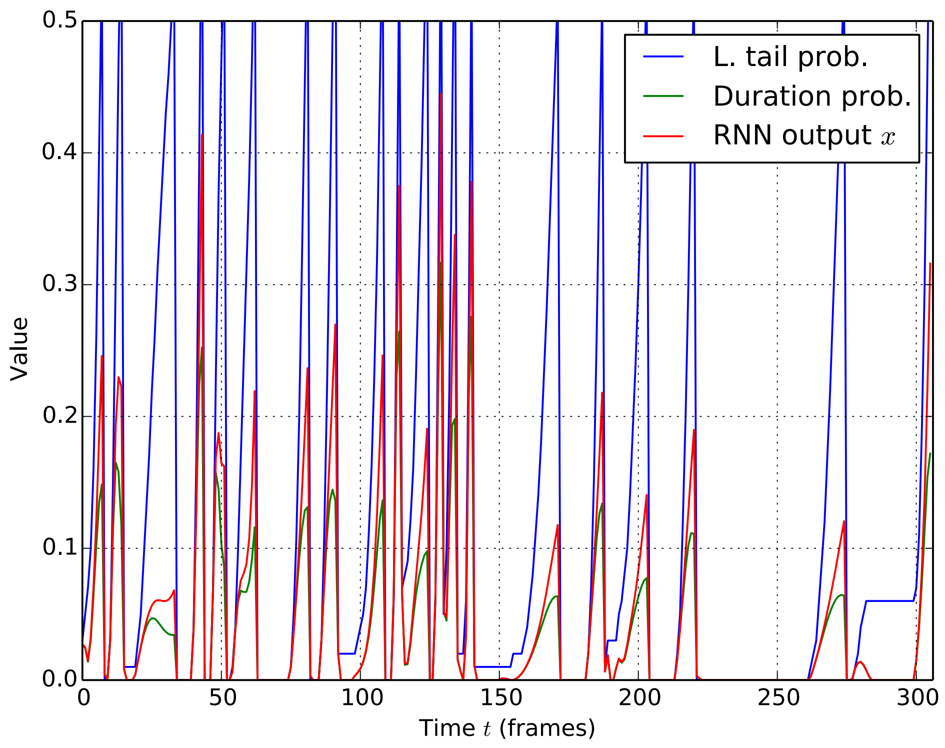


Figure 5.5: Example output for frame-level prediction with Frame-LSTM-E

5.6 Other related work on duration modelling

As discussed in Section 2.2.3 and 5.3.2, conventional approaches to duration modelling with DNNs predict duration per phoneme based on its context [Zen and Sak (2015), for example]. In this section, I describe some of my earlier work (in collaboration with others) on duration modelling with improvements to conventional approaches using: a) Methods from robust statistics for training with mixture density networks (MDN); b) Multi-task approach for duration modeling; The experiments were conducted on audiobooks from Librivox[2] and Blizzard Challenge data from 2016 (King and Karaiskos, 2016).

5.6.1 Robust duration models

In this section, the investigation reported by Henter et al. (2016) for robust duration modelling using DNNs is summarized. My contribution to this work was discussions, implementation of duration modelling and experimental setup including systems training. The perceptual listening test was carried by Henter.

[2]https://librivox.org

Theory

When modelling duration for audiobooks, it is known that DNNs/RNNs trained with the squared loss function (L2 loss) can suffer from the effect of outliers. These outliers generally can come from forced alignments as a result of mismatch between recordings and transcriptions. In the work by Henter et al. (2016), we therefore focused on statistical techniques for improved duration modelling, using DNNs. To cope with errors and empirical variability that cannot be modelled with standard set-ups, we introduced estimation procedures from the field of robust statistics to automatically identify and disregard dubious or unhelpful datapoints. As a side benefit, we obtained models that better estimate peak probabilities in the data, which is appealing since standard output-generation methods depend on these peaks for synthesis.

Unfortunately, found audiobook datasets are seldom subject to the quality-control that traditional synthesis methods expect. Common issues likely to affect duration modelling include transcription errors, reductions, filled pauses, and forced-alignment inaccuracies. Due to inaccuracy in the modelling, the forced-alignment durations may contain errors unless adjusted manually. To combat this, we propose to improve modelling and prediction of speech durations using methods from robust statistics, which are able to disregard ill-fitting points in the training material. To produce more stable durations from potentially faulty data, we first considered an MDN-based heuristic drawing on previous work in acoustic modelling by Zen et al. (2013). We will assume that durations are distributed according to a K-component Gaussian mixture distribution with diagonal covariance matrices,

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^{K} \omega_k \cdot f_{\mathcal{N}}(\boldsymbol{x};\boldsymbol{\mu}_k, \mathrm{diag}(\boldsymbol{\sigma}_k^2)) \tag{5.24}$$

where $f_{\mathcal{N}}$ is the Gaussian pdf and the component masses ω_k are nonnegative and sum to one. The individual phone durations $\boldsymbol{x}_p$ are assumed to be mutually statistically independent given the set of linguistic features $\{\boldsymbol{l}_p\}_p$ over the training data.

We improve duration prediction through a mismatch between training and generation principles: while the traditional likelihood optimised during training fits multiple components to the data, only a single mixture component $k(p)$ is selected to generate each phone p at generation time. Datapoints which were attributed to other components during training are effectively ignored during synthesis: the unused components thus act as a garbage model. We have chosen to generate durations based on the mode of the heaviest mixture component (greatest ω_k) for each phone p. The idea of fitting a mixture model and only using the heaviest component was previously described in Zen et al. (2013) for acoustic modelling.

Similar to duration modelling in early HMM-based synthesis systems, the key idea for the work in Henter et al. (2016) was to use statistical robustness to focus on getting better descriptions of typical durations from the model at prediction time, when facing issues with both the models and the data. For this purpose, we further considered robust regression techniques such as minimum density power divergence estimator to produce more natural synthetic-speech rhythm and durations by learning from large and less artificial speech datasets, whilst being robust against the inevitable errors. We consider minimising the density power divergence of Basu et al. (1998), also known as the beta-divergence (Eguchi and Kano, 2001). This leads to the following estimation principle:

$$\widehat{\boldsymbol{W}}_{\beta}(\mathcal{D}) = \underset{\boldsymbol{W}}{\operatorname{argmin}} \sum_{p} \Bigg(f(\boldsymbol{x}_p; \boldsymbol{\theta}(\boldsymbol{l}_p; \boldsymbol{W}))^{\beta} \\ - \frac{\beta}{1+\beta} \int f(\boldsymbol{x}; \boldsymbol{\theta}(\boldsymbol{l}_p; \boldsymbol{W}))^{1+\beta} \, \mathrm{d}\boldsymbol{x} \Bigg), \quad (5.25)$$

where β is a positive tuning parameter. Unlike the generation-time heuristic introduced earlier, the density power divergence offers a principled approach to robust estimation with theoretical guarantees on its performance. Beta estimation from this technique actually reduces the mismatch between training and generation time and was only used to fit a single Gaussian in a robust way. A more detailed explanation of this work along with the motivation behind using beta-divergence for robust methods is given in Henter et al. (2016).

Models

We built a number of different robust and non-robust DNN-based predictors of duration. As non-robust baselines, we used a traditional minimum mean square error (MMSE) optimised DNN, labelled "MSE", and a single-component, maximum-likelihood Gaussian MDN, labelled "MLE1".

Against these standard approaches, we contrast the two robust methods:

Model	BOT	MSE	MLE1	MLE3	B75	B50
Correlation	0.55	0.80	0.79	0.79	**0.81**	0.80
RMSE (100%)	9.17	6.70	6.69	6.86	**6.46**	6.56
RMSE (90%)	5.87	4.02	3.95	3.83	**3.45**	3.50

Table 5.3: Pearson correlation and RMSE (frames per phone) between predicted and forced-aligned durations. RMSE is reported for all data (100%) and for the 90% of the test data with the smallest prediction residual.

a three-component MDN using maximum likelihood, where only the component with maximum distribution was used for synthesis. This system was labelled "MLE3". For the robustly trained models using the power divergence in equation 5.25, we tried $\beta = 0.358$ and 0.663. These β-values were selected based on the asymptotic variance formula in (Basu et al., 1998, Sec. 4.2.d), such that approximately 75 or 50% of the original datapoints would be retained in the case of Gaussian-distributed observations. The associated systems were labelled "B75" and "B50", respectively.

We also included a topline system using reference durations from forced alignment on the test-set recordings (labelled "FRC"), and a bottom line (labelled "BOT") simply predicting durations to be equal to the corresponding mean monophone duration in the training data. No reasonable duration prediction system should be worse than this bottom line.

Evaluation and results

To evaluate the different duration models, we conducted an objective evaluation using the root-mean-square error (RMSE) in frames per phone (excluding silences and pauses), as well as the Pearson correlation coefficient, for each different system in relation to the reference durations (FRC) from forced-alignment. The results are reported in the first two rows of Table 5.3.

Since robust methods are designed to ignore extreme examples to describe the remainder of the data better, an interesting measure to consider is the RMSE on the X% of the test data with the smallest prediction residual. This is plotted in Figure 5.6, relative to the error of the monophone bottom line, with absolute numbers for $X = 90\%$ provided in the final row of Table 5.3. It is seen that the robust methods (solid lines), especially the ones based on the

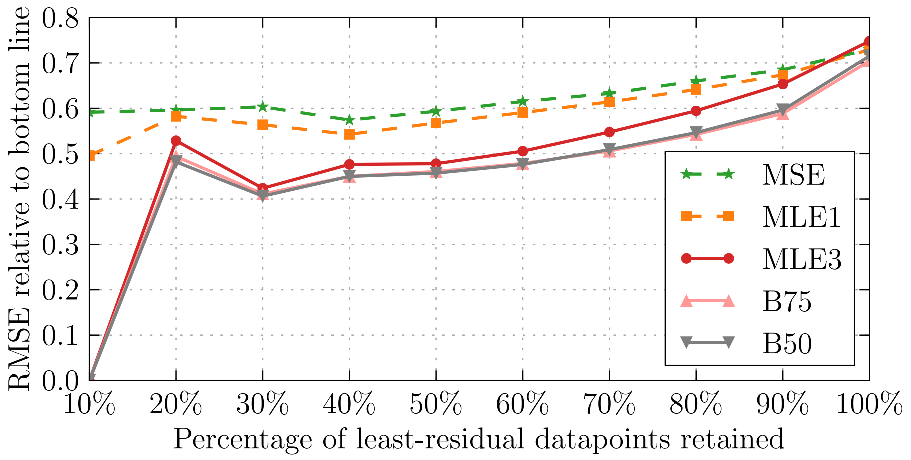


Figure 5.6: Relative RMSE (frames per phone) models on progressively larger and less well explained test-data subsets. (BOT is at 1.0.)

β-divergence, consistently outperform the non-robust baselines (dashed lines) for subset sizes less than 100%. This suggests that the robust methods were able to learn better models of the typical durations in the data.

A perceptual experiment was also conducted to evaluate all systems side-by-side using a hybrid between a MUSHRA and a preference test. Listeners ranked several parallel, unlabelled stimuli from the same text but different systems in terms of preference. No reference was given as there is no *one* correct prosodic (durational) realisation of a sentence, but rather numerous correct productions. Nevertheless, subjects were told to give at least one stimulus in every set a rating of 100.

Each listener rated 18 sets of 8 stimuli each, corresponding to the 6 methods in Table 5.3 plus FRC and vocoded natural speech (VOC). 21 normal-hearing native English speakers participated in the test, conducted in sound-insulated booths. For analysis, each set of eight parallel listener scores was converted to ranks from 1 (lowest) to 8 (highest), with tied ranks set to the mean of the tied position. A box plot of these rank scores aggregated across all prompts and listeners is shown in Figure 5.7.

The box plot indicates that robust methods perform better than non-robust ones, though a substantial gap remains to the vocoded natural speech. Mann-Whitney U significance tests with Holm-Bonferroni correction (Holm, 1979) applied to keep the familywise error rate below 5% showed most system pairs to be significantly different, except the sets {FRC, MSE, MLE1}, {FRC, MLE3},

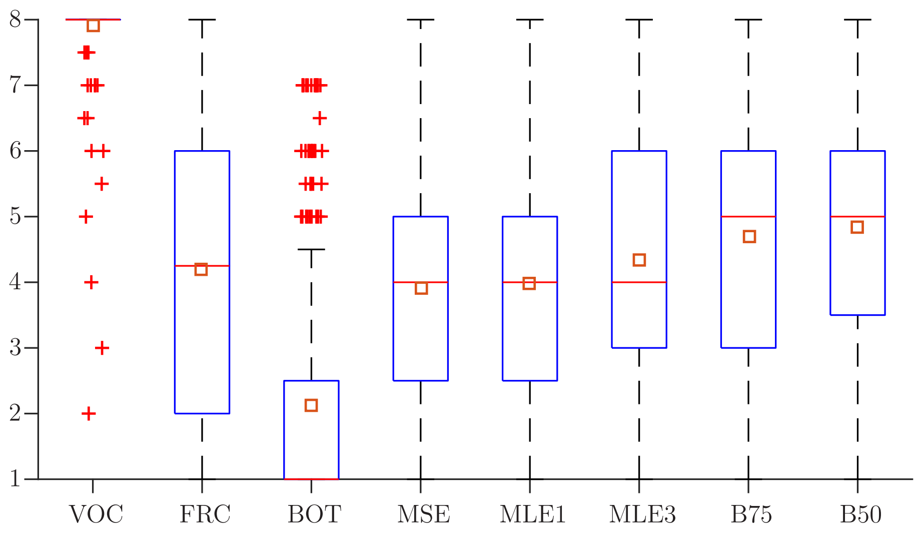


Figure 5.7: Aggregated ranks in listening test. Red lines are medians, orange squares denote means; box edges are at 25 and 75% quantiles.

and {B75, B50}.

It is clear that robustly predicted durations, particularly those based on the β-divergence, were preferred over their non-robust counterparts. Interestingly, durations generated using the non-robust baselines were not significantly worse than oracle durations based on forced-alignment. It should be noted that speech from robust models is faster than that of other systems: because of the skewness of the empirical data, rejecting extremely long durations noticeably reduces the average duration of certain speech sounds, making robust methods produce more phones per second. As speech rate affects perception (Dall et al., 2014), this might contribute to the robust methods scoring better than FRC.

5.6.2 A multi-task approach for duration modeling

In this section, we describe the duration model trained for our entry to the Blizzard Challenge 2016 (Merritt et al., 2016b). The details of the overall system architecture and results for the Blizzard Challenge 2017 (Ronanki et al., 2017a) will be discussed in Chapter 9.

In this work (Merritt et al., 2016b), we investigated the benefits of including long-range dependencies and multi-task learning in duration prediction using a) recurrent mixture density networks and b) simultaneously predicted durations on many levels, all the way up to word level prediction, and combining all these interrelated predictions at synthesis time. In the previous work by Henter et al. (2016), phone-level duration was used as a multi-task side-objective when predicting state-durations. Here, we extended this to syllable and word level not only to encourage learning of longer-range properties, but also to integrate their predicted values into the synthesis procedure. Whilst analysing the word-duration predictions (per every phoneme), we saw a pattern where predictions would consistently be too short at the edges of words, and too long in the middle. This observation was also demonstrated by the difference in RMSE for word-duration predictions (per every phoneme) on a very small sample word set (10 long words from our development set). By comparing instantaneous word-level predictions with the (more accurate) average over the entire word, and scaling in the generated sub-phone durations in the opposite direction of the disparity, this pattern could be counteracted. This scheme both integrates predictions at multiple levels into the generated durations, and introduces long-range dependencies by leveraging predictions over the entire word into each sub-phone duration.

For each phoneme, the scaling factor ($\tilde{s}_w$) is computed as:

$$\tilde{s}_w = \frac{2 * \widehat{D}_w}{\widehat{D}_w + \widehat{d}_w} \tag{5.26}$$

where $\widehat{d}_w$ denotes the instantaneous word-level prediction per phoneme and $\widehat{D}_w$ denotes its average over the entire word. The phone duration ($\tilde{d}_p$) is computed as a scaled sum of state-durations as:

$$\tilde{d}_p = \sum_{i=1}^{5} max(1, [\tilde{s}_w * \widehat{d}_{s_i}]) \tag{5.27}$$

where $\widehat{d}_{s_i}$ denotes predicted state-duration and square brackets denote the nearest integer function.

Furthermore, the duration model used is *statistically robust*: by training a multi-component bi-directional recurrent MDN, some components can be used to account for bad data (garbage components). By then synthesising from a single mixture component (e.g. the one with the largest mixture coefficient), datapoints that trained the other components – and the behaviours that led to those datapoints – are ignored in synthesis. The samples from these experiments can be accessed at this URL[3].

[3]Samples are available at `http://srikanthr.in/prosody-demo/duration`

5.7 Conclusions

This chapter described a new duration modelling paradigm with LSTMs which operates at frame-level for duration prediction in speech synthesis. Experiments were conducted on an audiobook data and the proposed systems were found to perform competitively when compared with a baseline phone-level LSTM system. The binary representation for duration used in this chapter has also shown promising results in predicting the long tail distribution of audiobook speech durations. We will discuss alternative representations of duration at frame-level and how this work can be extended to novel paradigms in the next chapter.

Chapter 6

Summary of investigations in Part-I

6.1 Discussion

In part-I of this thesis, the two important contributing factors of prosody, namely intonation and duration, were thoroughly investigated in the context of audiobooks. The initial investigations helped us in understanding the different causes for reduced quality in the statistical parametric speech synthesis (SPSS) framework. Based upon some of the findings, we proposed new approaches for better modelling of intonation and duration in chapters 4 and 5 respectively. This chapter summarizes the critical investigations from the proposed approaches and presents alternative methods for addressing some of the key problems. Also, the investigation of these causes will provide us with the focus of the improvements to be investigated in part-II of the thesis.

6.1.1 F0 reconstruction from templates degrades the quality

In chapter 4, we described an approach for speech synthesis intonation generation using syllable-level templates. We assumed that the use of syllable-level templates would overcome the smoothing problem (i.e., generally flat F0 across an utterance), which may happen if neural networks are trained to optimize mean F0 prediction. Although the use of syllable templates has the potential to overcome the over-smoothing problem (see Oracle model in Table 4.2) to some extent, it suffers from discontinuity between the templates and hence degrades the quality. In order to overcome this, we performed join-smoothing by predicting frame-level F0 (jointly with other acoustic features) given linguistic features and reconstructed F0 from syllable templates. Such a myopic approach has the demerits of both unit-selection and parametric synthesis and offers little room for improvement in naturalness. Also, the smaller number of templates for intonation reconstruction decreased the potential variation within each syllable and the use of fixed-number of DCT coefficients per each syllable for clustering lead to a small inventory of smoothed syllable templates.

There are several ways to improve upon the proposed approach of template reconstruction and some of them are outlined below:

- **Auxiliary input features:** An alternative approach is to use the probabilities of predicted templates as input features along with linguistic features in a conventional frame-level acoustic modelling. This approach thus avoids F0 reconstruction from templates as it uses prediction probabilities as additional input features. This may also provide us a way to variable prosodic synthesis by fine-tuning the probabilities.

- **Prosodic embeddings:** The aforementioned auxiliary features still require an additional network to train a classifier and predict the templates at run-time. The use of prediction probabilities as auxiliary features limits the overall performance of the acoustic model as these features largely depend upon the accuracy of the classifier. The model performance also restricts the number of templates to be used based on the amount of data used for training such a classifier. In order to overcome this, Ribeiro et al. (2017a) recently proposed a frequency distribution-based approach for learning word vector representations. The distribution matrix is computed by recording the co-occurrence of words and discretized acoustic features. The embeddings are derived from syllable and word-level F0 clusters by reducing the dimensionality with singular-value matrix decomposition and are used as auxiliary features in parametric synthesis framework. The results indicate that such learned representations are useful for generation of acoustic parameters in speech synthesis.

- **Autoregressive networks:** During training as well as at inference time, we provided F0 as input which was reconstructed from syllable templates for join-smoothing. The correlation between input and target F0 depends on the accuracy of the neural network-based template classifier. In a similar direction, Wang et al. (2017a) recently proposed an RNN-based quantized F0 model using autoregressive networks to provide frame-level correlation. For audiobook modelling, the prosodic embeddings proposed by Ribeiro et al. (2017a) may be used as additional input features and the acoustic model can be trained using autoregressive networks by feeding the oracle F0 at time $t-1$ as input. This makes the output prediction at time t dependent on previous predictions and thus enables continuous and smooth prediction of F0.

