Fundamental Frequency Modeling for Neural-Network-Based Statistical Parametric Speech Synthesis

Xin Wang

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

Fundamental Frequency Modeling for Neural-Network-Based Statistical Parametric Speech Synthesis

by

Xin Wang

Dissertation

submitted to the Department of Informatics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

SOKENDAI (The Graduate University for Advanced Studies)

September 2018

Committee

Advisor	Dr. Junichi Yamagishi Associate professor of SOKENDAI/National Institute of Informatics
Subadvisor	Dr. Yusuke Miyao Professor of SOKENDAI/National Institute of Informatics
Subadvisor	Dr. Isao Echizen Professor of SOKENDAI/National Institute of Informatics
Examiner	Dr. Keiichi Tokuda Professor of Nagoya Institute of Technology
Examiner	Dr. Nobuaki Minematsu Professor of The University of Tokyo

Abstract

The fundamental frequency (F0) of speech, which determines the perceived relative highness or lowness of the sound, plays an indispensable role in both the segmental and suprasegmental aspects of human languages. How to generate natural F0 contours from linguistic features of a text is one of the cruces in text-to-speech (TTS) systems, especially in the TTS system that has been able to synthesize speech with a natural segmental quality.

A TTS system may produce unnatural speech if the generated F0 contour of that speech is incompatible with the prosodic system of the language (e.g., an incorrect F0 pattern in a tonal language), or if it is bland and monotonous (e.g., an over-smoothed F0 contour). Even though a TTS system correctly plans the prosodic structure and linguistic specification for the text to be uttered, it may generate an unnatural F0 contour when the internal F0 model fails to fulfill the plan. This problem is crucial in common TTS systems that use statistical F0 models, particularly using the so-called black-box neural networks.

This thesis focuses on the neural-network-based F0 models for TTS systems, with the goal to identify potential limitations of conventional neural F0 models and propose better solutions. Specifically, this thesis treats F0 modeling as a sequential conversion problem where the input linguistic feature sequence is converted by a neural F0 model into an F0 contour frame by frame. Through interpreting and analyzing common neural F0 models or models that generate F0 with other spectral features, this thesis identifies three potential limitations: (1) it may be a suboptimal F0 modeling strategy to jointly model F0 with other acoustic features; (2) common models such as the recurrent neural network (RNN) is imperfect in modeling the temporal correlation of the F0 contour; (3) common neural F0

models are inefficient if they process the linguistic features frame by frame.

On issue (1), this thesis conducted experiments using highway neural networks and found that the network prioritized the high-dimensional spectral features over the F0. It also found that the F0 and other acoustic features required different hidden representations. These results cast doubt on the common practice to model F0 and spectral features jointly. On issue (2), this thesis used random sampling to visualize the limitation of the conventional neural F0 models such as the RNN. It then introduced the autoregressive (AR) dependency and defined a new model called shallow AR (SAR) model, which can be interpreted and extended based on the theories of digital filters and feature transformation. This thesis further identified the limitation of the SAR and proposed a deep AR (DAR) model. As experiments showed, the DAR outperformed previous models and enabled random F0 contour sampling, which has never been achieved by other F0 models. On issue (3), this thesis borrowed the idea of variational auto-encoder (VAE) and decomposed a neural F0 model into an F0 contour coding part and a linguistic association part. The coding part efficiently represents the F0 contour of a linguistic unit using one codeword, and the association part directly links the F0 code space and the linguistic space for each linguistic unit. Experiments found that the VAE-based F0 model learned an interpretable F0 code space and achieved a better objective performance than the DAR even though the VAE-based model was smaller and faster.

Although this thesis deals with the F0 and conducted experiments mainly on the English and Japanese data, it does not assume a specific linguistic theory about the F0. The proposed methods and models can hopefully be applied to other speech corpora and other acoustic feature sequences. All the models mentioned in this PhD thesis were implemented in C++/CUDA. The codes, scripts, and related slides are uploaded online as open source softwares (http://tonywangx.github.io).

Acknowledgments

I would like to express my gratitude to:

- Dr. Junichi Yamagishi, who is my supervisor. I sincerely appreciate your decision to accept me as a Ph.D. student 4 years ago (Interspeech 2014, at Singapore) and deeply thank you for all your efforts to guide me through this Ph.D. course, a period that is short in time but invaluable in life;

- Prof. Nobutaka Ono, who was my subadvisor during the first two years of the Ph.D. course at SOKENDAI. I sincerely thank you for your technical comments and suggestions on the work related to this thesis;

- Prof. Isao Echizen and Prof. Yusuke Miyao, who are my subadvisors at SOKENDAI and members of the dissertation committee. I sincerely thank you for your technical comments and suggestions on this thesis and the presentations;

- Prof. Keiichi Tokuda of Nagoya Institute of Technology, who is the member of the dissertation committee. I sincerely thank you for all your technical comments on this thesis and the presentations. I also thank you for accepting me as a short-term visiting student at NIT, during which period I started the work on Chapter 7;

- Prof. Nobuaki Minematsu of the University of Tokyo, who is the member of the dissertation committee. I sincerely thank you for all your technical comments on this thesis and the presentations, especially the issues related to the speech prosody;

- Dr. Shinji Takaki of NII, who gives me suggestions on my work and always supports my work from various perspectives during the Ph.D. course. I thank you for managing all the resources without which I could do nothing;
- Dr. Gustav Eje Henter and Dr. Jaime Lorenzo Trueba, who were project researchers at NII. I thank you for all the discussion and comments on various research topics, and also your support for my work;
- Prof. Simon King of the University of Edinburgh, who gave me constructive suggestions on the work related to Chapter 7 during his stay at NIT;
- Mr. Juvela Lauri, the Ph.D. student of Aalto University, who shared with me many wonderful ideas on speech synthesis.
- Ms. Mika Sasaki, Ms. Yuka Takedomi, and Ms. Makiko Kuwahara, who is or used to be the secretary of Yamagishi-lab. Thank you for your work without which I could not concentrate on the research work.

I also express my gratitude to:

- SOKENDAI, my official affiliation as a Ph.D. student, for recommending me as the MEXT scholarship student and creating a wonderful platform for Ph.D. students like me in national institutes of Japan;
- Ministry of Education, Culture, Sports, Science and Technology, Japan, for accepting me as a MEXT scholarship student. I couldn't complete the Ph.D. course without the support from MEXT;
- National Institute of Informatics, my physical affiliation, for providing the best research environment for me. Special thanks to the International Affairs and Education Support Team of SOKENDAI in NII for supporting my study.

Please forgive me for not being able to remember all the names to whom I owe this thesis. Finally, I thank my parents for supporting my decision to study abroad.

Contents

1 Introduction

Speech is an innate human capability for communication. A speaker encodes the message into speech by varying what is being said and how it is said. Accordingly, a listener decodes the information from the content and the form of speech. For an inanimate agent such as a machine, it must know how to speak and listen well in order to interact with human beings.

This thesis is about the techniques that enable a machine to speak naturally. Specifically, this thesis is dedicated to the methods that improve the fundamental frequency (F0) of the machine-produced speech, a property that affects both the content and form of the speech. Although this topic has been explored from other perspectives, this thesis treats it as a machine learning task and explores it on basis of neural-network-based machine learning techniques.

As an introduction, this chapter briefly explains the background of this thesis in Section 1.1. It then gives an overview of this thesis in Section 1.2, explaining the motivation and the topic, the potential issues with conventional methods and models, and the proposed solutions. The thesis outline is described in Section 1.3.

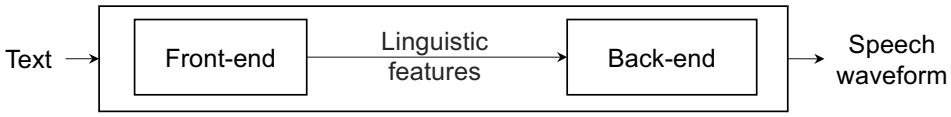


Figure 1.1: Diagram of a TTS pipeline system.

1.1 Background

The process in which machines produce human speech is called speech synthesis. Depending on the source information, it may refer to concept-to-speech synthesis (CTS) that converts abstract messages into speech [1] or text-to-speech synthesis (TTS) that reads a text string aloud [2]. This thesis is concerned with TTS, with the goal to improve TTS methods so that a machine can read the text in as natural a manner as human beings.

When a human being reads a text string, he or she may parse the text mentally, unconsciously predicting the pronunciation of each word and the manner in which speak. By manner, we mean the pace, pause, intonation, and other things rather than the word pronunciation. Given the inferred information about the text, the speaker can drive the speech organs to speak.

TTS can be implemented as a similar process. In a conventional pipeline TTS system plotted in Figure 1.1, a front-end parses the input text and plans what to speak (i.e., pronunciation) and how to speak (e.g., pace, pause, and intonation). A back-end then converts the speaking plan into a speech waveform. In TTS systems, the speaking plan is organized and represented as linguistic features.

TTS is challenging because of two difficulties, one for the front and the other for the back-end. First, it is difficult to parse the text and derive the linguistic features such as the pronunciation of abbreviation, the word to emphasize, the decision to use a question or a sarcastic voice and so on. This difficulty is unavoidable because a text may not encode all the information that decides what to speak and how to speak [3]. Even with accurate and sufficient linguistic features, the second difficulty is to convert them into a natural sounding speech waveform. After all, the linguistic features are a set of symbols or numbers while a speech waveform is a physical signal.

1.2 Thesis overview

1.2.1 Motivation

This thesis focuses on issues in the TTS back-end, assuming linguistic features from the front-end are available. More specifically, this thesis deals with the process that converts the linguistic features to an acoustic feature of the speech waveform called fundamental frequency (F0). This conversion process is referred to as F0 modeling.

The F0 is essential for TTS because it influences not only how accurately each word can be heard but also how other messages encoded in the text can be understood. As we will explain with more details in Chapter 2, the F0 is the acoustic cue that helps the listener to perceive the lexical tone and discriminate words with the same pronunciation; it helps the listener to find the beginning and end of a phrase; it hints the salient word that is emphasized or contrasted with previous words. In addition to these linguistic functions, the F0 may also help the listener to perceive the mood, the dialect, and the social status of the speaker. In general, a TTS system may only produce boring or even unintelligible speech if it lacks a good model for F0 modeling.

In the recent years, there were research works showing that a TTS system can produce speech waveforms with a natural quality [4]. However, those works tackle the TTS issues in an end-to-end paradigm, without explicitly modeling the F0 as a conventional TTS system does. Of course, the end-to-end paradigm is a promising direction for TTS research and application, but the difficulties in the pipeline TTS system still require investigation since most current TTS systems have a pipeline structure and need an F0 model.

1.2.2 Topic and scope

F0 modeling is a classical problem in TTS, or actually a cluster of problems. As mentioned above, the F0 relates not only to the segmental property but also the suprasegmental and para-linguistic aspects of speech. Researchers from various fields have worked under the name of F0 modeling but on different sub-topics, such as predicting more informative linguistic features for F0 modeling [5], finding

better F0 representations (e.g., a sequence of values measured every 5ms or more structural representation [6]), and defining better analytic and deterministic models that describe the F0 shape [7, 8].

Due to the above reason, it is not easy to pin down the core issue addressed by a paper on F0 modeling after reading it for the first time. To avoid such confusion, we now explain the topic of this thesis using a more technical wording:

> Given a sequence of linguistic features $\boldsymbol{x}_{1:T} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T\}$ provided by the TTS front-end in T frames, can we find a better statistical model $p_{\boldsymbol{\Phi}}$ to convert the linguistic feature sequences $\boldsymbol{x}_{1:T}$ into an F0 contour $\boldsymbol{o}_{1:T}$ so that the speech synthesized based on the F0 contour sounds more natural?

Note that this thesis treats the duration model as part of the front-end for the purpose of explanation. Definition of the input $\boldsymbol{x}_{1:T}$ and output $\boldsymbol{o}_{1:T}$ will be further explained in the next chapter.

Among various statistical models that can be used for F0 modeling, this thesis considers the neural-network-based ones. Although normal neural networks such as feedforward and recurrent networks appear to be deterministic, they are implicit statistical models as Chapter 3 will explain. In fact, it is based on the statistical interpretation of neural networks that this thesis identifies potential limitations of conventional neural F0 models and proposes new ones. This thesis does not consider other popular models such as hidden Markov model (HMM) because neural networks are more flexible and were found to work better on the speech corpus used in this thesis [9].

This thesis is dedicated to improving the statistical F0 models. It doesn't explore the design and acquisition of linguistic features but just uses existing front-end and the automatically derived linguistic features[1]. Neither does this thesis use other F0 representations or other expert-knowledge-based models to parameterize F0. It only considers F0 generation from linguistic features directly. Furthermore, although the data used are in a reading and neutral style, this thesis doesn't assume a particular domain of the speech data for the proposed models and methods.

[1] List of linguistic features can be found in Appendix A

1.2.3 Issues to be addressed

Using neural networks for F0 modeling is not a new idea [10, 11, 12]. Although it is straightforward and convenient to use neural networks for this task (or any task), whether the method takes full advantage of neural networks is open to question. Furthermore, it requires theoretical and empirical investigation in order to find a good network structure and specific modeling strategy for F0 modeling.

This thesis analyzes recent neural-network-based F0 modeling approaches and identifies three potential issues:

- Issue 1: whether it is appropriate to jointly model F0 and other spectral features using a normal neural network as many TTS back-ends do;
- Issue 2: whether a normal neural F0 model ignores the temporal correlation of F0 contours. If yes, how a model can learn the correlation;
- Issue 3: whether it is efficient for a neural F0 model to process linguistic features frame-by-frame. If not, how a more efficient and interpretable model can be designed.

Because the topic of this thesis is on the F0 modeling, we must answer Issue 1 first. If we simply investigate the F0 modeling part in the context of normal TTS back-ends, the unknown impact of the joint modeling may interfere with the F0 modeling performance, which may lead to unreliable results and conclusion. With Issue 1 addressed, we can focus on the F0 model. As Chapter 3 will explain, a neural network can be interpreted as a probabilistic model, in which the input features are transformed into a parameter set of a specific type of distribution for the target data. Accordingly, the performance of such a neural probabilistic model is affected by two points: 1) whether the assumed distribution is compatible with the target data; 2) whether the hidden layers are well designed to transform the input data into the parameter set. These two points motivate us to ask Issue 2 and 3 above.

1.2.4 Contribution

On the issues above, this thesis conducted new analyses and proposed new models:

- On issue 1:
  - An empirical evaluation using highway neural networks was conducted, and the results showed that a neural network prioritized the spectral features over the F0 when it jointly modeled both features as the target;
  - Analyses on the hidden features further illustrated different network behaviors in modeling the F0 and the spectral features. The results provide a rationale to separate F0 modeling from spectral feature modeling in order to improve F0 modeling performance;
- On issue 2:
  - Theoretical and empirical analyses were conducted to show how the normal neural networks ignore the temporal correlation of the target sequential data such as the F0 contour;
  - A neural F0 model called shallow autoregressive model (SAR) was proposed to amend the missing temporal correlation in F0 modeling. Interpretations based on digital filter and feature transformation were provided to understand and extend the SAR;
  - The shortcoming of the SAR was analyzed, and a deep autoregressive F0 model (DAR) was proposed. Based on quantized F0 representation and other techniques, the DAR outperformed previous models. It also generated smooth F0 contours by random sampling, which has never been achieved by other neural F0 models;
- On issue 3:
  - Based on the DAR, a new VAE-based F0 modeling framework was proposed. This framework encodes the F0 contour into a self-learned compact code space. Specifically, the encoder assigns only one codeword to each linguistic unit in the utterance no matter what the duration of that unit is. With the sparse codes and duration of units, the decoder can reconstruct the entire F0 contour accurately;

– The proposed framework further uses a separate network to learn the mapping from the linguistic features to the codewords directly for every linguistic unit. Combined with the VAE decoder, this network can be used for neural F0 modeling in TTS systems. As experiments demonstrated, this model processed the linguistic features more efficiently, learned interpretable and sparse F0 codes, and outperformed the DAR.

1.3 Outline of thesis

The thesis is organized according to the roadmap in Figure 1.2.

Chapter 2 will describe the TTS front-end and back-end used in this work. It also introduces the F0 from multiple perspectives and explains typical F0 modeling methods for TTS, including neural-network-based ones. Chapter 3 will introduce and interpret neural networks as statistical models. After that, it will re-iterate the three issues to be investigated in this thesis.

Chapter 4 to Chapter 7 will look into three issues. Specifically, Chapter 4 reports on empirical studies of the behavior of the neural network when it models F0 jointly with other spectral features. Results from this chapter motivate the way to model F0 separately from spectral features for the following chapters. Chapter 5 and 6 focus on the temporal correlation of F0 models. Particularly, Chapter 5 will analysis the shortcoming of normal neural works and define a simple and interpretable F0 model called SAR. Chapter 6 further analyzes the shortcoming of the SAR and proposes a better model called DAR for F0 modeling. Based on the DAR, Chapter 7 focuses on more efficient linguistic feature processing and introduces a variational auto-encoder (VAE)-based framework.

Chapter 8 will show the performance of TTS systems equipped with the proposed F0 models and other advanced modules. It will also explain the remaining gap between synthetic and natural speech and mention potential directions to further improve F0 modeling for the pipeline TTS systems.

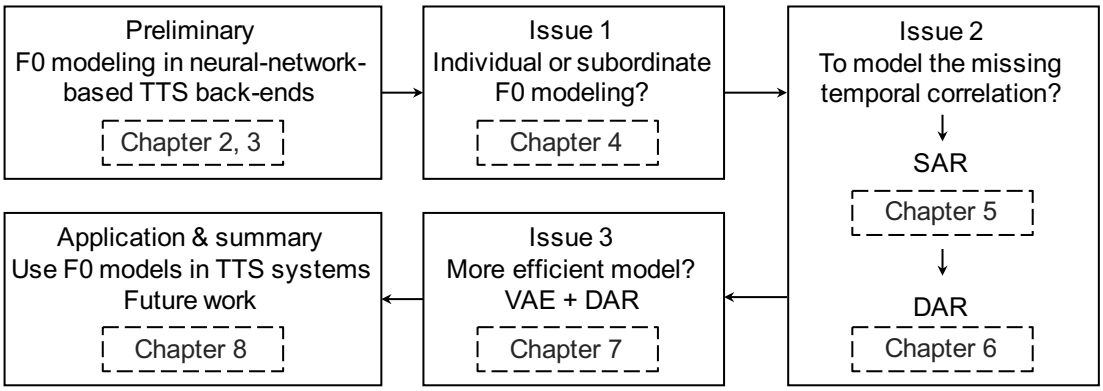


Figure 1.2: Thesis outline

2

F0 Modeling for TTS

F0 modeling for TTS involves not only signal processing and statistical modeling technologies but also natural and social sciences. Compared with other acoustic features, the F0 is closely related to the linguistic side of speech and language, which makes it notoriously difficult to model.

This chapter tries to introduce the F0 and F0 modeling from various perspectives. Section 2.1 introduces the F0 in the linguistic hierarchy, including its acoustic property and linguistic roles in speech and language. It also explains confusing terminologies related to the F0. Section 2.2 then introduces pipeline TTS systems and the statistical parametric speech synthesis (SPSS) framework for the TTS back-end. With the TTS systems in mind, Section 2.3 introduces the existing F0 modeling methods proposed for general TTS systems or specific TTS systems using the SPSS back-end. This section also defines the metrics to evaluate F0 modeling performance.

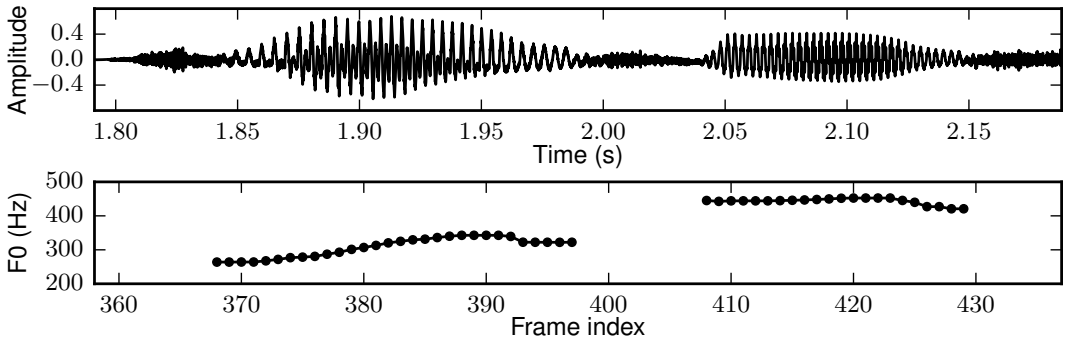


Figure 2.1: Speech waveform and F0 contour.

2.1 Introduction to F0

F0 is not a tangible object. Even under the same name, the F0 can be measured, interpreted, and reproduced from different viewpoints. This section briefly introduces the F0 using some of the viewpoints.

2.1.1 F0 in signal processing

If we take the perspective of signal processing, we often use the F0 to denote the greatest common divisor (GCD) of all the frequency components contained in a periodic signal such as a sine signal [13]. However, this definition may not apply to the speech waveform, which contains quasi- and non-periodic components. For a speech waveform, it is more practical to define the F0 value as the inverse of the smallest period in the speech segment being analyzed [14]. The rationale for this 'engineering' point of view is rooted in speech production theory. Nevertheless, given a small segment (e.g., 10ms in length) that is windowed from the original waveform, an F0 value can be acquired by calculating the inverse of the period of the repeating patterns in the segment. This speech segment is referred to as one frame. If the window is shifted by 5ms, another F0 value can be calculated for that frame. Similarly, a sequence of F0 values can be obtained for a whole speech waveform and is referred to as the F0 contour. For specific algorithms to calculate the F0 value, the readers may refer to other literature [14, 15, 16, 17, 18].

In this thesis, we use $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_T\}$ to denote an F0 contour, where T is the number of frames. We use a bold letter $\boldsymbol{o}$ rather than a scalar to denote the

F0 of one frame in order to represent the F0 of speech in a general manner. As the section on speech production will explain, a natural speech waveform may contain aperiodic frames for unvoiced sounds in which the periodicity and F0 do not exist. For those frames, $\boldsymbol{o}_t$ does not denote a scalar value but a special 'unvoiced' symbol.

2.1.2 F0 in speech production

From the perspective of speech production, the F0 is determined by the vibration frequency of the vocal folds when a human speaker utters a voiced sound [19]. The vibrated vocal folds convert the air flows from the lungs into a quasi-periodic sequence of air pulses, after which this pulse sequence is modulated by the vocal tract into a speech waveform. This physical process can be described by a source-filter model [20]. Since the sequence of pulses (source) is a quasi-periodic signal, the waveform signal coming out from the vocal tract (filter) is also quasi-periodic. This is the reason that we can calculate the F0 value from a voiced speech segment.

Unvoiced speech segments are uttered without the vibration of the vocal folds. Depending on the articulation gesture, the air flow may be stopped and released somewhere in the vocal tract. Or it may be constricted somewhere from the glottis to the lip. In general, an unvoiced sound does not display periodicity, for example, as the initial part of Figure 2.1 shows. For these frames, the F0 is not defined.

Notice that a speech sound can be between the 'voiced' and 'unvoiced' states. However, in most of the F0 extractors, a sound may simply be treated as either voiced or unvoiced [14]. This thesis also adopts this binary division to represent F0 contours. Accordingly, an F0 contour is plotted as a set of disconnected curves, for example, as Figure 2.1 shows.

2.1.3 F0 in speech perception

When a speech waveform reaches the perception system of a human listener, the F0 is perceived as the relative highness or lowness of speech, i.e., the pitch [21]. Although the pitch is affected by many factors such as the sound intensity, it is mainly decided by the F0 value. Accordingly, the F0 contour is an important acoustic cue for pitch perception.