6.1.2 Suboptimal modelling of frame-level durations

In chapter 5, we described an approach for frame-level duration modelling by predicting a transition probability at each time frame. The transition probability is defined as 0 for all frames except for the final one in each phoneme, denoted by a probability of 1. The neural network was trained using minimum mean square error loss (L2 loss) and the prediction probabilities are used to compute the right tail probability of the phone duration distribution to yield median durations.

The advantage of such a model is that it doesn't assume any predetermined duration distribution and can effectively model the long-tail distribution by generating durations from the median quantile. However, the accuracy of predicted durations is still suboptimal. Below, we present alternative representations of frame-level duration models and some of the effective methods for modelling such representation.

- **Sigmoid cross-entropy loss:** The neural networks used in Chapter 5 were trained using L2 loss with a linear layer as final output layer. Since the output is represented with only zeros and ones, the model can be effectively trained using sigmoid cross-entropy loss which measures the error as in a discrete classification task. Shen et al. (2017) effectively used similar loss with a sigmoid activation to predict the probability of end of output sequence.

- **Continuous representation:** As an alternative to median-based estimation, we can explore the generation based on other quantiles such as mean to enable efficient control in rate of speaking. In this approach, we model duration as a continuous variable (see equation 6.1) varying slowly from 0 to 1 as shown in Figure 6.1 and hence named under the pseudonym of *ramp model*.

$$x_t = \begin{cases} 1 & \text{if } t = t_0 + d_p \\ \frac{t-t_0}{d_p} & \text{otherwise.} \end{cases} \tag{6.1}$$

However, there might be some problems with this representation. The fact that the ramp output value can decrease may introduce multimodal behaviour with our duration generation criterion, since the ramp output may first increase, but then temporarily show a decreasing trend later on. The generated duration will then differ substantially depending on whether or not the critical value was reached in the first peak of the ramp output.

- **Joint-modelling of duration and acoustic features:** The continuous representation also effectively allows us to perform joint-modelling of duration and acoustic features. The model can be trained using either L1 or L2 loss since the timescales of both duration and acoustic features are matched and represented on a continuous scale. We conducted a few experiments in this direction using L2 loss but the performance of acoustic features such as mel-cepstral distortion and F0 correlation was quite limited in the parametric synthesis framework as the acoustic model also relies on frame-level duration features. As for future work, the use of L1 loss with proper regularization may help us to circumvent the application of MLPG and also aligns with our motivation of using median-based duration prediction.

- **Hidden-state upsampling:** The linguistic features are upsampled at the input to predict the transition probability of each frame. Although LSTMs are capable of modelling hidden-state transition, an additional feature indicating the position of frame within phone was also used to induce frame-level correlation. This can be enhanced by replacing the upsampling at the input to one of the hidden layers. This allows effective modelling of phone-level linguistic features and is further investigated in Chapter 7 of this thesis.

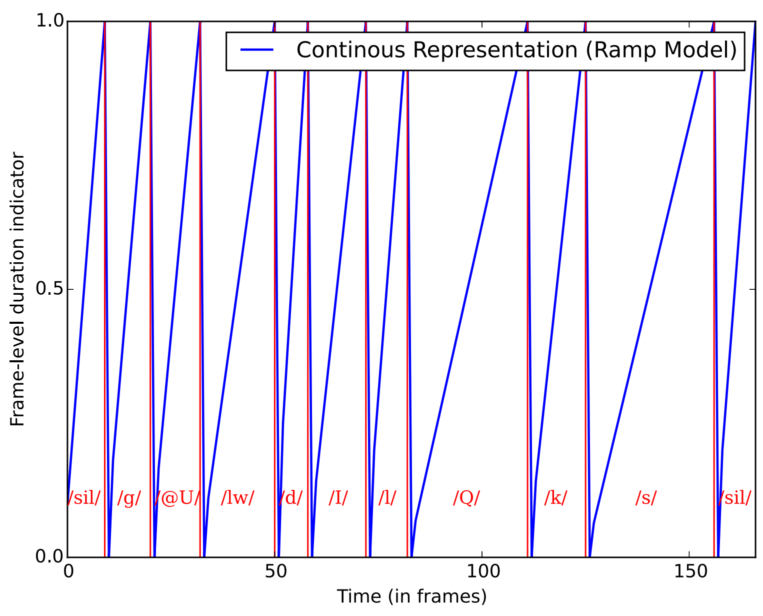


Figure 6.1: An example sentence showing the frame-level transition indicator (x_t) using continous representation. Red lines indicate phone level segmentation matching with frame-level transition.

- **Attention networks:** Considering recent approaches in neural speech synthesis (Wang et al., 2017c; Shen et al., 2017), the upsampling may be altogether replaced by using sequence-to-sequence networks. In these approaches, the duration is learnt implicitly along with acoustic features. However, a multi-prediction model of frame-level transition probability and acoustic features may allow explicit control of duration of individual phones. Also, the frame-level joint acoustic and duration model should allow iterative refinement of the data alignments.

6.2 Hypotheses for the work in Part II

Part I of the thesis has focused on two individual contributing factors of prosody within SPSS: intonation and duration. Given the findings from Part I of the thesis and by reviewing the literature on deep-learning based approaches for parametric synthesis (see Section 2.1.2), we present three propositions to mitigate some of the main shortcomings of SPSS:

- We hypothesize that upsampling linguistic features at the input makes

LSTMs less effective for modelling long-term context. This intuition was drawn based on the small incremental improvements presented in LSTM-based approaches (Zen and Sak, 2015; Wu and King, 2016, for example) and also from our results in Chapter 4.

- Given that linguistic features are of different time-scales, they are quite ineffective when used at frame-level, as the LSTM processes the same information for several frames. The use of multiple encoders at different time-scales to handle these linguistic features may improve the overall performance of SPSS. This intuition was drawn based on the work presented by Ribeiro et al. (2016) and also from our results in Chapter 7.
- The synthesis quality of audiobooks using parametric models is quite limited due to the effect of vocoding. Therefore, we hypothesize that the use of either parametric-guided unit selection or neural waveform generation are some of the possible ways to address the problems with vocoding. The intuition for using hybrid unit selection was drawn based on the results presented in Merritt et al. (2016b) for Blizzard Challenge 2016.

The main aim in Part II of the thesis is to present the motivation for these propositions and a clear understanding of how these approaches can be used to effectively model the prosody in children's storybooks, while building complete text-to-speech systems.

Part II

Improving prosody in parametric speech synthesis

Chapter 7

A Hierarchical Encoder-Decoder Model for Statistical Parametric Speech Synthesis

This chapter is an expanded version of the work in Ronanki et al. (2017b) and therefore the text is closely related to that.

This work was completed in collaboration with others. Discussion of ideas was between myself, Oliver Watts and Simon King. The code for the hierarchical encoder-decoder model was implemented by myself and I provided the code and a recipe to enable this system to be reproduced using the Merlin toolkit. Fine-tuning of the systems was then done by myself with the help of Oliver Watts.

7.1 Introduction

Statistical parametric speech synthesis (SPSS) continues to make significant improvements every year, yet is rarely convincingly natural (Zen et al., 2009; King, 2011). The absence of variation in the predicted intonation makes synthetic speech dull and therefore inappropriate for audiobooks. We introduced novel approaches for modelling F0 at syllable-level and duration at frame-level in Chapters 4 and 5. Although the intonation modelling introduced in Chapter 4 is promising, the synthesized speech suffered from poor segmental quality. Therefore, in this chapter, we continue to focus on intonation modelling at frame-level within SPSS.

The conventional approaches to statistical parametric speech synthesis using Neural Networks (Zen et al., 2013; Qian et al., 2014; Wu et al., 2015b) generally require input at the same temporal resolution as the output, typically a frame every 5ms, or in some cases at waveform sampling rate (van den Oord et al., 2016). It is therefore necessary to fabricate highly-redundant frame-level (or sample-level) linguistic features at the input. This chapter proposes the use of a hierarchical encoder-decoder model to perform the sequence-to-sequence regression in a way that takes the input linguistic features at their original timescales, and preserves the relationships between words, syllables and phones. The proposed model is designed to make more effective use of supra-segmental features than conventional architectures, as well as being computationally efficient.

7.1.1 Relation to prior work

As discussed in Section 2.1.2, Neural networks are rapidly coming to dominate SPSS, for both duration and acoustic modelling (including F0). In the work by Zen et al. (2013), a Deep Neural Network (DNN) was used as a fairly straightforward drop-in replacement for the usual regression tree of HMM-based SPSS, performing the complex mapping (i.e., regression) from linguistic features to speech parameters on a frame-by-frame basis. Later, others (Qian et al., 2014; Wu et al., 2015b, for example) explored variants on this basic idea. Given that speech is a time series, and SPSS is a sequence-to-sequence regression problem, it was then natural to apply various recurrent architectures, including Long Short-Term Memory (LSTM) units and Gated Recurrent Units (GRUs) (Fan et al., 2014; Zen and Sak, 2015; Wu and King, 2016; Zen et al., 2016, for example), which we can group under the general category of Recurrent Neural Networks (RNNs). Nevertheless, these approaches still all require input at the same frame-rate as the output, which is determined by the needs of the vocoder.

Focusing on intonation, DNNs offer improvements over regression trees (Watts et al., 2015b; Ribeiro et al., 2016) and RNNs improve over DNNs (Fernandez et al., 2014; Wang et al., 2017b). It seems likely that long-range dependencies are of greater importance in predicting intonation than when predicting the spectral envelope. By “long-range” we mean over sequences of linguistic units such as syllables, words and phrases. This suggests the use of a hierarchical model that directly exploits such linguistic structure, instead of merely "flattening" it, first on to the segment (phone) and then further to the acoustic frame.

We identify two potential limitations of the usual approach taken in prior work (Zen et al., 2013; Wu et al., 2015b). First, the use of frame-level inputs is highly redundant because the majority of input features remain constant for 10-100 consecutive frames, incurring unnecessary computations both in training and when performing synthesis. Second, representing supra segmental features associated with syllables, words or phrases, at segmental or sub-segmental timescales may actually hinder the model from learning to use such features (Wang et al., 2015).

There are other recent attempts at hierarchical modelling, although only of F0 contours. Yin et al. (2016) proposes two different model structures – cascade and parallel DNNs – to embody hierarchical and additive properties. Ribeiro et al. (2016) also proposed parallel and cascaded DNNs to model supra-segmental and segmental features separately. Both approaches use multiple networks, incurring additional computational cost compared to non-hierarchical approaches.

7.1.2 Novelty of this work

We present a hierarchical framework which exploits linguistic structure in order to model long-term dependencies, whilst at the same time being computationally efficient.

New techniques have recently emerged for mapping between two sequences of arbitrary lengths and potentially unknown alignment, including sequence-to-sequence neural networks (Sutskever et al., 2014), attention-based models (Chan et al., 2016), and sequence transduction networks (Graves, 2012) that we investigate in this work. We combine a hierarchical *encoder* to model linguistic structure at multiple time-scales, with a *decoder* that predicts speech parameters for each output acoustic frame.

Through experiments, we first compare our proposed hierarchical encoder-decoder (HED) architecture with the usual approach. We then examine the effectiveness of additional supra-segmental word-level features, when added to a standard set of linguistic features, comparing conventional frame-level combination with their use directly at the word level in our proposed hierarchical encoder.

The rest of the chapter is organized as follows: Section 7.2 describes the usual approach for frame-level acoustic modelling within SPSS. Section 7.3 presents a vanilla encoder-decoder architecture adapted for parametric speech synthesis. The proposed approach for hierarchical encoding of linguistic features is introduced in Section 7.4. Section 7.5 presents the experimental setup and the systems trained for evaluation of usual approach, vanilla encoder-decoder and hierarchical encoder-decoder. The experimental evaluation and results are interpreted in Section 7.6 and finally, Section 7.7 presents the conclusions.

7.2 The Usual Approach

This section describes the procedure for the conventional approach in SPSS i.e., upsampling phone, syllable and word-level linguistic features to frame-level at the input and performing regression using neural networks. This approach is used to train a baseline model for experiments presented in Section 7.5.

7.2.1 Pre-processing input features

The alignments between words, syllables, and phones are given by the linguistic specification, provided by the TTS front end. The usual approach in neural network-based TTS – regardless of neural network architecture – is to pre-process the input representation by **flattening** followed by **upsampling** (Zen et al., 2013; Wu et al., 2015b).

Flattening: This step is performed by attaching linguistic features to the phone, and thereby creating a linear sequence of context-dependent phones, and discarding explicit structure (e.g., that phones belong to syllables).

Upsampling: This is done by duplicating linguistic features for a number of consecutive acoustic frames, to map from linguistic timescale to vocoder frame rate (or possibly to waveform sampling rate, if directly generating a waveform). Note that upsampling cannot *add* information; in fact, it results in the same amount of information being represented less efficiently.

It is common practice to add within-phone positional features, derived from existing features, when upsampling, to compensate for limitations of the regression model. For the work presented in this chapter, we only considered phone-level alignment (discarding HMM states) and thus used only 4 numerical features as within-phone positional features (three numerical features for coarse coded position and one for phoneme duration). Features like position of frame within each state (forward and backward), state-duration are not considered. This is consistent with our longer-term goal of freeing ourselves from depending on often arbitrary HMM sub-phone alignments.

7.2.2 Input-to-output alignment in the usual approach

By making an assumption that is almost universal in speech technology – viz. that a speech signal is a sequence of non-overlapping units – the input-output alignment can be pre-computed using HMM-based forced alignment for the training data, and can be determined during synthesis of test utterances using a duration model learned from the same data.

7.2.3 Regression

There are many possibilities for the architecture of the network used to perform regression from flattened-and-upsampled linguistic features to either frame-level vocoder speech parameters (Zen et al., 2013; Wu and King, 2016; Watts et al., 2015b, for example) or to waveform samples (van den Oord et al., 2016). Nevertheless, what all of these architectures have in common is the requirement for input and output to be at the same rate, meaning that the input must be upsampled. We considered both feed-forward neural networks and simplified LSTM networks to perform regression for the baseline experiments presented in Section 7.5.

7.3 Vanilla Encoder-Decoder

In this section, we'll present the key modifications performed to adapt a vanilla encoder-decoder architecture for speech synthesis. Figure 7.1 provides a schematic diagram of the basic vanilla encoder-decoder neural network for SPSS. The key idea is to integrate the frame-level upsampling into the model itself. There are three components of this architecture: 1) Vanilla encoder, 2) Input-to-output alignment and 3) Recurrent decoder. We'll discuss these components in each of the following sub-sections.

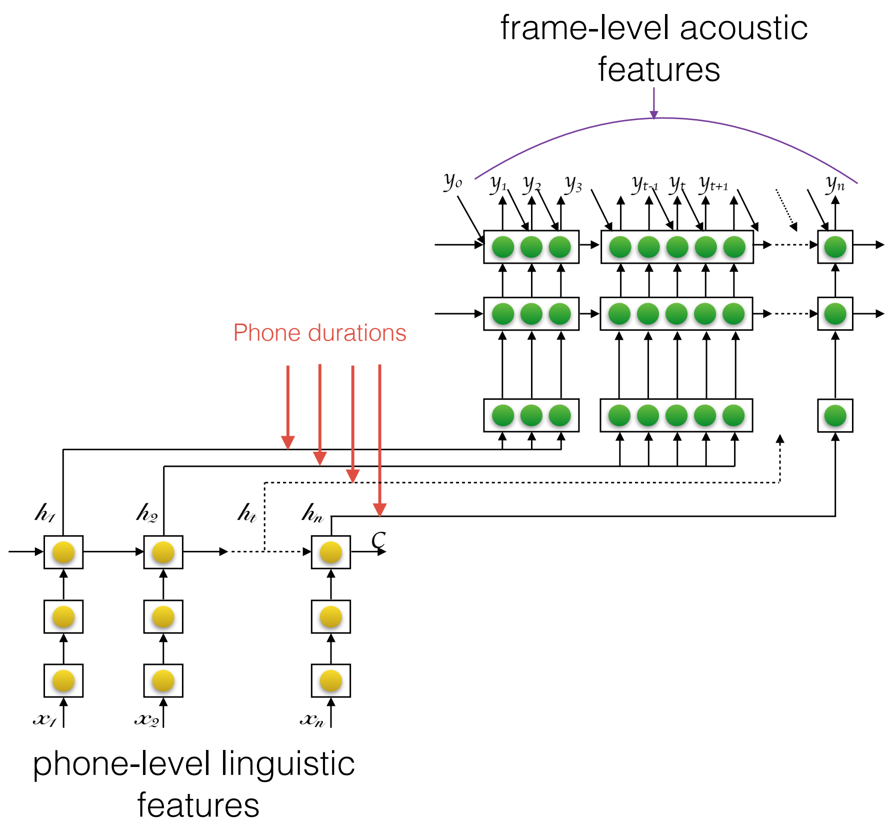


Figure 7.1: Schematic diagram of the simplest form of encoder-decoder for SPSS. The lower part of the network is the phoneme encoder and the upper part of the network is the recurrent decoder, generating speech parameters using frame-level recurrence.

7.3.1 Vanilla encoder

Traditionally, in machine translation (Sutskever et al., 2014; Cho et al., 2014), an encoder-decoder architecture processes word-level inputs from a source language and outputs words in a target language. The task of speech synthesis requires output to be modelled at a very small temporal resolution, typically either at a frame-level or sample-level, in order to generate high-quality speech. However, decoding frame-level acoustic information from only word-level or syllable-level linguistic inputs might be a very challenging and arduous task for an encoder-decoder architecture. For this reason, we append the upsampled syllable and word-level features to phoneme linguistic features as in the usual approach for duration modelling.

The input to the vanilla encoder is a sequence of phone-level linguistic features as shown in Figure 7.1 (represented in yellow). The vanilla encoder processes the input linguistic features through a series of hidden layers which apply a set of non-linear transformations. Since the input linguistic features contains both binary and numerical features (see Appendix A), we considered first few hidden layers to be feed-forward with non-linear activation (typically TANH) to embed into a continuous vector. The top most layer of the vanilla encoder is configured with a recurrent layer in order to extract sequential features.

To test the effectiveness of linguistic features extracted at different timescales, the experiments presented in Section 7.5 also considered decoding frame-level acoustic information from only syllable or word-level features. However, irrespective of the nature of input and length of sequence, the architecture of the encoder remains same for all the experiments.

7.3.2 Input-to-output alignment

In machine translation, where encoder-decoder architectures are more commonly used (Sutskever et al., 2014; Cho et al., 2014), alignment between input and output sequences is non-trivial: it may be non-monotonic because of word re-ordering. The situation in speech synthesis is more straightforward because the alignment is strictly monotonic. However, unlike machine translation, in speech synthesis the input and output sequences are at different timescales.

In its simplest form, the encoder can be fed with a sequence of upsampled phone-level linguistic features and the decoder can be used to predict frame-level acoustic features. In the work presented here, we employ methods from the usual approach for neural network-based TTS (Section 7.2.2): forced alignment during training, and externally-provided oracle durations for synthesis. This is done to measure the contribution of our proposed architecture independently of issues of duration modelling. The hidden state of each phoneme from the vanilla encoder is simply broadcasted to multiple frames based on its duration as shown in Figure 7.1.

Integration of alignment (during training) (Wang et al., 2016) or duration prediction (for synthesis) are possible extensions in future work, and the proposed encoder-decoder architecture may offer ways to do this that are not available in the usual approach. We'll discuss these extensions in Chapter 8.

7.3.3 Recurrent decoder

The output from the vanilla encoder is at phonetic timescale (h_t in Figure 7.1). Using externally-provided duration information, this is upsampled to the acoustic frame rate used by the vocoder.

The input to the decoder is a sequence of upsampled phonetic encoder state outputs (represented in green blocks) and is processed by a series of recurrent layers. Each block of green units represents a phone, with the number of units corresponding to its duration in frames (although not drawn to scale). The recurrent state in each of these recurrent layers is generally initialized as zero vector. We also tried final encoder representation (h_n, also known as context vector C) as initialization for recurrent layers and found no improvements over default initialization. This may be because of monotonic alignments between encoder and decoder.

When training the recurrent decoder, past output (y_{t-1} in Figure 7.1) can either be ground-truth from the training data (a method known as "teacher forcing" (Pascanu et al., 2013)) or the prediction of the network itself. However, the experiments tested in this chapter used prediction of the network during training as well as inference. The non-recurrent output y_0 is also initialized as the zero vector. We provide the complete details of the system architecture (number of layers, nodes etc.,) used for training in Section 7.5.

7.4 Hierarchical Encoder-Decoder

In this section, we improve upon our vanilla encoder-decoder architecture and present hierarchical encoding of linguistic features. Figure 7.2 provides a schematic diagram of the proposed hierarchical encoder-decoder neural network. The key ideas are to avoid the flattening pre-processing step entirely, and to integrate the upsampling into the model itself. This means even the syllable and word-level linguistic features are not duplicated at the input. Similar to Section 7.3, we'll discuss each component of the encoder-decoder architecture and their differences in the following subsections.

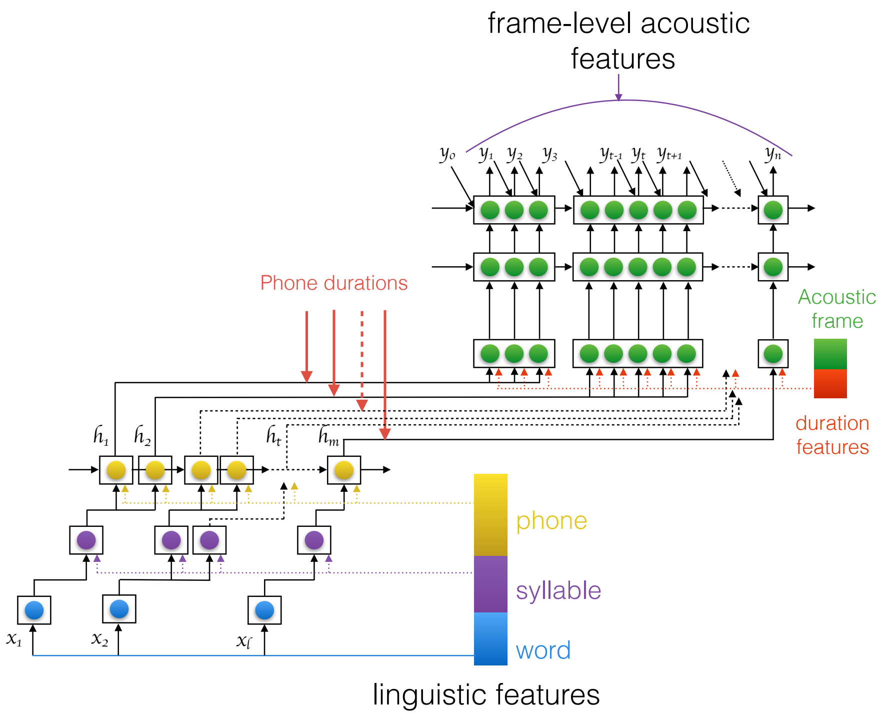


Figure 7.2: Schematic diagram of a hierarchical encoder-decoder for SPSS. The lower part of the network is the hierarchical encoder, with each layer operating at a particular linguistic level, and phone-level recurrence as its final encoded output. The upper part of the network is the decoder, generating speech parameters using frame-level recurrence.

7.4.1 Hierarchical encoder

The goal of hierarchical encoder is to process each of the phone, syllable and word-level linguistic features at their natural timescales and integrate the up-sampling while performing forward propagation. As shown in Figure 7.2, each level of linguistic features are color coded, with blue representing linguistic features extracted for each word, violet representing linguistic features extracted at syllable-level and yellow representing phone-level linguistic features. Solid black lines indicate the propagation of hidden activations between layers, and dashed colored lines indicate the injection of linguistic features at the appropriate level.

The input to the hierarchical encoder is a sequence of word-level linguistic features ($\boldsymbol{l}_w \times d_w$), where $\boldsymbol{l}_w$ is the length of word sequence and d_w is the dimension of word-level features. The linguistic features extracted at word-level and at other higher levels (phrase-level and utterance-level, but duplicated for each word) are concatenated to form a 53-dimensional word vector (see Appendix A). These word-level features are processed using a couple of feed-forward layers with TANH activation. The output from second hidden layer is broadcast-concatenated with $\boldsymbol{l}_s \times d_s$-dimensional syllable-level linguistic features, where $\boldsymbol{l}_s$ is the length of syllable sequence and d_s is the dimension of syllable-level features. The concatenated syllable-level features are further processed using another two feed-forward layers. Similar to word-level features, the processed syllable-level features are broadcast-concatenated with $\boldsymbol{l}_p \times d_p$-dimensional phone-level linguistic features, where $\boldsymbol{l}_P$ is the length of phone sequence and d_P is the dimension of phone-level features. These phone-level features are further processed using a feed-forward layer and a final recurrent layer to extract sequential features, as in vanilla encoder. The sequence mapping and the patterns of connections (shown in Figure 7.2) between word, syllable and phone layers is determined by the known linguistic structure of the current utterance.

Figure 7.2 shows how the proposed method employs a hierarchical encoder that accepts input at the original linguistic timescales of word, syllable and phone. Unlike the vanilla encoder, the upsampling between these levels is performed progressively, rather than all at once. Features at each level are injected into the model at the appropriate timescale by appending them to the hidden representation at that level. This has a variety of possible advantages over the usual approach.

Features from longer timescales (e.g., word), have a potentially weakened or diluted effect in the usual approach because they are constant for many consecutive frames, yet must share the same projections (hidden layers) as features that are locally more predictive of the output. In the proposed encoder architecture, there is a hierarchy of projections (layers) that introduces information from one linguistic timescale at a time.

The proposed architecture substantially reduces the number of inputs because the upsampling takes places within the model architecture rather than the pre-processing of features in the usual approach. This reduces the number of model parameters and consequently the computational cost.

7.4.2 Input-to-output alignment and Recurrent decoder

The input-to-output alignment is very much similar to the vanilla encoder-decoder. For each utterance, the d_h-dimensional hidden output from hierarchical encoder is simply broadcast-concatenated with $\boldsymbol{l}_f \times d_f$-dimensional frame-level features derived from phone duration (also known as within-phone positional features and are represented in red color), where $\boldsymbol{l}_f$ is the length of output sequence and d_f is the dimension of duration features.

Similar to the vanilla decoder, the $d_h + d_f$-dimensional sequence of upsampled encoder outputs are processed by a series of recurrent layers, with a final recurrent layer generating a sequence of frame-level acoustic features. Both vanilla and hierarchical encoder-decoder networks are trained in an end-to-end fashion with the data described in Section 7.5.1 and no pre-training is applied to any of the individual components. Frame-level error is computed and accumulated for each utterance using L2 loss. This loss is then backpropagated to compute the gradients and update the weights in each of the hidden layers of the decoder and encoder.

7.4.3 Additional word-level features

In addition to the usual linguistic and positional features, other features can be derived from text that may improve the naturalness of generated speech. Embeddings (Turian et al., 2010) are one of the most promising and generally-applicable ways to derive new features from text. For example, bottleneck features (Wu et al., 2015b) (a kind of supervised embedding) work well for TTS.

Unsupervised word embeddings (Wang et al., 2015), learned only from text, are another promising idea. Unlike bottleneck features, which are extracted at frame level, word embeddings are derived for each word. As discussed earlier, upsampling word-level features to the vocoder frame rate is rather unsatisfactory and can be ineffective for learning. In the proposed hierarchical encoder-decoder architecture it is simple to input these features directly at the word level, without upsampling.

7.5 Experimental Setup

In this section, we present the data used for training the proposed encoder-decoder architectures and the procedure for feature extraction in detailed. We also present further details on number of layers and hidden nodes used for each of the proposed architectures and other details of the system training.

7.5.1 Data

We tested our proposed method on the speech database released for the 2016 Blizzard Challenge (King and Karaiskos, 2016) consisting of the written and spoken versions of 50 children's books. The details of the data and the split for train-validation-test was provided in Section 3.1.

7.5.2 Feature extraction

Phone sequences were obtained from the text using Festival and each phone was then characterised by a vector of 481 text-derived binary and numerical features – a subset of the features used as decision-tree clustering questions in the HTS demo (Zen et al., 2007), adapted for our phoneset. The procedure for extraction of linguistic features is detailed in Section 3.3.1.

The questions included linguistic contexts such as quinphone identity which are added at phone-level, and part-of-speech, positional information relating to syllables, words, phrases, *etc.* which are given as input at syllable and word-level. All numerical features are input (after appropriate normalisation) directly to the network, and not encoded as (for example) 1-of-K. Similar to Zen et al. (2013), three numerical features for coarse-coded position of the current frame in the current phoneme and duration are computed and appended as frame-level additional features. All inputs were normalised to the range $[0.01, 0.99]$.

Word embeddings and word-quotation features were used as additional word-level features in some of the systems (with *WF* in the system name). We used 300-dimensional word embeddings[1][2] obtained as described in Pennington et al. (2014). We also added a word-quotation feature to words within

[1]We made use of the embeddings from 6B tokens made available at Glove

[2]`https://nlp.stanford.edu/projects/glove`

double quotes; this generally indicates direct speech vs. narration, in the data we used.

The speech data was analysed with STRAIGHT (Kawahara, 2006), and each 5ms frame was represented using 60 mel cepstral coefficients (MCC), measures of aperiodicity in 25 frequency bands (BAP), logarithmic F0 interpolated through unvoiced regions, and a binary voicing feature. These 87 static features were supplemented with delta and delta-delta features, and per-component mean and variance normalisation was performed. The procedure for extraction of these acoustic features is detailed in Section 3.3.2.

7.5.3 System training

We performed multiple experiments mainly based on the approaches described in Section 7.2, Section 7.3 and Section 7.4. We group them into three systems and we discuss several variants tested for optimizing each of these architectures.

Systems trained based on the usual approach

Four variant systems were trained based on the approach described in Section 7.2 to establish a strong baseline system. The baseline system is trained in a conventional way and is typical of the usual method as described in several other works (Zen et al., 2013; Wu et al., 2015b, for example). We tested two types of architecture: feed-forward and recurrent neural networks. For feed-forward networks, we trained two models, one with four layers of 512 nodes and the other is configured with six layers of 1024 nodes. Two deep uni-directional LSTMs were also trained for testing recurrent neural networks: the first one was configured with three feed-forward layers of 1024 nodes and two uni-directional simplified long short-term memory (SLSTM) (Wu and King, 2016) hidden layers consisting of 512 nodes and the other system was configured with two feed-forward and three SLSTM hidden layers. The first two layers of the recurrent systems were configured with feed-forward layers in order to embed both categorical and numerical features onto continuous scale. The architecture details of these various baseline models tested are also listed in the 2nd column of Table 7.1.

Systems trained using vanilla encoder-decoder

To evaluate the effect of various linguistic features for $F0$ modelling, five variant systems were trained with a vanilla encoder-decoder architecture (*ED-LSTM*) without any model hierarchy as described in Section 7.3 and as shown in Figure 7.1. The input features tested were word-level, syllable-level and phone-level linguistic features. The various input features (along with their dimension in brackets) for systems trained with *ED-LSTM* are listed in the 2nd column of Table 7.2. For word-level system, both categorical and numerical features of word and other higher level linguistic features (see Appendix A) were concatenated to form word-level input features. For syllable input features, two systems were prepared: one system with only syllable-level numerical and categorical features; for the second system, an additional syllable identity with 300-dimensional vector was added which consists of 1-of-K phonemes with syllable nucleus at center. For phoneme input features, all 481 features (both categorical and numerical) were upsampled to phoneme level; an additional system was also prepared at phoneme-level with upsampled word features (as explained in Section 7.5.2) added as auxiliary features.

For the systems trained with *ED-LSTM*, the input features were fed through an encoder configured with two feed-forward layers and a final uni-directional SLSTM layer. The upsampled hidden features from encoder were concatenated to within-phone positional features and passed through the decoder. As shown in Figure 7.1, the decoder has one hidden recurrent layer and an output layer whose predicted output at time t is given as additional input for predicting frame $t+1$. Both layers have 512 uni-directional SLSTM units.

Systems trained using hierarchical encoder-decoder

For final evaluation, four systems were tested: identifiers are listed in the left-hand column of Table 7.3. We use one of the best performing baseline variants with recurrent layers from Table 7.1 and name it as *Frame-LSTM* from here onwards – which provide frame-rate recurrence – and a final linear output layer. It is trained to regress directly from a sequence of flattened-and-upsampled frame-level linguistic feature vectors to a sequence of vocoder speech parameters.

HED-LSTM is a hierarchical encoder-decoder which implements our proposed idea. The encoder was configured with five feed-forward layers of 1024 nodes each and a uni-directional SLSTM. Similar to *ED-LSTM*, the decoder is configured with two recurrent layers with each layer consisting of 512 SLSTM units.

Two variant systems were built to test whether the proposed hierarchical encoder can better exploit word-level features than the usual approach. *Frame-LSTM+WF* and *HED-LSTM+WF* are identical to *Frame-LSTM* and *HED-LSTM* respectively, except that distributional word representations were added (Section 7.5.2). In *Frame-LSTM+WF* these were added to every frame of input, whereas in the hierarchical encoder-decoder *HED-LSTM+WF*, they were added at the word level, with other word-level linguistic and positional features.

All networks were initialised using small random weights; no pre-training was used. Each system was trained with a fixed learning rate of 0.001 and a batch size of 1, manually tuned to yield close-to-optimal results on the development set in 25 epochs or less. Early stopping was used to avoid overfitting, by aborting training once the objective function on the development set had failed to improve for five consecutive epochs.

7.5.4 Synthesis

At synthesis time, `ehmm` phone sequences derived from the test data were used as input to each model. This corresponds to an oracle pausing strategy. For all systems, natural durations were used during testing, since we are interested only in regression from linguistic features to speech parameters. In the proposed hierarchical encoder-decoder systems, durations are used when upsampling from phones to acoustic frames at the interface between encoder and decoder. Replacing oracle durations with predictions by an external duration model would be trivial.

In all systems, maximum likelihood parameter generation (MLPG) (Tokuda et al., 2000) using variances computed from the training data was applied to output features; spectral enhancement post-filtering was applied to the resulting MCC trajectories (Wu et al., 2015b). The STRAIGHT vocoder (Kawahara, 2006) was used to synthesize the waveform which was then normalised according to ITU P.56 (2011).

7.6 Results

7.6.1 Objective measures

We objectively compared the speech parameters generated by each system for the test set against held out natural examples. We calculated Mel cepstral distortion (MCD), band aperiodicity distortion (BAPD), and root mean square error of F0 (RMSE), Pearson correlation of F0 (Corr.) and voiced/unvoiced error rate (V/UV). The equations for computing these measures are given in Section 2.3.1.

We first conducted objective evaluations to realize the best performing baseline system among all system variants trained based on the usual approach. The results for the baseline system with various types of architecture tested are presented in Table 7.1. For comparison, the MCD from a feed-forward model with 6 layers is slightly lower than the model trained with 4 layers and, as also expected, the MCD of recurrent models is marginally lower compared to feed-forward models. Although the MCD of baseline system improved slightly with the increase in number of recurrent layers, it also made the system computationally inefficient. The difference in F0 RMSE and correlation between all four tested systems is quite minimal. Based on these objective results, we speculate that the data for training deep recurrent neural networks may not be sufficient.

The results (particularly F0 RMSE) from Table 7.1 are a little worse than the results presented in Table 4.2 from Chapter 4. This is for two reasons: first, the baseline systems presented in Table 7.1 are trained without any state-level information; second, the F0 values in Chapter 4 were derived using STRAIGHT with a much lower range (max. F0 value set to 300 Hz) and were not replaced with a more accurate F0 (max. F0 value was allowed till 400 Hz accounting for character imitations and other creaky sounds) extracted from GlottalHMM toolkit. Therefore, correlation numbers are more suitable for drawing any comparisons.

After establishing a best-performing baseline (last row from Table 7.1 based on lowest MCD), we now assess the performance of vanilla encoder-decoder system with various input linguistic features. We tested the effectiveness of word, syllable and phoneme-level linguistic features using the vanilla encoder-

System	Model Type	Spectral		F0 Measure		
		MCD	BAPD	RMSE	Corr.	V/UV
Baseline	4*TANH	5.518	0.075	50.988	0.430	5.594%
	6*TANH	5.505	0.075	50.829	0.430	5.552%
	3*TANH + 2*LSTM	5.467	0.075	50.834	0.432	5.437%
	2*TANH + 3*LSTM	5.438	0.075	51.847	0.432	5.448%

Table 7.1: Objective results for baseline system with feed-forward and recurrent architectures.

System	Input Features	Spectral		F0 Measure		
		MCD	BAPD	RMSE	Corr.	V/UV
ED-LSTM	Word (53)	9.155	0.094	54.016	0.338	26.005%
	Syllable (131)	8.378	0.090	52.510	0.385	21.048%
	Syllable + syl-identity (431)	7.168	0.082	52.431	0.366	11.269%
	Phoneme (481)	5.513	0.075	51.012	0.440	5.602%
	Phoneme + WF (782)	5.588	0.075	49.942	0.463	5.581%

Table 7.2: Objective results for vanilla encoder-decoder system with several variants of input linguistic features.

decoder architecture and the results are shown in Table 7.2. The systems that are trained only with word-level or syllable-level features performed very poorly in both spectral and F0 measures. This is however expected, as the input features doesn't contain any segmental information and the network is unable to decode frame-level acoustic information from such higher-level linguistic features. The experiments are rather performed to test the effectiveness of each individual feature for F0 prediction. Comparing syllable and word-level features, the systems trained with syllable features provided higher F0 correlation. However when syllable identity (concatenated 1-hot-k phoneme vectors) is added, the performance of F0 prediction didn't improve. From this, we interpret that the use of linguistic information at its natural timescale may be useful for better prediction of F0.

The systems with phone-level input features (last two rows from Table 7.2) provided the best numbers in F0 correlation among all the variants tested for vanilla encoder-decoder architecture. The F0 correlation for the phoneme

	Spectral		**F0 Measure**		
System	**MCD**	**BAPD**	**RMSE**	**Corr.**	**V/UV**
Frame-LSTM	5.44	0.075	51.85	0.432	5.49%
HED-LSTM	5.48	0.074	50.20	0.453	5.48%
Frame-LSTM + WF	5.50	0.075	50.22	0.454	5.44%
HED-LSTM + WF	5.53	0.075	49.68	0.473	5.47%

Table 7.3: Objective results for best-performing baseline model from Table 7.1 and hierarchical encoder-decoder system and their variants with auxiliary features.

encoder-decoder (Corr. = 0.440) was slightly higher than the baseline system (Corr. = 0.432) from Table 7.1. Although syllable and word-level features are upsampled to phone-level, the marginal improvement in F0 correlation over baseline indicate that modelling linguistic features at phone-level may be useful compared to upsampled frame-level linguistic features. Also, the improvements in F0 RMSE and correlation (Corr. = 0.463) when adding word features (word-embeddings and word quotation features) shows the effectiveness of vanilla encoder-decoder architecture (*ED-LSTM*, last row from Table 7.2) in processing higher-level features.

The results for the baseline model (last row from Table 7.1, renamed as *Frame-LSTM*) and hierarchical encoder-decoder systems are shown in Table 7.3. Although *HED-LSTM* slightly increases MCD from 5.44 dB to 5.48 dB compared to *Frame-LSTM*, it makes more accurate predictions of F0: RMSE dropped from 51.85 Hz to 50.20 Hz and Correlation increased from 0.432 to 0.453. When word-level features are added, both frame-level and hierarchical systems' F0 predictions improved, while the MCD deteriorated slightly.

Usefully, *HED-LSTM* is computationally cheap, both in training and generation, compared to *Frame-LSTM*. We computed generation time for both the systems, which is the total time to generate all the 387 utterances in both development and testing sets. The generation time for *Frame-LSTM* with two feedforward layers and three recurrent layers is 1910 seconds while *HED-LSTM* with the same architecture took 1566 seconds.