The perceived variation of pitch further contributes to the perception and understanding of the speech. In tonal languages such as Mandarin, the perceived pitch change over a syllable is recognized as one of the categorical lexical tones. Such lexical tones help the listener to differentiate the syllables with otherwise the same sequence of phones. For example, the syllable [ma] with a flat tone type means 'mother', while the same syllable with a decreasing tone type means 'scold'. In pitch-accent languages such as Japanese (Tokyo Yamanote dialect), the rise-fall pitch pattern signals the location of an accent nucleus, which also discriminates lexical words with different meanings [22].

Another major use of the pitch is to signal post-lexical nuances. In English, a finite set of pitch contour patterns are the acoustic cues for (post-lexical) pitch-accents[1]. These pitch accents help the listener to understand the meaning beyond the surface words. For example [24], let's consider dialogues below, where the displaying of the word 'Marianna' shows the pitch accents:

Speaker A: Who made the marmalade?
Speaker B: $^{\text{Mari}}\text{an}_{\text{na}}$ made the marmalade.

Speaker A: Bob made the marmalade.
Speaker B: (No,) $_{\text{Mari}}{}^{\text{an}}\text{na}$ made the marmalade.

In the first case, speaker B just uses a normal high pitch accent to answer 'who did that'. In the second one, however, speaker B uses a low-high pitch accent to indicate that speaker A's statement is incorrect and emphasizes Marianna rather than Bob made the marmalade. Notice that the words uttered by speaker B are the same in the two dialogues, but they show a semantic difference through different pitch accents.

Beyond the lexical and post-lexical roles, the pitch may also indicate the para-linguistic and social-linguistic information such as the social status of the speaker [33, 34]. With other perceptual features, the pitch may also signal the

[1] 'Pitch-accent' in English and that in Japanese are quite different. Although both 'accents' are signaled by the pitch contour, they have different functions, distributions in utterance, and categorical types [23]

Table 2.1: Functions of speech prosody and acoustic cues

Level	Function	(Part of) Acoustic cues
Word	Segment words [25]	spectral tilt, duration
	Distinguish lexical words [22]	pitch pattern (tone)
Phrase	Signals the phrase boundary [25]	pause, pitch pattern
	Shows prominence in phrase [26]	pitch pattern
	Encodes the speech act [27]	pitch pattern
Discourse	Segments the discourse [28][28]	pitch range, loudness
	Indicates the focus in discourse [29]	pitch pattern, duration
	Manages conversation [30]	pitch level, *mm-hm, yeah,*
Global	Implies the speaker's attitude [31][32]	pitch range, duration
	Implies the social status [33, 34]	rhythm, accent

speaker's mood and emotion [35]. On these aspects, the pitch is used not as part of a linguistic structure but as a type of animal communication [22].

Nevertheless, the pitch is an essential component of speech. Together with timing, loudness, articulation effort, and other acoustic cues, the pitch determines the speech prosody that covers almost every corner of speech and language understanding. Table 2.1 lists some of the functions played by the speech prosody, together with some of the acoustic cues including the pitch.

2.1.4 Terminology

In the introduction to speech perception, we have distinguished 'F0' from 'pitch'. Although in many papers 'pitch' is used interchangeably with 'F0', we only use 'pitch' to denote the perceived highness or lowness of sound and 'F0' to denote the inverse of the basic period in waveforms. Accordingly, F0 modeling for TTS refers to the generation of F0 contours given linguistic features.

In the previous section, we also used terms such as 'prosody', 'intonation', 'tone', and 'accent'. Here, we just casually define them for explanation in this thesis. Readers may refer to other books for thorough definition and explanation [22].

First, we use 'tone' to refer to the lexical tones in tonal-languages, which is mainly the categorical patterns of pitch movement. For English, the 'tone' is also used to refer to the pitch movement in an intonational structure, which may also

be called intonational tone (e.g., phrase boundary tone in the Tone and Break Index (ToBI) [36] labeling protocol). We use 'pitch-accent' to refer to the pitch movement that discriminate lexical words in Japanese [23]. It is also used to name the English pitch movement that consists of a single or two intonational tones. For example, the L-H pitch accent that consists of the L intonational tone and H intonational tone [36]. The word 'intonation' is used to denote the sequence of pitch-accent in English. Finally, 'prosody' is the most general word to denote the suprasegmental properties of the speech. Not only does it include pitch but also duration, spectral tilt, or other perceived features.

2.2 TTS systems

2.2.1 TTS front-end

In Section 1.1, we briefly mentioned the TTS systems with a pipeline structure [2, 37]. Such a TTS system consists of a front-end and a back-end, where the front-end parses the text and decides what to speak and how to speak while the back-end converts the speaking plan into a speech waveform.

Suppose we are asked to read a text in a foreign language (so that we consciously know what we are doing), for example English. We may first recognize the word in the text, guess its pronunciation, and decide the location of primary stress. This process is quite easy if we are familiar with the words. But real cases may be much more difficult. For example, we have to decide the way to read a date, number (e.g., 1990), abbreviation (e.g., Dr.), non-standard words, and homographs (e.g., 'lives'). For some words, we need to decide the stress based on its syntactic role (e.g., 'project' as a noun and 'project' as a verb). If there is an out-of-vocabulary word, we may try our best to guess its pronunciation based on the stem and affixes.

A TTS front-end needs to do a similar process [38]. In technical terms, a TTS front-end needs to recognize individual words through text tokenization, process non-standard words through text normalization [39], and convert the letter sequence into the phoneme sequence through letter-to-sound conversion [40]. When discriminating homographs, the front-end also conducts part-to-speech (POS) tagging [41] and decides the pronunciation using syntactic information.

Besides 'what to speak', we human beings also plan 'how to speak', i.e., decide the speech prosody, including the intonation, phrasing, duration and so on. This part is the crux of the front-end. While speech prosody is essential, there is no consensus about what should be planned, how to represent them, and from what source information to make the plan. In practice, the POS tags and the syntactic structure of the text may be used to decide the location and the length of pauses (i.e., phrasing) [42], the location and type of pitch accents (i.e., intonation in symbols) [43], and the duration of each word [44].

The above description is by no means comprehensive. On one hand, a front-end is language dependent; on the other hand, there is no standard about what should be predicted from the front-end for speech prosody as aforementioned. Nevertheless, we can at least know the basic steps in the TTS front-end. The output of the front-end is referred to as the linguistic feature. Appendix A lists the linguistic features used in this thesis for English and Japanese TTS.

2.2.2 TTS back-end using SPSS

Given the linguistic features, the TTS back-end converts them into a speech waveform. A few back-end frameworks have been proposed in the past, including unit-selection [45] and formant synthesis [46]. Here we only introduce the statistical parametric speech synthesis (SPSS) framework [47, 48] since it is used in this thesis and many other TTS systems. A diagram of SPSS is plotted in Figure 2.2.

Parametric side of SPSS

SPSS is parametric in the sense that it produces the speech waveform through parametric speech production models. One of the typical models is the source-filer model, which contains a source component to produce the excitation signal and a filter component to modulate the excitation into a speech waveform. In practice, this model is implemented by a digital signal processing system called a vocoder.

The parameters of the source and filter components, which are referred to as acoustic features, mainly include the spectral features and the F0. While the F0 determines the periodicity and voicing status of the source excitation signal, the spectral features depict the frequency response of the filter compoment. Commonly

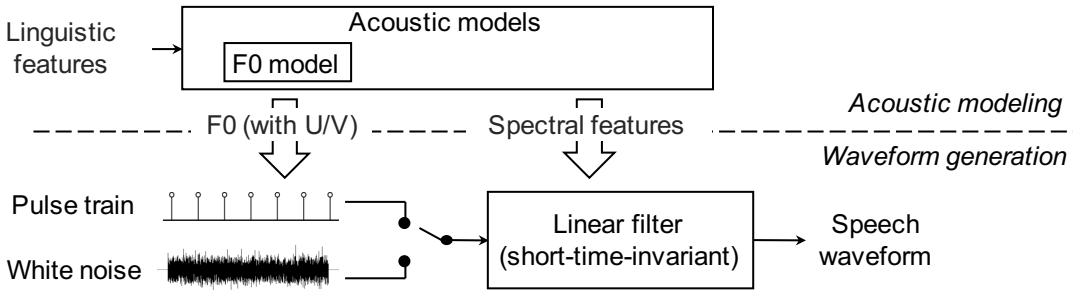


Figure 2.2: SPSS using a source-filter-based waveform generator

used spectral features include the Mel-generalized cepstral coefficients (MGC) [49] and line-spectrum-pairs (LSP) [50]. Some vocoders also include the band aperiodicity features (BAP), binary noise mask [51], and other spectral features that depict the degree of aperiodicity in each frequency band.

Suppose the acoustic features, including the F0 and spectral features, have been given as a sequence of vectors $\boldsymbol{a}_{1:T} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_t, \cdots, \boldsymbol{a}_T\}$ in T frames. In practice, the vocoder may generate a small segment of the waveform using the spectral features in each $\boldsymbol{a}_t$. It then may overlap and add all the T segments based on the period specified by the F0 information. Alternatively, a vocoder may overlap and add the excitation pulses based on the F0 and generate the output waveform by filtering the excitation signal. Readers may find more details in commonly used vocoders such as STRAIGHT [15] and WORLD [18].

Note that there are also vocoders incorporating more complicated source components, for example the glottal excitation [52, 53, 54] and mixed excitation [55, 56]. In addition to the source-filter-based vocoder, there are also vocoders based on the sinusoidal model [57, 58] and neural networks [59].

Statistical side of SPSS

The statistical side of SPSS is to predict the acoustic features based on the linguistic features. Statistical approaches are necessary because there is no simple rule to implement this conversion. For example, in order to generate the waveform for phoneme /a/, we may need to analyze some waveform samples of /a/, calculate the acoustic features from these samples, and store them into a lookup table. However, due to the co-articulation effect, we need to analyze a huge number of

samples and prepare the entries in the lookup table for /a/ in every phonemic context (i.e., allophones). The cost for such a manual process is prohibitive, before even making the rules for prosody. Even if a set of rules can be built, we would have to repeat this process if we want to build the system using another voice.

The statistical framework provides a flexible and effective solution: first, it does not require too much human effort to build a TTS-back-end; second, it can leverage large speech corpora and learn the mapping automatically; third, it allows voice interpolation, adaption, and other manipulation applications [47, 48].

The process to convert the linguistic features to the acoustic features is referred to as acoustic modeling, and the statistical models used for this process are called acoustic models. Similar to other statistical modeling approaches, acoustic modeling for SPSS requires three steps:

1. Collect a corpus $\mathcal{D} = \{\{\boldsymbol{a}^{(1)}_{1:T_1}, \boldsymbol{x}^{(1)}_{1:T_1}\}, \{\boldsymbol{a}^{(2)}_{1:T_2}, \boldsymbol{x}^{(2)}_{1:T_2}\}, \cdots, \{\boldsymbol{a}^{(N)}_{1:T_N}, \boldsymbol{x}^{(N)}_{1:T_N}\}\}$, where $\{\boldsymbol{a}^{(n)}_{1:T_n}, \boldsymbol{x}^{(n)}_{1:T_n}\}$ denotes the acoustic and linguistic feature sequences of the n-th utterance;

2. Define a probabilistic model $p(\boldsymbol{a}_{1:T}|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta})$ and learn the parameter set $\boldsymbol{\Theta}$ using the maximum likelihood training criterion

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \sum_{n=1}^{N} \log p(\boldsymbol{a}^{(n)}_{1:T_n}|\boldsymbol{x}^{(n)}_{1:T_n}; \boldsymbol{\Theta}); \tag{2.1}$$

3. Given $\boldsymbol{x}_{1:T}$ of a new input text, generate $\widehat{\boldsymbol{a}}_{1:T}$ by using the maximum output probability criterion

$$\widehat{\boldsymbol{a}}_{1:T} = \arg\max_{\boldsymbol{a}_{1:T}} p(\boldsymbol{a}_{1:T}|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta}^*). \tag{2.2}$$

The first and second steps define the process to build the acoustic models while the third step uses the trained acoustic model for acoustic feature generation.

One crucial point of SPSS is the definition of the acoustic models. HMM plus the decision trees is one of the typical acoustic models for SPSS[2] [47, 48]. In recent

[2] For HMM-based SPSS, the linguistic feature sequence $\boldsymbol{x}_{1:N}$ is usually a set of context-dependent phones. Hidden state $\boldsymbol{q}_{1:T}$ is introduced to model the alignment between $\boldsymbol{x}_{1:N}$ and $\boldsymbol{a}_{1:T}$, and the acoustic model is written as $p(\boldsymbol{a}_{1:T}|\boldsymbol{x}_{1:N}; \boldsymbol{\Theta}) = \sum_{\boldsymbol{q}_{1:T}} p(\boldsymbol{a}_{1:T}, \boldsymbol{q}_{1:T}|\boldsymbol{x}_{1:N}; \boldsymbol{\Theta})$.

years, neural-network-based acoustic models are becoming popular [60, 61, 62] because they alleviate the inappropriate model assumptions in HMMs and decision trees [63]. We will briefly explain these acoustic models later.

Note that the above framework is not the only solution to SPSS. For example, in the so-called 'end-to-end' approaches [64], the acoustic features may be mel-spectrogram without explicit F0 features. It also integrates the front-end into the SPSS framework, without using the aligned linguistic and acoustic feature pairs.

2.3 F0 modeling for TTS

Having introduced the TTS and SPSS, this section looks into F0 modeling. By F0 modeling, we mean the process to generate the F0 contour $\boldsymbol{o}_{1:T}$ given the linguistic feature sequence $\boldsymbol{x}_{1:T}$. For this task, we may directly use the HMM- or neural-network-based SPSS and generate the F0 together with spectral features. However, because of the F0's special role in speech and language, F0 modeling has been investigated using various approaches beyond the SPSS-based framework. We use $\boldsymbol{s}_{1:T}$ to denote the spectral feature sequence.

2.3.1 Deterministic F0 representation

One class of the research is concerned with the compact representation of F0. While an F0 contour is a sequence of values measured in every frame, the tone and intonation are defined over syllables or longer linguistic units. What's more, linguistic units may span different numbers of frames. If representations can be found to describe F0 contours for linguistic units, they would favor the association between the F0 and the linguistic structure of the utterance.

One typical representation is based on the Autosegmental-Metrical (AM) model, a model that describes the American English F0 contour as a sequence of high (H) and low (L) movement [26]. Specifically, the AM model assumes that an F0 contour of one utterance consists of three parts: the pitch accents signaled by the excursion of F0 contours on the accented syllables, the F0 movement at the end of the phrase, and the F0 curve linking the last pitch accent to the phrase end. The AM model further assumes that F0 contours in the three parts can be

abstracted into a finite set of categories. This AM-based representation is adopted in the ToBI labeling protocol [36].

Another typical of representation is based on the Tilt model [7]. Similar to the AM model, the Tilt model also decomposes an F0 contour into a sequence of events, including pitch accents, boundary tones, connections, and silence. However, the Tilt model uses continuous parameters rather than categorical symbols to describe each event. For example, a pitch accent event is described by a starting F0 value, duration, absolute F0 amplitude, position of the rise-all boundary, and a special tilt parameter [7]. In addition to the Tilt model, other representations have been proposed on the basis of the Fujisaki model [65, 66] and Parallel Encoding and Target Approximation (PENTA) model [8]. One common characteristic of these models is that the F0 curve is assumed to be produced from a configurable function, for example, the exponential functions in the PENTA and the Fujisaki model. It has been argued that these functions follow the physiology of F0 production.

Other representations may be purely engineering approaches. For example, three target points can be used to represent a coarse F0 shape for each syllable [11, 67]. Similarly, continuous-wavelet-transformation and discrete-cosine-transformation can also be used to represent F0 contours [68, 69]. It has been shown that a certain degree of linguistic regularity can be observed from the F0 representation derived using these engineering approaches [6]. Another special work is the F0 stylization for perceptual experiments, a representation that converts the F0 contour into a sequence of connected straight lines [70].

In general, the purpose of F0 representation and abstraction is to derive a compact representation $\boldsymbol{l}_{1:N} = \{\boldsymbol{l}_1, \cdots, \boldsymbol{l}_N\}$ from the F0 contour $\boldsymbol{o}_{1:T}$. Meanwhile, the representation $\boldsymbol{l}_{1:N}$ should be easy to predict from linguistic features and sufficiently detailed to recover the F0 contour. Depending on the methods, the sequence length N may be equal to the number of accented syllables, phones, or frames, and each $\boldsymbol{l}_n$ may be a categorical value or a set of continuous values.

2.3.2 F0 modeling based on compact F0 representation

The above F0 representations can be used for F0 modeling in TTS. After F0 representations are extracted from the F0 training data, statistical models can be

trained to map the linguistic features into the F0 representations. After that, predicted F0 representations for a new text can be used to produce the F0 contour.

For example, F0 modeling for TTS can be constructed on the basis of the Tilt model [71]. After the Tilt parameters are extracted from training data, regression trees can be built to predict the Tilt parameters for each Tile event. The inputs to the regression trees are the linguistic features of the text, including the lexical stress of syllable, the position of the syllable within a phrase, and composition of the syllable structure. During generation, the predicted Tilt parameters are used to drive the Tilt synthesis model and generate the F0 contour syllable-by-syllable for a new test. Similar F0 generation methods can be used on the basis of the Fujisaki model [72] and the PENTA model [73].

However, not every type of F0 representation can be directly used for TTS. The AM model represents the F0 contour in a quite sparse and abstract manner, i.e., only a few symbols on accented syllables. Thus, a different model is required to generate the F0 contour after the pitch accent symbols are predicted from the linguistic features [74]. For example, a simple linear regression model can be used to convert the pitch accents and other linguistic features into the F0 values of the starting, middle, and ending target points for each syllable. This step-wise F0 contour can then be low-pass filtered to produce a smooth F0 contour [67].

2.3.3 Frame-level statistical F0 modeling

The deterministic representations based on the Tilt and other models assume specific functions to represent the shape of F0 contours, which may not cover all possible F0 curve shapes. Or, in the case of target F0 points representation, the accuracy of the reconstructed F0 is limited.

A more flexible approach is to use statistical models to directly map the linguistic features into the F0 contour. An F0 model for this purpose may automatically learn the F0 shapes from the training data. To differentiate it from the previous approaches using F0 representations, we refer to this approach as the frame-level statistical F0 modeling. The basic procedure of this approach is identical to the SPSS framework described in Section 2.2.2. In fact, frame-level statistical F0 modeling is usually integrated into the HMM and neural-network-based SPSS

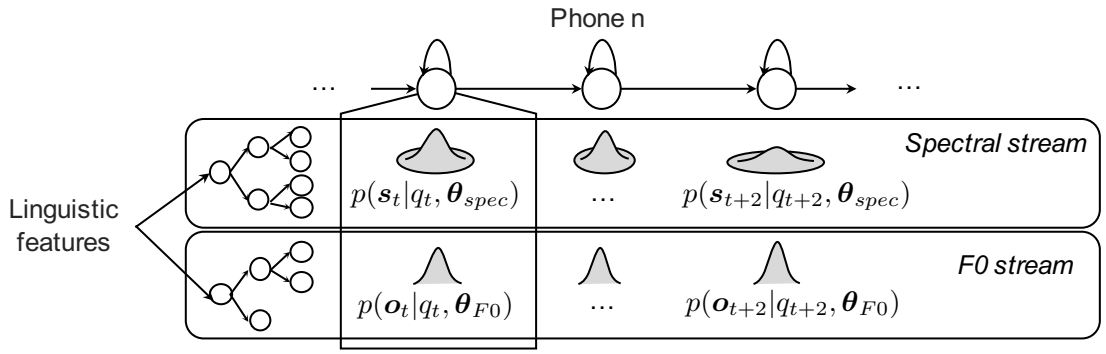


Figure 2.3: HMM-based SPSS with multi-streams, where $\boldsymbol{o}_t$ and $\boldsymbol{s}_t$ denotes the F0 and spectral feature at the t-th frame.

frameworks, where the F0 and other spectral features are modeled simultaneously. However, since the F0 is quite different from other spectral features, statistical F0 modeling usually requires special modeling methods.

HMM-based approach

Let's take the HMM-based approach as an example. The major problem is the F0 in unvoiced frames. Since these F0 values are undefined, they cannot be modeled together with the F0 values in voiced frames. In the classical HMM-based approach, an F0 value $\boldsymbol{o}_t$ is defined as either a real-valued number or a special discrete symbol NULL, i.e., $\boldsymbol{o}_t \in \{\text{NULL}\} \cup \mathbb{R}$. A model called multi-space distribution HMM (MSD-HMM) then can be used to describe the distribution of the F0 values in the voiced frames and the probability of being unvoiced [75]. Given a hidden state at one frame, the MSD-HMM first decides whether the frame is unvoiced. If not, it then predicts the F0 value for that frame.

Another method is to assign artificial F0 values to the unvoiced frames and model the interpolated F0 contour together with the binary voicing status of every frame. In such a case, every frame has a real F0 value, i.e., $o_t \in \mathbb{R}$, and a normal HMM can be used for statistical F0 modeling. This approach is known as the explicit continuous F0 modeling [76].

In both approaches above, the F0 data are modeled with other spectral features $\boldsymbol{s}_{1:T}$ using a multi-stream HMM-structure [77]. In the generation stage, after the hidden state sequence $\{\boldsymbol{q}_1, \cdots, \boldsymbol{q}_T\}$ is determined by a Gaussian-based duration

model, the linguistic features can be used to find the distribution of the spectral feature $p(\boldsymbol{s}_t|\boldsymbol{q}_t, \boldsymbol{\theta}_{\text{spec}})$ and F0 $p(\boldsymbol{o}_t|\boldsymbol{q}_t, \boldsymbol{\theta}_{\text{F0}})$ from the leaf nodes of the decision trees. Given the sequence of distributions, the maximum likelihood parameter generation algorithm [78] can be used to generate the F0 and spectral feature sequences. Notice that F0 and spectral features are described by independent distributions given the hidden state[3].

The independence of F0 and spectral features in each HMM state makes it feasible to use more specific approaches for F0 modeling. For example, syllable-level F0 features (e.g., mean F0 value) can be integrated into the HMM as another stream [80, 81, 82, 83].

Neural-network-based approach

Despite the good performance of the HMM-based approach, many recent statistical models are based on deep neural networks [61, 84, 85, 86, 62]. Compared with the HMM-based approach, the neural-network-based approach avoids unnecessary assumptions[4]. For example, it doesn't assume that the same distribution is used for all the frames belonging to the same hidden state. Furthermore, the decision-tree-based model clustering process is replaced by the non-linear transformation in neural networks, which avoids the data-fragmentation problem.

Another advantage of the neural-network-based approach is that the model can be trained in a straightforward manner. Given the F0 and spectral features as the target, the network can be simply trained under a minimum square error criterion on a data corpus[5]. In the generation stage, outputs can be directly taken from the last layer of the neural network. Usually, the F0 contour are interpolated before modeling. Figure 2.4 shows one recurrent neural network (RNN) that models both the F0 and the spectral features.

In both HMM- and neural-network-based frameworks, the F0 is usually modeled with other acoustic features jointly. In the HMM-based SPSS framework, the F0 is somewhat separated from spectral features because of the multi-stream HMM as

[3]There are also methods that model the dependency of the spectral features on the F0 [79].

[4]Not all the assumptions are avoided. For example, the independence between F0 and spectral features is still held by a normal neural network, which will be explained in Chapter 3.

[5]This is equivalent to the maximum likelihood criterion as Chapter 3 will show.

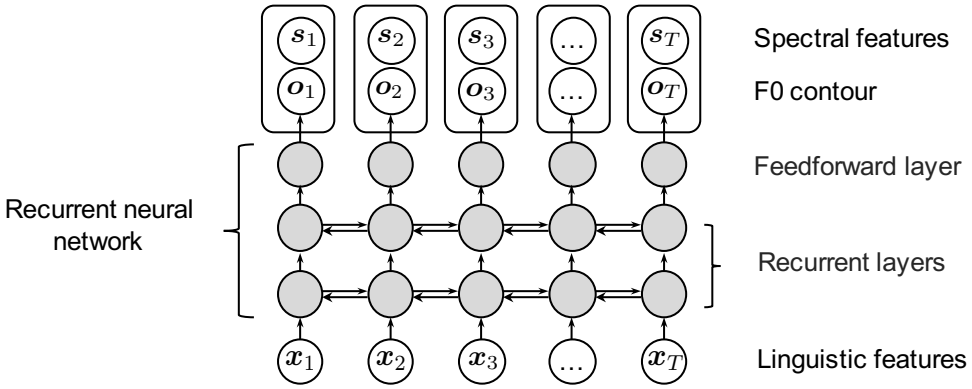


Figure 2.4: RNN-based acoustic model

Figure 2.3 shows. In neural networks, however, the F0 and other acoustic features share the hidden layers. As Chapter 4 will argue, this joint training method may be suboptimal for F0 modeling.

We leave the detailed explanation of neural networks to the next chapter. Here we only briefly mention some works using neural networks exclusively for F0 modeling. One work uses the RNN with long-short-term-memory (LSTM) units [87], which is also used as the baseline model in this thesis. Other F0 models based on neural networks were proposed around 1990 [10, 12]. One of the models includes a macro-phonemic network to predict the average F0 value for each phoneme in addition to the frame-level network. However, the network size and design of linguistic features are quite limited. There are other works using neural networks for F0 modeling [88, 11]. However, these works do not model the F0 contour at the frame level but just generate a fixed number of F0 values for each syllable.

2.3.4 Evaluation metrics

After introducing the F0 models, this section explains the common metrics used to evaluate the performance of F0 models. These metrics are also used in this thesis.

Objective metrics

As objective metrics, we can calculate the distance between natural F0 contours $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$ and generated versions $\hat{\boldsymbol{o}}_{1:T} = \{\hat{\boldsymbol{o}}_1, \cdots, \hat{\boldsymbol{o}}_T\}$ on the test set. We use the natural duration T of the test utterance to generate $\hat{\boldsymbol{o}}_{1:T}$.

Since F0 contours contain unvoiced frames, we first extract the frames t where both $\boldsymbol{o}_t$ and $\widehat{\boldsymbol{o}}_t$ are voiced. Let's use $\boldsymbol{o}_{1:T_v} = \{o_1, \cdots, o_{T_v}\}$ and $\widehat{\boldsymbol{o}}_{1:T_v} = \{\widehat{o}_1, \cdots, \widehat{o}_{T_v}\}$ to denote the extracted frames from $\boldsymbol{o}_{1:T}$ and $\widehat{\boldsymbol{o}}_{1:T}$, respectively. Here, T_v denotes the number of frames where both $\boldsymbol{o}_t$ and $\widehat{\boldsymbol{o}}_t$ are voiced. We then can calcualte:

- Root mean square error (**RMSE**):

$$\text{RMSE}(\boldsymbol{o}_{1:T_v}, \widehat{\boldsymbol{o}}_{1:T_v}) = \sqrt{\sum_{t=1}^{T_v}(o_t - \widehat{o}_t)^2 / T_v}; \tag{2.3}$$

- Correlation (**CORR**):

$$\text{CORR}(\boldsymbol{o}_{1:T_v}, \widehat{\boldsymbol{o}}_{1:T_v}) = \frac{\sum_{t=1}^{T_v}(o_t - \bar{o}_{1:T_v})(\widehat{o}_t - \bar{\widehat{o}}_{1:T_v})}{\sqrt{\sum_{t=1}^{T_v}(o_t - \bar{o}_{1:T_v})^2}\sqrt{\sum_{t=1}^{T_v}(\widehat{o}_t - \bar{\widehat{o}}_{1:T_v})^2}}, \tag{2.4}$$

  where $\bar{o}_{1:T_v}$ and $\bar{\widehat{o}}_{1:T_v}$ denote the mean of $o_{1:T_v}$ and $\widehat{o}_{1:T_v}$, respectively;

- Voicing decision error (**U/V**):

$$\text{U/V}(T, T_v) = 100\% - T_v/T. \tag{2.5}$$

RMSE (the lower the better) and **CORR** (the higher the better) measures the accuracy of generated F0 contours while **U/V** (the lower the better) measures the voicing accuracy. Their average values on the test set are used as objective metrics.

Subjective metrics

We can also use subjective evaluation metrics. After using the generated F0 contours to synthesize the speech waveform, we evaluate the synthetic and natural samples through the Mean-Opinion-Score (MOS) test [89]. In this test, native speakers will rate the sample from 0 (unacceptable) to 5 (perfect) in terms of the perceived quality. Another test is the preference test, in which the participant listens to a pair of synthetic samples and chooses the better one. Details of the subjective tests will be given in the following chapters.

3 Neural Networks

This chapter introduces another keyword of this thesis: neural network. In recent years, it has become easy to train and use neural networks using various tools. We may think that a neural network is a production line that can converts the input materials into a desired output. However, this view may not explain how a neural network works when the mapping between the input and output is ambiguous. Neither may it explain the neural networks that receive no input. For a better explanation, this chapter tries to interpret the neural works as probabilistic models.

The chapter focuses on the neural networks with input and output features because they are widely used in recent TTS systems. Section 3.1 reviews the common practice to use feedforward and recurrent neural networks as trainable yet deterministic functions. Section 3.2 then interprets these networks as probabilistic models, which justifies the common practice. Based on this interpretation, Section 3.2 discusses the shortcomings of conventional neural networks when we use them for F0 modeling.

It is impractical to cover most of the interesting aspects of neural networks

in a short chapter. This chapter does not explain fundamental techniques such as back-propagation and momentum. Neither does it explain the details of implementation. Most contents of this chapter are written based on the works [90, 91]. More general and up-to-date introduction on neural networks can be found in other literature [92, 93].

3.1 Neural networks as deterministic functions

Since the 1940s, various types of neural networks have been proposed for different learning tasks [92]. The type concerned with in this thesis is designed for supervised learning tasks. It learns the mapping from input to output data in a training set and stores the learned 'knowledge' as the trainable network weights. It then converts a new input datum into an appropriate output datum. Accordingly, such a network can be treated as a function $f_{\boldsymbol{\Theta}} : \mathcal{X} \to \mathcal{Y}$, which, given the trainable weights $\boldsymbol{\Theta}$, converts the input data of domain $\mathcal{X}$ into the output data of domain $\mathcal{Y}$.

3.1.1 A simple network for the XOR problem

Let us use a network to solve the XOR problem by learning the ideal input and output data pairs listed in Figure 3.1. Here we use a vector $\boldsymbol{x} = [x_1, x_2]$ and a scalar o to denote an input datum and an output datum, respectively. To learn the XOR problem, we use a network that has a single hidden unit. As plotted in Figure 3.1, this network defines a function $f_{\boldsymbol{\Theta}} : \mathbb{R}^2 \to \mathbb{R}$ that can be written as:

$$f_{\boldsymbol{\Theta}}([x_1, x_2]) = \sigma(w_1 x_1 + w_2 x_2 + w_3 \sigma(w_4 x_1 + w_5 x_2 + b_2) + b_1), \tag{3.1}$$

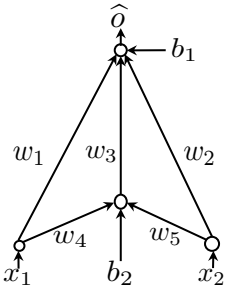
where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic function, and $\boldsymbol{\Theta} = \{w_1, w_2, w_3, w_4, w_5, b_1, b_2\}$ is the set of network weights.

The network weights $\boldsymbol{\Theta}$ can be learned from the four pairs of ideal input and output data using a mean-square-error (MSE) criterion

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} E(\boldsymbol{\Theta}) = \arg\min_{\boldsymbol{\Theta}} \frac{1}{2 \times 4} \sum_{n=1}^{4} \left(f_{\boldsymbol{\Theta}}([x_1^{(n)}, x_2^{(n)}]) - o^{(n)} \right)^2, \tag{3.2}$$

Ideal input/output pairs

x_1	x_2	o
0	0	0
1	0	1
0	1	1
1	1	0



Network's weights and output after learning

$w_1 = w_2 = -4.2$
$w_4 = w_4 = -6.4$
$w_3 = -9.4$
$b_1 = 6.3$
$b_2 = 2.2$

x_1	x_2	$\widehat{o}$
0	0	0.084
1	0	0.871
0	1	0.871
1	1	0.109

Figure 3.1: A toy neural network to solve the XOR problem [94].

where the superscript $^{(n)}$ denotes the n-th training data pair. Unfortunately, the best $\boldsymbol{\Theta}^*$ cannot be found analytically because of the non-linear activation functions inside the network. A practical method is to iteratively adjust $\boldsymbol{\Theta}$ so that the MSE gradually decreases. This is known as the gradient-descent method. Starting with a randomly initialized $\boldsymbol{\Theta}$, this method calculates the MSE and adjusts a network weight w on the basis of its gradient with respect to the MSE. This updating procedure can be written as $w \leftarrow w - \eta \frac{\partial E(\boldsymbol{\Theta})}{\partial w}$, where η is a learning rate parameter. By repeating this updating process for multiple iterations, an estimated w can be acquired. For network weights near the input side, e.g., w_4, w_5, and b_2 in Figure 3.1, their gradients are calculated based on the gradients of w_3. The process to calculate the gradients is referred to as the back-propagation algorithm [94, 95].

Figure 3.1 also plots a set of learned weights for the toy network and the output value of the network using the learned weights. The network seems to learn the XOR operation if we think 0.87 and 0.10 are approximately equal to 1 and 0, respectively. Nevertheless, this network shows the power of a neural network to learn the XOR problem, which is impossible for a network without the hidden unit. The back-propagation-based gradient descent algorithm 'answered Minsky and Papert's challenge' of finding an effective learning method for networks with hidden units [94] and has been the keystone of deep learning.

3.1.2 Feedforward neural network

The toy network for the XOR problem can be extended to a more general network and used for more complex tasks. Let's consider a sequential supervised learning task where a network needs to convert a sequence of input data $\boldsymbol{x}_{1:T} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$

into a target sequence $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$ with T steps. For explanation, we define $\boldsymbol{o}_t \in \mathbb{R}^{D_o}$ and $\boldsymbol{x}_t \in \mathbb{R}^{D_x}$. Since the example on the XOR problem suggests that a learned network may not generate the target data exactly equal to the ideal value, we use $\widehat{\boldsymbol{o}}_t$ to denote the actual output of the network at the t-th step.

The first type of network for the sequential learning task is the feedforward neural network (FNN), a network assuming that the mapping from $\boldsymbol{x}_{1:T}$ to $\boldsymbol{o}_{1:T}$ can be conducted for each $t \in \{1, \cdots, T\}$ independently. Let's consider an FNN with a hidden layer and a linear output layer, which is shown in Figure 3.2. Similar to the network for the XOR problem, this FNN can be treated as a deterministic function $f_{\boldsymbol{\Theta}} : \mathbb{R}^{D_x} \to \mathbb{R}^{D_o}$, which can be written as

$$\widehat{\boldsymbol{o}}_t = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t) = \boldsymbol{W}_o \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{b}_i) + \boldsymbol{b}_o. \tag{3.3}$$

Here $\sigma(\cdot)$ is the logistic activation function, and $\boldsymbol{\Theta} = \{\boldsymbol{W}_o, \boldsymbol{W}_i, \boldsymbol{b}_o, \boldsymbol{b}_i\}$ is the set of network weights. This network generates a sequence of output $\widehat{\boldsymbol{o}}_{1:T} = \{\widehat{\boldsymbol{o}}_1, \cdots, \widehat{\boldsymbol{o}}_T\} = \{f_{\boldsymbol{\Theta}}(\boldsymbol{x}_1), \cdots, f_{\boldsymbol{\Theta}}(\boldsymbol{x}_T)\}$ by repeating the conversion process for each time step $t \in \{1, \cdots, T\}$.

The network weights $\boldsymbol{\Theta}$ can be learned similarly to the toy network on the previous page. Without loss of generality, we consider a training data set with one pair of input and output sequences $\mathcal{D} = \{\{\boldsymbol{o}_{1:T}, \boldsymbol{x}_{1:T}\}\}$. On this tiny corpus, $\boldsymbol{\Theta}$ can be learned on the basis of the MSE-based criterion

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} E_{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \sum_{t=1}^{T} ||\boldsymbol{o}_t - f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t)||^2. \tag{3.4}$$

In practice, a good but not necessarily the best solution can be found by using the gradient-descent method in a similar manner to the toy XOR network.

Note that the style of Figure 3.2 is different from that in Figure 3.1 in order to illustrate this sequential conversion process. Each circle in Figure 3.2 denotes a scalar or vector calculated by the network. The link connecting circles denotes the connection parameterized by the transformation weight matrix and bias vector. Furthermore, each link is plotted repeatedly for every time step.

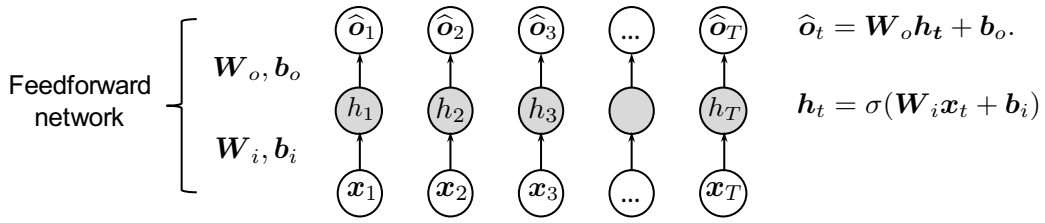


Figure 3.2: Using a feedforward network on sequential data.

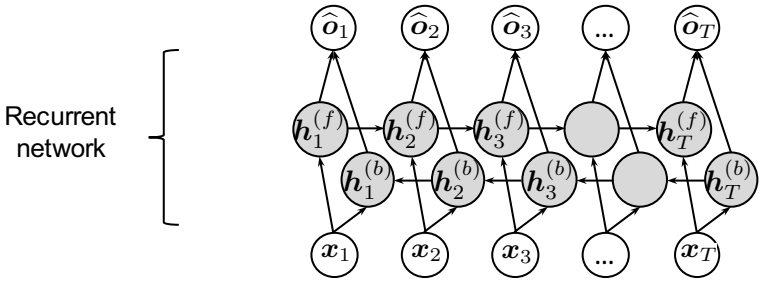


Figure 3.3: Using a recurrent network on sequential data.

3.1.3 Recurrent neural network

The FNN has been used for sequential learning tasks such as the letter-to-sound conversion [96]. However, since it treats the input and output sequences as a set of time-independent data, it ignores the temporal information in the sequential data. A better type of network is the recurrent neural network (RNN). Let us consider an RNN with a bi-directional recurrent hidden layer [97] and a linear output layer, which is shown in Figure 3.3. Given the input $\boldsymbol{x}_{1:T}$, this RNN calculates the output as:

$$\widehat{\boldsymbol{o}}_t = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t) = \boldsymbol{W}_o^{(f)} \boldsymbol{h}_t^{(f)} + \boldsymbol{W}_o^{(b)} \boldsymbol{h}_t^{(b)} + \boldsymbol{b}_o, \tag{3.5}$$

where

$$\boldsymbol{h}_t^{(f)} = \sigma(\boldsymbol{W}_h^{(f)} \boldsymbol{h}_{t-1}^{(f)} + \boldsymbol{W}_i^{(f)} \boldsymbol{x}_t + \boldsymbol{b}_h^{(f)}), \tag{3.6}$$

$$\boldsymbol{h}_t^{(b)} = \sigma(\boldsymbol{W}_h^{(b)} \boldsymbol{h}_{t+1}^{(b)} + \boldsymbol{W}_i^{(b)} \boldsymbol{x}_t + \boldsymbol{b}_h^{(b)}), \tag{3.7}$$

$\sigma(\cdot)$ is an activation function, and $\boldsymbol{\Theta} = \{\boldsymbol{W}_h^{(*)}, \boldsymbol{W}_i^{(*)}, \boldsymbol{W}_o^{(*)}, \boldsymbol{b}_h^{(*)}, \boldsymbol{b}_o\}$ denotes network weights. Notice that we use $f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t)$ to denote the network output at time t because it depends on not only $\boldsymbol{x}_t$ but also the whole sequence $\boldsymbol{x}_{1:T}$. Such an RNN then can be considered as a function $f_{\boldsymbol{\Theta}} : \mathbb{R}^{D_x \times T} \rightarrow \mathbb{R}^{D_o \times T}$.

The network weights of an RNN can be trained using the gradient descent method. However, the gradient for each weight must be computed and accumulated over the whole sequence. This can be conducted using an algorithm called back-propagation through time (BPTT) [98]. Its basic idea is to follow the graph in Figure 3.3, propagate the gradient through the recurrent link, and accumulate the gradient for each network weight.

3.2 Neural networks as probabilistic models

Deterministic interpretation is insufficient and inappropriate

The previous section interprets neural networks as deterministic functions, which allows us to use the MSE criterion to train the network and treat the output of a network as the predicted target data.

Let us consider the network for the XOR problem again. In that task, the number of possible input and output pairs is finite; the mapping from the input to the output can be expressed as a surjective function[1]. Even though the learned network does not generate the exact value of 1 or 0 because of the non-linear activation functions and the gradient learning method, such a network can be interpreted as a deterministic function. In fact, the toy XOR network can generate the exact and correct 1 or 0 if it uses a binary step activation function[2].

However, the deterministic interpretation cannot fully explain the 'meaning' of the network's output in more complex scenarios. For more complex tasks, the mapping from the input to the output doesn't satisfy the definition of a deterministic function when two different output data correspond to the same input datum, i.e., $y_1 = f(x_1), y_2 = f(x_1)$ while $y_1 \neq y_2$. For example, the same phoneme /t/ may be uttered by a human being as different allophones. In such a case, a non-linear network may not simply make a compromise and generate $f(x_1) = (y_1 + y_2)/2$. Furthermore, the average $(y_1 + y_2)/2$ may be meaningless for specific tasks, e.g., the average of a flap [t] and non-flap [t].

[1] A function $f : \mathcal{X} \to \mathcal{Y}$ is surjective if every element y in $\mathcal{Y}$ can find at least one element x in $\mathcal{X}$ that satisfies $y = f(x)$.

[2] See Figure 2 of [94]. However, such a network cannot be trained using gradient descent method since the binary step activation function is not differentiable.

If we want the network to learn from such data as a deterministic function, we may add discriminant information as the additional input, e.g., adding the phoneme context to the identity /t/. This method forces the mapping from the input to the output to be surjective, and the neural network may be trained to perfectly produce the expected output given the input. However, this method ignores that the input/output data may contain noise or error of measurement. The 'function' learned by the network may simply overfit to the observation noise and be unable to explain the authentic generative process. In all, the deterministic interpretation of neural networks is insufficient and inappropriate.

Probabilistic interpretation

An alternative approach is to interpret neural networks from the perspective of statistical modeling [91]. Let's consider the sequential learning task in the previous section and use the toy FNN to learn it. Note that there may be ambiguous input/output training pairs, i.e., $\boldsymbol{x}_i = \boldsymbol{x}_j$, $\boldsymbol{o}_i \neq \boldsymbol{o}_j$, $j \neq i$. To deal with such cases, we can assume that true $\boldsymbol{o}_t$ is equal to the sum of the network's output $f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t)$ and a noise component $\boldsymbol{\epsilon}_t$, which can be written as

$$\boldsymbol{o}_t = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t) + \boldsymbol{\epsilon}_t. \tag{3.8}$$

This formula attributes the ambiguity to the observation noise in the output values, which will lead to interpretations on the common usage of neural networks. For analytic simplicity, we assume that $\boldsymbol{\epsilon}_t$ is drawn from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\epsilon}_t; \mathbf{0}, \sigma^2\boldsymbol{\mathcal{I}})$ with zero mean and a diagonal covariance matrix $\sigma^2\boldsymbol{\mathcal{I}}$, where $\boldsymbol{\mathcal{I}}$ denotes an identity matrix. Accordingly, we implicitly define a conditional probabilistic distribution for $\boldsymbol{o}_t$ as

$$p(\boldsymbol{o}_t|\boldsymbol{x}_t; \boldsymbol{\Theta}) = \frac{1}{(\sqrt{2\pi\sigma^2})^{D_o}} \exp\Big(- \frac{||\boldsymbol{o}_t - f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t)||^2}{2\sigma^2}\Big), \tag{3.9}$$

where D_o is the dimension of $\boldsymbol{o}_t$. It suggests that the network's output determines the mean of the distribution $p(\boldsymbol{o}_t|\boldsymbol{x}_t)$ rather than a value $\widehat{\boldsymbol{o}}_t$ that we get from this distribution. This interpretation is quite reasonable as we will explain later.

On the basis of Equation (3.9), the network weights $\boldsymbol{\Theta}$ can be trained using a

maximum-likelihood criterion

$$\begin{aligned}\boldsymbol{\Theta}^* &= \arg\max_{\boldsymbol{\Theta}} \log p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta}) \\ &= \arg\max_{\boldsymbol{\Theta}} \sum_{t=1}^{T} \log p(\boldsymbol{o}_t|\boldsymbol{x}_t;\boldsymbol{\Theta}) \\ &= \arg\max_{\boldsymbol{\Theta}} \sum_{t=1}^{T} \log \Big[\frac{1}{(\sqrt{2\pi\sigma^2})^{D_o}} \exp\Big(-\frac{||\boldsymbol{o}_t - f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t)||^2}{2\sigma^2}\Big)\Big] \\ &= \arg\min_{\boldsymbol{\Theta}} \sum_{t=1}^{T} \frac{1}{2\sigma^2}||\boldsymbol{o}_t - f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t)||^2.\end{aligned} \tag{3.10}$$

Note that we drop the additive terms unrelated to $\boldsymbol{\Theta}$ in the last line; σ is kept in the equation, but it doesn't affect the estimation of $\boldsymbol{\Theta}^*$. By comparing Equations (3.10) and (3.4), we can notice that the maximum-likelihood criterion under the assumption of Gaussian distribution is equivalent to the MSE criterion. From another perspective, we may think that the MSE-based training method implicitly assumes a Gaussian distribution $p(\boldsymbol{o}_t|\boldsymbol{x}_t)$ with the mean equal to $f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t)$ and a fixed covariance matrix. Note that the best parameter set $\boldsymbol{\Theta}^*$ of a non-linear neural network cannot be found analytically. Only a good solution can be iteratively updated using the gradient descent method.

The next interpretation is on the generation method. Suppose the task is to find a good output $\hat{\boldsymbol{o}}_t$ given the input $\tilde{\boldsymbol{x}}_t$ and a learned parameter set $\boldsymbol{\Theta}^\dagger$. Intuitively, we may let $\hat{\boldsymbol{o}}_t$ be equal to the mean of the Gaussian distribution $p(\hat{\boldsymbol{o}}_t|\tilde{\boldsymbol{x}}_t;\boldsymbol{\Theta}^\dagger)$ because it maximizes the value of the probability density function (PDF). We refer to this generation method as the **mean-based generation method**. Since the mean of $p(\hat{\boldsymbol{o}}_t|\tilde{\boldsymbol{x}}_t;\boldsymbol{\Theta}^\dagger)$ is equal to the network's output, we can directly use the network's output as the generated output, i.e., $\hat{\boldsymbol{o}}_t = f_{\boldsymbol{\Theta}^\dagger}(\tilde{\boldsymbol{x}}_t)$.