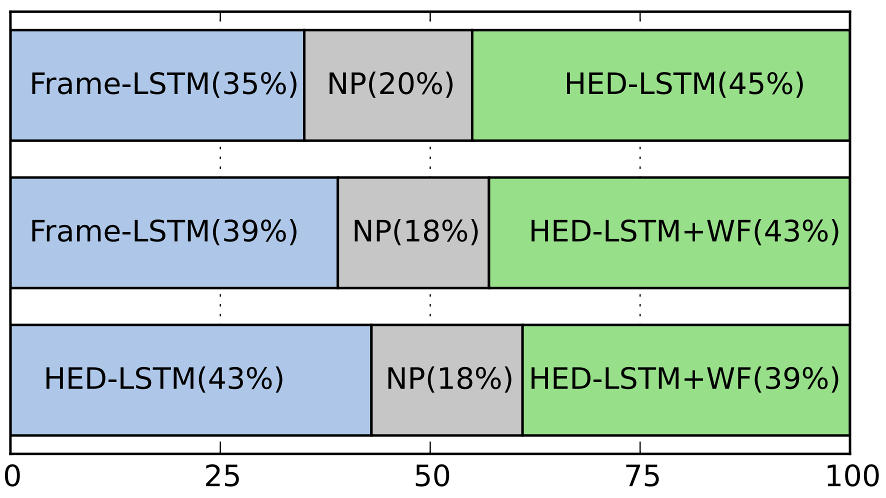


Figure 7.3: Preference test results for naturalness with percentage in brackets, where NP denotes "no preference"

7.6.2 Subjective evaluation

We assessed the naturalness of the synthesised speech via a pairwise preference listening test. Three pairs of systems were considered: *Frame-LSTM* vs. *HED-LSTM*, *Frame-LSTM* vs. *HED-LSTM+WF*, and *HED-LSTM* vs. *HED-LSTM+WF*. 16 paid native English speakers with no known hearing impairments participated in the listening test. Each listener was asked to listen to 75 randomly selected pairs. Thus each condition received a total of 400 judgements(16*25). Listeners were told that the synthesized audio files were from children's story books. Within each pair, the listener was played two synthesised versions of the same sentence and asked to choose which one sounded more natural. They were asked to focus on prosody (tune) and only to select the "no preference" option when both sounded the same. Some samples from the listening test can be accessed at this URL[3].

Results of the preference test are presented in Figure 7.3. First, examine the effectiveness of the proposed hierarchical encoder-decoder: *HED-LSTM* sounds significantly more natural than *Frame-LSTM* with $\alpha = 0.05$ under a binomial test with an expected 50% split. *HED-LSTM+WF* is slightly, but not significantly preferred over *Frame-LSTM* and similarly, *HED-LSTM* is slightly preferred over *HED-LSTM+WF* but not significant. Based on the results from objective and subjective evaluation and also through informal listening, we

[3]Samples are available at: http://srikanthr.in/prosody-demo/intonation_s2s

interpret that although there is an improvement in F0 prediction for the hierarchical system (higher F0 correlation numbers but not statistically significant), it is masked by the degradation in prediction of spectral features (also, higher MCD numbers for hierarchical models).

Based on recent work in the literature (as detailed in Section 2.1.3), we make the following speculations: with each linguistic feature on its own time-scale, the depth of the network significantly increases compared to the usual approach. This results in a *vanishing gradient problem* and the network weights from initial layers such as word and syllable-level may receive very small updates in each iteration of training. This prevents the network from further training by preventing any change in the weights of these layers. One possible solution is to train these linguistic features independently through multiple parallel encoders. This approach is further investigated in next chapter.

7.7 Conclusions

We have described a hierarchical encoder-decoder to improve intonation modelling in an effective and computationally efficient way. Unlike the usual approach which requires linguistic feature flattening and upsampling our proposed encoder accepts linguistic features at their natural timescales, and performs upsampling within the model architecture.

Experiments were conducted on prosodically-varied audiobook material because the use of supra-segmental features is thought to be particularly important in this case. Objective results show an improvement in F0 and the results from subjective listening tests, which asked listeners to focus on prosody, show that the proposed method performs significantly better than a conventional architecture that requires the linguistic input to be at the acoustic frame rate.

Chapter 8

Multi-scale parallel encoder for parametric synthesis

This chapter covers preliminary work on a multi-scale parallel encoder for statistical parametric speech synthesis (SPSS).

This work was completed in collaboration with others. Discussion of ideas was between myself, Sam Ribeiro, Oliver Watts and Simon King. The code for the multi-scale parallel encoder was implemented by myself and made open-source within the Merlin toolkit.

8.1 Introduction

In Chapter 7, we discussed a hierarchical encoder which replaced linguistic feature flattening and input upsampling in the conventional approach for SPSS. The hierarchical architecture brought an improvement in intonation modelling and was also shown to be computationally efficient. The work in this chapter aims at further improvements to intonation modelling for audiobooks with another novel encoder.

The hierarchical encoder presented in Chapter 7 consists of multiple levels of hierarchy with each level of features feeding into subsequent lower levels. The hierarchical encoder is also quite deep because of its hierarchy of projections (layers) that introduces information from one linguistic timescale at a time. As a result, features from longer timescales (e.g., word), may have a potentially weakened or diluted effect in the hierarchical encoder approach. We therefore introduce a multi-scale parallel encoder consisting of multiple par-

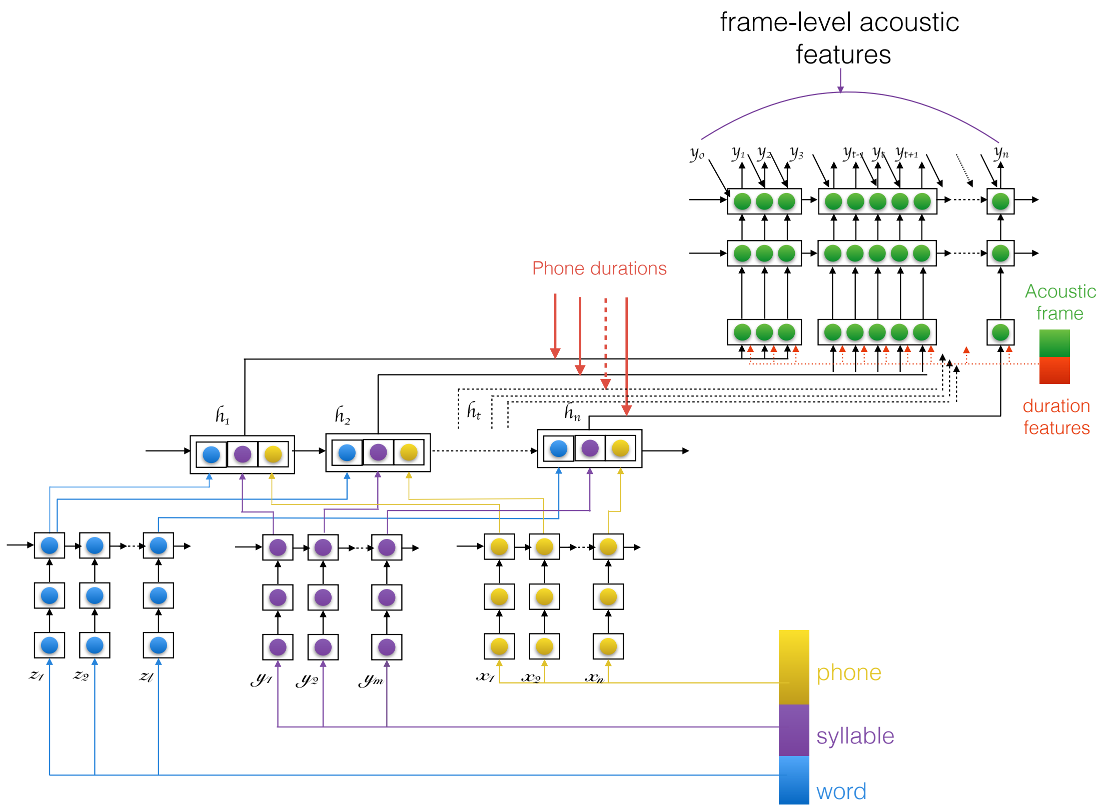


Figure 8.1: Schematic diagram of a multi-scale parallel encoder-decoder for SPSS.

allel encoders with each encoder processing each level of linguistic features independently.

The rest of the chapter is organized as follows: Section 8.2 describes the proposed approach for multi-scale parallel encoding of linguistic features and hidden feature upsampling via a broadcast-copy mechanism. Section 8.3 and 8.4 describes the experimental setup and results respectively. Our initial investigation of end-to-end attention based models for children's storybooks is presented in Section 8.5. Finally, Section 8.6 presents the conclusions and the scope for future work.

8.2 Multi-scale Parallel Encoder-Decoder

Figure 8.1 provides a schematic diagram of the proposed multi-scale parallel encoder-decoder neural network. The key ideas are to process the linguistic features and other suprasegmental features of different scale independently through multiple encoders in parallel and upsample the hidden features altogether at once.

8.2.1 Multi-scale parallel encoder

Figure 8.1 shows how the proposed method employs a multi-scale parallel encoder that accepts input at the original linguistic timescales of word, syllable and phone making use of multiple parallel encoders. Each linguistic feature is represented by a different color and is similar to the hierarchical encoder. The advantage of a multi-scale parallel encoder is that it processes linguistic features at different time-scales independently and separately through multiple encoders. The use of separate encoders for high-level features enables the network to capture long-term context by allowing each individual encoder to process the information through its own time-scale.

As shown in Figure 8.1, the input to the phone encoder (represented in yellow color) is a sequence of phone-level linguistic features and the input to the syllable encoder (represented in violet color) is a sequence of syllable-level linguistic features. Similar to the hierarchical encoder, the input to word encoder (represented in blue color) is a sequence of word-level features and other higher-level features (phrase-level and utterance-level, but duplicated for each word). The sequence lengths of each of these encoders are different, as they operate at different timescales ($\boldsymbol{l}_p$ for phone, $\boldsymbol{l}_s$ for syllable and $\boldsymbol{l}_w$ for word). The architecture for each of these encoders is identical and consists of feed-forward layers (typically TANH) and a final uni-directional recurrent layer, whose dimension is d_h.

The output from syllable and word-level encoders can be processed in three different ways: a) upsampling to phone-level features; b) fixed-dimensional embedding by passing the final state of the recurrent layer in each encoder; c) use of an attention layer to create a weighted context vector also known as variable-length embedding. Recently[1], the use of fixed-dimensional embedding was demonstrated by Skerry-Ryan et al. (2018) in an end-to-end prosody transfer for expressive speech synthesis with Tacotron. The use of an attention layer to create a weighted context vector may also have adverse affects on the decoder as the training may get unstable. One advantage of using an attention layer is that it can be generalized to very long utterances.

However, the work in this chapter focuses on upsampling of higher-level features to phone-level features as shown in Figure 8.1 as it aligns with our pre-

[1]After the work in this chapter was completed

vious study of hierarchical encoder. The upsampling procedure is explained in the next section.

In terms of processing time, the network architecture is still computationally efficient since the encoders are processed in parallel. However, the multiple parallel encoders increases the number of model parameters by three times compared to hierarchical encoder architecture and is relatively slow.

8.2.2 Hidden-feature upsampling

Similar to the hierarchical encoder system (HED-LSTM) proposed in Chapter 7, the upsampling takes place within the model architecture but is performed in parallel rather than progressively. The outputs from syllable-level ($\boldsymbol{l}_s \times d_h$) and word-level ($\boldsymbol{l}_w \times d_h$) encoders are upsampled to phone-level ($\boldsymbol{l}_p \times d_h$) using the input mapping between syllables to phonemes (number of phones in the corresponding syllable) and words to phonemes (number of phonemes in the corresponding word) respectively. Similar to the hierarchical encoder, this mapping is in general determined by the known linguistic structure of the current utterance. Once the sequence lengths from each of these two encoders are matched to phone encoder, one of the following operations can be performed to form the final phone-level encoder output.

Addition: In this approach, the upsampled hidden features from each of the encoders are added together to form a single hidden vector ($\boldsymbol{l}_p \times d_h$) which is of same dimension as of each of the encoders. For this approach, the dimension of each of the encoders has to be identical.

Concatenation: In this approach, the upsampled hidden vectors from each of the three encoders are concatenated together to form a large-dimensional hidden vector ($\boldsymbol{l}_p \times 3 * d_h$) and is symbolically represented by a rectangular block consisting of three colors in Figure 8.1. Unlike addition, the hidden dimension of each of these encoders can be configured with a different value. However, for experiments presented in this chapter, we used a fixed dimension (d_h) for all the encoders.

8.2.3 Recurrent decoder

Similar to the hierarchical encoder, the output from the multi-scale parallel encoder is also at phonetic timescale (h_t as shown in Figure 8.1). Using externally-provided duration information, this is upsampled to the acoustic frame rate used by the vocoder and then augmented by appending frame-level features derived from duration. Similar to our previous work in Chapter 7, the experiments in this chapter used a feedback mechanism i.e., predictions at the previous time step from the final layer of the network being fed back into the recurrent decoder during training as well as for inference.

8.3 Experimental Setup

In this section, we present the data used for training the proposed encoder-decoder architecture and the procedure for feature extraction. We also present further details on number of layers and hidden nodes used for the proposed multi-scale parallel encoder-decoder architecture and other details of the system training.

8.3.1 Data and feature extraction

We tested our proposed method on the speech database released for the 2017 Blizzard Challenge (King et al., 2017) consisting of 56 children's books – this includes 50 books released for the 2016 Blizzard Challenge and an additional 6 books which were officially used as test set for 2016 Challenge evaluation. The details of the data and the split for train-validation-test was provided in Section 3.1.

The feature extraction procedure for input linguistic features and output acoustic features hasn't changed and is same as previously explained in Section 7.5.2.

8.3.2 System training

For final evaluation, four systems were trained: identifiers listed in the left-hand column of Table 8.1. The baseline system (*Frame-LSTM*) and the hierarchical encoder-decoder system (*HED-LSTM*) were repeated from Chapter 7.

For our proposed approach (Section 8.2), the encoder was configured with four feed-forward layers of 1024 nodes each with a TANH activation and a uni-directional SLSTM. Similar to *HED-LSTM*, the decoder was configured with two recurrent layers with each layer consisting of 512 SLSTM units.

Two variant systems were built to test whether the proposed parallel encoder can better exploit linguistic features at different time-scales than the usual approach. *PED-LSTM-CC* is a parallel encoder-decoder which implements our proposed idea with each of its encoder features upsampled to phone-level and concatenated together to form a large-dimension hidden vector of size 1536 (512 from phone encoder, 512 from syllable encoder and 512 from word encoder). *PED-LSTM-ADD* is also a parallel encoder-decoder with each of its encoder output features upsampled to phone-level and summed to form a 512 dimensional hidden vector.

All networks were initialised using small weights sampled from a Gaussian distribution and trained with a fixed learning rate of 0.001; no pre-training was used and no regularization was applied for any of the layers in the network. Unlike the baseline and hierarchical systems, the proposed systems were trained with a batch size of 3, which reduced the training time of the proposed system considerably from 48 hours to 4 hours. A batch size of more than 3 couldn't be used for training since the amount of data used for training is considerably small (7000 utterances) and the network doesn't generalize well for larger batch sizes. Also, the batch training for recurrent systems is not straight-forward as the sequence lengths vary for each utterance. For the implementation in this work, the input linguistic features and the output acoustic features were zero padded to the maximum sequence length within each batch. This implementation along with other variants of batch training are now made open-source within the Merlin toolkit.

8.3.3 Synthesis

The synthesis procedure remained as previously explained in Section 7.5.4. The STRAIGHT vocoder (Kawahara, 2006) was used to synthesize the waveform which was then normalised according to ITU P.56 (P.56, 2011).

8.4 Results

In this section, we will first present objective results for baseline (trained using conventional approach in SPSS), hierarchical system presented in Chapter 7 and our proposed architecture in Section 8.4.1. Subjective results from listening

	Spectral		F0 Measure		
System	**MCD**	**BAPD**	**RMSE**	**Corr.**	**V/UV**
Frame-LSTM	5.44	0.075	51.85	0.432	5.49%
HED-LSTM	5.48	0.074	50.20	0.453	5.48%
PED-LSTM-CC	5.60	0.075	51.56	0.405	5.79%
PED-LSTM-ADD	5.66	0.075	51.08	0.410	6.26%

Table 8.1: Objective results for baseline and hierarchical encoder-decoder systems with and without auxiliary features.

tests follows in Section 8.4.2.

8.4.1 Objective measures

We objectively compared the speech parameters generated by each system for the test set against held out natural examples. We calculated Mel cepstral distortion (MCD), band aperiodicity distortion (BAPD), and root mean square error of $F0$ (RMSE), Pearson correlation of $F0$ (Corr.) and voiced/unvoiced error rate (V/UV).

The results for the proposed systems along with the baseline (*Frame-LSTM*) and hierarchical encoder-decoder system (*HED-LSTM*) from Chapter 7 are shown in Table 8.1.

Comparing the two variants of the proposed system, the concatenation approach (*PED-LSTM-CC*) provided lower mel-cepstral distortion (MCD). The F0 RMSE and correlation numbers are similar for both the variants. However, the voiced-unvoiced error (VUV) is much lower (5.79%) in *PED-LSTM-CC* compared to *PED-LSTM-ADD* (6.26%). From the overall objective scores, we interpret that the concatenation approach fared slightly better compared to the addition approach for the proposed architecture.

Compared to baseline system (Frame-LSTM) and hierarchical system (HED-LSTM), the MCD for the best-performing variant (PED-LSTM-CC) has deteriorated slightly. The F0 RMSE and correlation also suggest that intonation hasn't improved either. We hypothesize that one of the possible reasons for not achieving improved results is that the training paradigm for recurrent neural networks has changed from single utterance to batch training and is not completely optimized.

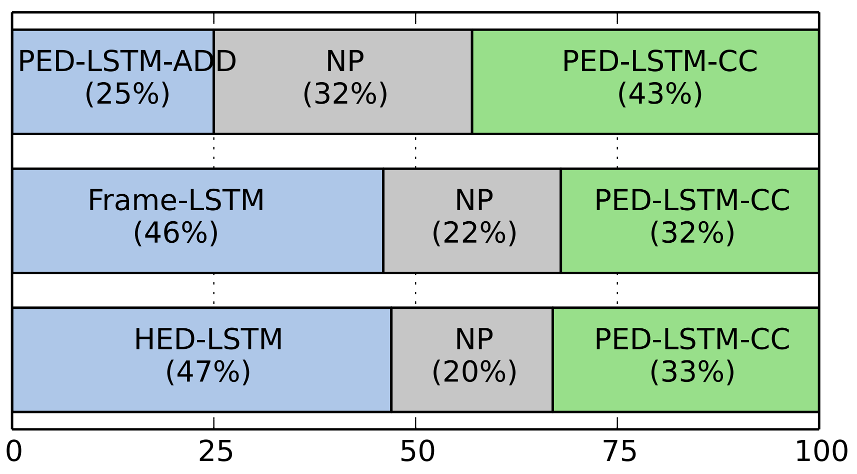


Figure 8.2: Preference test results for naturalness with percentage in brackets, where NP denotes “no preference”

8.4.2 Subjective evaluation

We assessed the naturalness of the synthesised speech via pairwise preference listening tests. Three pairs of systems were considered: *PED-LSTM-ADD* vs. *PED-LSTM-CC*, *Frame-LSTM* vs. *PED-LSTM-CC*, and *HED-LSTM* vs. *PED-LSTM-CC*. 20 paid native English speakers with no known hearing impairments, participated in the listening test. Each listener was asked to listen to 75 randomly selected pairs. Thus each condition received a total of 500 judgements(20*25). Listeners were told that the synthesized audio files were from children's story books. Within each pair, the listener was played two synthesised versions of the same sentence and asked to choose which one sounded more natural. They were asked to focus on prosody (tune) and only to select the “no preference” option when both sounded the same.

Results of the preference test are presented in Figure 8.2. First, examine the effectiveness of the two different variants of the proposed multi-scale parallel encoder-decoder: *PED-LSTM-CC* sounds significantly more natural than *PED-LSTM-ADD* with $\alpha = 0.01$ under a binomial test with an expected 50% split. This suggests that the approach that used the concatenation of output vectors from multiple parallel encoders has outperformed the addition of upsampled hidden-features.

The best performing variant *PED-LSTM-CC* is now compared with both baseline and previously proposed hierarchical system. The baseline system

(*Frame-LSTM*) is significantly preferred over *PED-LSTM-CC* and similarly, *HED-LSTM* is significantly preferred over *PED-LSTM-CC*. The results from these listening tests correlated well with the observations we have seen from objective scores of these systems.

Based on the poor performance of these systems (*PED-LSTM-CC* and *PED-LSTM-ADD*) and literature review of the recent end-to-end systems for acoustic modelling (Wang et al., 2017c; Shen et al., 2017, for example), we speculate that the performance of the proposed architecture may be hindered by several factors including upsampling of hidden-features, number of hidden nodes used for modelling phone, syllable and word-level hidden features (each of these features were given equal weight and not fine-tuned) and poor regularization of the network. Some of these factors are merely drawn based on intuition and require further investigation. The following section on sequence-to-sequence (Seq2Seq) modelling further highlights the limitations with some of the underlying assumptions in the current approach.

8.5 Sequence-to-sequence modelling

In the current approach for multi-scale parallel encoder, the durations from forced-alignments are used for upsampling phone-level hidden-features and these state-aligned durations are obtained from a pre-trained monophone hidden Markov model. Such an approach has two main disadvantages as identified in Bruguier et al. (2018): a separate forced-alignment model has to be trained and any errors that appear from alignment are propagated through encoder-decoder network; the forced-alignment model and the encoder-decoder model for predicting vocoder features would be separately optimized. This means that the forced-alignment model may not be optimal for the final prediction of acoustic features.

The Tacotron2 (Shen et al., 2017) suggests that an end-to-end model can be jointly trained and optimized for improved naturalness. Therefore, we also considered to replace the hidden-feature upsampling with an attention based sequence-to-sequence model as shown in Figure 8.3. This means the features from each of these three encoders are still upsampled and concatenated to form a sequence of phone-level hidden features (h_t). But, the final upsampling to frame-level is replaced by an attention, which produces a context vector for

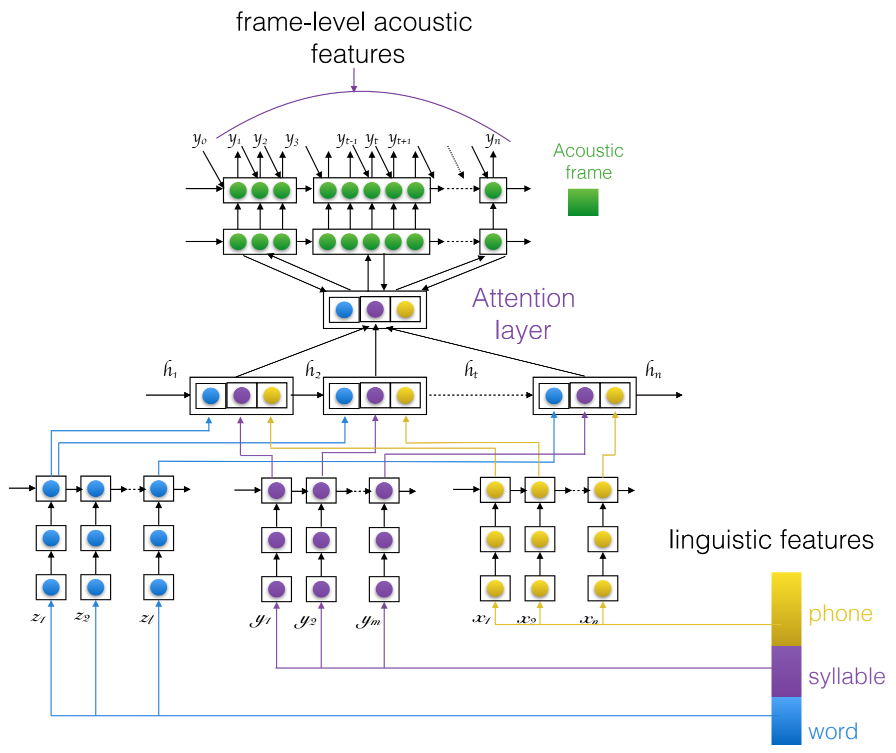


Figure 8.3: Schematic diagram of a multi-scale parallel encoder with attention layer replacing hidden-feature upsampling.

every decoder step (i.e., for each frame in the decoder).

An open-source implementation of Tacotron[2] was used for initial investigation of an attention-based approach. However, our data is quite limited (6.5 hours) for an end-to-end model and the alignments didn't appear even though the network was trained for several hundred epochs. The sequence-to-sequence models therefore require further investigation in the context of limited data for audiobooks. Some ideas to improve sequence-to-sequence modelling are proposed as future work (Section 10.3).

8.6 Conclusions

We have described a multi-scale parallel encoder-decoder to process the linguistic features at different time-scales separately, making use of multiple parallel encoders. Experiments were conducted on prosodically-varied audiobook material because the use of supra segmental features is thought to be particularly important in this case. Although the proposed architecture has

[2]https://github.com/keithito/tacotron

potential to model long-term context, both objective and subjective results suggested that the preliminary work requires further investigation.

We conjecture that the hierachical encoder proposed in Chapter 7 and multi-scale parallel encoder proposed in this chapter may be benefited by using a Tacotron2-like paradigm (Shen et al., 2017). Section 10.3 presents future work on improving Seq2Seq models and discusses the most recent work[3] on utilizing multi-scale parallel encoder for end-to-end training.

[3]Published after the work in this chapter was completed

Chapter 9

DNN-guided speech synthesis system

This chapter is an expanded version of the work in Ronanki et al. (2017a) and therefore the text is closely related to that.

The work in this Chapter is based on the CSTR entry for the Blizzard Challenge 2017 and was completed in collaboration with others. Discussion of ideas was between myself, Thomas Merritt, Sam Ribeiro, Felipe Espic and Oliver Watts. The code for parametric-guided unit selection was initially implemented by Rob Clark in Festival. I made improvements in duration modelling to the initial implementation and provided a recipe in Merlin to reproduce the system. Concatenation and join smoothing between selected waveform units was done by Felipe Espic. Subjective evaluation including other participants was done as part of the Blizzard Challenge listening test evaluation, by the challenge organisers.

9.1 Introduction

In Chapters 7 and 8, we discussed novel encoder-decoder systems in statistical parametric speech synthesis. Although the hierarchical encoder-decoder has shown promising results (see Figure 7.3) over the conventional baseline, the segmental quality of the synthesized speech is still suboptimal and is greatly limited by the use of the STRAIGHT vocoder. To overcome this problem, there are two possible solutions: parametric guided unit selection system (also known as Hybrid Unit Selection) and direct waveform generation (sample by sample generation, typically using autoregressive models).

The most common hybrid approaches, which use parametric models to guide unit selection, has drawn a lot of attention in text-to-speech synthesis

(Yan et al., 2010; Qian et al., 2013; Merritt et al., 2016a, for example). In order to improve the segmental quality of synthesized speech, these approaches typically use a parametric-based target cost to select the best possible units for concatenation and minimize the distortions between selected units. Although Merritt et al. (2016a) used DNN-guided unit selection system, it was not tested for synthesis of audiobooks. For the work in this Chapter, we discuss a hybrid system which was built on the same basis using children's storybooks for the Blizzard Challenge 2016-17. The main motivation for introducing a hybrid unit selection system in this thesis is to show the kind of segmental quality we can achieve with audiobooks especially with highly expressive children's storybooks that were used for experiments throughout in this thesis.

The second alternative is to use a neural vocoder (Tamamori et al., 2017; Shen et al., 2017; Lorenzo-Trueba et al., 2018) in place of the conventional STRAIGHT vocoder, as discussed in Section 2.1.3. However, the second approach is quite new at the time of writing this thesis and therefore is out of scope for discussion here.

The rest of the chapter is organized as follows: Section 9.2 presents the task for the 2016-17 Blizzard Challenge and discusses the main differences between CSTR entries in the years 2016 and 2017. Section 9.3 describes the hybrid unit selection system submitted for the 2017 Blizzard Challenge. The results provided by challenge organizers are analysed and discussed in detail in Section 9.4 and finally, section 9.5 presents the conclusions.

9.2 CSTR entries to the Blizzard Challenge

The annual Blizzard Challenge conducts side-by-side testing of a number of speech synthesis systems trained on a common set of speech data. The task for the 2016-17 Blizzard Challenge is to train on expressively-read children's storybooks, and to synthesise speech in the same domain. The Challenge therefore presents an opportunity to investigate the effectiveness of several techniques we have developed when applied to expressive and prosodically-varied audiobook data.

As discussed previously in Section 2.1.1.2, hybrid synthesis systems operating on the basis of target cost functions employ statistical models to predict acoustic properties of speech thereby bringing the benefits of extremely natural-sounding unit selection (which is unaffected by the degradations introduced by vocoding (Zen et al., 2013; Henter et al., 2014b)). This section further describes the text-to-speech system entered by The Centre for Speech Technology Research (CSTR) into the 2016-17 Blizzard Challenge. The system submitted for the challenge was a hybrid synthesis system which drives a unit selection synthesiser using the output from neural network acoustic and duration models. We assess the performance of our system by reporting the results from formal listening tests provided by the 2017 challenge in Section 9.4.

9.2.1 CSTR entry for the 2016 Blizzard Chalenge

My contribution to the 2016 entry was discussions, integration of DNN-based duration prediction in Festival Multisyn and experimental setup including systems training for parametric models. Building unit selection models and fine-tuning the weights of target cost and join cost was carried by Thomas Merritt.

The data used for the Challenge was obtained from professionally-read child-directed audio books and is therefore much more prosodically rich than more standard prompt-based speech data. The experiments presented in Qian et al. (2013) and Merritt et al. (2016a) established that improving the underlying SPSS of a hybrid synthesiser results in improvements to the concatenated output speech. Therefore, for our entry in 2016 (Merritt et al., 2016b), we have incorporated two major improvements to the underlying SPSS model compared to the system presented in Merritt et al. (2016a): the decision tree duration model was replaced with a bi-directional long short-term memory (LSTM) recurrent neural network, and the feed-forward DNN acoustic model was replaced with a LSTM network. The bi-directional LSTM-based duration model is previously described in Section 5.6.2 and the LSTM-based acoustic model is very much similar to the conventional baseline described in Section 7.2.

9.2.2 CSTR entry for the 2017 Blizzard Chalenge

The CSTR entry to 2017 Blizzard Challenge builds on the hybrid Multisyn (Clark et al., 2004, 2007b) system submitted for 2016 (Merritt et al., 2016b). The schematic diagram of the hybrid speech synthesis framework used for the challenge submission in 2017 is presented in Figure 9.1. The text processing and speech parameterization steps are similar to the feature extraction proce-

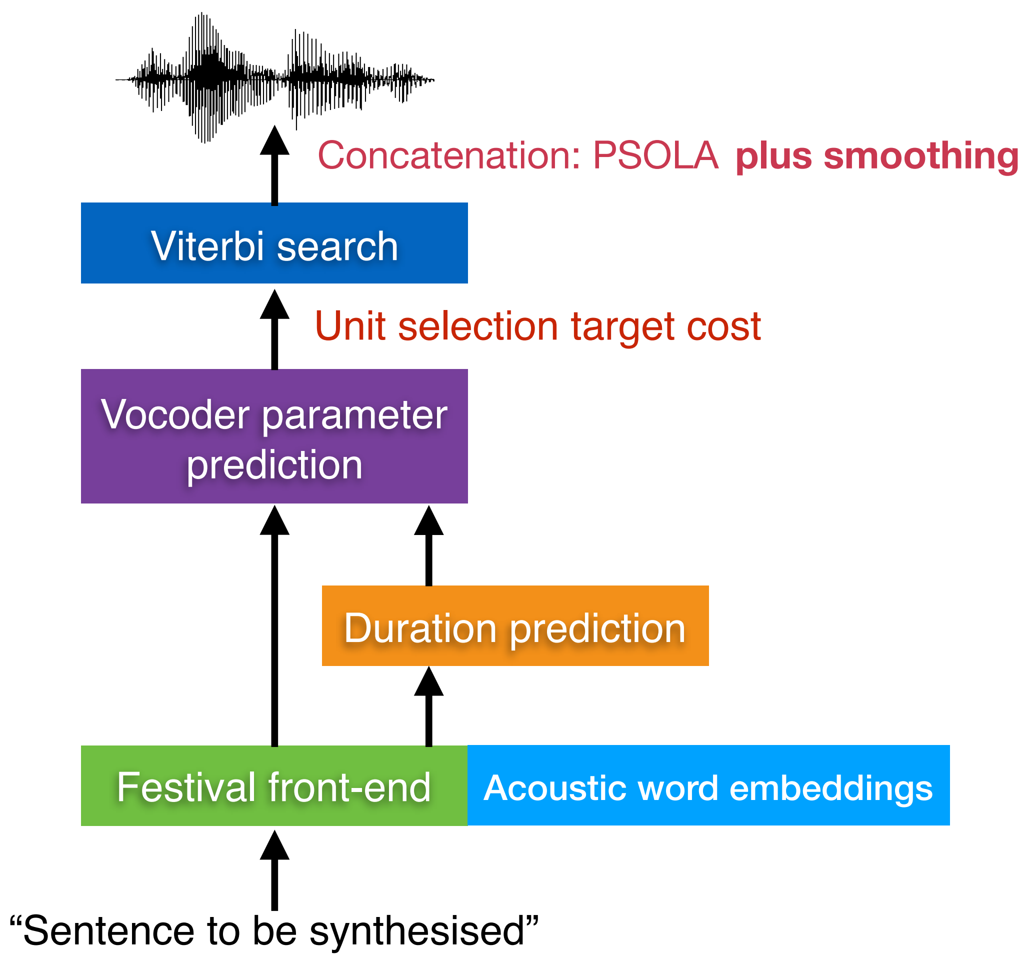

Concatenation: PSOLA plus smoothing
Viterbi search
Unit selection target cost
Vocoder parameter prediction
Duration prediction
Festival front-end
Acoustic word embeddings
"Sentence to be synthesised"

Figure 9.1: Schematic diagram of the hybrid speech synthesis framework used for the Blizzard Challenge 2017 entry.

dure explained in Section 3.3 and are largely unchanged across our entries in 2016 and 2017. Acoustic model prediction is slightly optimised for better prediction of fundamental frequency by adding supra-segmental features based on acoustic counts (represented by a blue block with name "Acoustic word embeddings") and is explained in Section 9.3.2. However, in contrast to the 2016 submission, feed-forward neural networks were utilised for training both acoustic and duration models due to lack of time for training and proper optimization of recurrent networks for the additional supra-segmental features. Compared to the 2016 CSTR submission (Merritt et al., 2016b), another notable change is our attempt to smooth the joins: this new development is described in Section 9.3.6. The neural networks used in this entry were trained using the open-source Merlin speech synthesis toolkit (Wu et al., 2016).

9.3 System Description

In this section, we describe each of the components used for building hybrid unit selection system entry to the 2017 Blizzard Challenge. The procedure for data processing, sentence selection and text processing is explained in Section 9.3.1 and Section 9.3.2 respectively. Section 9.3.3 describes the parametric models trained for predicting acoustic and duration features. Section 9.3.4 describes the procedure for selecting diphone units given the parameters predicted from acoustic model. Finally, the procedure for concatenation of units and join smoothing is explained in Section 9.3.5 and Section 9.3.6 respectively.

9.3.1 Data

The database – provided to the Challenge by Usborne Publishing Ltd. – consists of the speech and text of 56 children's audiobooks spoken by a British female speaker. The total duration of the audio is approximately 6 hours after segmentation. Three audiobooks from the given corpus were held out to act as an internal development set to gauge system performance before generating the final test data. The details of the data and the split for train-valid-test was provided in Section 3.1.

9.3.1.1 Sentence selection

For sentence selection, we have followed a similar approach to 2016. Harnessing the variety of speaking styles present in expressively-read audiobooks might enable us to produce less robotic-sounding TTS systems. However, initial experiments showed that the extreme variation in parts of the training data for the Challenge resulted in poor unit selection. We therefore filtered the data using the active learning approach described in Watts et al. (2013): 198 utterance-level acoustic features are extracted, and 15 sentences initially labelled as *keep* or *too expressive* by an expert listener. Uncertainty sampling (Lewis and Gale, 1994) using an ensemble of decision trees was then used to select a further informative sample to be hand-labelled; this process continued for 20 minutes (real time). A classifier built on the entire set of hand-labelled data was then used to determine the subset of available sentences to be used for training. 20% of the training sentences were discarded in this way; informal comparison suggested this resulted in more stable synthesis with fewer unwarranted prosodic excursions.

9.3.2 Text processing

We have used Festival's English front-end with the British Received Pronunciation version of the Combilex lexicon (Fitt and Richmond, 2006). 163 items were added to cover words appearing in the training data but otherwise absent from the dictionary. There were slight differences in the lexicon-lookup procedures used in preparing the annotation for training the SPSS model and those employed by the Festival front-end used for Multisyn. The lexicon-lookup for SPSS was derived using a HTS combilex voice developed in 2011 (from voice clone toolkit), which has few additional post-processing rules, whereas the lexicon lookup for Multisyn uses a 2009 unit-selection voice with additional addenda (local dictionary for a few specific set of words). The resulting inconsistencies were dealt with by aligning the DNN's phone sequences to those expected by Multisyn in an ad hoc fashion – either by merging the duration of phones (in case of deletions) or by splitting the phone duration (in case of insertions).

For acoustic word embeddings shown in Figure 9.1, word and syllable level vector representations were learned, according to the method described in Ribeiro et al. (2017b) and were appended to the conventional linguistic features described in Appendix A. These embeddings were learned by finding the singular value decomposition (SVD) of a co-occurrence matrix M (defined as $|V| \times w|A|$, where V is a pre-defined vocabulary set, A is an acoustic class set and w is the window size). We find the SVD of M and take k left singular vectors such that:

$$M = USV^T \tag{9.1}$$

where U is defined as $|V| \times k$. The co-occurrence matrix M is computed by taking counts of acoustic events of $F0$ and energy for each syllable or word and its context defined by a window size. Both these acoustic events are stylized by clustered vectors and mean values defined over syllables or words. For clustering F0 and energy, the template-based approach proposed in Section 4.2.1 was used. A total of 20 templates and a window span of 3 was used to compute the counts of each template for all the words in vocabulary set. The training data available for the Challenge was used to learn these matrices. Experiments were also conducted using vector representations learned over a larger database of a different speaker, but we have observed that results were comparable with speaker-dependent vectors learned on a smaller database.

9.3.3 Parametric system

The parametric system was implemented using DNNs in a conventional two-stage approach. In the first stage, a duration model is used to predict phone durations to form frame-level linguistic features. In the second stage, an acoustic model is used to generate parameters from those linguistic features.

9.3.3.1 Feature extraction

The procedure for feature extraction is described in several previous chapters and is also described in detail in Section 3.3. A state-level forced alignment of phone sequences with the sentence-segmented audio was obtained using context-independent HMMs and the procedure for extraction of linguistic features is detailed in Section 3.3.1. The speech data was analysed with STRAIGHT (Kawahara, 2006) and the procedure for extraction of these acoustic features is detailed in Section 3.3.2.

For duration modelling, the output for training is a five-dimensional vector of durations for every phone, comprising five sub-state durations. For acoustic modelling, the output for training is 259-dimensional vector of acoustic features, which includes delta and delta-delta features.

9.3.3.2 Duration model

The duration model trained for our entry to the challenge made use of a simple and straightforward approach with feed-forward neural networks (DNNs) as demonstrated in Ronanki et al. (2016d,c). The duration model is trained on the force-aligned data and generates state-level durations given phone-level linguistic features. For the duration model, we have used 481-dimensional binary and numerical valued feature vectors as input. The model was defined to be 6 feedforward hidden layers, each with 1024 nodes, using the *tanh* activation function. Mini batch size was set to 64 and learning rate was set to 0.002, being reduced by 50% with each epoch after the first 10 training epochs.

The described approach was used only to generate durations, which were then used to form frame-level linguistic features used as input in the generation of acoustic parameters. The hybrid Multisyn unit-selection system, however, does not make use of any duration-related features in its target cost function.

9.3.3.3 Acoustic model

For the acoustic model, we have used the same 481-dimensional feature vector representing linguistic features. To these, we appended the syllable and word level vector representations based on acoustic counts (Ribeiro et al., 2017b) and is described in Section 9.3.2. Nine frame-level features were also included according to Wu et al. (2015b) and available from Wu et al. (2016). The durations generated by the duration model described above were used to upsample all features to frame-level. These frame-level feature vectors were then used as input to an acoustic model. The input vector to the acoustic models consisted of a total of 1900 dimensions. A feedforward neural network was trained at the frame level to map linguistic inputs to vocoder parameters consisting of static and dynamic (delta and delta-delta) features. The model consisted of 6 feedforward hidden layers, each with 1024 nodes, using the *tanh* activation function. Mini-batch size was 256 and remaining parameters were identical to the duration model.