The above interpretations on the conditional distribution, the MSE training criterion, and the mean-based generation method also apply to RNNs. If we take the toy RNN in Equation (3.5) as an example, we can get the distribution implemented by the RNN as

$$p(\boldsymbol{o}_t|\boldsymbol{x}_{1:T};\boldsymbol{\Theta}) = \frac{1}{(\sqrt{2\pi\sigma^2})^{D_o}} \exp\Big(-\frac{||\boldsymbol{o}_t - f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t)||^2}{2\sigma^2}\Big), \tag{3.11}$$

where $f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t)$ denotes the output of the RNN at time t. Since this RNN uses a bi-directional recurrent layer, the distribution of $\boldsymbol{o}_t$ becomes dependent on the whole input data sequence $\boldsymbol{x}_{1:T}$ rather than $\boldsymbol{x}_t$ only. This RNN can be trained using the same maximum-likelihood training criterion

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \log p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta}) = \arg\max_{\boldsymbol{\Theta}} \sum_{t=1}^{T} \log p(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta}). \tag{3.12}$$

It can generate an output sample by taking the mean of the inferred distribution in a similar manner to the FNN, i.e., $\widehat{\boldsymbol{o}}_t = f_{\boldsymbol{\Theta}^\dagger}(\tilde{\boldsymbol{x}}_{1:T})$.

Assess the probabilistic interpretation

The probabilistic interpretation is quite general and can be applied to FNNs or RNNs with more complex network structures. Particularly, the mean-based generation method suggests that the network's output can be interpreted as the mean of the conditional distribution of the target data. This interpretation is more understandable than treating the network's output as a fixed approximation to an unknown target datum.

The probabilistic interpretation, however, raises one question: is it appropriate to assume a Gaussian distribution in Equation (3.9)? We cannot answer it completely since we don't know the true conditional distribution $\mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_t)$ or $\mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T})$[3][4]. However, we use a Gaussian distribution rather than other distributions because there is a correspondence between the distribution assumed in the probabilistic interpretation and the error measure used in common scenarios [99]. For example, maximizing the likelihood on a Gaussian distribution leads to the MSE-criterion as Equation (3.10) shows. If a Poisson distribution is assumed, maximizing its likelihood is equivalent to minimize a generalized Kullback-Leibler divergence between the natural data vectors and the outputs of the network [100]. In general, the Gaussian distribution is assumed not merely for mathematic simplicity.

[3] We use the special font $\mathfrak{p}$ to denote a true but unknown distribution.

[4] The distribution should be written as $p(\boldsymbol{O}_t = \boldsymbol{o}_t|\boldsymbol{X}_t = \boldsymbol{x}_t)$ where $\boldsymbol{O}_t$ and $\boldsymbol{X}_t$ denote the random variables while $\boldsymbol{o}_t$ and $\boldsymbol{x}_t$ denote the values taken by the random variables. To avoid dense notation, we drop the symbols of random variables when the meaning of a mathematic symbol is self-evident.

Another question is whether the network output should be interpreted as the conditional mean of the target data in Equation (3.9). To answer this question, we need to define a general error function based on the square error. Suppose we are using an RNN, whose output at time t is $\widehat{\boldsymbol{o}}_t \triangleq f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t)$. Accordingly, the square error for the time t can be measured using the true distribution $\mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T})$ as

$$SE(t, \boldsymbol{\Theta}) = \int_{\boldsymbol{o}_t} ||\widehat{\boldsymbol{o}}_t - \boldsymbol{o}_t||^2 \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t. \tag{3.13}$$

Let $\mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}} \triangleq \int_{\boldsymbol{o}_t} \boldsymbol{o}_t \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t$ denote the mean of the true conditional distribution. We can then derive $SE(t, \boldsymbol{\Theta})$ as

$$\begin{aligned}
SE(t, \boldsymbol{\Theta}) &= \int_{\boldsymbol{o}_t} ||\widehat{\boldsymbol{o}}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}} + \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}} - \boldsymbol{o}_t||^2 \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t \\
&= \int_{\boldsymbol{o}_t} ||\widehat{\boldsymbol{o}}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}||^2 d\boldsymbol{o}_t + \int_{\boldsymbol{o}_t} ||\boldsymbol{o}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}||^2 \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t \\
&\quad + 2 \int_{\boldsymbol{o}_t} (\widehat{\boldsymbol{o}}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}})^\top (\boldsymbol{o}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}) \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t \\
&= \int_{\boldsymbol{o}_t} ||\widehat{\boldsymbol{o}}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}||^2 d\boldsymbol{o}_t + \int_{\boldsymbol{o}_t} ||\boldsymbol{o}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}||^2 \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t,
\end{aligned} \tag{3.14}$$

where the third line is derived based on the fact that

$$\begin{aligned}
&\int_{\boldsymbol{o}_t} (\widehat{\boldsymbol{o}}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}})^\top (\boldsymbol{o}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}) \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t \\
=&(\widehat{\boldsymbol{o}}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}})^\top \Big(\int_{\boldsymbol{o}_t} \boldsymbol{o}_t \mathfrak{p}(\boldsymbol{o}_t|\boldsymbol{x}_{1:T}) d\boldsymbol{o}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}} \Big) \\
=&(\widehat{\boldsymbol{o}}_t - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}})^\top (\mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}} - \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}) = 0
\end{aligned} \tag{3.15}$$

Since the second term in the last line of Equation (3.14) is the intrinsic variance of the data, it is unrelated to the RNN. Therefore, $SE(t, \boldsymbol{\Theta})$ is minimized when $\widehat{\boldsymbol{o}}_t \triangleq f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t) = \mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}$. This result suggests that a network's output will approximate the true mean of target data if two conditions can be satisfied: the square-error calculated over a training corpus approximates the true square error $SE(t, \boldsymbol{\Theta})$; the RNN is perfectly trained to minimize the square error over the training data. This result also indicates that it is reasonable to interpret the network's output as the approximation to the true conditional mean $\mathbb{E}_{\boldsymbol{o}_t|\boldsymbol{x}_{1:T}}$. It is thus also reasonable to use the mean-based generation method.

3.3 Mixture density network

A normal FNN or RNN can be considered as a probabilistic model that uses a Gaussian distribution to approximate the target data distribution. The true data distribution, however, may not be Gaussian or even uni-mode. The assumed Gaussian distribution may poorly fit the training data [101], and the mean of the Gaussian distribution may be far away from any mode of the true data distribution. Another shortcoming is that the variance of the assumed distribution is not updated and used at all in a normal FNN or RNN.

To model the multi-mode data distribution, we need a probabilistic model using a more flexible distribution. One candidate is the mixture density network (MDN) [102], a type of probabilistic model that combines normal neural networks and Gaussian mixture models (GMMs). Let's consider an MDN that combines a bi-directional RNN and a GMM with M mixture components and use it in the sequential learning task. Accordingly, we can write down the conditional PDF $p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta})$ as

$$p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{x}_{1:T};\boldsymbol{\Theta}) = \prod_{t=1}^{T}\sum_{m=1}^{M} \omega_t^m \mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_t^m,\boldsymbol{\Sigma}_t^m), \tag{3.16}$$

where $\{\omega_t^m, \boldsymbol{\mu}_t^m, \boldsymbol{\Sigma}_t^m\}$ denotes the weight, mean, and covariance matrix of the Gaussian distribution $\mathcal{N}(\cdot)$ in the m-th mixture component at time t. At the time t, the parameter set of the GMM $\boldsymbol{\mathcal{M}}_t$ is generated by the RNN given $\boldsymbol{x}_{1:T}$ as input, which can be written as

$$\boldsymbol{\mathcal{M}}_t \triangleq [\omega_t^1, (\boldsymbol{\mu}_t^1)^\top, \mathrm{vec}(\boldsymbol{\Sigma}_t^1)^\top, \cdots, \omega_t^M, (\boldsymbol{\mu}_t^M)^\top, \mathrm{vec}(\boldsymbol{\Sigma}_t^M)^\top] = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t), \tag{3.17}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix.

Note that the MDN is defined as the combination of a normal neural network and a GMM, rather than being interpreted as a probabilistic model. An MDN may also use an FNN to generate parameter set of GMM. To differentiate the MDN using an FNN and that using an RNN, we refer to the former case as FMDN while the latter as recurrent MDN (RMDN)[5]. An RMDN based on the toy RNN network

[5] To our knowledge, the name was first used in [103].

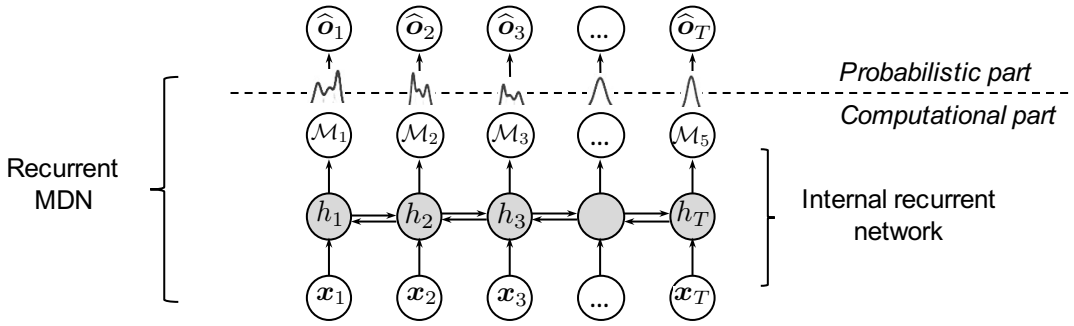


Figure 3.4: A toy RMDN with an internal RNN.

in Figure 3.3 is plotted in Figure 3.4. Notice that the forward and backward branches of the recurrent layer are denoted by $\rightarrow$ and $\leftarrow$, and $\{\boldsymbol{h}_t^{(f)}, \boldsymbol{h}_t^{(b)}\}$ are concatenated as $\boldsymbol{h}_t$.

The trainable parameter set $\boldsymbol{\Theta}$ of an MDN consists of the network weights of the internal network. They can be learned from a corpus using the gradient descent approach under the maximum likelihood criterion. In the generation stage, the MDN converts the input data $\tilde{\boldsymbol{x}}_{1:T}$ into the parameters of GMMs $\boldsymbol{\mathcal{M}}_t = f_{\boldsymbol{\Theta}}(\tilde{\boldsymbol{x}}_{1:T}, t)$. The GMM then can generate the output $\widehat{\boldsymbol{o}}_t$ by using the mean of the most probable mixture component:

$$\begin{aligned} \widehat{\boldsymbol{o}}_t &= \arg\max_{\boldsymbol{o}_t} \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_t^{m^*}, \boldsymbol{\Sigma}_t^{m^*}) = \boldsymbol{\mu}_t^{m^*}, \\ \text{where} \quad m_* &= \arg\max_m w_t^m. \end{aligned} \tag{3.18}$$

This is the mean-based generation method for MDN.

3.4 Neural classifier models for classification

Motivation

Neural networks or MDN reviewed in the previous sections are defined for regression tasks, where the target data are assumed to be real continuous-valued numbers. This section considers the models for classification tasks. By saying classification tasks, we mean that the target datum o_t at time t can only take one of the K possible outcomes from a set of categorical data $\{1, \cdots, K\}$.

The reader may think that the network for the XOR problem in Section 3.1.1 is a classification task. In fact, although the target for that task is either '0' or '1', the network implicitly assumes that the target data are two continuous numbers 0 and 1. Otherwise, the network should not be trained using the MSE-criterion because the square error in the Euclidean space is only defined for continuous numbers but not for categorical symbols.

The reader then may wonder why not treat the classification task as a regression one and treat the symbols '0' and '1' as the real numbers 0 and 1. Indeed, there is no problem to do so. However, this strategy may suffer from the mismatch between model assumption and data distribution. As we explained in Section 3.2, a neural network trained with the MSE criterion implicitly assume a Gaussian for the target data. However, the distribution of the categorical data cannot be assumed to be Gaussian. First of all, the support of the categorical distribution is not from $-\infty$ to $+\infty$, and the Gaussian distribution wastes probability mass to observations that do not belong to $\{1, \cdots, K\}$.

Definition

A better approach is to design a model using a categorical distribution under the MDN framework. Let's consider a sequential classification task, where the target o_t takes one of K possible outcomes, i.e., $o_t \in \{1, \cdots, K\}$. Given an input sequence $\boldsymbol{x}_{1:T}$, the model computes the probability mass function (PMF) for o_t as

$$P(o_t|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta}) = p_{o_t,t}, \tag{3.19}$$

where the probabilities of the K outcomes are computed by an internal RNN as

$$\boldsymbol{\mathcal{M}}_t \triangleq [p_{1,t}, \cdots, p_{K,t}] = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, t). \tag{3.20}$$

Since $\{p_{1,t}, \cdots, p_{K,t}\}$ are probability values between $[0, 1]$ and $\sum_{k=1}^{K} p_{k,t} = 1$, the internal RNN needs to use a softmax output layer [104] to calculate $\boldsymbol{\mathcal{M}}_t$. The whole model can be illustrated using a figure similar to Figure 3.4. The difference is the assumed data distribution, the meaning of $\boldsymbol{\mathcal{M}}_t$, and the way to calculate $\boldsymbol{\mathcal{M}}_t$. This neural model is referred to as 'neural classifier' [104, 105].

The MDN for classification tasks can be trained using the maximum-likelihood criterion. Suppose a training corpus contains one pair of input and output data $\mathcal{D} = \{\boldsymbol{x}_{1:T}, \boldsymbol{o}_{1:T}\}$, where $o_t \in \{1, \cdots, K\}$. The log-likelihood of the MDN on the corpus can be written as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}) &\triangleq \log P(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta}) = \log \prod_{t=1}^{T} P(o_t|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta}) = \log \prod_{t=1}^{T} \prod_{k=1}^{K} P(k|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta})^{\delta_{ko_t}} \\ &= \sum_{t=1}^{T} \sum_{k=1}^{K} \delta_{ko_t} \log p_{k,t},\end{aligned} \tag{3.21}$$

where the Kronecker delta function δ_{ko_t} is equal to 1 only when $k = o_t$; otherwise $\delta_{ko_t} = 0$. The negative of the last line of Equation (3.21) is also known as the cross-entropy between the true data distribution and the categorical distribution calculated by the MDN. A trained MDN can be used for the classification task by assigning the output class

$$\widehat{o}_t = \arg \max_{o_t \in \{1, \cdots, K\}} P(o_t|\boldsymbol{x}_{1:T}, \boldsymbol{\Theta}^*). \tag{3.22}$$

This is a theoretically sound method because $P(o_t|\boldsymbol{x}_{1:T}, \boldsymbol{\Theta}^*)$ approximates the true probability $\mathbb{P}(o_t|\boldsymbol{x}_{1:T})$ [106]. The above framework of MDN has been used in many tasks especially in natural language processing [105, 107, 108].

3.5 Baseline neural F0 modeling method

Based on the introduction to neural networks in this chapter and TTS in the last chapter, we can introduce various neural models for F0 modeling in TTS. Table 3.1 lists some of the recent neural F0 models or acoustic models for SPSS-based TTS. Because researchers usually model F0 together with spectral features, we include those acoustic models as long as they can generate the F0 contour given input linguistic features.

Table 3.1: Some of recent neural F0 models or acoustic models with F0 modeling capability

Neural model	Target F0 representation	Model time resolution	Joint spectral and F0 modeling
FNN [61, 109] FMDN [86] GRU-RNN [110] LSTM-RNN [62]	Interpolated F0 $o_t \in \mathbb{R}$	Frame-by-frame $\boldsymbol{x}_{1:T} \rightarrow \boldsymbol{o}_{1:T}$	Joint modeling
LSTM-RNN [111]	Interpolated F0	From unit-by-unit to frame-by-frame	Joint modeling
DBN [112]	Interpolated F0 of 100 frames in syllable	Syllable-by-syllable	Joint modeling
FNN [113]	Interpolated F0 and wavelet parameters of F0 contour	Frame-by-frame	Joint modeling
LSTM-RNN [87]	Interpolated F0	Frame-by-frame	Only F0
FNNs (parallel) [114]	Interpolated F0 with DCT coefficients of syllable, word, phrase F0 contour	Frame-by-frame	Only F0
LSTM-RNN [115]	Interpolated F0 and F0 templates of syllables	Frame-by-frame	Only F0

FNN: feedforward neural network
RNN: recurrent neural network
GRU: gated recurrent unit [117]
FMDN: mixture density network based on FNN
DBN: deep belief network [116]
LSTM: long-short-term-memory unit [118, 119]

Table 3.1 also shows a few characteristics of each model, including

- Type of target F0 representation: whether it is the original F0 value or other representations derived from the F0 contour;
- Time resolution of model: does the model predicts the F0 value frame-by-frame or at higher linguistic level. Note that, even if the model includes components operating at a higher linguistic level, as long as it generates the final F0 per frame, it is considered as frame-by-frame;
- Joint spectral and F0 modeling: whether F0 and spectral features are used at the same time as the target of the model.

The target F0 representation is important because it determines the accuracy of F0 modeling and the model time resolution. As the table shows, most of the methods use the interpolated F0 contour as the target data. Accordingly, these models must operate frame-by-frame so that the F0 value of each frame can be generated. There is one DBN-based case where the F0 contour is modeled at the syllable level. Specifically, it uses evenly sampled 100 frames from each syllable as the target F0 of that syllable. During generation, the F0 contour at the frame level has to be interpolated from the syllable-level F0 values.

The third point is on the acoustic modeling strategy. Notice that most of the acoustic models jointly model the F0 and spectral features. For the last two methods that only model F0, they introduce additional neural models dedicated to F0 features. There is only one model which uses the common configuration (interpolated F0, frame-by-frame) for F0 modeling only.

The baseline model in this thesis is the RNN [87] that models the interpolated F0 at frame level without spectral features. The reason is that the RNN is the most used, efficient and powerful neural model. Furthermore, the baseline model is assumed to only model the F0 because the network may prioritize spectral features in the case of joint modeling, which has been noticed in [112] and will be further explored in Chapter 4. Finally, a baseline model is not required to use additional F0 representations since this thesis focuses on the model rather than F0 representations. However, methods and models proposed in this thesis can be directly used to model F0 and other sequential F0 representations.

3.6 Limitations of baseline neural F0 models

Although the baseline F0 and other models that can generate the F0 performed very well, we wonder whether F0 modeling performance can be further improved. We are motivated by a few potential issues:

1. **Issue 1: joint modeling or only F0 modeling?** As Table 3.1 shows, neural F0 modeling is usually integrated into the acoustic model where F0 and other acoustic features are modeled at the frame level. It is questionable whether the network could well model F0 when it has to pay attention to other acoustic features. Although simultaneous modeling of multiple types of target features resemble the multi-task learning [120], whether different types of acoustic features could share the hidden features is unaddressed. In fact, when the joint modeling strategy was used, it has been noticed that the FNN didn't outperform HMM in terms of F0 modeling [61].

2. **Issue 2: do the common neural models describe the temporal dependency of F0 contours?** Probably not. As the introduction to neural networks explains, the FNN, RNN and RMDN assume that one target datum $\boldsymbol{o}_{t_1}$ is statistically independent from $\boldsymbol{o}_{t_2}$ given the condition $\boldsymbol{x}_{1:T}$, $\forall t_2 \neq t_1$. This assumption is held whatever the type of F0 representation or the time resolution is. The assumption made by FNN can be observed from Equations (3.10), where the PDF of the target sequence $\boldsymbol{o}_{1:T}$ is decomposed as $p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{x}_{1:t})$. We may tend to believe that an RNN or RMDN should avoid the assumption because of the recurrent layers. However, an RNN or RMDN is only superior to an FNN in modeling the dependency between the target datum $\boldsymbol{o}_t$ and the entire input sequence $\boldsymbol{x}_{1:T}$. This dependency allows the model to assume $p(\boldsymbol{o}_t|\boldsymbol{x}_{1:T})$ rather than $p(\boldsymbol{o}_t|\boldsymbol{x}_t)$, but it cannot build the temporal dependency across $\boldsymbol{o}_{1:T}$.

3. **Issue 3: frame-by-frame processing even for linguistic features?** For the neural models operating frame-by-frame, the input linguistic feature vector is assigned to every frame. However, the linguistic features mainly include the phone identity, prosodic tags, and other features above the frame level. The linguistic feature vectors of adjacent frames may be almost

identical, and processing the linguistic features frame-by-frame is unnecessary. It also makes it more difficult for a recurrent layer to retrieve linguistic features of adjacent linguistic units. Using a hierarchical model [111] or a clockwork RNN [121] may increase the efficiency. But we further wonder whether the model can derive meaningful hidden features when it processes the linguistic features more efficiently.

We will try to answer the first question using a specific type of neural network in Chapter 4. Then, we will address the second question and propose new models in Chapter 5 and 6. The last question will be explored in Chapter 7 by using a new modeling strategy.

4

Investigating F0 Modeling Using Highway Networks

Modeling F0 together with other spectral features seems to be the default strategy for neural SPSS. There may be a historical reason since the initial works on HMM-based SPSS proposed to do so [77]. However, whether this strategy is optimal for F0 modeling remains unclear. This is **Issue 1** investigated in this thesis.

This chapter empirically examines the above issue using a non-recurrent neural network called highway network [122]. After introducing the highway network in Section 4.2, this chapter uses Section 4.3 to show how a neural network prioritizes spectral features rather than F0 in the common SPSS. It further tests how F0 is differently modeled from other acoustic features in Section 4.4 on the basis of two type of statistics, namely the highway gate activation histogram and the sensitivity to the input linguistic features. These results suggest that the F0 may be better modeled when it is modeled separately from other acoustic features.

4.1 Joint modeling of F0 and spectral features?

In addition to the historical reason, one common belief of the default strategy is that it could take advantage of multi-task learning so that the F0 and spectral features can share hidden features. Another belief is that a single network can learn the correlation between spectral and F0 features [123].

The above beliefs may be doubtful. As Chapter 3 explained, a normal FNN or RNN assumes a Gaussian distribution with a diagonal covariance matrix for target data vector. Accordingly, the model assumes the distribution of the F0 and those of the spectral features are independent from each other. The other belief on shared hidden features is also doubtful as we will show in this chapter.

In fact, there has been some literature showing evidence against the default strategy. In [61], it was shown that the F0 RMSE achieved by the FNN was similar to that of the HMM even though the FNN outperformed HMM in spectral features with a large margin. Another work also showed that the performance of F0 modeling became worse when spectral features were better generated [109].

In this chapter, we decided to use the highway neural network to test the default strategy thoroughly. Although its performance may be inferior to RNN, it is prohibitive to train and evaluate many RNNs, especially very deep and wide ones, for the experiments planned in this chapter. Using the highway network allows us to do the experiments quickly. Compared with the FNN, the highway network is easier to train. More importantly, the highway network provides hidden features allowing us to look inside the network.

4.2 Definition of highway network

4.2.1 Computation flow

A highway network is based on an FNN and special gating mechanism. Let's consider a highway network based on the toy FNN in Section 3.1.2. As Figure 4.1 plots, this highway network extracts two hidden vectors on the basis of the input

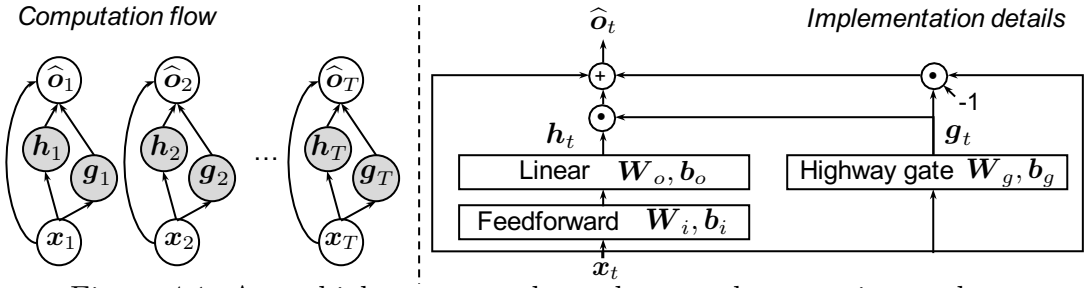


Figure 4.1: A toy highway network. $\odot$ denotes element-wise product.

vector $\boldsymbol{x}_t$

$$\boldsymbol{h}_t = \boldsymbol{W}_o f(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{b}_i) + \boldsymbol{b}_o, \tag{4.1}$$

$$\boldsymbol{g}_t = \sigma(\boldsymbol{W}_g \boldsymbol{x}_t + \boldsymbol{b}_g), \tag{4.2}$$

where $\sigma(\cdot)$ is the logistic activation function and $f(\cdot)$ denotes an activation function which can be tanh. The network then merges these two vectors using a element-wise product $\odot$ and derive the final output as

$$\widehat{\boldsymbol{o}}_t = \boldsymbol{g}_t \odot \boldsymbol{h}_t + (\boldsymbol{1} - \boldsymbol{g}_t) \odot \boldsymbol{x}_t. \tag{4.3}$$

Each dimension of $\boldsymbol{g}_t$ is constrained by the sigmoid activation function and lies between 0 and 1. When each dimension of $\boldsymbol{g}_t$ is close to 0, $\widehat{\boldsymbol{o}}_t$ is roughly equal to $\boldsymbol{x}_t$; otherwise $\widehat{\boldsymbol{o}}_t$ approximates $\boldsymbol{h}_t$. The path that delivers $\boldsymbol{x}_t$ to the output is referred to as the highway connection. Since $\boldsymbol{g}_t$ controls how much information from the input $\boldsymbol{x}_t$ is directly sent to the output $\widehat{\boldsymbol{o}}_t$, it is referred to as a highway gating vector; the layer that computes $\boldsymbol{g}_t$ is referred to as a highway gate.