In SPSS, the predicted parameters would be passed through the vocoder to synthesize a speech waveform. Instead, we use them as targets for selecting waveform units. Within each phone unit (from beginning frame to end frame of a phoneme), generated parameters are split uniformly across time into 4 sections. A Gaussian distribution is then fitted for each sub-phone section of acoustic parameters by computing means and variances across all the frames present in each section. The variances of these Gaussian distributions are floored at 1% of the global variance per parameter (Merritt et al., 2016a).

Similar to Merritt et al. (2016a), diphones are chosen as units for concatenation in this work. The distributions associated with each of the 4 sub-phone sections are used to construct a diphone representation for the target utterance. To construct a diphone representation, we consider sub-phone sections from two consecutive phonemes. We take the first 2 sections of current phoneme and last 2 sections from its previous phoneme (or contextual phoneme). Comparable distributions were generated for the diphone candidates in the unit database, based on parameters predicted for all the training data but using natural durations obtained by forced alignment. We also considered vocoder parameters extracted from the training data for target cost computation but the generated parameters were found to provide better selection of units (through informal listening) due to the matching conditions.

9.3.4 Unit selection waveform renderer

For unit selection, a modified form of Festival's Multisyn engine (Clark et al., 2007b) is used. To compare the suitability of a given diphone candidate in the unit database with the 4 sub-phone distributions representing a synthesised diphone, the symmetrised Kullback Leibler divergence (KLD) (Hershey and Olsen, 2007) is used. The KLD is computed between each of the 4 candidate unit's distributions and the corresponding target unit distributions individually. The resulting 4 scores are then summed to produce the final target score.

Similar to Merritt et al. (2016a), the pre-selection criterion of candidate units is chosen by matching diphone identity and the standard Multisyn join cost (sum of distances between 12 MFCCs, $F0$ and energy from the frame either side of the join) is applied. The Multisyn Viterbi search (with pruning to reduce the search time) is performed in order to optimise target cost and join cost. Also the standard Multisyn back-off rules are used where the target diphone to be synthesised is not present in the training data and in exceptional cases, a diphone with pauses ($/pau - pau/$) is used as the back-off unit.

9.3.5 Speech synthesis

At synthesis time, duration is predicted first, and is used as an input to the acoustic model to predict the speech parameters. Similar to our previous work in Chapters 7 and 8, maximum likelihood parameter generation (MLPG) (Tokuda et al., 2000) using variances computed from the training data was applied to the output features for synthesis, and spectral enhancement postfiltering was applied to the resulting MCC trajectories. These parameter trajectories are then used to produce diphone coefficients. The Festival Multisyn

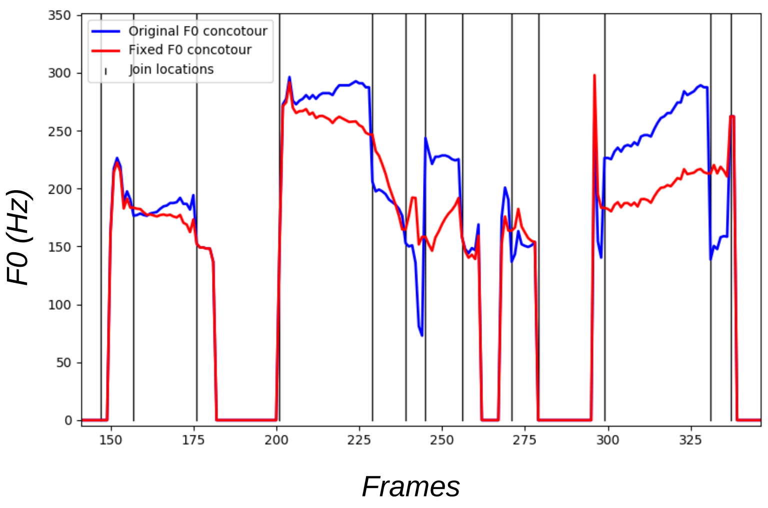


Figure 9.2: An example F0 contour with concatenation and correction of F0 from diphone segments.

engine was used to compute the target and join cost between target unit and pre-selected candidate units to select the final candidate, as explained above. The final waveform synthesis was done by joining the selected units. An additional smoothing and post-modification of prosody was performed during joining the units and is explained below.

9.3.6 Concatenation and join smoothing

The selected waveform units are parameterised using the method proposed in Espic et al. (2017). It extracts pitch synchronous speech features on a frame-by-frame basis, describing the complex spectra and F0 contour. The correction/smoothing operations are performed over these features to produce seamless concatenation of units.

9.3.6.1 Concatenation and correction of F0 contours

The $F0$ mid point ($F0_m$) between two consecutive units is given by $F0_m = (F0_p[N_p - 1] + F0_c[0])/2$, where p means preceding unit, c current unit, and N is the unit length in frames.

Then, the slope of the F0 contours of both units are adjusted to reach the $F0_m$ just at the join location. The corrected $F0$ contours are computed by Equations 9.2 and 9.3.

$$F0'_c[n_c] = F0_c[n_c] + (F0_m - F0_c[0]) \cdot \left(\frac{n_c}{1 - N_c} + 1\right) \tag{9.2}$$

$$F0'_p[n_p] = F0_p[n_p] + (F0_m - F0_p[N_p - 1]) \cdot \frac{n_p}{N_p - 1} \tag{9.3}$$

Where $F0'$ is the corrected $F0$, and n is the frame index within each unit. An example F0 contour after applying above smoothing techniques is shown in Figure 9.2. After obtaining all the corrected $F0$ contours for all the units, these are appended building a single $F0$ contour for the whole sentence.

9.3.6.2 Spectral concatenation and smoothing

Spectral concatenation is done by overlapping and crossfading the complex FFT spectra of two consecutive units. Some extra frames are extracted from the sources, so the units can be overlapped without affecting their expected locations in the synthesised waveform. Three extra frames on each side of the units are extracted from the sources, thus an overlap of seven frames around the joins is produced.

The FFT complex spectrum S is derived from the parameters proposed in Espic et al. (2017), M, R, and I, by $S = M \cdot (R + Ij)$. The crossfade is linearly applied to mix the FFT complex spectra of two consecutive units, progressively. It is seven frames length, and in case where a unit is too short, the crossfade is shortened accordingly.

After performing this operation on every join, the FFT complex spectra of all the units are concatenated producing a single complex spectra stream, that describes the whole utterance. Finally, the signal is synthesised by converting the FFT complex spectra to time domain, and applying Pitch Synchronous Overlap-Add as explained in Espic et al. (2017), using the corrected $F0'$ contour. An example waveform from children's storybooks before and after smoothing can be accessed at this URL[1].

[1]Smoothing samples: `http://srikanthr.in/Blizzard2017`

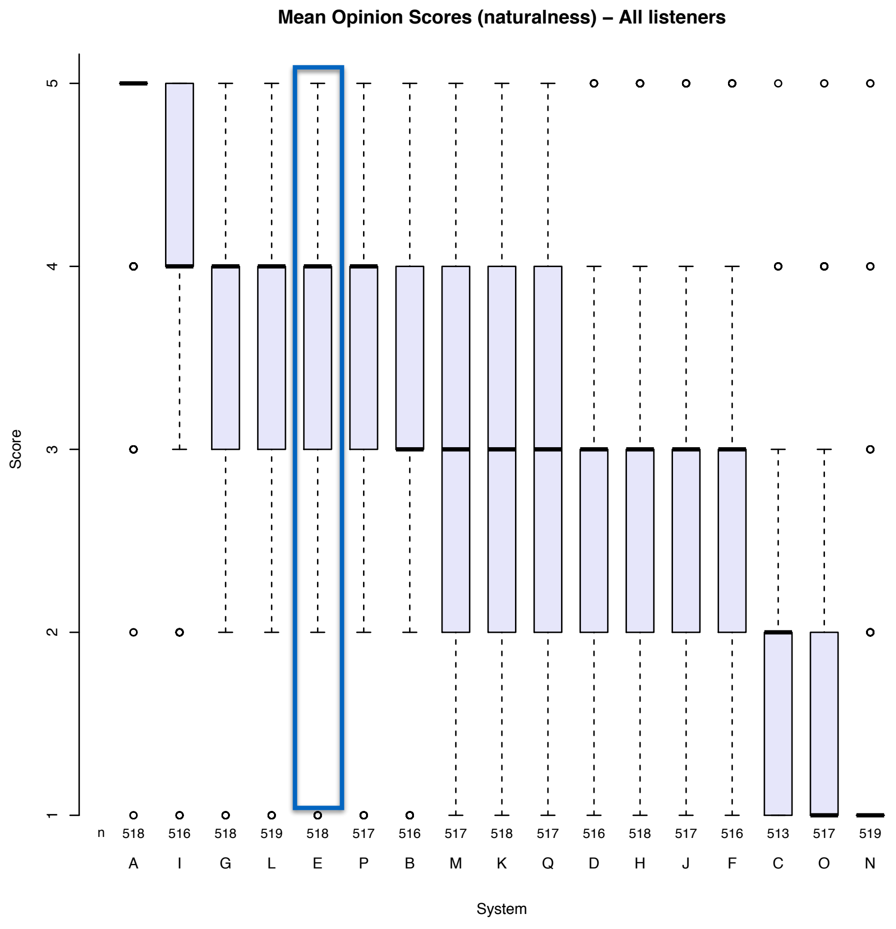
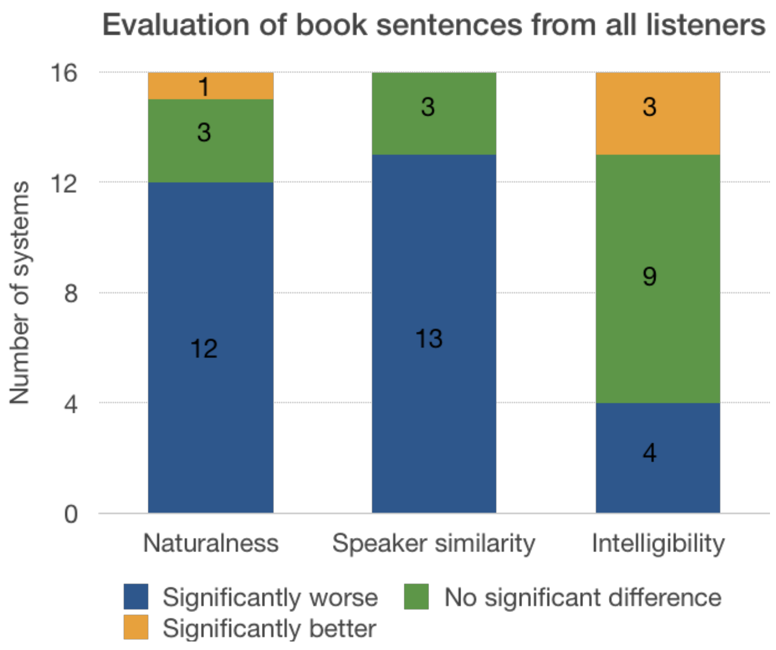


Figure 9.3: Our system(E): Mean opinion score for naturalness of the synthesized speech with ratings from all listeners.

9.3.7 Paragraph-level synthesis

From the sentences synthesised in this way, files were made containing whole paragraphs, chapters and books as required by the Challenge by simply concatenating the waveforms. While proper exploitation of long-distance contexts ought to improve synthesis quality, no contexts outside the current sentence were used for the challenge submission. Six books were generated as part of the the Blizzard Challenge evaluation and the audio files can be accessed from this URL[2].



1

Figure 9.4: Significant differences between our system and other Challenge participants.

9.4 Results

The submissions to the Blizzard Challenge were generally evaluated using MOS for naturalness and speaker similarity and word error rate (WER) for intelligibility. For the 2017 Blizzard Challenge, the book sentences were evaluated on two factors: naturalness and speaker similarity. The audiobook paragraphs were additionally evaluated based on: emotion, pleasantness, listening effort, intonation, stress, speech pauses and overall impression. Intelligibility was also evaluated but on a specially designed semantically unpredictable sentences (SUS) test set.

There were three baseline systems: A unit-selection system based on Multisyn (identifier B), HMM-based SPSS built using HTS (identifier C), DNN-based SPSS built using Merlin (identifier D), which were also evaluated along with other challenge participants. The identifier for our system in the published results is E. All systems were evaluated by three listener groups: paid listeners, on-line volunteers and speech experts.

[2]Samples are available at: http://srikanthr.in/Blizzard2017/wav/Hybrid

9.4.1 Naturalness

Mean opinion scores for naturalness (on a scale from 1 to 5) from all listeners on book sentences are shown in Figure 9.3. In our discussion, we make use of the published statistical analysis of the results at 1% level with Bonferoni corrected alpha (Clark et al., 2007a). Our system achieved a mean opinion score of 3.5 (mean), 4 (median) and outperformed all three baselines (systems B, C and D). Among the 12 other challenge participants, our system is outperformed only by a single system (see Figure 9.4 for naturalness). The same trend was seen across the scores given by paid listeners and on-line volunteers. When considering ratings only from speech experts, no other system was significantly better than ours. Overall, our system outperformed 11 out of 15 other systems (including three baselines) evaluated in the listening test.

9.4.2 Speaker similarity

We now consider mean opinion scores for speaker similarity. The mean opinion scores for speaker similarity from all listeners on book sentences are shown in Figure 9.5. Considering ratings from all listeners (or any other listener group), no other system was significantly better than ours and our system was in turn significantly better than 13 other systems. These results show the effectiveness of our system and, in general, waveform concatenation systems for speaker similarity.

9.4.3 Intelligibility (SUS)

We now consider the results for intelligibility of semantically unpredictable sentences (making use of the published statistical analysis of significant difference between word error rates of the systems). Taking into account ratings from all listeners, there are only three other systems out of 15 (D, L, and M) significantly better than ours. Considering only paid listeners, there are only two other systems (D and L) significantly better than ours. Out of 15 other systems evaluated by paid listeners, 10 were not significantly more or less intelligible than ours, 3 were significantly less intelligible, and only 2 significantly more intelligible. The results show that our system is quite effective on intelligibility as well.

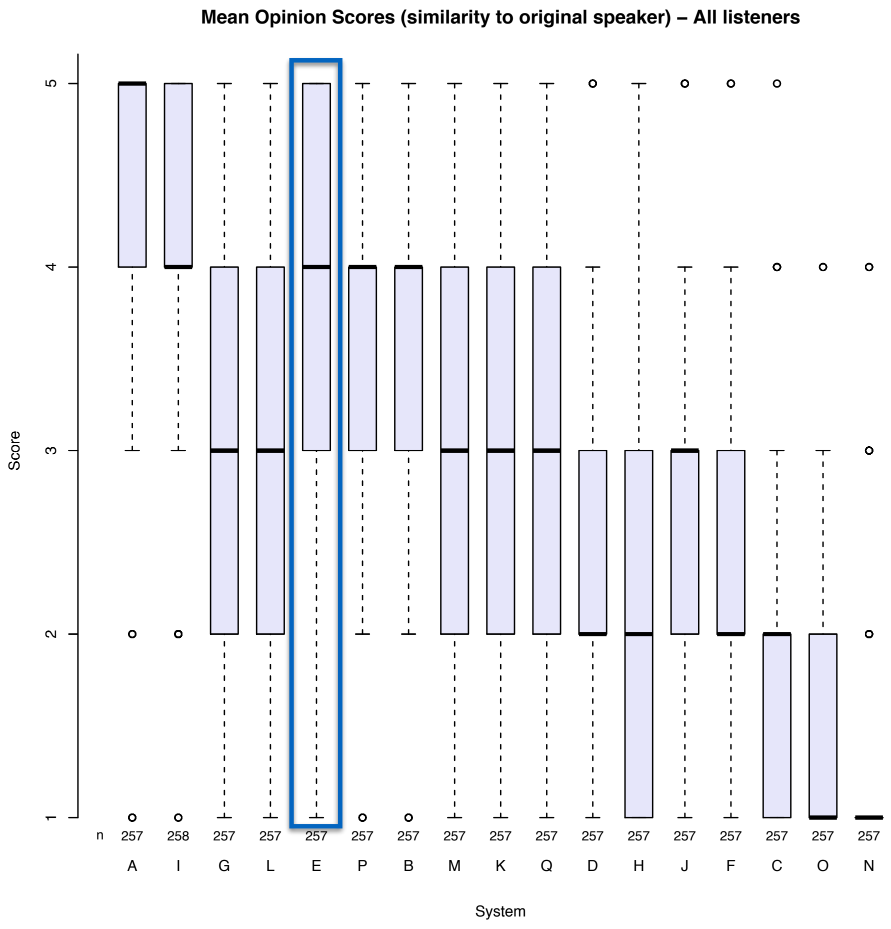
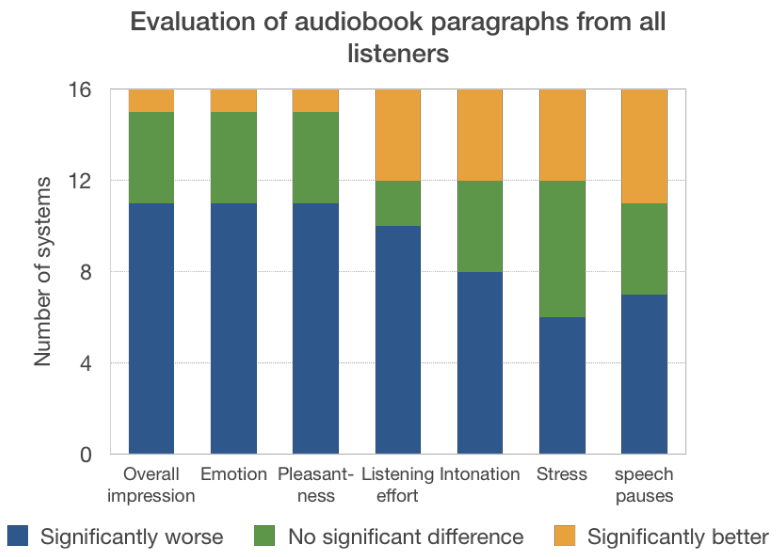


Figure 9.5: Our system(E): Mean opinion score for speaker similarity with ratings from all listeners.

Figure 9.4 shows the significance of our system compared to all other systems in each of the three factors i.e., naturalness, speaker similarity and intelligibility. Overall, our system has shown consistent performance (standing in the top four) in all the factors evaluated for the Challenge on isolated sentences.

9.4.4 Evaluation of audiobook paragraphs

We now consider the results for evaluation of audiobook paragraphs that have been evaluated on several other factors like stress, intonation, emotion, pleasantness, listening effort, speech pauses and overall impression. These ratings reveal whether the synthesized systems focused on such underlying aspects of the speech. Considering ratings from all listeners on overall impression, our system showed similar performance as in the case of the isolated sentence



1

Figure 9.6: Significant differences between our system and other Challenge participants in each of the factors evaluated for audiobook paragraphs.

evaluation of naturalness and speaker similarity. Only one system (I) outperformed us and our system was significantly better in turn than 11 other systems (cf. Figure 9.6). A similar trend was seen across the scores made by speech experts, online volunteers and paid listeners. Considering ratings for other individual factors (e.g., intonation, emotion and pleasantness) from all listeners, again only system I consistently outperformed ours.

Figure 9.6 shows the significance of our system compared to all other systems in each of the six factors evaluated for audiobook paragraphs i.e., emotion, pleasantness, listening effort, intonation, stress and speech pauses. Overall, our system outperforms between 7 and 11 other systems in evaluation of each of these factors, performing amongst the best in emotion and pleasantness.

We didn't present the results from the 2016 Blizzard Challenge for a number of reasons: first, the number of participants between 2016 and 2017 remained the same and the performance of our system compared to other challenge participants is quite similar. Second, there is no easy way to compare the performance of our system from 2016 and 2017, as the test data used is different and the 2017 Blizzard Challenge training data also includes the test data released for 2016. Further, we have not done any evaluations using our proposed systems from Chapter 7 or Chapter 8 and compare with the 2017 CSTR submission. We performed synthesis of few audiobook paragraphs with the same unit database and found performance to be very similar (through informal listening) and hence didn't investigate much in this direction.

9.5 Conclusions

For the 2017 CSTR Blizzard Challenge entry, the hybrid system submitted for 2016 submission (Merritt et al., 2016b) was optimized in acoustic modelling for better prediction of F0 and performed smoothing between joins.

The results of the evaluation are on the whole very positive, but there are still a number of potential future improvements which could be made to the hybrid synthesis system described here. These include adopting consistent lexicon-lookup for both the SPSS and unit selection systems, making use of the same acoustic features for both join and target cost, prediction of phrase breaks, and the explicit inclusion of predicted duration in the unit selection synthesis target cost.

Chapter 10

Conclusions and future work

The first section of this chapter provides a summary of main contributions from this thesis. The code and recipes for reproducing the results of the main contributions are detailed in the following section. Finally, the chapter concludes with future work focusing on modelling limited amount of data for audiobooks with end-to-end neural text-to-speech methods (such as Tacotron).

10.1 Main contributions

The first part of this thesis aims at investigating two important factors of prosody: intonation and duration through individual studies. Motivated by the findings from Part-I and through some speculations drawn from other work, the second part of this thesis aims to generate more natural prosody making use of linguistic features and other suprasegmental features at their natural timescales in statistical parametric speech synthesis.

The overall contributions and conclusions of this thesis and its corresponding chapters are summarized below:

- **Template based approach to prosody modelling**
  - Chapter 4 presented an approach for unsupervised clustering of F0 contours at syllable-level using discrete cosine transformation and demonstrated the use of LSTM-CTC models for classification of syllable-level templates from phone-level linguistic features.
  - The use of syllable-level templates for F0 reconstruction may not be ideal and probably should be used as additional input features for

F0 prediction. We hypothesize that the F0 templates derived at syllable level or word level may also be used for other speech tasks such as inverse text normalization (for prediction of punctuation marks) and emotion recognition or generation.

- **Median based duration generation**
  - Frame-level duration modelling using LSTM-RNNs and synthesising speech based on the median predicted duration is presented in Chapter 5.
  - The DNN methods utilized for prediction of median based duration (such as using L2 loss instead of L1 or cross-entropy during training and not using autoregressive networks) may not be optimal. However, it's worth investigating frame-based models (such as Pole model and Ramp model) for duration prediction in encoder-decoder based approaches. Recently, neural text-to-speech approaches utilized similar such models for end of sentence prediction (Shen et al., 2017; Latorre et al., 2019; Prateek et al., 2019) and may also be extended for prediction of phone labels (useful for inserting pauses, and speech fillers between words).
- **Encoder-decoder approaches for SPSS**
  - A novel hierarchical encoder-decoder architecture for parametric speech synthesis to process linguistic features at their natural timescales is presented in Chapter 7. Similarly, an investigation of a multi-scale parallel encoder-decoder architecture for parametric speech synthesis is also carried in Chapter 8.
  - Although the subjective listening test with the hierarchical encoder demonstrated a significant preference over baseline ($\alpha = 0.05$), the difference is rather marginal. Also, the speech samples evaluated in the listening test were generated using natural durations. However, we speculate that both hierarchical and multi-scale parallel encoders have potential to generate more natural and dynamic prosody but the performance of these architectures in this thesis may be hindered by sub-optimal utilization of neural network layers and poor optimization of training through model upsampling. Recently, neural

text-to-speech approaches utilized linguistically driven hierarchical conditional variational network for varying prosody (Kenter et al., 2019) and multi-scale encoder conditioning for modelling speaking styles and improving naturalness (Prateek et al., 2019; Ming et al., 2019).

- **Hybrid unit selection system for audiobooks**
  - An application of parametric guided unit-selection approach to achieve high degree of naturalness for children's storybooks is presented in Chapter 9. Hybrid unit-selection system performed well overall by combining the benefits of unit selection and parametric synthesis even for expressive audiobooks.
  - Although hybrid unit selection systems can produce better segmental quality by concatenating appropriate units, the generated speech is sporadically unintelligible with respect to unseen contexts. Also, the generated prosody is often inappropriate for the given context and sounds appalling. Based on recent work, we speculate that neural vocoder approaches (Shen et al., 2017; Lorenzo-Trueba et al., 2018) may be more suitable for generating natural prosody in children's storybooks.
- **Merlin speech synthesis toolkit**
  - The code for duration modelling, encoder-decoder paradigms, batch-training of LSTM-RNNs is made open-source within the Merlin toolkit. The parametric-guided unit selection system was submitted for Blizzard Challenge 2017. The recipes for the system along with benchmarks used in the challenge are made available through the Merlin toolkit.
  - By making the code and recipes used for training models open-source, we hope that it enables various academic and research groups to replicate the results and benefit from it.

10.2 Reproducibility

The code and recipes for various modules in this thesis are made open-source within the Merlin toolkit[1] (Wu et al., 2016; Ronanki et al., 2016d).

- The code for unsupervised clustering of syllable templates is available in a Github repository (Ronanki, 2015).
- The scripts for building a new voice is provided within the Merlin toolkit (Ronanki, 2016).
- The recipe for building encoder-decoder models with Blizzard Challenge 2017 data is provided in Merlin (Ronanki, 2017b).
- The recipe for building DNN-guided unit-selection system is also provided in Merlin (Ronanki, 2017a).

10.3 Future work

In this section, we suggest directions for future research focusing on modelling limited amounts of data for audiobooks. The ideas proposed here aim at achieving natural generation of prosody in text-to-speech synthesis with end-to-end neural methods and go beyond the scope of this thesis.

Sequence-to-sequence modelling:

In Chapter 8, we made some preliminary experiments using a multi-scale parallel encoder with hidden-feature upsampling and the results suggest that the upsampling is the bottleneck to achieving a high degree of naturalness, along with the vocoder. We also investigated an attention layer replacing upsampling but didn't achieve a stable alignment model.

Although end-to-end neural TTS systems (such as Tacotron2) achieve state-of-the-art performance, they still require a considerable amount of data (more than 20 hours) for building a stable alignment model. The variability of audiobooks makes it hard to get good alignments, compared to carefully recorded utterances typically used in speech synthesis databases.

[1] https://github.com/CSTR-Edinburgh/merlin/

Tachibana et al. (2017) introduced *guided attention* which places constraints on the attention matrix to prompt it to be monotonic. The loss function computed on the attention matrix is jointly optimized with the main loss (on acoustic parameters estimation) with equal weight. It was claimed that with guided attention loss, the alignment becomes correct within a few iterations of training. Although guided attention was introduced to reduce the amount of time to train a neural system, it might also be useful in the context of limited amount of training data for audiobooks.

Transfer learning:

The work in this thesis is limited to single-speaker training in the framework of statistical parametric speech synthesis. From Chapter 7 and Chapter 8, it is learnt that the improvements are limited by separately trained and optimized models. The joint optimization of forced-alignment model and acoustic model may be required to achieve high degree of robustness. However, it would require a large amount of data to train such an end-to-end system from scratch. An alternate approach is to do transfer learning from a pre-trained system (which was trained on an abundance of data). This direction of work was never attempted in this thesis but might be required in the case of insufficient data for training.

In such cases, fine-tuning remains to be the conventional method of choice for transfer learning: a model is pre-trained on a large amount of data in a source domain, and the output layers of the model are adapted to the target domain. In neural text-to-speech systems, the source domain can be selected in two ways: single-speaker domain or multi-speaker domain.

In a single-speaker source domain, the network is pre-trained on a large amount of data to learn the alignment from text-based input features. The target speaker can be trained using two approaches. The first approach is to initialise the weights of the network from a pre-trained source model. The network is then trained either with weights from all the layers updated or by freezing the layers in the encoder so that only the layers in the decoder are updated to adapt to the target speaker. The second approach is to do transfer learning by reducing the amount of data from source speaker and increase the amount of data for target speaker after the source model is trained for a certain number of epochs. The transfer of data can be done in a similar way to scheduled decaying of learning rate (e.g., step-decay or exponential decay).

There has been a considerable amount of work done in a multi-speaker domain for neural text-to-speech synthesis. Deep voice 2 (Arık et al., 2017) introduced multi-speaker neural Text-to-Speech system which can learn hundreds of unique voices from less than half an hour of data per speaker while achieving high quality audio synthesis. In a similar direction, Jia et al. (2018) described a multi-speaker system that is able to generate high-quality speech in the voice of different speakers, including those unseen during training. However, the nuances of prosody and style may not be learned unless the network is adapted to the target speaker.

The multi-speaker approaches mentioned above have achieved state-of-the-art performance by generating high quality audio with limited amount of data from each speaker. However, they are yet to be tested in the context of audiobook domain. The multi-speaker systems can be trained with multiple audiobooks to achieve higher robustness against the variability seen in audiobooks and then the model can also be used for transfer learning for a target speaker with limited amount of training data.

Long-form reading:

The audiobook sentences are in general very long compared to isolated sentences that are typically used in speech synthesis databases. The conventional attention mechanisms used in end-to-end systems such as Bahadanu attention (Bahdanau et al., 2015), location-sensitive attention (Chorowski et al., 2015; Shen et al., 2017) may not be stable enough when modelling utterances that are beyond 30 seconds. Therefore, they are usually chopped into multiple sentences of shorter segments based on punctuation marks.

Baljekar (2018) investigated different attention mechanisms with sequence-to-sequence models for long-form reading – attention mechanisms such as multi-hop and multi-head attention were used to learn different aspects of the utterance. Recently, Li et al. (2018) introduced a multi-head mechanism replacing traditional RNNs in a Tacotron-like architecture. The multi-head attention splits the attention into several subspaces so that it can model different aspects of the utterance and is quite effective for synthesizing prosody when the utterances are long. Further investigation into these attention mechanisms is required for modelling prosody of audiobook paragraphs.

10.4 Concluding remarks

The thesis investigated prosody generation for parametric speech synthesis from different standpoints. Although the methods are not always successful, we provided key insights for improving the generation of natural prosody. This was given through the contributions made in the first part of this thesis by conducting independent studies of intonation and duration modelling and also through the contributions made in second part of this thesis by investigating novel encoder-decoder architectures in the framework of statistical parametric speech synthesis.

High-quality generation of speech prosody for audiobooks and controllability remain the fundamental problems in neural text-to-speech systems. A stronger understanding of the speech variability in audiobooks is required to bridge the gap between natural and synthetic speech. This thesis has sought to provide insights into novel encoder-decoder architectures and hopefully foster a discussion that could lead to more natural synthetic speech for audiobooks.

Appendix A

Linguistic Features

A.1 Binary linguistic features

- **Phone-level features (348)**
  - identity of phoneme (quinphone)
  - position and manner of articulation of current phoneme
- **Syllable-level features (57)**
  - identity of the nucleus of current syllable
  - articulatory features of the nucleus of current syllable
- **Word-level features (33)**
  - guessed parts-of-speech of word (previous, current, next)

A.2 Numerical linguistic features

- **Phone-level features (2)**
  - position of current phoneme in syllable (forward, backward)
- **Syllable-level features (21)**
  - syllable stress (previous, current, next)
  - syllable accent (previous, current, next)
  - number of phonemes in syllable (previous, current, next)

  - position of current syllable in current word (forward, backward)
  - position of current syllable in current phrase (forward, backward)
  - number of stressed syllables before current syllable in current phrase
  - number of stressed syllables after current syllable in current phrase
  - number of accented syllables before current syllable in current phrase
  - number of accented syllables after current syllable in current phrase
  - number of syllables from previous stressed syllable
  - number of syllables to next stressed syllable
  - number of syllables from previous accented syllable
  - number of syllables to next accented syllable

- **Word-level features (9)**
  - number of syllables in word (previous, current, next)
  - position of current word in current phrase (forward, backward)
  - number of content words before current word in current phrase
  - number of content words after current word in current phrase
  - number of words from previous content word
  - number of words to next content word

- **Phrase-level features (8)**
  - number of syllables in phrase (previous, current, next)
  - number of words in phrase (previous, current, next)
  - position of current phrase in utterance (forward, backward)

- **Utterance-level features (3)**
  - number of syllables in utterance
  - number of words in utterance
  - number of phrases in utterance

Bibliography

Agüero, P. D., Wimmer, K., and Bonafonte, A. (2004). Automatic analysis and synthesis of Fujisaki's intonation model for TTS. In *Speech Prosody*.

Anumanchipalli, G. K., Oliveira, L. C., and Black, A. W. (2011). A statistical phrase/accent model for intonation modeling. In *Proc. Interspeech*, pages 1813–1816.

Arik, S. O., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Raiman, J., Sengupta, S., et al. (2017). Deep voice: Real-time neural text-to-speech. *arXiv preprint arXiv:1702.07825*.

Arık, S. O., Diamos, G., Gibiansky, A., Miller, J., Peng, K., Ping, W., Raiman, J., and Zhou, Y. (2017). Deep voice 2: Multi-speaker neural text-to-speech. *arXiv preprint arXiv:1705.08947*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Baljekar, P. (2018). *Speech Synthesis from Found Data*. PhD thesis, School of Computer Science, Carnegie Mellon University.

Basu, A., Harris, I. R., Hjort, N. L., and Jones, M. C. (1998). Robust and efficient estimation by minimising a density power divergence. *Biometrika*, 85(3):549–559.

Basu, S. and DasGupta, A. (1997). The mean, median, and mode of unimodal distributions: a characterization. *Theory Probab. Appl.*, 41(2):210–223.

Bisani, M. and Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Commun.*, 50(5):434–451.

Black, A., Taylor, P., Caley, R., Clark, R., Richmond, K., King, S., Strom, V., and Zen, H. (2001). The Festival Speech Synthesis System, Version 1.4.2. *Unpublished document available via http://www.cstr.ed.ac.uk/projects/festival.html*.

Black, A. W. and Muthukumar, P. K. (2015). Random forests for statistical speech synthesis. In *Proc. Interspeech*, pages 1211–1215.

Black, A. W. and Taylor, P. (1997a). Automatically clustering similar units for unit selection in speech synthesis. In *Proc. Eurospeech*, pages 601–604.

Black, A. W. and Taylor, P. A. (1997b). The Festival Speech Synthesis System: System documentation. Technical Report HCRC/TR-83, Human Commununciation Research Centre, University of Edinburgh, Scotland, UK. Avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

Black, A. W., Zen, H., and Tokuda, K. (2007). Statistical parametric speech synthesis. In *Proc. ICASSP*, volume 4, pages 1229–1232.

Bruguier, A., Zen, H., and Arkhangorodsky, A. (2018). Sequence-to-sequence neural network model with 2D attention for learning Japanese pitch accents. In *Proc. Interspeech*, pages 1284–1287.

BS.1534-3, I. R. I.-R. (2015). Method for the subjective assessment of intermediate quality levels of coding systems. *International Telecommunication Union Radiocommunication Sector*.

Bulyko, I. and Ostendorf, M. (2001). Joint prosody prediction and unit selection for concatenative speech synthesis. In *Proc. ICASSP*, volume 2, pages 781–784.

Campbell, W. N. (1989). Syllable-level duration determination. In *Proc. Eurospeech*, pages 2698–2701.

Capes, T., Coles, P., Conkie, A., Golipour, L., Hadjitarkhani, A., Hu, Q., Huddleston, N., Hunt, M., Li, J., Neeracher, M., et al. (2017). Siri on-device deep learning-guided unit selection text-to-speech system. In *Proc. Interspeech*, pages 4011–4015.

Carlson, R. and Granstrom, B. (1976). A text-to-speech system based entirely on rules. In *Proc. ICASSP*, volume 1, pages 686–688.

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proc. ICASSP*, pages 4960–4964.

Chen, L.-H., Raitio, T., Valentini-Botinhao, C., Ling, Z.-H., and Yamagishi, J. (2015). A deep generative architecture for postfiltering in statistical parametric speech synthesis. *IEEE T. Audio Speech*, 23(11):2003–2014.

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. *arXiv preprint arXiv:1406.1078*.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *Proc. Advances in neural information processing systems*, pages 577–585.

Clark, R. A., Podsiadlo, M., Fraser, M., Mayo, C., and King, S. (2007a). Statistical analysis of the Blizzard Challenge 2007 listening test results. *Proc. Blizzard Challenge Workshop*.

Clark, R. A., Richmond, K., and King, S. (2004). Festival 2–build your own general purpose unit selection speech synthesiser. In *Proc. SSW*.

Clark, R. A., Richmond, K., and King, S. (2007b). Multisyn: Open-domain unit selection for the Festival speech synthesis system. *Speech Commun.*, 49(4):317–330.

Dall, R. and Gonzalvo, X. (2016). JNDSLAM: A SLAM extension for speech synthesis. In *Proc. Speech Prosody*, pages 1024–1028.

Dall, R., Wester, M., and Corley, M. (2014). The effect of filled pauses and speaking rate on speech comprehension in natural, vocoded and synthetic speech. In *Proc. Interspeech*, pages 56–60.

Donovan, R. E. and Eide, E. (1998). The IBM trainable speech synthesis system. In *Proceedings of Int. Conf. Spoken Language Processing*.

Eddy, S. R. et al. (1995). Multiple alignment using hidden Markov models. In *Ismb*, volume 3, pages 114–120.

Eguchi, S. and Kano, Y. (2001). Robustifying maximum likelihood estimation. Technical Report Research Memo 802, Institute of Statistical Mathematics, Tokyo, Japan.

Escudero, D., Cardeñoso, V., and Bonafonte, A. (2002). Corpus based extraction of quantitative prosodic parameters of stress groups in Spanish. In *Proc. ICASSP*, volume 1, pages I–481.

Espic, F., Valentini-Botinhao, C., and King, S. (2017). Direct modelling of magnitude and phase spectra for statistical parametric speech synthesis. In *Proc. Interspeech*, Stochohlm, Sweden.

Fan, X., Monti, E., Mathias, L., and Dreyer, M. (2017). Transfer learning for neural semantic parsing. *arXiv preprint arXiv:1706.04326*.

Fan, Y., Qian, Y., Xie, F.-L., and Soong, F. K. (2014). TTS synthesis with bidirectional LSTM based recurrent neural networks. In *Proc. Interspeech*, pages 1964–1968.

Fernandez, R., Rendel, A., Ramabhadran, B., and Hoory, R. (2014). Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks. In *Proc. Interspeech*, pages 2268–2272.

Fernandez, R., Rendel, A., Ramabhadran, B., and Hoory, R. (2015). Using deep bidirectional recurrent neural networks for prosodic-target prediction in a unit-selection text-to-speech system. In *Proc. Interspeech*, pages 1606–1610.

Fitt, S. and Richmond, K. (2006). Redundancy and productivity in the speech technology lexicon - can we do better? In *Proc. Interspeech*.

Fons-ant, H. and Naessentialr, S. (1969). A model for the synthesis of pitch contours of connected speech. *annual report of the engineering research institute*, 23.

Fonseca De Sam Bento Ribeiro, M. (2018). Suprasegmental representations for the modeling of fundamental frequency in statistical parametric speech synthesis. *The University of Edinburgh*.

Fujisaki, H. and Hirose, K. (1982). Modelling the dynamic characteristics of voice fundamental frequency with application to analysis and synthesis of intonation. In *Proceedings of 13th International Congress of Linguists*, pages 57–70.

Fujisaki, H., Ohno, S., and Wang, C. (1998). A command-response model for F0 contour generation in multilingual speech synthesis. In *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*.

Graves, A. (2012). Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proc. ICML*, pages 369–376.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Greenwood, A. R. (1997). Articulatory speech synthesis using diphone units. In *Proc. ICASSP*, volume 3, pages 1635–1638.

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

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

Henter, G. E., Ronanki, S., Watts, O., Wester, M., Wu, Z., and King, S. (2016). Robust TTS duration modelling using DNNs. In *Proc. ICASSP*, volume 41, pages 5130–5134, Shanghai, China.

Hershey, J. R. and Olsen, P. A. (2007). Approximating the Kullback-Leibler divergence between Gaussian mixture models. In *Proc. ICASSP*.

Hill, J. H. (2002). "Expert rhetorics" in advocacy for endangered languages: Who is listening, and what do they hear? *Journal of linguistic anthropology*, 12(2):119–133.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scand. J. Stat.*, 6(2):65–70.