The toy highway network uses the highway gate at the output layer and requires that $\boldsymbol{x}_t$, $\boldsymbol{g}_t$, $\boldsymbol{h}_t$, and $\widehat{\boldsymbol{o}}_t$ have the same dimension. In practice, a deep highway network may use the gating mechanism among the hidden layers as Figure 4.2 shows. For convenience, we refer to a highway gate layer and the hidden layers covered by the gate as a highway block. Such a deep highway network only requires $\{\boldsymbol{x}_t, \boldsymbol{h}_t, \boldsymbol{g}_t\}$ within a highway block to have the same dimension. The network can easily change the dimension of different highway blocks using a linear layer.

Compared with FNNs, highway networks ease the training process even when they are very deep. One reason is that they alleviate the gradient vanishing

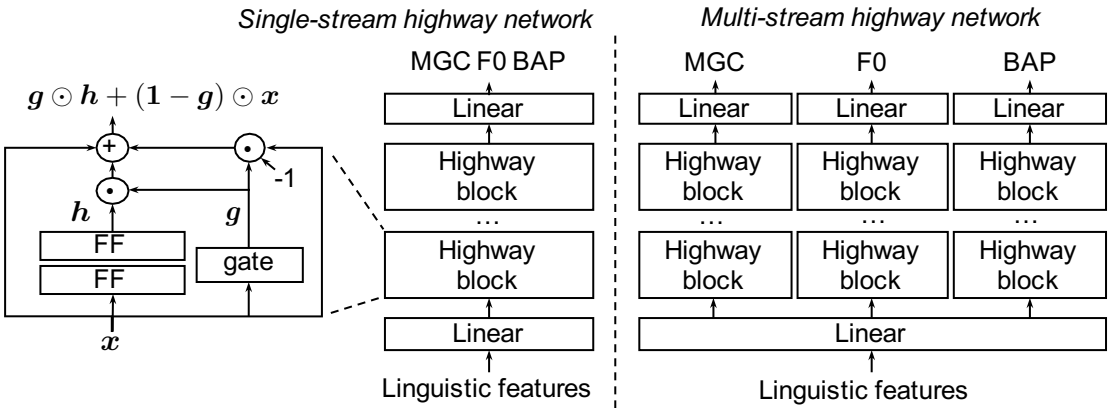


Figure 4.2: Single-stream (left) and multi-stream (right) highway network structure. 'FF' and 'linear' denotes a non-linear and a linear feedforward layer, respectively.

problem using the highway gate and links [124] . When the highway gate opens, the input of each highway block is propagated by the highway link to the next block without being non-linearly transformed. In such a case, the gradients propagated backward are not attenuated by the non-linear transformation layers.

The definition of highway network is not unique. A more general definition may introduce different formulae to calculate the gating and output vectors [122]. On the other hand, a more specific definition may exclude the trainable highway gate and directly sum the $\boldsymbol{h}_t$ and $\boldsymbol{x}_t$, which is known as the residual network [125]. We choose the definition in Equation (4.3) because it allows us to easily analyze the behavior of the network as we will mention in the next subsection.

4.2.2 Multi- and single-stream highway network for SPSS

A highway network can be used as the acoustic model for SPSS in the same manner to an FNN. Suppose the target vector $\boldsymbol{o}_t$ at the t-th time step consists of F0, MGC, and BAP features, which can be written as $\boldsymbol{o}_t = [\boldsymbol{o}_t^{(F0)\top}, \boldsymbol{o}_t^{(MGC)\top}, \boldsymbol{o}_t^{(BAP)\top}]^\top$. Accordingly the corresponding output of a highway network can be denoted by $\hat{\boldsymbol{o}}_t = [\hat{\boldsymbol{o}}_t^{(F0)\top}, \hat{\boldsymbol{o}}_t^{(MGC)\top}, \hat{\boldsymbol{o}}_t^{(BAP)\top}]^\top = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_t)$, where $\boldsymbol{\Theta}$ and $\boldsymbol{x}_t$ denotes the weights of the highway network and the input linguistic feature vector, respectively. This highway network can be trained using the MSE-based criterion (Equation (3.4)) on the training set; it then can generate the output $\hat{\boldsymbol{o}}_t$ for $\boldsymbol{x}_t$ of a new text.

Such a highway network can be used to jointly model F0 and other acoustic features. For example, if the network uses a linear transformation output layer and the mean-based generation method, it calculates the output as

$$\widehat{\boldsymbol{o}}_t = \begin{bmatrix} \widehat{\boldsymbol{o}}_t^{(F0)} \\ \widehat{\boldsymbol{o}}_t^{(MGC)} \\ \widehat{\boldsymbol{o}}_t^{(BAP)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{W}_{o,11} & \boldsymbol{W}_{o,12} & \boldsymbol{W}_{o,13} \\ \boldsymbol{W}_{o,21} & \boldsymbol{W}_{o,22} & \boldsymbol{W}_{o,23} \\ \boldsymbol{W}_{o,31} & \boldsymbol{W}_{o,32} & \boldsymbol{W}_{o,33} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_{t,1} \\ \boldsymbol{h}_{t,2} \\ \boldsymbol{h}_{t,3} \end{bmatrix} = \boldsymbol{W}_o \boldsymbol{h}_t, \tag{4.4}$$

where $\boldsymbol{W}_{o,ij}$ denotes a block matrix in the transformation matrix $\boldsymbol{W}_o$ of the output layer, and $\boldsymbol{h}_{t,j}$ denotes a sub-part of the hidden feature vector $\boldsymbol{h}_t$. In this modeling strategy, the generated F0, MGC and BAP share the same set of base vectors in $\boldsymbol{h}_t$. Similarly, $\boldsymbol{h}_t$ may be transformed from a hidden feature vector in the previous layer. It means that $\{\boldsymbol{o}_t^{(F0)}, \boldsymbol{o}_t^{(MGC)}, \boldsymbol{o}_t^{(BAP)}\}$ share the same hidden feature vectors in the network. This strategy is commonly used in neural SPSS as Section 3.5 explained. However, it is different from the strategy in HMM-based SPSS framework where F0 and other acoustic features are modeled in independent streams [77]. For this reason, we refer to this network structure as a **single-stream network**, no matter whether it is a highway network or an FNN.

As a comparison, we consider a **multi-stream** structure, where different output acoustic feature vectors are separately modeled. For example, a multi-stream network may compute the output vectors as

$$\widehat{\boldsymbol{o}}_t = \begin{bmatrix} \widehat{\boldsymbol{o}}_t^{(F0)} \\ \widehat{\boldsymbol{o}}_t^{(MGC)} \\ \widehat{\boldsymbol{o}}_t^{(BAP)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{W}_{o,11} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{o,22} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{W}_{o,33} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_{t,1} \\ \boldsymbol{h}_{t,2} \\ \boldsymbol{h}_{t,3} \end{bmatrix} = \boldsymbol{W}_o \boldsymbol{h}_t, \tag{4.5}$$

where $\widehat{\boldsymbol{o}}_t^{(F0)}$, $\widehat{\boldsymbol{o}}_t^{(MGC)}$, and $\widehat{\boldsymbol{o}}_t^{(BAP)}$ use different base vectors and transformation matrices. With this sparse transformation being used in the highway gates and other hidden layers, the hidden features for different target features are separated.

A multi-stream highway network can be simply implemented as a combination of multiple single-stream networks, where each single-stream network models only one specific acoustic feature type. Examples of the single- and multi-stream highway network are plotted in Figure 4.2. Notice that the multi-stream one uses a shared linear layer to change the dimension of input linguistic features.

4.3 Evaluation methodology and analysis tools

4.3.1 Evaluation methodology

As we questioned in Section 4.1, using the single-stream structure may be unwise for F0 modeling in SPSS. In a single-stream network, the hidden feature vector encodes all the information for generating both F0 and other acoustic features. It may prioritize the part for the high-dimensional acoustic features rather than the low-dimensional F0 because the former contributes more to the training error. If the empirical evaluation supports the hypothesis, it will suggest that the hidden features for the F0 are different from those for other acoustic features. It further indicates that the F0 and other acoustic features may rely on different input linguistic features.

The hypothesis on the potential limitation of the single-stream structure may be justified by evaluating a single-stream network against a multi-stream one. However, we cannot directly compare the performance of these two network types. As Equations (4.4-4.5) suggest, the two types of network have different numbers of network weights. Even though the network size can be set to be roughly equal by adjusting the network width, in a single-stream network, we cannot know how much 'bandwidth' of the hidden vector is used for the F0. It thus meaningless to compare the accuracy of different acoustic features.

Since our goal is to investigate the potential limitation of the single-stream structure, we can examine how the performance of a network changes with a varying factor. For example, we can examine the error curve of the generated F0 after evaluating a specific network type with different numbers of hidden layers or varied layer sizes. For this experiment, we will compare error curves of the single- and multi-stream highway networks while using the single-stream FNN as a reference.

We mainly test the highway networks because a deep FNN is not easy to train due to the gradient vanishing problem. Another reason is that the highway network is an ideal testbed to investigate the hidden features, as will be pointed out in the next section. Even though the highway network is different from FNN, we think the evaluation results can generalize to FNNs.

To verify the hypothesis on the hidden features, i.e., whether F0 and other acoustic features rely on different hidden features and input linguistic features, we use a histogram-based analysis and an entropy-based analysis on the highway network. These analysis tools are explained in the next section.

4.3.2 Analysis tools

Histogram of highway gating vectors

A highway block uses a gating vector to merge the output vector of non-linear transformation layers with the input vector (i.e., $\boldsymbol{g}_t \odot \boldsymbol{h}_t + (\mathbf{1} - \boldsymbol{g}_t) \odot \boldsymbol{x}_t$). The value of the gating vector indicates the degree of non-linear transformation exerted by the highway block on the input feature vector. Therefore, we can draw a histogram of the gating vector values for each highway block in a multi-stream highway network. By comparing the histograms from different sub-streams, we can evaluate the difference of the hidden features derived for different sub-streams.

Here, we draw histograms on the test set. Specifically, we feed in the input feature vectors $\boldsymbol{x}_{1:T}$ of each test utterance to the network, collect the highway gating vectors $\boldsymbol{g}_{1:T}$ from one specific highway block, and assign each $\boldsymbol{g}_t$ to a phoneme-dependent histogram based on the phoneme identity at time t. In total, we can draw M histograms for each highway block, where M is equal to the number of phonemes in the corpus.

Sensitivity to input linguistic features

The second tool investigates the contribution of the linguistic features in each sub-stream of the multi-stream network. This tool measures the sensitivity of a highway gate to a certain class of input linguistic features, for example the phoneme identity and the part-of-speech tags.

Similar to the steps in drawing histograms, we use the input feature vector $\boldsymbol{x}_{1:T}$ to excite the network and collect the highway gating vectors $\boldsymbol{g}_{1:T}$ from a specific highway block. Suppose the dimension of the gating vector is K. We calculate a statistic for each dimension $k \in \{1, \cdots, K\}$ with respect to each linguistic feature

value s in a feature class $\boldsymbol{S}$. Specifically, we compute an average vector

$$\tilde{a}_k^{(s)} = \frac{\sum_{t=1}^{T} \delta(t,s) g_{t,k}}{\sum_{t=1}^{T} \delta(t,s)}, \tag{4.6}$$

where the indicator function $\delta(t,s)$ is equal to 1 if the input linguistic feature $\boldsymbol{x}_t$ contains a specific feature value s, for example the phoneme identity $/a/$. In other words, $\tilde{a}_k^{(s)}$ denotes the average value of $g_{t,k}$ over the time t where $\boldsymbol{x}_t$ contains s. In practice, we compute $\tilde{a}_k^{(s)}$ over the test set. Given $\tilde{a}_k^{(s)}$ for every s in a linguistic feature class $\boldsymbol{S}$ such as the phoneme set $\{/a/, /t/, \cdots\}$, we normalize it as

$$a_k^{(s)} = \frac{\exp(\tilde{a}_k^{(s)})}{\sum_{s \in \boldsymbol{S}} \exp(\tilde{a}_k^{(s)})}. \tag{4.7}$$

We finally calculate an entropy value for each k and $\boldsymbol{S}$ as

$$E_{\boldsymbol{S},k} = \frac{-\sum_{s \in \boldsymbol{S}} a_k^{(s)} \log a_k^{(s)}}{-\sum_{s \in \boldsymbol{S}} \frac{1}{|\boldsymbol{S}|} \log \frac{1}{|\boldsymbol{S}|}}, \tag{4.8}$$

where $|\boldsymbol{S}|$ denotes the number of values that s can take.

We calculate $E_{\boldsymbol{S},k}$ for each linguistic feature class listed in Appendix A.3. For a quantitive linguistic feature $\boldsymbol{S}$ such as the number of phonemes in a syllable, the possible value of s is quantized into finite categories $\{1, 2, \cdots, N-1, \geq N\}$, where N is a manually defined number[1]. The value of $E_{\boldsymbol{S},k}$ is maximized only when $a_k^{(s)} = \frac{1}{|S|}$, i.e., $\tilde{a}_k^{(s)}$ is equal for each $s \in \boldsymbol{S}$. In this case, the k-th dimension of the highway gate vector does not change no matter what the input linguistic feature s is. In other words, the k-th dimension is insensitive to the feature class $\boldsymbol{S}$. Overall, $\boldsymbol{E}_{\boldsymbol{S}} = [E_{\boldsymbol{S},1}, \cdots, E_{\boldsymbol{S},K}]$ indicates the sensitivity of the highway gate to $\boldsymbol{S}$.

This sensitivity measure was originally defined for neural networks in speech recognition systems [126]. The difference is that $\tilde{a}_k^{(s)}$ is an average rather than a sum of the vectors. Taking the average ensures that $\tilde{a}_k^{(s)}$ is less relevant to the number of times that s appears in the data set, which is more suitable for the linguistic features that appear only a few times in a corpus.

[1] We directly used the N defined in the question set of HTS-demo for English and Japanese. The HTS-demo can be found at http://hts.sp.nitech.ac.jp/?Download.

4.4 Results and analyses on the English corpus

In this section, we conducted experiments on the English corpus using the single- and multi-stream highway networks. The single-stream FNN was included as a reference network. Details about the speech corpus and input/output figure configuration can be found in Appendix A. Specifically, the input linguistic features and output acoustic features (MGC, F0, BAP) are listed in Table A.3 and A.5. The corpus is discussed in Table A.4.

The experiments evaluated the networks with different depth (experiment I) or width (experiment II). Network configuration is detailed in each experiment. For both experiments, the stochastic gradient descent (learning rate 1e-6) with early stopping was used for training. The objective metrics defined in Section 2.3.4 were calculated on the test data, given natural duration.

4.4.1 Results of objective evaluation

Experiment I: Varying network depth

The first experiment varied the network depth. In the case of highway networks, the network depth refers to the number of non-linear transformation layers, excluding the highway gate layers; in the case of the FNN, it refers to the total number of non-linear transformation hidden layers. The range of the network depth was set to $\{2, 4, 8, 14, 20, 40\}$.

Specifically, the single-stream highway network used a layer size of 382 for all the highway blocks, which was equal to the dimension of the input linguistic feature vector. Each highway block contained 2 non-linear transformation layers using a tanh activation function and a gate layer using a logistic activation function. The multi-stream highway network contained three sub-networks for MGC, F0, and BAP, where each sub-network had the layer size of 256. Other configurations of the highway block was the same as that of the single-stream one. Note that, for a highway network of depth d, the number of highway blocks was equal to $d/2$. Finally, the single-stream FNN had a layer size of 382 and used the tanh activation function. All the networks were initialized using the normalized initialization strategy [127]. The bias vector in the highway gate ($\boldsymbol{b}_g$ in Equation (4.2)) was

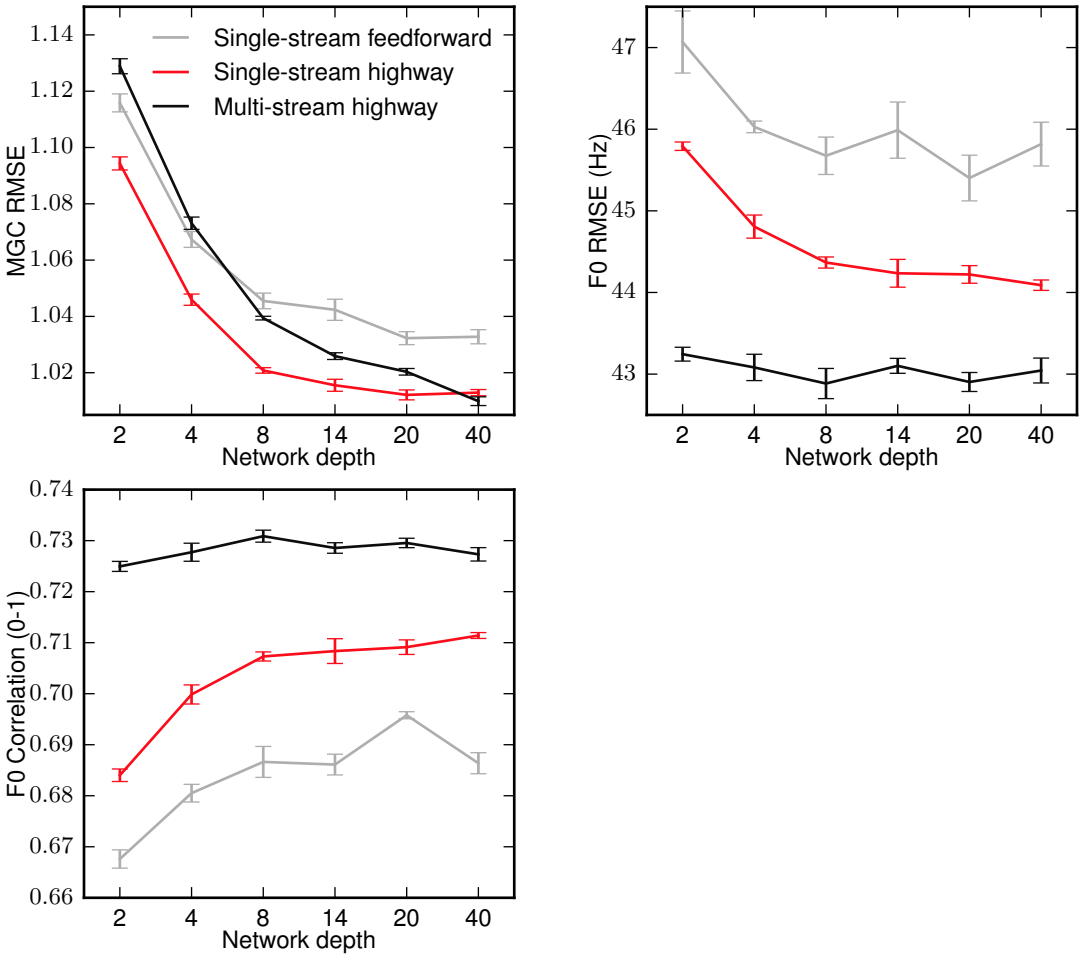


Figure 4.3: Objective results of single- and multi-stream networks. Network depth means the total number of transformation layers using tanh activation function.

initialized as -1.5 according to the original paper of highway network [122].

The results are shown in Figure 4.3, where the y-axis is the objective metric and x-axis is the network depth. These results show that the MGC RMSE decreased when the network depth was increased from 2 to 40 in all the three types of networks. Obviously, the performance of MGC modeling can be consistently improved if the network depth is increased.

However, different trends can be observed from the results on F0. For example on F0 correlation, while the results of the single-stream highway and FNN started from a low point and gradually improved as the network became deeper, the

Table 4.1: Network structure of multi-stream highway networks in Figure 4.4

	Layer size of the sub-network		
	MGC stream	F0 stream	BAP stream
MS_1	256	256	256
MS_2	382	256	256
MS_3	512	382	256
MS_4	768	512	256

curve of the multi-stream network was quite flat. These results suggest that the performance of a multi-stream highway network on F0 modeling may not be further improved even if the network depth is increased.

Note that we cannot directly compare the multi- and single-stream networks in terms of the objective results because the multi-stream network had a larger layer width (256 * 3) than the single-stream one (382). Nevertheless, the results at least suggest that the network depth has different effects on the multi- and single-stream networks in terms of F0 modeling.

Experiment II: Varying network width

The second experiment varied the network width, i.e., the size of all the hidden layers. The depth of every network was fixed to 14 based on the results of Experiment I. The experimental network width was $\{382, 482, 582, 782, 1024\}$ for the single-stream highway network and $\{382, 782, 8821024\}$ for the single-stream FNN. The range of width for the FNN was selected so that the single-stream FNN and highway network can be compared in terms of the network size. The width configuration for the multi-stream highway networks is listed in Table 4.1. The training recipe was the same as Experiment I.

Objective results are shown in Figure 4.4 where the y-axis is the objective metric while the x-axis is the number of network weights. Similar to the results in the previous section, the multi-stream highway network achieved similar results on F0 whatever the network width was. In contrast, the single-stream networks achieved better results on F0 only when the network width increased. On MGC, all the networks performed better when the width was increased.

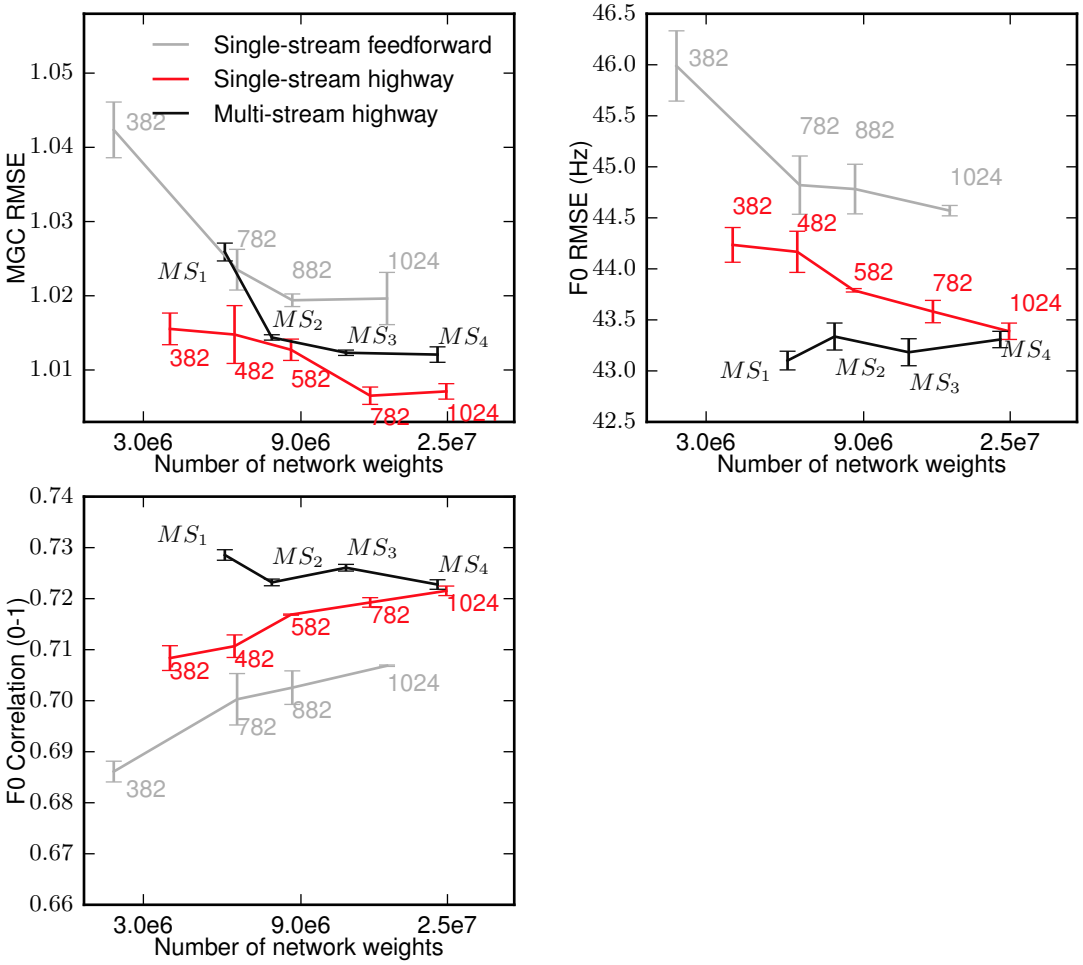


Figure 4.4: Objective evaluation on single-stream and multi-stream networks. The integer number on the red and grey lines denotes the layer size; MS_x denotes the multi-stream network configuration listed in Table 4.1.

The results of Experiment I and II are consistent. For both multi- and single stream networks, a larger network can improve the performance on MGC modeling. However, a multi-stream network does not require a larger network size for F0 modeling. One hypothesis is that a single-stream network may be dominated by the hidden features of MGC. If this hypothesis was correct, a network must be sufficiently large in order to assign some of the network capacity to model the F0. The hypothesis casts doubt on the default strategy of neural SPSS to model the F0 together with spectral features.

4.4.2 Analyses of hidden representations

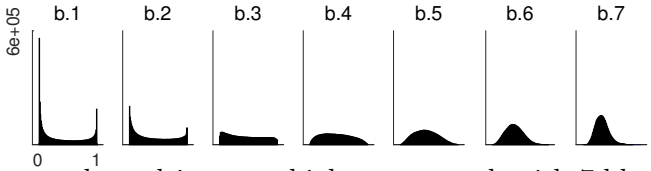
Empirical results above also suggest that the F0 and spectral features may not share a common hidden representation inside the single-stream network. Otherwise, the performance of F0 modeling would be similar whether the network is single- or multi-stream. These results further cast doubt on the default strategy of neural SPSS: If different types of target features do not share a common hidden feature, there is no advantage in modeling them in a single-stream network as a multi-task learning task. This section analyzes the hidden features to find the answer.

On multi-stream highway networks

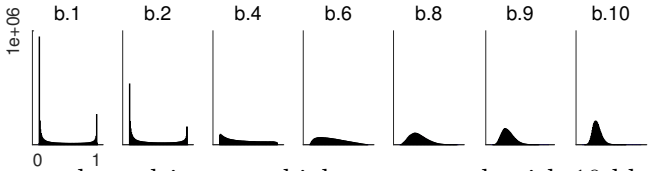
We used the histogram defined in Section 4.3.2 to analyze the multi-stream highway networks in Figure 4.3. Specifically, we collected the histograms on the test data for each highway block in each sub-network. Figure 4.5 (a) shows the histograms for the 7 highway blocks in the MGC sub-network of the multi-stream network with depth 14. Interestingly, the histogram of the first highway block near the input side (b.1 on the left-hand side) shows a binomial distribution. From the second block to the last one, the shape of the histogram gradually changes into a bell. Figure 4.5 (d) shows the histogram for the highway blocks in the F0 sub-network. Contrary to those in the MGC sub-network, none of the histograms in the F0 sub-network show a binomial distribution. From the input to the output side, the histograms had a increased degree of sharpness. In general, the histograms in the MGC and F0 sub-networks show different patterns.

In the highway block near the input side of the MGC sub-network (e.g., b.1), the histogram has one peak near 0 and another peak near 1, which indicates that the highway block used non-linear transformations on some of feature dimensions. The histogram b.7 in the F0 sub-network has a single spike near 0.2, which was determined by the initial value -1.5 of the bias vector $\boldsymbol{b}_g$ in the highway gate $(1/(1+\exp(1.5)))$. A histogram with a shape spike around this location indicates that parameters in that highway gate were not updated intensively. Meanwhile, this highway gate delivered most of the information from the input to the output without transformation. Such a highway block was inactive.

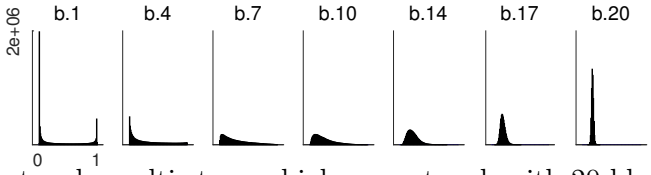
From the explanation above we can know that different histogram patterns in



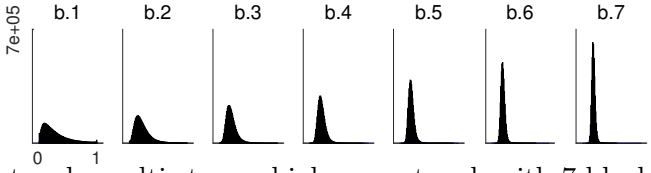
(a) MGC sub-network, multi-stream highway network with 7 blocks (depth 14)



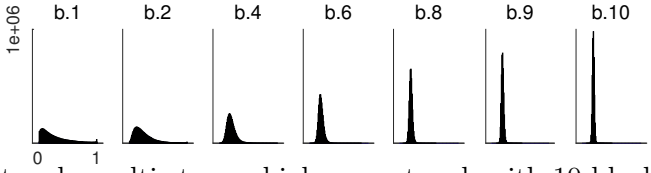
(b) MGC sub-network, multi-stream highway network with 10 blocks (depth 20)



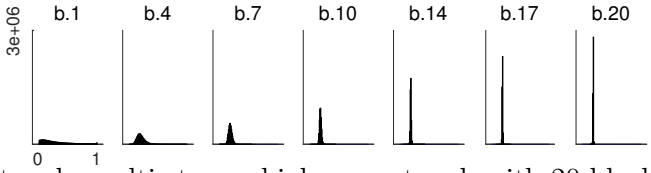
(c) MGC sub-network, multi-stream highway network with 20 blocks (depth 40)



(d) F0 sub-network, multi-stream highway network with 7 blocks (depth 14)



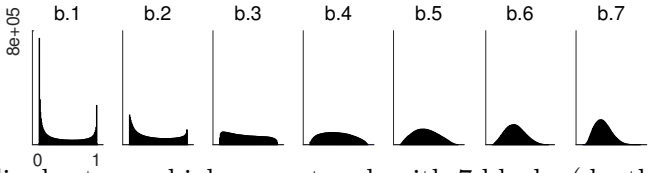
(e) F0 sub-network, multi-stream highway network with 10 blocks (depth 20)



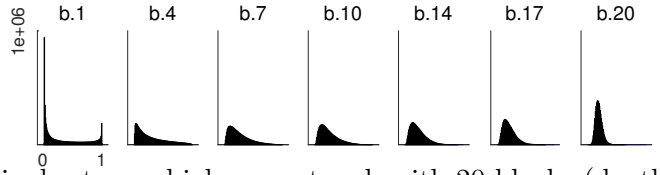
(f) F0 sub-network, multi-stream highway network with 20 blocks (depth 40)

Figure 4.5: Histogram of highway gating vectors for MGC and F0 sub-networks in multi-stream highway networks. Note that **b.1** denotes the first block near the network's input. These histograms are counted on test set for phoneme /a/.

the MGC and F0 sub-networks indicate the difference of the hidden features for the MGC and F0. Specifically, the MGC sub-network extracted hidden features



(a) Single-stream highway network with 7 blocks (depth 14)



(b) Single-stream highway network with 20 blocks (depth 40)

Figure 4.6: Histogram of highway gating vectors for single-stream highway networks. Note that **b.1** denotes the first block near the network's input. These histograms are counted on test set for phoneme /a/.

from a few dimensions of the linguistic features by non-linear transformations. However, the F0 sub-network used fewer non-linear transformations.

Another interesting difference is the number of highway blocks used to derive the hidden features. The F0 sub-network had a few inactive highway blocks near the output side, which indicates that the hidden features cannot be further refined. This observation is consistent with the results in Figure 4.3, where adding more highway blocks (increasing the depth) did not improve the F0 RMSE and CORR of the multi-stream network. On the other hand, the highway blocks in the MGC sub-network were quite active. If the MGC sub-network has more highway blocks, it may further improve the MGC modeling performance, which is consistent with the MGC RMSE curve in Figure 4.3.

Note that, when the network depth was increased to 40, the last block (b.20) in the MGC sub-network had a sharp histogram as Figure 4.5 (c) shows. This suggests that 40 highway blocks may be more than enough for MGC modeling on this English corpus.

On single-stream highway networks

Next, we plotted the histograms for the single-stream highway networks of depth 14 and 40 in Figure 4.6. Note that in this case we cannot differentiate the histograms for the MGC and F0. Interestingly, the histograms resemble those in the MGC

sub-network of the multi-stream one (Figure 4.5 (a) and (c)). This result suggests that the single-stream highway network mainly focused on MGC modeling. It may explain why the single-stream network's performance on the F0 was worse.

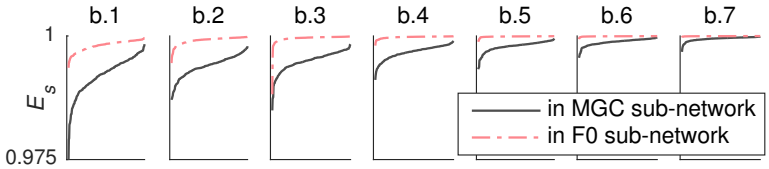
A single-stream network could improve F0 modeling if the network was deeper. One reason may be that a larger network has additional capacity to model the F0. This assumption is somewhat supported by the histogram in Figure 4.6 (b). Notice how the last highway block did not produce a very sharp histogram if we compare it with the b.20 in Figure 4.5 (c). This indicates that this single-stream network fully used the network capacity to model the F0 and spectral features.

4.4.3 Analyzing sensitivity to input features

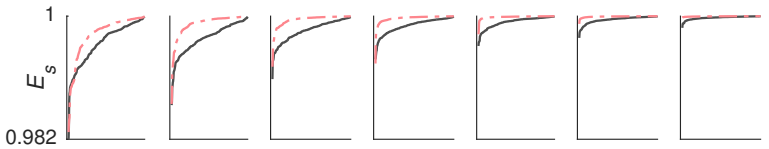
Histograms extracted from multi-stream highway networks suggest that the hidden features for F0 may be different from those for MGC. Since the hidden features were derived from the same input linguistic features vectors, the MGC and F0 sub-networks may focus on different linguistic features.

We used the sensitivity measure defined in Section 4.3.2 and analyzed the multi-stream network with depth 14 Figure 4.3. Linguistic feature classes are listed in Appendix A.3. For each feature class $\boldsymbol{S}$, we calculated a sensitivity score $E^{(l)}_{\boldsymbol{S},k}$ for the k-th neuron in the l-th highway block of a sub-network. Then, we sorted the vector $E^{(l)}_{\boldsymbol{S}} = [E^{(l)}_{\boldsymbol{S},1}, \cdots, E^{(l)}_{\boldsymbol{S},K}]$ based on the value of $E^{(l)}_{\boldsymbol{S},k}$ for the l-th block. We further calculated the average score $\bar{E}_{\boldsymbol{S}} = \frac{1}{KL}\sum_k \sum_l E^{(l)}_{\boldsymbol{S},k}$ as the overall sensitivity of a sub-network to the linguistic feature class $\boldsymbol{S}$.

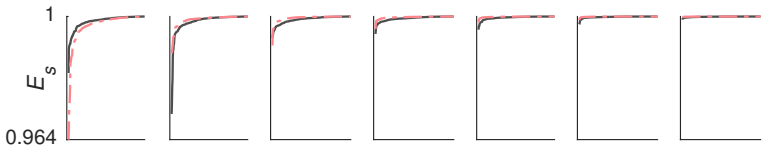
Figure 4.7 plots the vector $E^{(l)}_{\boldsymbol{S}}$ for the 7 highway blocks in MGC and F0 sub-networks and for different linguistic features classes $\boldsymbol{S}$. Table 4.2 lists the overall sensitivities of the MGC and F0 sub-networks to some linguistic feature classes. First, Figure 4.7 shows that the sensitivity of a sub-network varied for different linguistic feature classes. For example, the MGC sub-network had a lower score (high sensitivity) to the phoneme identity than to the accent type. This is expected because the MGC correlates mainly with the segmental features. Meanwhile, the F0 sub-network had a higher sensitivity to the features related to the accent type. Since the MGC and F0 sub-networks acquired different sensitivity scores, we can infer that the two sub-networks extracted different information from



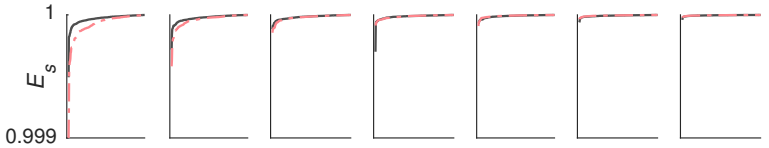
(a) Feature class $\boldsymbol{S}$: central phoneme identity



(b) Feature class $\boldsymbol{S}$: position of the phoneme in the current syllable



(c) Feature class $\boldsymbol{S}$: accent type of the next syllable



(d) Feature class $\boldsymbol{S}$: ToBI boundary tone

Figure 4.7: Sensitivity measure $E^{(l)}_{\boldsymbol{S},k}$ for the MGC (solid line) and F0 (dash line) sub-networks of multi-stream highway networks with 7 blocks (depth 14), $l \in \{1, \cdots, 7\}$. **b.1** denotes the first block near the network's input. The x-axis k, which ranges from 1 to 256, is sorted based on the value of $E_{\boldsymbol{S},k}$. A low value of $E_{\boldsymbol{S},k}$ denotes higher sensitivity.

the input linguistic vectors and thus derived different hidden features.

Figure 4.7 also indicates that the F0 sub-network was generally less sensitive to the linguistic features than the MGC sub-network. In other words, the F0 sub-network found the linguistic features uninformative for F0 modeling. This may be one reason of the inactive highway blocks in the F0 sub-network. Such a result may be reasonable because all the linguistic features were automatically annotated by the text-analyzer and may be too noisy for F0 modeling. Particularly, the ToBI

Table 4.2: $\bar{E}_S$ to each linguistic feature class S (English corpus).

MGC sub-network

	Feature class S	$\bar{E}_S$
Top five entries	Current phoneme identity	0.9923
	Position of current phoneme in syllable	0.9961
	Position of current phoneme in syllable (backward)	0.9967
	Number of preceding lexically stressed syllables in phrase	0.9974
	Number of following lexically stressed syllables in phrase	0.9975
Bottom five entries	Position of phrase in utterance (backward)	0.9999
	Number of words in phrase	0.9999
	Number of phrases in utterance	0.9999
	Number of syllables in previous phrase	1.0000
	ToBI boundary tone	1.0000

F0 sub-network

	Feature class S	$\bar{E}_S$
Top five entries	Position of phoneme in syllable (forward)	0.9973
	Is the next syllable bearing an English pitch-accent	0.9974
	Is the previous syllable bearing an English pitch-accent	0.9980
	Position of current syllable in the word	0.9980
	Current phoneme identity	0.9981
Bottom five entries	Number of words in previous phrase	0.9999
	Number of words in next phrase	0.9999
	ToBI boundary tone	0.9999
	Number of syllables in next phrase	0.9999
	Number of syllables in previous phrase	1.0000

boundary tone annotated by the text-analyzer may be useless and thus ranked as one of the least sensitive features for the F0 sub-network as Table 4.2 shows. On the other hand, the MGC sub-network showed high sensitivity at least to the phoneme identity in all the highway blocks. This may explain the binomial shape of the histogram in the MGC sub-network.

In general, the results based on the sensitivity measure are consistent with those from the histogram analysis: first, the MGC and F0 sub-networks rely on different input features; second, they may use a different number of hidden layers and extract different hidden features.

4.5 Results and analyses on the Japanese corpus

The experiment on the English corpus has shown the different behaviors of the F0 and MGC sub-networks. However, this result may be corpus or language dependent. Therefore, we conducted a similar experiment on a Japanese corpus. Details of this corpus can be found in the Appendix A.4.

Similar to the experiments on the English data, we mainly compared the multi- and single stream highway networks with the single-stream FNN as a reference. To simplify the experiment design, we directly adjusted the width of a multi-stream highway network so that the network size was comparable to that of a single-stream network with the same depth. The networks were trained using the same recipe as that of the experiment on the English corpus.

The objective results are plotted in Figure 4.8. Similar to those on the English corpus, the results indicate that a single-stream highway network prioritized the MGC modeling. Especially when the network depth was less than 8, the single-stream highway network's performance on F0 was much worse than that of the multi-stream one, even though they achieved similar error scores on MGC modeling. Only when the depth of the single-stream highway network was beyond 8 did the single-stream highway network achieve a similar F0 modeling performance to the multi-stream highway network.

The histogram analysis on the highway gating vectors shows similar patterns to those found on the English corpus. Typically in the multi-stream highway networks, the F0 sub-networks had more inactive blocks than the MGC sub-networks. What's more, F0 and MGC sub-networks acquired different sensitivity scores to the linguistic features as Table 4.3 listed.

However, one interesting observation from Table 4.3 is on the sensitivity score to the linguistic features related to the pitch contour. In the case of English[2], the ToBI boundary tone was ranked as the least sensitive feature, the Japanese accent-type phrase, which specifies the location of pitch fall, was among the most sensitive ones. This difference is reasonable in a linguistic sense. The Japanese accent type is based on the accent type of individual words and can be somewhat

[2]For English systems, the question 'is a syllable bearing an English pitch-accent' does not specify the accent type (high, low, or the combination).

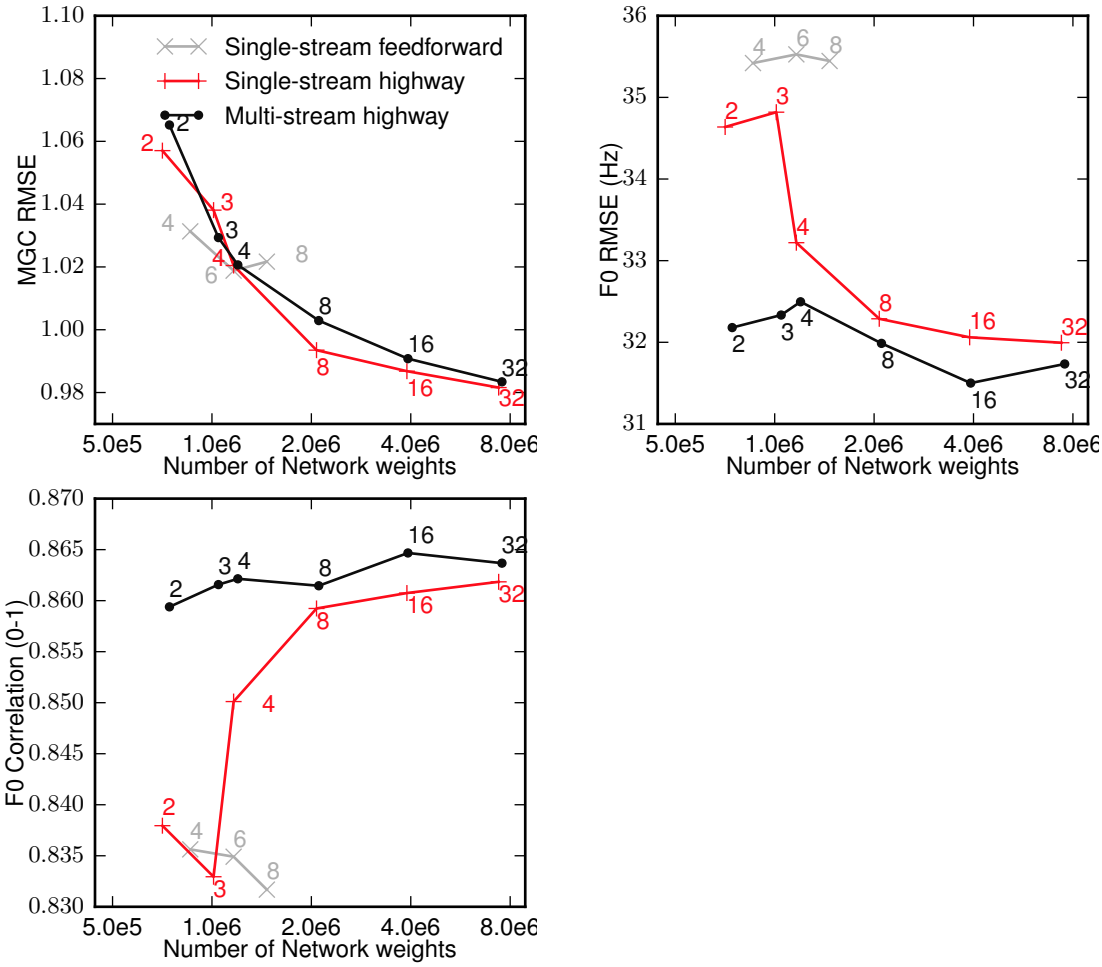


Figure 4.8: Objective evaluation on single-stream and multi-stream networks. The number with each point on the error curve denotes the network depth. Note that, for the single-stream feedforward networks with depth larger than 8, training failed on the Japanese corpus.

predicted using a lexicon. In addition, the F0 contour is mainly depicted by the accent type, which differentiates words with the same phoneme sequences. In contrast, the English boundary tone is not specified by the lexicon. Therefore, the accent type and the F0 contour may be more correlated in the Japanese corpus than that between the ToBI boundary tone and the F0 contour in the English corpus.

Table 4.3: $\bar{E}_S$ to each linguistic feature class S (Japanese corpus).

MGC sub-network

	Feature class S	$\bar{E}_S$
Top five entries	Current phoneme identity	0.977
	Inflected forms of the current word	0.979
	Position of mora in accent phrase (forward)	0.979
	Number of accent phrases in breath group	0.980
	Position of accent phrase in breath group (backward)	0.980
Bottom five entries	Position of breath group (backward)	0.995
	Position of accent phrase in breath group (forward)	0.996
	Number of accent phrases in this utterance	0.997
	Number of breath groups in this utterance	0.999
	Number of moras in this utterance	0.999

F0 sub-network

	Feature class S	$\bar{E}_S$
Top five entries	Conjugation type of the next word	0.979
	Position of mora in accent phrase (forward)	0.983
	Part-of-speech of the next word	0.984
	Accent type (Japanese pitch-accent) of next accent phrase	0.986
	Accent type (Japanese pitch-accent) of accent phrase	0.986
Bottom five entries	Position of breath group (backward)	0.996
	Position of breath group (forward)	0.997
	Number of accent phrases in this utterance	0.997
	Number of breath groups in this utterance	0.999
	Number of moras in this utterance	1.000

4.6 Summary

In this chapter, we explored **Issue 1: is it appropriate in the neural SPSS to jointly model the F0 with other acoustic features?** The answer suggested by this chapter is negative.