Hsia, C.-C., Wu, C.-H., and Wu, J.-Y. (2010). Exploiting prosody hierarchy and dynamic features for pitch modeling and generation in hmm-based speech synthesis. *IEEE T. Audio Speech*, 18(8):1994–2003.

Hu, Q., Richmond, K., Yamagishi, J., and Latorre, J. (2013). An experimental comparison of multiple vocoder types. In *Proc. SSW*.

Hu, Y. J. and Ling, Z. H. (2016). DBN-based spectral feature representation for statistical parametric speech synthesis. *IEEE Signal Processing Letters*, 23(3):321–325.

Huber, K. (1990). A statistical model of duration control for speech synthesis. In *Proc. EUSIPCO*, pages 1127–1130.

Hunt, A. J. and Black, A. W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In *Proc. ICASSP*, volume 1, pages 373–376.

Ito, K. (2017). The LJ speech dataset. https://keithito.com/LJ-Speech-Dataset/.

Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Moreno, I. L., et al. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *arXiv preprint arXiv:1806.04558*.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*, volume 3. Pearson London:.

Kaneko, T., Kameoka, H., Hojo, N., Ijima, Y., Hiramatsu, K., and Kashino, K. (2017). Generative adversarial network-based postfilter for statistical parametric speech synthesis. In *Proc. ICASSP*, pages 4910–4914.

Kawahara, H. (2006). STRAIGHT, exploitation of the other aspect of VOCODER: Perceptually isomorphic decomposition of speech sounds. *Acoustic Science Technology*, 27(6):349–353.

Kenter, T., Wan, V., Chan, C.-A., Clark, R., and Vit, J. (2019). CHiVE: Varying prosody in speech synthesis with a linguistically driven dynamic hierarchical conditional variational network. In *International Conference on Machine Learning*, pages 3331–3340.

King, S. (2011). An introduction to statistical parametric speech synthesis. *Sadhana*, 36(5):837–852.

King, S. (2014). Measuring a decade of progress in text-to-speech. *Loquens*, 1(1).

King, S., Crumlish, J., Martin, A., and Wihlborg, L. (2018). The Blizzard Challenge 2018. In *Proc. Blizzard Challenge Workshop*.

King, S. and Karaiskos, V. (2013). The Blizzard Challenge 2013. In *Proc. Blizzard Challenge Workshop*.

King, S. and Karaiskos, V. (2016). The Blizzard Challenge 2016. In *Proceedings of Blizzard Challenge Workshop*.

King, S., Wihlborg, L., and Guo, W. (2017). The Blizzard Challenge 2017. In *Proceedings of Blizzard Challenge Workshop*.

Klatt, D. (1980). Software for a cascade/parallel formant synthesizer. *The Journal of the Acoustical Society of America*, 67.

Klatt, D. H. (1987). Review of text-to-speech conversion for English. *J. Acoust. Soc. Am.*, 82(3):737–793.

Klimkov, V., Moinet, A., Nadolski, A., and Drugman, T. (2018). Parameter generation algorithms for text-to-speech synthesis with recurrent neural networks. In *2IEEE workshop on Spoken Language Technology (SLT)*, pages 626–631.

Klimkov, V., Nadolski, A., Moinet, A., Putrycz, B., Barra-Chicote, R., Merritt, T., and Drugman, T. (2017). Phrase break prediction for long-form reading TTS: exploiting text structure information. In *Proc. Interspeech*, pages 1064–1068.

Kominek, J. and Black, A. W. (2004). The CMU Arctic speech databases. In *Fifth ISCA workshop on speech synthesis*.

Kominek, J., Schultz, T., and Black, A. W. (2008). Synthesizer voice quality of new languages calibrated with mean mel cepstral distortion. In *Spoken Languages Technologies for Under-Resourced Languages*.

Kubichek, R. (1993a). Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, volume 1, pages 125–128.

Kubichek, R. F. (1993b). Mel-cepstral distance measure for objective speech quality assessment. In *Proc. IEEE Pac. Rim Conf. Commun. Comput. Signal Process.*, volume 1, pages 125–128.

Latorre, J. and Akamine, M. (2008). Multilevel parametric-base F0 model for speech synthesis. In *Proc. Interspeech*, pages 2274–2277.

Latorre, J., Lachowicz, J., Lorenzo-Trueba, J., Merritt, T., Drugman, T., Ronanki, S., and Klimkov, V. (2019). Effect of data reduction on sequence-to-sequence neural TTS. In *Proc. ICASSP*, pages 7075–7079.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12.

Li, N., Liu, S., Liu, Y., Zhao, S., Liu, M., and Zhou, M. (2018). Neural speech synthesis with Transformer network. *arXiv preprint arXiv:1809.08895*.

Ling, Z., Deng, L., and Yu, D. (2013). Modeling spectral envelopes using restricted Boltzmann machines and deep belief networks for statistical parametric speech synthesis. *IEEE T. Audio Speech*, 21(10):2129–2139.

López, G., Quesada, L., and Guerrero, L. A. (2017). Alexa vs. Siri vs. Cortana vs. Google Assistant: a comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics*, pages 241–250.

Lorenzo-Trueba, J., Drugman, T., Latorre, J., Merritt, T., Putrycz, B., and Barra-Chicote, R. (2018). Robust universal neural vocoding. *arXiv preprint arXiv:1811.06292*.

MacGillivray, H. L. (1981). The mean, median, mode inequality and skewness for a class of densities. *Australian Journal of Statistics*, 23(2):247–250.

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., and Bengio, Y. (2017). SampleRNN: An unconditional end-to-end neural audio generation model. In *Proc. ICLR*, Toulon, France.

Merritt, T., Clark, R. A. J., Wu, Z., Yamagishi, J., and King, S. (2016a). Deep neural network-guided unit selection synthesis. In *Proc. ICASSP*.

Merritt, T., Ronanki, S., Watts, O., , Wu, Z., and Clark, R. A. J. (2016b). The CSTR entry to the Blizzard Challenge 2016. In *Proc. Blizzard Challenge Workshop*.

Ming, H., He, L., Guo, H., and Soong, F. (2019). Feature reinforcement with word embedding and parsing information in neural TTS. *arXiv preprint arXiv:1901.00707*.

Mixdorff, H. (2015). Extraction, analysis and synthesis of Fujisaki model parameters. In *Speech Prosody in Speech Synthesis: Modeling and generation of prosody for high quality and flexible speech synthesis*, pages 35–47.

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*, 99(7):1877–1884.

Muthukumar, P. K. and Black, A. W. (2016). Recurrent neural network postfilters for statistical parametric speech synthesis. *arXiv preprint arXiv:1601.07215*.

Nooteboom, S. (1997). The prosody of speech: melody and rhythm. *The handbook of phonetic sciences*, 5:640–673.

Obin, N., Beliao, J., Veaux, C., and Lacheret, A. (2014). SLAM: Automatic stylization and labelling of speech melody. In *Proc. Speech Prosody*, pages 246–250.

Ostendorf, M., Price, P. J., and Shattuck-Hufnagel, S. (1995). The Boston University radio news corpus. *Linguistic Data Consortium*, pages 1–19.

P.56, I. R. I.-T. (2011). Objective measurement of active speech level. *International Telecommunication Union, Telecommunication Standardization Sector*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proc. EMNLP*, pages 1532–1543.

Ping, W., Peng, K., Gibiansky, A., Arik, S., Kannan, A., Narang, S., Raiman, J., and Miller, J. (2018). Deep voice 3: Scaling text-to-speech with convolutional sequence learning. In *Proc. 6th International Conference on Learning Representations*.

Pollet, V. and Breen, A. (2008). Synthesis by generation and concatenation of multiform segments. In *Proc. Interspeech*.

Prahallad, K., Black, A. W., and Mosur, R. (2006). Sub-phonetic modeling for capturing pronunciation variations for conversational speech synthesis. In *Proc. ICASSP*, pages 853–856.

Prateek, N., Łajszczak, M., Barra-Chicote, R., Drugman, T., Lorenzo-Trueba, J., Merritt, T., Ronanki, S., and Wood, T. (2019). In other news: A bi-style text-to-speech model for synthesizing newscaster voice with limited data. *arXiv preprint arXiv:1904.02790*.

Qian, Y., Fan, Y., Hu, W., and Soong, F. K. (2014). On the training aspects of deep neural network (DNN) for parametric TTS synthesis. In *Proc. ICASSP*, pages 3829–3833.

Qian, Y., Soong, F. K., and Yan, Z.-J. (2013). A unified trajectory tiling approach to high quality speech rendering. *IEEE T. Audio Speech*, 21(2):280–290.

Qian, Y., Wu, Z., Gao, B., and Soong, F. K. (2011). Improved prosody generation by maximizing joint probability of state and longer units. *IEEE T. Audio Speech*, 19(6):1702–1710.

Raitio, T., Suni, A., Yamagishi, J., Pulakka, H., Nurminen, J., Vainio, M., and Alku, P. (2010). HMM-based speech synthesis utilizing glottal inverse filtering. *IEEE T. Audio Speech*, 19(1):153–165.

Rao, K., Peng, F., Sak, H., and Beaufays, F. (2015). Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Proc. ICASSP*, pages 4225–4229.

Ribeiro, M. S. and Clark, R. A. J. (2015). A multi-level representation of $f0$ using the continuous wavelet transform and the discrete cosine transform. In *Proc. ICASSP*, pages 4909–4913.

Ribeiro, M. S., Watts, O., and Yamagishi, J. (2016). Parallel and cascaded deep neural networks for text-to-speech synthesis. In *Proc. SSW*, pages 107–112.

Ribeiro, M. S., Watts, O., and Yamagishi, J. (2017a). Learning word vector representations based on acoustic counts. In *Proc. Interspeech*.

Ribeiro, M. S., Watts, O., and Yamagishi, J. (2017b). Learning word vector representations based on acoustic counts. In *Proc. Interspeech*, Stockholm, Sweden.

Ribeiro, M. S., Yamagishi, J., and Clark, R. A. J. (2015). A perceptual investigation of wavelet-based decomposition of $f0$ for text-to-speech synthesis. In *Proc. Interspeech*, volume 16, pages 1586–1590.

Richmond, K., Clark, R. A., and Fitt, S. (2009). Robust LTS rules with the Combilex speech technology lexicon. In *Proc. Interspeech*.

Ronanki, S. (2015). Github repository for hierarchical clustering. https://github.com/ronanki/Hybrid_prosody_model/blob/master/src/py/clustering.py.

Ronanki, S. (2016). Merlin recipe for building custom voice using SPSS. https://github.com/CSTR-Edinburgh/merlin/tree/master/egs/build_your_own_voice/s1.

Ronanki, S. (2017a). Merlin recipe for hybrid unit selection synthesis. https://github.com/CSTR-Edinburgh/merlin/tree/master/egs/hybrid_synthesis/s1.

Ronanki, S. (2017b). Merlin recipe for the Blizzard Challenge 2017 voice building. https://github.com/CSTR-Edinburgh/merlin/tree/master/egs/fls_blizzard2017.

Ronanki, S., Henter, G. E., Wu, Z., and King, S. (2016a). A template-based approach for speech synthesis intonation generation using LSTMs. In *Proc. Interspeech*, San Francisco, USA.

Ronanki, S., Reddy, S., Bollepalli, B., and King, S. (2016b). DNN-based speech synthesis for Indian languages from ASCII text. In *Proc. SSW9*, Sunnyvale, USA.

Ronanki, S., Ribeiro, S., Espic, F., and Watts, O. (2017a). The CSTR entry to the Blizzard Challenge 2017. In *Proc. Blizzard Challenge Workshop*.

Ronanki, S., Watts, O., and King, S. (2017b). A hierarchical encoder-decoder model for statistical parametric speech synthesis. In *Proc. Interspeech*, pages 1133–1137.

Ronanki, S., Watts, O., King, S., and Henter, G. E. (2016c). Median-based generation of synthetic speech durations using a non-parametric approach. In *IEEE workshop on Spoken Language Technology*, San Diego, California.

Ronanki, S., Wu, Z., Watts, O., and King, S. (2016d). A demonstration of the Merlin open source neural network speech synthesis system. In *Proc. SSW9*, Sunnyvale, USA.

Sagisaka, Y., Kaiki, N., Iwahashi, N., and Mimura, K. (1992). ATR v-TALK speech synthesis system. In *Proceedings of Int. Conf. Spoken Language Processing*, pages 483–486.

Scully, C. (1990). Articulatory synthesis. In *Speech production and speech modelling*, pages 151–186. Springer.

Sejnowski, T. J. and Rosenberg, C. R. (1988). Neurocomputing: Foundations of research. chapter NETtalk: A Parallel Network That Learns to Read Aloud, pages 661–672. Cambridge, MA, USA.

Shattuck-Hufnagel, S. and Turk, A. E. (1996). A prosody tutorial for investigators of auditory sentence processing. *Journal of psycholinguistic research*, 25(2):193–247.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R., et al. (2017). Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. *arXiv preprint arXiv:1712.05884*.

Silverman, K., Beckman, M., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J., and Hirschberg, J. (1992). ToBI: A standard for labeling English prosody. In *Second international conference on spoken language processing*.

Skerry-Ryan, R., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., Weiss, R. J., Clark, R., and Saurous, R. A. (2018). Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron. *arXiv preprint arXiv:1803.09047*.

Smith III, J. O. and Abel, J. S. (1999). Bark and ERB bilinear transforms. *IEEE T. Speech Audi. P.*, 7(6):697–708.

Sorin, A., Shechtman, S., and Pollet, V. (2011). Uniform speech parameterization for multi-form segment synthesis. In *Proc. Interspeech*, pages 337–340.

Sproat, R. (1996). Multilingual text analysis for text-to-speech synthesis. *Natural Language Engineering*, 2(4):369–380.

Suni, A., Aalto, D., Raitio, T., Alku, P., and Vainio, M. (2013). Wavelets for intonation modeling in HMM speech synthesis. In *Proc. SSW*, volume 8, pages 285–290.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tachibana, H., Uenoyama, K., and Aihara, S. (2017). Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. *arXiv preprint arXiv:1710.08969*.

Takaki, S. and Yamagishi, J. (2016). A deep auto-encoder based low-dimensional feature extraction from FFT spectral envelopes for statistical parametric speech synthesis. In *Proc. ICASSP*, pages 5535–5539.

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

Taylor, P. (1998). The Tilt intonation model. In *International Conference on Spoken Language Processing*.

Taylor, P. (2000). Analysis and synthesis of intonation using the Tilt model. *The Journal of the Acoustical Society of America*, 107(3):1697–1714.

Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge university press.

Teutenberg, J., Watson, C., and Riddle, P. (2008). Modelling and synthesising F0 contours with the discrete cosine transform. In *Proc. ICASSP*, pages 3973–3976.

Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., and Kitamura, T. (2000). Speech parameter generation algorithms for HMM-based speech synthesis. In *Proc. ICASSP*, volume 3, pages 1315–1318.

Tuerk, C. and Robinson, T. (1993). Speech synthesis using artificial neural networks trained on cepstral coefficients. In *Third European Conference on Speech Communication and Technology*.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Association for computational linguistics*, pages 384–394, Stroudsburg, PA, USA.

Turk, A. and Shattuck-Hufnagel, S. (2014). Timing in talking: what is it used for, and how is it controlled? *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1658):20130395.

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*.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., et al. (2017). Parallel WaveNet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*.

Wagner, M. and Watson, D. G. (2010). Experimental and theoretical advances in prosody: A review. *Language and cognitive processes*, 25(7-9):905–945.

Wang, P., Qian, Y., Soong, F. K., He, L., and Zhao, H. (2015). Word embedding for recurrent neural network based TTS synthesis. In *Proc. ICASSP*, pages 4879–4883.

Wang, W., Xu, S., and Xu, B. (2016). First step towards end-to-end parametric TTS synthesis: Generating spectral parameters with neural attention. In *Proc. Interspeech*, pages 2243–2247.

Wang, X., Takaki, S., and Yamagishi, J. (2017a). An autoregressive recurrent mixture density network for parametric speech synthesis. In *Proc. ICASSP*, pages 4895–4899.

Wang, X., Takaki, S., and Yamagishi, J. (2017b). An RNN-based quantized F0 model with multi-tier feedback links for text-to-speech synthesis. In *Proc. Interspeech*, pages 1059–1063.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al. (2017c). Tacotron: A fully end-to-end text-to-speech synthesis model. *arXiv preprint arXiv:1703.10135*.

Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R., Battenberg, E., Shor, J., Xiao, Y., Ren, F., Jia, Y., and Saurous, R. A. (2018). Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. *arXiv preprint arXiv:1803.09017*.

Watts, O., Henter, G. E., Merritt, T., Wu, Z., and King, S. (2016). From HMMs to DNNs: where do the improvements come from? In *Proc. ICASSP*, pages 5505–5509.

Watts, O., Ronanki, S., Wu, Z., Raitio, T., and Suni, A. (2015a). The NST–GlottHMM entry to the Blizzard Challenge 2015. In *Proc. Blizzard Challenge Workshop*.

Watts, O., Stan, A., Clark, R., Mamiya, Y., Giurgiu, M., Yamagishi, J., and King, S. (2013). Unsupervised and lightly-supervised learning for rapid construction of TTS systems in multiple languages from 'found' data: evaluation and analysis. In *Proc. SSW8*, pages 121–126, Barcelona, Spain.

Watts, O., Wu, Z., and King, S. (2015b). Sentence-level control vectors for deep neural network speech synthesis. In *Proc. Interspeech*, pages 2217–2221.

Weijters, T. and Thole, J. (1993). Speech synthesis with artificial neural networks. In *Neural Networks, 1993., IEEE International Conference on*, pages 1764–1769.

Wu, Z. and King, S. (2016). Investigating gated recurrent networks for speech synthesis. In *Proc. ICASSP*, pages 5140–5144.

Wu, Z., Qian, Y., Soong, F. K., and Zhang, B. (2008). Modeling and generating tone contour with phrase intonation for mandarin chinese speech. In *2008 6th International Symposium on Chinese Spoken Language Processing*, pages 1–4. IEEE.

Wu, Z., Swietojanski, P., Veaux, C., Renals, S., and King, S. (2015a). A study of speaker adaptation for DNN-based speech synthesis. In *Proc. Interspeech*.

Wu, Z., Valentini-Botinhao, C., Watts, O., and King, S. (2015b). Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Proc. ICASSP*, pages 4460–4464.

Wu, Z., Watts, O., and King, S. (2016). Merlin: An open source neural network speech synthesis system. In *Proc. SSW9*, Sunnyvale, USA.

Yamagishi, J. (2006). An introduction to HMM-based speech synthesis. *Technical Report*.

Yan, Z.-J., Qian, Y., and Soong, F. K. (2010). Rich-context unit selection (RUS) approach to high quality TTS. In *Proc. ICASSP*, pages 4798–4801.

Yin, X., Lei, M., Qian, Y., Soong, F. K., He, L., Ling, Z.-H., and Dai, L.-R. (2014). Modeling DCT parameterized F0 trajectory at intonation phrase level with DNN or decision tree. In *Proc. Interspeech*, pages 2273–2277.

Yin, X., Lei, M., Qian, Y., Soong, F. K., He, L., Ling, Z.-H., and Dai, L.-R. (2016). Modeling F0 trajectories in hierarchically structured deep neural networks. *Speech Commun.*, 76:82–92.

Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., and Kitamura, T. (1999). Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis. In *Proc. Eurospeech*, pages 2347–2350.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2002). The HTK book. *Cambridge university engineering department*, 3:175.

Yu, K. and Young, S. (2011). Continuous F0 modeling for HMM based statistical parametric speech synthesis. *IEEE T. Audio Speech*, 19(5):1071–1079.

Zen, H., Agiomyrgiannakis, Y., Egberts, N., Henderson, F., and Szczepaniak, P. (2016). Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices. In *Proc. Interspeech*.

Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A., and Tokuda, K. (2007). The HMM-based speech synthesis system (HTS) version 2.0. In *Proc. SSW*, volume 6, pages 294–299.

Zen, H. and Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Proc. ICASSP*, pages 4470–4474.

Zen, H. and Senior, A. (2014). Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *Proc. ICASSP*, pages 3844–3848.

Zen, H., Senior, A., and Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *Proc. ICASSP*, pages 7962–7966.

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

Zen, H., Tokuda, K., Masuko, T., Kobayashi, T., and Kitamura, T. (2004). Hidden semi-Markov model based speech synthesis. In *Proc. Interspeech*, pages 1393–1396.