To investigate this issue, we introduced the multi- and single-stream highway networks that were easy to analyze and train. We further defined a histogram-based analysis tool and a sensitivity measure to investigate the behavior of the highway network. The experimental results suggest rethinking the common strategy of joint modeling. First, a single-stream network may prioritize the spectral features over

the F0. The network must be sufficiently large in order to assign model capacity to the F0. Furthermore, the analyses on the histogram and sensitivity indicate that a network may focus on different linguistic features and derive different hidden features for the F0 and spectral features. It casts doubt on the argument that the neural SPSS could leverage the framework of multi-task learning through joint modeling of the F0 and spectral features. A similar result has already been observed in the HMM-based SPSS [128].

These results suggest that the common strategy may not be the best idea at least for neural F0 modeling. Since the F0 and MGC rely on different hidden representations, a single-stream network must be large enough to encode the hidden features for both F0 and other acoustic features. Since we focus on F0 modeling in this thesis, we have to be careful about the F0 modeling results if we model the F0 with spectral features. To avoid factors that possibly affect the F0 modeling performance, we separate the F0 model from the acoustic models for other acoustic features in the following chapters.

5

Shallow Autoregressive Neural F0 model

As Section 3.6 mentioned, a conventional neural F0 model such as an RNN assumes a conditional independence when it models the F0 contour. However, this assumption may be inappropriate because the F0 values of adjacent frames are usually similar. This inappropriate assumption of temporal independence is **Issue 2** addressed in this thesis.

To revise the model assumption, this chapter proposes the shallow autoregressive (SAR) model, a new model that describes the autoregressive (AR) temporal dependency for F0 contours. This chapter first explores the limitation caused by the conditional independence assumption in Section 5.1, using a random-sampling-based generation method. It then introduces the concept of AR dependency and defines the SAR in Section 5.2. Interestingly, an SAR can be interpreted from two perspectives: Section 5.3 interprets the SAR as the combination of baseline RMDN and time-invariant filter pairs; Section 5.4 interprets the SAR as a baseline RMDN augmented with a special normalizing flow. The latter interpretation also brings in the idea to extend SAR to a more generalized model.

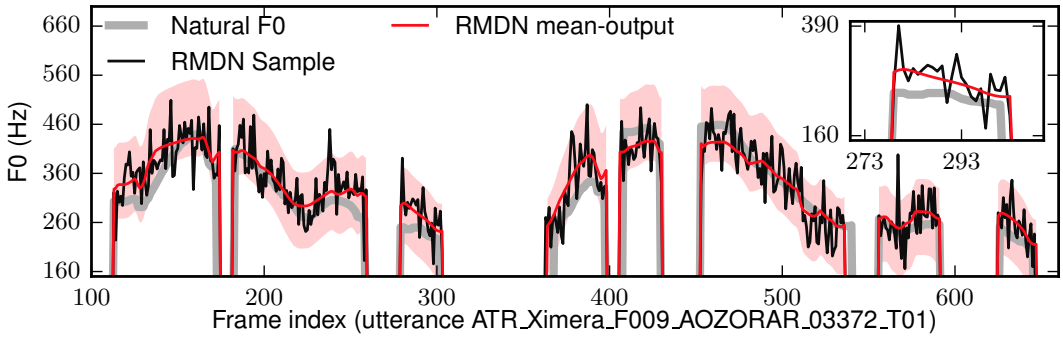


Figure 5.1: Generated F0 from RMDN using mean-based generation and random sampling on Japanese corpus. The red area shows the range of standard deviation.

5.1 Conditional independence in baseline models

Investigation using random sampling

According to Chapter 3, a baseline neural F0 model such as an RNN or RMDN assumes that the distribution of an F0 contour $\boldsymbol{o}_{1:T}$ conditioned on the linguistic features $\boldsymbol{x}_{1:T}$ can be written as $p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{x}_{1:T};\boldsymbol{\Theta})$, where $\boldsymbol{\Theta}$ denotes the network weights. Therefore, it assumes that $p(\boldsymbol{o}_{t_1}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta})$ and $p(\boldsymbol{o}_{t_2}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta})$ are independent conditional distributions, $\forall t_1 \neq t_2$.

Such an assumption may be inappropriate for modeling natural F0 contours, which are smooth contours. To verify this point, we can draw a sample F0 contour $\widehat{\boldsymbol{o}}_{1:T}$ from the distribution $p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta})$ inferred by the model. If the baseline model depicts the generative process of F0 contours authentically, the sampled contour $\widehat{\boldsymbol{o}}_{1:T}$ should resemble natural F0 contours.

Here, we use a well-trained RMDN for F0 modeling as an example (`RMDN` trained in Section 5.5.4). Given the input $\boldsymbol{x}_{1:T}$ from the test set, we can let the RMDN do the forward computation and infer the distribution $p(\boldsymbol{o}_t|\boldsymbol{x}_{1:T})$, $t \in \{1, \cdots, T\}$. Since the model assumes the conditional independent across time, we can draw the sample $\widehat{\boldsymbol{o}}_t$ from $p(\boldsymbol{o}_t|\boldsymbol{x}_{1:T};\boldsymbol{\Theta})$ for $t \in \{1, \cdots, T\}$ simultaneously. Figure 5.1 plots one sampled F0 contour $\widehat{\boldsymbol{o}}_{1:T}$ and the natural $\boldsymbol{o}_{1:T}$. Obviously, the sampled sequence is very noisy. In Figure 5.1, we further plot the mean trajectory and the standard deviation range specified by the mixture components with the largest weight at each time step. Although the mean trajectory is smooth

and resembles the natural contour, the standard deviation range is quite large. Consequently, even samples drawn from two adjacent distributions may be quite different from each other, which results in the noisy $\widehat{\boldsymbol{o}}_{1:T}$.

Note that if the model uses the mean-based generation method, it will generate the smooth F0 mean trajectory. However, the smoothness of the mean trajectory does not mean the appropriateness of the model [129]. It is by random sampling that the capability of the generative model can be assessed [130].

Towards AR dependency

Why did the previous model infer a large standard deviation? One intuitive explanation is that the model was trained to learn the ambiguous mapping between input and target training data, particularly for the F0 modeling task where similar input data sequences may correspond to quite different F0 contours. However, a model could reduce the ambiguity if it takes into consideration the intrinsic pattern of the target data sequences. For example, F0 contours are smooth because they are produced under the physiological constraints of human organs. If the F0 value at time t is known, the F0 at time $t+1$ is likely to have a similar value. Similarly, if an F0 model observed $\boldsymbol{o}_t$, it should be more confident in predicting $\boldsymbol{o}_{t+1}$ than the case when it ignores the temporal dependency.

A better model should take into account the temporal correlation in the target feature sequence $\boldsymbol{o}_{1:T}$. One idea is to re-define the baseline model as a Markov random field where $\boldsymbol{o}_{1:T}$ form a single clique [131]. This re-defined model can be written as $p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T}) = \phi_c(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T, \boldsymbol{x}_{1:T})/Z$, where $\phi_c(\cdot)$ is a potential function and $Z = \int_{\boldsymbol{o}_{1:T}} \phi_c(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T, \boldsymbol{x}_{1:T}) d\boldsymbol{o}_{1:T}$ is a normalization constant. The model can cover the dependency between any pair of $\boldsymbol{o}_{t_1}$ and $\boldsymbol{o}_{t_2}$ if the potential function $\phi_c(\cdot)$ is sufficiently complex. Unfortunately, training such a model may be intractable because of the complicated Z. In practice, we may only define $\phi_c(\cdot)$ to cover the temporal dependency within a local time window in a similar manner to the trajectory model [132, 133, 134]. However, it is still difficult to train and use the model because it requires sophisticated training and generation algorithms.

The aforementioned model is impractical because it relies on un-directed temporal correlation among random variables, e.g., $\boldsymbol{o}_{t_1}$ can affect and be affected

by $\boldsymbol{o}_{t_2}$. Another idea is to model directed or AR dependency [135], i.e., the dependency of $\boldsymbol{o}_t$ on $\boldsymbol{o}_{1:t-1}$. Although the dependency may be less general, it may be more compatible with sequential data that are generated by a causal system. What's more, a model with the AR dependency can be factorized as $p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1},\boldsymbol{x}_{1:T};\boldsymbol{\Theta})$, which makes the model less complex than undirected graphic models. As we will see in the following sections, such an AR model can be trained and used in a similar manner to a baseline model.

5.2 Definition of shallow AR model

Definition

The SAR can be defined on the basis of the conventional RMDN and the AR dependency. Let's use $\boldsymbol{o}_{1:T} \in \mathbb{R}^{T\times D}$ and $\boldsymbol{x}_{1:T}$ to denote the target and the input data sequence, respectively. The SAR can be written as

$$\begin{aligned} p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T};\boldsymbol{\Theta},\boldsymbol{\Psi}) &= \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{t-K:t-1},\boldsymbol{x}_{1:T};\boldsymbol{\Theta},\boldsymbol{\Psi}) \\ &= \prod_{t=1}^{T}\sum_{m=1}^{M} \omega_t^m \mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_t^m + f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{t-K:t-1}),\boldsymbol{\Sigma}_t^m) \end{aligned}, \tag{5.1}$$

where $\boldsymbol{\Theta}$ is the set of the neural network weights, and $\boldsymbol{\Psi}$ is the parameter set of the function $f_{\boldsymbol{\Psi}}(\cdot)$. Let us define $\boldsymbol{\mathcal{M}}_t \triangleq \{\omega_t^1,\cdots,\omega_t^M,\ \boldsymbol{\mu}_t^1,\cdots,\boldsymbol{\mu}_t^M,\ \boldsymbol{\Sigma}_t^1,\cdots,\boldsymbol{\Sigma}_t^M\}$ as the parameter set of the GMM with M mixture components. The value of $\boldsymbol{\mathcal{M}}_t$ is calculated by the internal RNN as $\boldsymbol{\mathcal{M}}_t = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T},t)$ in the same manner as the baseline RMDN. In contrast to the baseline RMDN, however, the SAR defines the distribution of $\boldsymbol{o}_t$ as a conditional distribution that depends on the previous K observations $\boldsymbol{o}_{t-K:t-1}$. Specifically, the SAR introduces a function $f_{\boldsymbol{\Psi}}(\cdot)$ to summarize $\boldsymbol{o}_{t-K:t-1}$ and adjust the mean of each GMM mixture at time t. Note that K is a hyper-parameter of SAR.

The parametric form of $f_{\boldsymbol{\Psi}}(\cdot)$ can be defined flexibly. Here, we use a simple form

$$f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{t-K:t-1}) = \sum_{k=1}^{K} \boldsymbol{a}_k \odot \boldsymbol{o}_{t-k} + \boldsymbol{b}, \tag{5.2}$$

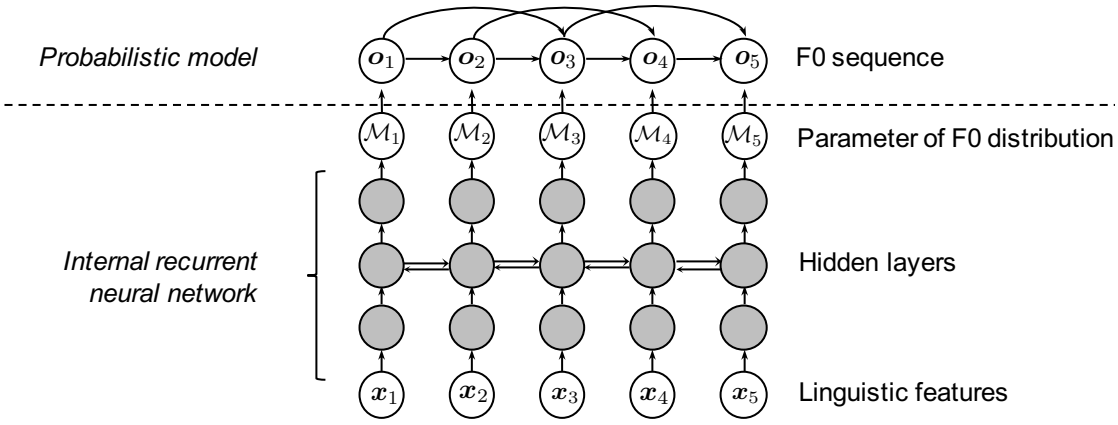


Figure 5.2: Example SAR with $K = 2$

where $\mathbf{\Psi} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_K, \boldsymbol{b}\}$ and $\odot$ is the element-wise product. We refer to the model defined based on Equation (5.1) and (5.2) as SAR because it only models the local AR dependency using a linear function $f_{\mathbf{\Psi}}(\cdot)$. An example SAR with $K = 2$ is plotted in Figure 5.2.

An SAR can be trained in a similar manner to a baseline RMDN, where the parameter set $\mathbf{\Theta}$ and $\mathbf{\Psi}$ can be estimated using the maximum-likelihood criterion and the gradient descent method. It only requires additional cost to calculate $f_{\mathbf{\Psi}}(\cdot)$ and the gradients of $\mathbf{\Psi}$. In the generation time, the mean-based or sampling-based generation method can be applied sequentially from $t = 1$ to $t = T$.

Comparison with extended RMDN

The SAR plotted in Figure 5.2 looks similar to a baseline RMDN with a recurrent output layer [85]. However, the AR dependency link in the SAR is different from the recurrent link in an RMDN. Let us explain this difference using two toy models in Figure 5.3. To simplify the explanation, we assume that all the layers use a linear activation function and set the bias to zero. What's more, we assume the distribution for o_t as a Gaussian distribution with a unit variance. The task is to model the target data sequence $\boldsymbol{o}_{1:2} = [o_1, o_2]$, where $o_1, o_2 \in \mathbb{R}$.

For the toy extended RMDN, it is easy to show that $\mathcal{M}_1 \triangleq \mu_1 = \boldsymbol{w}_m^\top \boldsymbol{h}_1$ and $\mathcal{M}_2 \triangleq \mu_2 = \boldsymbol{w}_m^\top \boldsymbol{h}_2 + w_\mu \mu_1$. Let $\tilde{\mu}_2 \triangleq \boldsymbol{w}_m^\top \boldsymbol{h}_2$, then the conditional distribution of

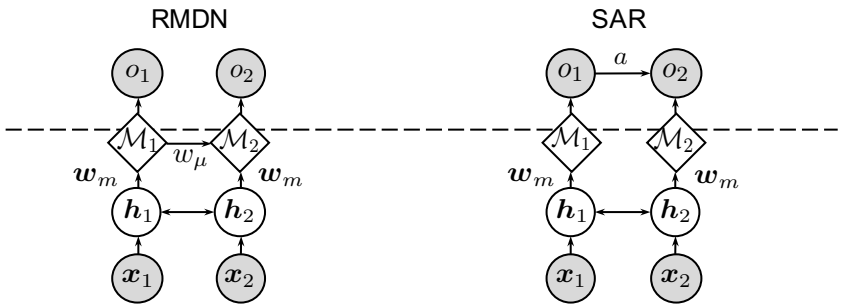


Figure 5.3: Toy SAR and RMDN

$\boldsymbol{o}_{1:2}$ can be written as

$$\begin{aligned} p(o_{1:2}|\boldsymbol{x}_{1:2}) =& \mathcal{N}(o_1;\mu_1,1)\mathcal{N}(o_2;\tilde{\mu}_2+w_\mu\mu_1,1) \\ =& \frac{1}{2\pi}\exp(-\frac{(o_1-\mu_1)^2}{2}-\frac{(o_2-\tilde{\mu}_2-w_\mu\mu_1)^2}{2}) \\ =& \frac{1}{2\pi}\exp(-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}-\boldsymbol{\mu})), \end{aligned} \tag{5.3}$$

where $\boldsymbol{o}=[o_1,o_2]^\top$, $\boldsymbol{\mu}=[\mu_1,\tilde{\mu}_2+w_\mu\mu_1]^\top$, and $\boldsymbol{\Sigma}=\begin{bmatrix}1&0\\0&1\end{bmatrix}$. Obviously, although the RMDN uses a recurrent link in the output layer, the covariance matrix $\boldsymbol{\Sigma}$ is diagonal, which suggests that o_1 and o_2 are treated as being independent.

The toy SAR computes $\mathcal{M}_1 \triangleq \mu_1 = \boldsymbol{w}_m^\top\boldsymbol{h}_1$ and $\mathcal{M}_2 \triangleq \mu_2 = \boldsymbol{w}_m^\top\boldsymbol{h}_2$. Accordingly, the distribution calculated with the model can be written as

$$\begin{aligned} p(o_{1:2}|\boldsymbol{x}_{1:2}) =& \mathcal{N}(o_1;\mu_1,1)\mathcal{N}(o_2;\mu_2+ao_1,1) \\ =& \frac{1}{2\pi}\exp(-\frac{(o_1-\mu_1)^2}{2}-\frac{(o_2-\mu_2-ao_1)^2}{2}) \\ =& \frac{1}{2\pi}\exp(-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}-\boldsymbol{\mu})), \end{aligned} \tag{5.4}$$

where $\boldsymbol{o}=[o_1,o_2]^\top$, $\boldsymbol{\mu}=[\mu_1,\mu_2+a\mu_1]^\top$, $\boldsymbol{\Sigma}=\begin{bmatrix}1&a\\a&1+a^2\end{bmatrix}$, and a is the trainable AR parameter. As long as $a\neq 0$, the covariance matrix $\boldsymbol{\Sigma}$ becomes a full matrix, which means that the correlation between o_1 and o_2 is not ignored by the model.

In general, the AR dependency link in the SAR is defined among the random variables $\boldsymbol{o}_{1:T}$ while the recurrent link in RMDN is among outputs of the neural layer. This difference affects the power of the model.

5.3 SAR as neural network plus digital filters

The SAR uses a linear function to capture the AR dependency. Although it is simple in definition, it allows us to interpret the model from different perspectives and understand the potential issues of the SAR.

5.3.1 Interpretation based on signal and filter

The first interpretation is from the perspective of digital filter and signal. Suppose that $\boldsymbol{o}_t \in \mathbb{R}^D$, and the SAR uses GMMs with a single mixture component and a diagonal covariance matrix. Then, the conditional distribution of $\boldsymbol{o}_t$ can be written as

$$p(\boldsymbol{o}_t|\boldsymbol{o}_{t-K:t-1}, \boldsymbol{x}_{1:T}) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_{t,d}^2}} \exp\left[-\frac{(o_{t,d} - \sum_{k=1}^{K} a_{k,d} o_{t-k,d} - \mu_{t,d} - b_d)^2}{2\sigma_{t,d}^2}\right], \tag{5.5}$$

where $o_{t,d}, \mu_{t,d}, a_{k,d}$, and b_d are the d-th dimensions of $\boldsymbol{o}_t, \boldsymbol{\mu}_t, \boldsymbol{a}_k$, and $\boldsymbol{b}$, respectively, and $\sigma_{t,d}$ is the d-th diagonal element of the diagonal matrix $\boldsymbol{\Sigma}_t$.

Let us define a new random variable as $c_{t,d} = o_{t,d} - \sum_{k=1}^{K} a_{k,d} o_{t-k,d}$. Then, we can find that the vector $\boldsymbol{c}_{1:T,d} = [c_{1,d}, \cdots, c_{T,d}]^\top$ and another vector $\boldsymbol{o}_{1:T,d} = [o_{1,d}, \cdots, o_{T,d}]^\top$ are related by a linear transformation

$$\boldsymbol{c}_{1:T,d} = \boldsymbol{A}^{(d)} \boldsymbol{o}_{1:T,d}, \tag{5.6}$$

where

$$\boldsymbol{A}^{(d)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ -a_{1,d} & 1 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ -a_{2,d} & -a_{1,d} & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ -a_{K,d} & \cdots & -a_{2,d} & -a_{1,d} & 1 & 0 & 0 & \cdots & 0 \\ 0 & -a_{K,d} & \cdots & -a_{2,d} & -a_{1,d} & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -a_{K,d} & \cdots & -a_{2,d} & -a_{1,d} & 1 & 0 \\ 0 & 0 & \cdots & 0 & -a_{K,d} & \cdots & -a_{2,d} & -a_{1,d} & 1 \end{bmatrix}. \tag{5.7}$$

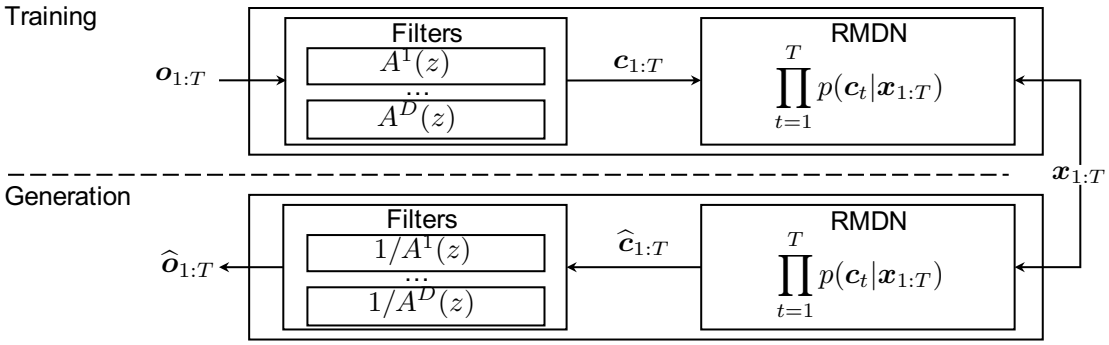


Figure 5.4: Interpretation of SAR based on filters and signals

Interestingly, Equation (5.6) is also a filtering process in which the input signal $\boldsymbol{o}_{1:T,d}$ of length T is converted into the output signal $\boldsymbol{c}_{1:T,d}$ by using a finite impulse response (FIR) filter. This FIR filter can be written in the z-domain as

$$A^d(z) = 1 - \sum_{k=1}^{K} a_{k,d} z^{-k}, \tag{5.8}$$

where $\{a_{1,d}, \cdots, a_{K,d}\}$ are the filter coefficients. Because the triangular matrix $\boldsymbol{A}^{(d)}$ is invertible, an 'inverse' filtering process can be defined as

$$\boldsymbol{o}_{1:T,d} = (\boldsymbol{A}^{(d)})^{-1} \boldsymbol{c}_{1:T,d} = \boldsymbol{H}^{(d)} \boldsymbol{c}_{1:T,d}, \tag{5.9}$$

and the 'inverse' filter written in the z-domain is

$$H^d(z) = \frac{1}{A^d(z)} = \frac{1}{1 - \sum_{k=1}^{K} a_{k,d} z^{-k}}. \tag{5.10}$$

The above filtering process is defined for the slice of the d-th dimension of $\boldsymbol{o}_{1:T}$. Generally, the filter pair $\{H^d(z), A^d(z)\}$ and the signal $\boldsymbol{c}_{1:T,d}$ can be defined for each dimension $d \in [1, D]$. If we replace $o_{t,d} - \sum_{k=1}^{K} a_{k,d} o_{t-k,d}$ in Equation (5.5) with $c_{t,d}$, we can re-write the distribution of $\boldsymbol{o}_t$ as

$$p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{t-K:t-1}, \boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{c}_t|\boldsymbol{x}_{1:T}) = p(\boldsymbol{c}_{1:T}|\boldsymbol{x}_{1:T}), \tag{5.11}$$

where $\boldsymbol{c}_{1:T}$ and $\boldsymbol{o}_{1:T}$ are related by the filters $\{A^1(z), \cdots, A^D(z)\}$. In other words,

as Figure 5.4 shows, the SAR can be interpreted as a combination of digital filters and an RMDN:

- in the training stage, the filters transform $\boldsymbol{o}_{1:T}$ into $\boldsymbol{c}_{1:T}$. The filter part ($\boldsymbol{\Psi}$ of $f_{\boldsymbol{\Psi}}$) and the RMDN part ($\boldsymbol{\Phi}$ of the internal RNN) can be trained jointly or iteratively by maximizing the likelihood of the RMDN over $\boldsymbol{c}_{1:T}$;
- in the generation stage, the RMDN generates $\widehat{\boldsymbol{c}}_{1:T}$, and the inverse filters de-transform $\widehat{\boldsymbol{c}}_{1:T}$ into $\widehat{\boldsymbol{o}}_{1:T}$.

Note that the combination of filters and RMDN is just for interpretation, and no actual filter is required in the implementation of the SAR. For this reason, we call $\{H^d(z), A^d(z)\}$ the **virtual filter** of SAR. The implementation of the SAR is still based on the original definition in Equation (5.1) and (5.2).

5.3.2 Stability of SAR

The above interpretation reveals one potential issue on the model stability. As shown in Equation (5.10), since the filters $\{H^1(z), \cdots, H^D(z)\}$ are infinite impulse response (IIR) filters, they must remain stable after the filter coefficients $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_K\}$ are estimated in model training. Otherwise, the output signal $\widehat{\boldsymbol{o}}_{1:T,d}$ from an unstable IIR filter $H^d(z)$ may acquire infinite values.

A causal stable IIR filter must have all the poles inside the unit-circle in the z-domain [136]. However, the gradient-descent training method directly updates the filter coefficients $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_K\}$ and does not constrain the location of poles. To ensure the filter stability, we propose three methods. While the first two **sufficient but unnecessary** methods directly parameterize the filters' poles, the last method uses log area ratio [137] to parameterize the filter coefficients.

Note that the proposed methods just parameterize the coefficients $\{a_{1,d}, \cdots, a_{K,d}\}$, $\forall b \in [1, D]$, using new parameters. This whole model is still trained using the back-propagation and gradient-descent method.

Method 1: IIR filters with real-valued poles

We now explain the first method for the d-th filter, and we will drop the index d to simplify the notation. The idea is to constrain the poles of an IIR filter to be

real-valued numbers between -1 and 1. Let us re-write an IIR-filter as a cascade of 1st order filters and apply the $\tanh(\cdot)$ function to constrain the value of the pole in each 1st order filter as $\alpha = \tanh(\tilde{\alpha})$. Accordingly we can get:

$$H(z) = \frac{1}{1 - \sum_{k=1}^{K} a_k z^{-k}} = \prod_{k=1}^{K} \frac{1}{1 - \alpha_k z^{-1}} = \prod_{k=1}^{K} \frac{1}{1 - \tanh(\tilde{\alpha}_k) z^{-1}}. \tag{5.12}$$

The real-valued poles $\{\alpha_1, \cdots, \alpha_K\}$ usually have different values from the filter coefficients $\{a_1, \cdots, a_K\}$, but there is a deterministic mapping from $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ to $\{a_1, \cdots, a_K\}$. In a word, the proposed method just parameterizes the original $\{a_1, \cdots, a_K\}$ using $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$. The parameter set $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ can be estimated using the gradient-descent method jointly with the neural network weights. Whatever the value of the estimated $\tilde{\alpha}_k$, the pole α_k is constrained by the $\tanh(\cdot)$ function to be within $(-1, 1)$, and the filter coefficients $\{a_1, \cdots, a_K\}$ calculated from $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ always form a stable IIR filter.

As implementation, we need two additional procedures to the vanilla SAR: one to map $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ to $\{a_1, \cdots, a_K\}$ and the other to map the gradients $\{\frac{\partial E}{\partial a_1}, \cdots, \frac{\partial E}{\partial a_K}\}$ to $\{\frac{\partial E}{\partial \tilde{\alpha}_1}, \cdots, \frac{\partial E}{\partial \tilde{\alpha}_K}\}$, where E is the negative log-likelihood. The first procedure is used when we need to calculate the $\{a_1, \cdots, a_K\}$ in the training and inference stages. The second procedure is required when updating $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ through the gradient-descent Method in the training stage.

To convert $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ into $\{a_1, \cdots, a_K\}$, we can re-write Equation (5.12) as:

$$1 - \sum_{k=1}^{K} a_k z^{-k} = \prod_{k=1}^{K} \left[1 - \tanh(\tilde{\alpha}_k) z^{-1}\right]. \tag{5.13}$$

We can then just unfold the right hand side of Equation (5.13) into a polynomial and set a_k equal to the coefficient of z^{-k}. This can be done by using Algorithm 1, the time and space complexity of which is only $\mathcal{O}(K)$. Note that the loop in lines 9-10 is only for explanation and can be parallelized.

To compute the gradients $\{\frac{\partial E}{\partial \tilde{\alpha}_1}, \cdots, \frac{\partial E}{\partial \tilde{\alpha}_K}\}$, we need to use the chain rule. For example, the gradient of $\tilde{\alpha}_j$ can be computed as

$$\frac{\partial E}{\partial \tilde{\alpha}_j} = \sum_{k=1}^{K} \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial \tilde{\alpha}_j}. \tag{5.14}$$

Algorithm 1: Forward computation for Method 1: converting $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ to $\{a_1, \cdots, a_K\}$ for filter $A(z) = 1 - \sum_{k=1}^{K} a_k z^{-1}$ and $H(z) = 1/A(z)$. This routine is similar to the 'poly' function of Numpy [138].

Input: $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}, K$

$[\alpha_1, \alpha_2, \cdots, \alpha_K] \leftarrow [\tanh(\tilde{\alpha}_1), \tanh(\tilde{\alpha}_1), \cdots, \tanh(\tilde{\alpha}_K)]$;

$a_1 \leftarrow \alpha_1$;

if $K = 1$ **then**

  Return $\{a_1\}$;

for $k \leftarrow 2$ **to** K **do**

  $\boldsymbol{a}^{(k)} \leftarrow [1, -a_1, \cdots, -a_{k-1}, 0]$; `/* a^(k) has k+1 dimensions */`

  $\boldsymbol{b}^{(k)} \leftarrow [0, 1, -a_1, \cdots, -a_{k-1}] * -\alpha_k$; `/* b^(k) has k+1 dimensions */`

  $\boldsymbol{a}^{(k)} \leftarrow \boldsymbol{a}^{(k)} + \boldsymbol{b}^{(k)}$;

  for $p \leftarrow 1$ **to** k **do**

    $a_p \leftarrow \boldsymbol{a}^{(k)}[p+1] * -1$; `/* a^(k)[p] is the p-th dimension */`

Return $\{a_1, \cdots, a_K\}$;

The gradient $\frac{\partial E}{\partial a_k}$ is acquired through back-propagation under the maximum likelihood criterion in a conventional manner, while $\frac{\partial a_k}{\partial \tilde{\alpha}_j}$ can be calculated using an additional iterative procedure. By taking the derivative w.r.t α_j on the two sides of Equation (5.13), we get

$$\sum_{k=1}^{K} \frac{\partial a_k}{\partial \tilde{\alpha}_j} z^{-k} = z^{-1} \frac{\partial \tanh(\tilde{\alpha}_j)}{\partial \tilde{\alpha}_j} \prod_{k=1, k\neq j}^{K} \left[1 - \tanh(\tilde{\alpha}_k) z^{-1}\right]. \tag{5.15}$$

We can then unfold the right hand side into a polynomial using a similar procedure to Algorithm 1 and set $\frac{\partial a_k}{\partial \tilde{\alpha}_j}$ equal to the coefficient of z^{-k}. Obviously, each $\frac{\partial a_k}{\partial \tilde{\alpha}_j}$ depends on the values of $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$.

In Summary, the proposed method ensures the filter stability by parameterizing the coefficients $\{a_1, \cdots, a_K\}$ into $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ on the basis of Equation (5.12). Compared with a vanilla SAR without any constraints, the SAR using the proposed method just needs a forward computation procedure to convert $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ into $\{a_1, \cdots, a_K\}$ and a backward computation procedure to calculate $\{\frac{\partial E}{\partial \tilde{\alpha}_1}, \cdots, \frac{\partial E}{\partial \tilde{\alpha}_K}\}$ given $\{\frac{\partial E}{\partial a_1}, \cdots, \frac{\partial E}{\partial a_K}\}$. These procedures are applicable to any K.

Let us consider an example where $K = 2$. For the forward computation (i.e.,

from $\{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_K\}$ to $\{a_1, \cdots, a_K\}$), we follow Equation (5.13) and get

$$1 - a_1 z^{-1} - a_2 z^{-2} = \Big[1 - \tanh(\tilde{\alpha}_1) z^{-1}\Big]\Big[1 - \tanh(\tilde{\alpha}_2) z^{-1}\Big]. \tag{5.16}$$

By unfolding the right hand side, we get

$$a_1 = \tanh(\tilde{\alpha}_1) + \tanh(\tilde{\alpha}_2), \quad a_2 = -\tanh(\tilde{\alpha}_1)\tanh(\tilde{\alpha}_2). \tag{5.17}$$

For the backward computation (i.e., from $\{\frac{\partial E}{\partial a_1}, \cdots, \frac{\partial E}{\partial a_K}\}$ to $\{\frac{\partial E}{\partial \tilde{\alpha}_1}, \cdots, \frac{\partial E}{\partial \tilde{\alpha}_K}\}$), we use Equation (5.15) and get

$$\begin{aligned}
\frac{\partial a_1}{\partial \tilde{\alpha}_1} z^{-1} + \frac{\partial a_2}{\partial \tilde{\alpha}_1} z^{-2} &= \frac{\partial \tanh(\tilde{\alpha}_1)}{\partial \tilde{\alpha}_1} z^{-1} - \frac{\partial \tanh(\tilde{\alpha}_1)}{\partial \tilde{\alpha}_1} \tanh(\tilde{\alpha}_2) z^{-2}, \\
\frac{\partial a_1}{\partial \tilde{\alpha}_2} z^{-1} + \frac{\partial a_2}{\partial \tilde{\alpha}_2} z^{-2} &= \frac{\partial \tanh(\tilde{\alpha}_2)}{\partial \tilde{\alpha}_2} z^{-1} - \frac{\partial \tanh(\tilde{\alpha}_2)}{\partial \tilde{\alpha}_2} \tanh(\tilde{\alpha}_1) z^{-2}.
\end{aligned} \tag{5.18}$$

Based on the above equation, we can get

$$\begin{aligned}
\frac{\partial a_1}{\partial \tilde{\alpha}_1} &= \frac{\partial \tanh(\tilde{\alpha}_1)}{\partial \tilde{\alpha}_1}, \quad \frac{\partial a_2}{\partial \tilde{\alpha}_1} = -\frac{\partial \tanh(\tilde{\alpha}_1)}{\partial \tilde{\alpha}_1} \tanh(\tilde{\alpha}_2) \\
\frac{\partial a_1}{\partial \tilde{\alpha}_2} &= \frac{\partial \tanh(\tilde{\alpha}_2)}{\partial \tilde{\alpha}_2}, \quad \frac{\partial a_2}{\partial \tilde{\alpha}_2} = -\frac{\partial \tanh(\tilde{\alpha}_2)}{\partial \tilde{\alpha}_2} \tanh(\tilde{\alpha}_1)
\end{aligned} \tag{5.19}$$

Finally, we can compute $\{\frac{\partial E}{\partial \tilde{\alpha}_1}, \frac{\partial E}{\partial \tilde{\alpha}_2}\}$ using the chain rule in Equation (5.14).

Method 2: High-order filter with complex-poles

The first method only allows the filters to have real-valued poles. A high-order IIR filter in practice could have one or more pairs of conjugate complex-valued poles. To allow such a case, we propose the second sufficient but unnecessary method to parameterize the filter coefficients for the SAR.

This method first constrains a K-order filter $H(z)$ to have at most one real-valued pole. It further constraints other poles of the filter to be pairs of conjugate complex-valued numbers. Accordingly, this method assume a K-order filter can be

written as

$$H(z) = \begin{cases} \dfrac{1}{\prod_{k=1}^{K/2}(1-\alpha_k z^{-1}-\beta_k z^{-2})} & \text{if } K \text{ is even} \\ \dfrac{1}{(1-\alpha_0 z^{-1})\prod_{k=1}^{(K-1)/2}(1-\alpha_k z^{-1}-\beta_k z^{-2})} & \text{if } K \text{ is odd} \end{cases}. \quad (5.20)$$

When K is odd, there are $(K-1)/2$ pairs of conjugated complex-valued poles, and the remaining one real-valued pole $\alpha_0 \in \mathbb{R}$ belongs to the filter $\frac{1}{1-\alpha_0 z^{-1}}$.

For a 2nd order filter $\frac{1}{1-\alpha_k z^{-1}-\beta_k z^{-2}}$, its poles can be generally written as $\{\frac{\alpha_k+\sqrt{\alpha_k^2+4\beta_k}}{2}, \frac{\alpha_k-\sqrt{\alpha_k^2+4\beta_k}}{2}\}$. Since we want these poles to be complex-valued and conjugated, α_k and β_k must satisfy

$$\alpha_k^2 + 4\beta_k < 0, \quad (5.21)$$

and the poles become $\{\frac{\alpha_k}{2} - \frac{\sqrt{-(\alpha_k^2+4\beta_k)}}{2}i, \frac{\alpha_k}{2} + \frac{\sqrt{-(\alpha_k^2+4\beta_k)}}{2}i\}$, where $i = \sqrt{-1}$. To ensure the filter stability, α_0 can be constrained by using a $\tanh(\cdot)$ function when K is odd. For a 2nd order filter, the two poles must be inside the unit circle, or equivalently, their amplitudes should be less than 1[1]. This requires that

$$\left|\frac{\alpha_k}{2} \pm \frac{\sqrt{-(\alpha_k^2+4\beta_k)}}{2}i\right|^2 = \frac{\alpha_k^2}{4} + \frac{-\alpha_4^2-4\beta_k}{4} = -\beta_k \in (0,1). \quad (5.22)$$

The constraints in Equations (5.21-5.22) can be summarized as

$$\beta_k \in (-1,0), \quad \alpha_k \in (-2\sqrt{-\beta_k}, 2\sqrt{-\beta_k}). \quad (5.23)$$

As implementation, β_k and α_k can be parameterized as:

$$\beta_k = -\text{sigmoid}(\tilde{\beta}_k), \quad \alpha_k = 2\sqrt{\text{sigmoid}(\tilde{\beta}_k)}\tanh(\tilde{\alpha}_k). \quad (5.24)$$

We can use the forward and backward computation procedures similar to those in the first method to calculate the filter coefficients $\{a_1, \cdots, a_K\}$ from $\{\tilde{\alpha}_1, \tilde{\beta}_1, \cdots, \tilde{\alpha}_K, \tilde{\beta}_K\}$ and the gradients of $\{\tilde{\alpha}_1, \tilde{\beta}_1, \cdots, \tilde{\alpha}_K, \tilde{\beta}_K\}$ from $\{\frac{\partial E}{\partial a_1}, \cdots, \frac{\partial E}{\partial a_K}\}$.

[1]For a complex number $x+yi$ where $[x,y] \in \mathbb{R}^2$, its amplitude is $|x+yi| = \sqrt{x^2+y^2}$.

Algorithm 2: Levinson recursion algorithm for Method 3: converting log area ratio $\{\tilde{\gamma}_1, \cdots, \tilde{\gamma}_K\}$ to filter coefficients $\{a_1, \cdots, a_K\}$ for the filter $A(z) = 1 - \sum_{k=1}^{K} a_k z^{-1}$ and $H(z) = 1/A(z)$. This implementation is based on `frwlev` in [140].

Input: $\{\tilde{\gamma}_1, \cdots, \tilde{\gamma}_K\}, K$

for $k \leftarrow 1$ **to** K **do**

  $a_k^{(k)} \leftarrow \tanh(\tilde{\gamma}_k)$; `/* -tanh(`$\tilde{\gamma}_k$`) if filter is` $1 + \sum_{k=1}^{K} a_k z^{-1}$ `*/`

  if $k \geq 2$ **then**

    for $p \leftarrow 1$ **to** $k - 1$ **do**

      $a_p^{(k)} \leftarrow a_p^{(k-1)} - \tanh(\tilde{\gamma}_k) a_{k-p}^{(k-1)}$;

Return: $\{a_1, \cdots, a_K\} \leftarrow \{a_1^{(K)}, \cdots, a_K^{(K)}\}$;

Method 3: Filter parameterized by log area ratio

The previous methods require unnecessary assumptions on the filter form and the location of the poles. A more flexible method is to use the log area ratio [137] to parameterize the filter. It is known that the coefficients $\{a_1, \cdots, a_K\}$ of an AR filter $H(z) = \frac{1}{1 - \sum_{k=1}^{K} a_k z^{-1}}$ have a deterministic relationship with reflection coefficients (RCs) $\{\gamma_1, \cdots, \gamma_K\}$, where $\gamma_k \in \mathbb{R}$. The values of $\{a_1, \cdots, a_K\}$ can be calculated from $\{\gamma_1, \cdots, \gamma_K\}$ using a Levinson recursion algorithm [139]. A nice property about RC is that, as long as $\gamma_k \in (-1, 1), \forall k \in \{1, \cdots, K\}$, the filter coefficients $\{a_1, \cdots, a_K\}$ calculated from $\{\gamma_1, \cdots, \gamma_K\}$ will form a stable filter $H(z)$, and the poles of the filter will be inside the unit circle [139] .

To ensure that $\gamma_k \in (-1, 1)$, we can simply set $\gamma_k = \tanh(\tilde{\gamma}_k)$, where $\tilde{\gamma}_k$ is referred to as the log area ratio[2]. Therefore, as long as $\{a_1, \cdots, a_K\}$ are calculated from $\{\tilde{\gamma}_1, \cdots, \tilde{\gamma}_K\}$ using the Levinson recursion algorithm and the tanh function, the stability of the filter is guaranteed.

To implement the SAR using the log area ratio, we need the Levinson recursion algorithm to convert $\{\tilde{\gamma}_1, \cdots, \tilde{\gamma}_K\}$ into $\{a_1, \cdots, a_K\}$ and another backward routine to calculate $\{\frac{\partial E}{\partial \tilde{\gamma}_1}, \cdots, \frac{\partial E}{\partial \tilde{\gamma}_K}\}$ on the basis of $\{\frac{\partial E}{\partial a_1}, \cdots, \frac{\partial E}{\partial a_K}\}$. For reference, the Levinson recursion algorithm is written above as Algorithm 2.

[2] According to the exact definition of log area ratio [137], the relationship between γ_k and $\tilde{\gamma}_k$ should be written as $\gamma_k = \tanh(\frac{\tilde{\gamma}_k}{2}) = \frac{1 - \exp(-\tilde{\gamma}_k)}{1 + \exp(-\tilde{\gamma}_k)}$. In implementation, we directly use the tanh function without the scaling factor $\frac{1}{2}$.

Algorithm 3: Calculating $\frac{\partial a_j}{\partial \tilde{\gamma}_i}$, $\forall j, i \in \{1, \cdots, K\}$, for Method 3.

Input: $\{a_1^{(1)}, a_1^{(2)}, a_2^{(2)}, a_1^{(3)} \cdots, a_1^{(K)}, \cdots, a_K^{(K)}\}, \{\tilde{\gamma}_1, \cdots, \tilde{\gamma}_K\}, K$

for $p \leftarrow 1$ **to** K **do**
  for $i \leftarrow 1$ **to** p **do**
    for $j \leftarrow 1$ **to** p **do**
      if i=p **then**
        if j=p **then**
          $\frac{\partial a_j{}^{(p)}}{\partial \gamma_i} \leftarrow 1$; /* $\frac{\partial a_p{}^{(p)}}{\partial \gamma_p}$ */
        else
          $\frac{\partial a_j{}^{(p)}}{\partial \gamma_i} \leftarrow -a_{p-j}^{(p-1)}$; /* $\frac{\partial a_j{}^{(p)}}{\partial \gamma_p}, j \in [1, p-1]$ */
      else
        if j=p **then**
          $\frac{\partial a_j{}^{(p)}}{\partial \gamma_i} \leftarrow 0$; /* $\frac{\partial a_p{}^{(p)}}{\partial \gamma_i}, i \in [1, p-1]$ */
        else
          $\frac{\partial a_j{}^{(p)}}{\partial \gamma_i} \leftarrow \frac{\partial a_j{}^{(p-1)}}{\partial \gamma_i} - \tanh(\tilde{\gamma}_p) \frac{\partial a_{p-j}{}^{(p-1)}}{\partial \gamma_i}$;

Return $\frac{\partial a_j}{\partial \tilde{\gamma}_i} \leftarrow \frac{\partial a_j^{(K)}}{\partial \gamma_i} \frac{\partial \tanh(\tilde{\gamma}_i)}{\partial \tilde{\gamma}_i}$, $\forall j, i \in \{1, \cdots, K\}$;

For the backward computation, we first need to calculate the gradients $\frac{\partial a_j}{\partial \tilde{\gamma}_i}$, $\forall j, i \in \{1, \cdots, K\}$. After that, the chain rule can be then used to calculate $\frac{\partial E}{\partial \tilde{\gamma}_k} = \sum_{j=1}^{K} \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial \tilde{\gamma}_k}$. Due to the complicated relationship between $\{\tilde{\gamma}_1, \cdots, \tilde{\gamma}_K\}$ and $\{a_1, \cdots, a_K\}$, $\frac{\partial a_j}{\partial \tilde{\gamma}_i}$ must be computed using a recursive procedure listed in Algorithm 3. This algorithm propagates the gradients backwards through the Levinson recursion algorithm. Note that the input of Algorithm 3 includes the intermediate filter coefficients $a_p^{(k)}$ for $p \in \{1, \cdots, k\}$ and $k = \{1, \cdots, K\}$, which are calculated during the Levinson recursion.

The log area ratio was proposed to parameterize the AR filter for the linear prediction coding of speech signals around 45 years ago [137]. The original motivation was to ensure the stability and accuracy of the filter when the parameters are quantized and distorted during speech compression and transmission. An alternative parameterization scheme is based on the line spectrum pairs (LSP) [50, 141]. We leave the LSP-based approach for further work.

5.4 SAR as neural network plus normalizing flow

The filter-based interpretation in Section 5.3 is not the only perspective to understand the SAR. This section gives another interpretation on the basis of the rule to change continuous random variables.

5.4.1 Rule of changing random variable

Suppose a random vector $\boldsymbol{c} \in \mathbb{R}^d$ is drawn from a distribution $p_{\boldsymbol{C}}$. If an invertible function $g : \mathbb{R}^d \to \mathbb{R}^d$ is used to convert $\boldsymbol{c}$ into another vector $\boldsymbol{o}$, i.e., $\boldsymbol{o} = g(\boldsymbol{c})$, the distribution of $\boldsymbol{o}$ can be acquired using the rule of changing random variable. Let's define $f(\cdot) \triangleq g^{-1}(\cdot)$; then $p_{\boldsymbol{O}}(\boldsymbol{o})$ can be written as

$$p_{\boldsymbol{O}}(\boldsymbol{o}) = p_{\boldsymbol{C}}(f(\boldsymbol{o}))\Big|\det\frac{\partial f(\boldsymbol{o})}{\partial \boldsymbol{o}}\Big|, \tag{5.25}$$

where $\frac{\partial f(\boldsymbol{o})}{\partial \boldsymbol{o}}$ denotes the Jacobian matrix.

Interestingly, if another invertible function $g_2 : \mathbb{R}^d \to \mathbb{R}^d$ is applied to convert $\boldsymbol{o}$ into $\boldsymbol{s}$, the distribution of $\boldsymbol{s}$ can be acquired by applying the rule one more time. Let's define $f_2(\cdot) \triangleq g_2^{-1}(\cdot)$, then $p_{\boldsymbol{S}}(\boldsymbol{s})$ can be written as

$$p_{\boldsymbol{S}}(\boldsymbol{s}) = p_{\boldsymbol{C}}(f(f_2(\boldsymbol{s})))\Big|\det\frac{\partial f(\boldsymbol{o})}{\partial \boldsymbol{o}}\Big|\Big|\det\frac{\partial f_2(\boldsymbol{s})}{\partial \boldsymbol{s}}\Big|. \tag{5.26}$$

In a similar manner, the rule of changing random variable can be applied multiple times as long as the transformation function is invertible. Such a technique is referred to as normalizing flow in some literature [142, 143]. It is also related to the feature transformation Method 1n HMM-based speech processing tasks such as speaker adaption for speech recognition [144] and synthesis [145]. We use the name normalization flow because it is applied to the entire sequential data and can be applied multiple times (as a flow).

5.4.2 SAR as neural network plus normalizing flow

The rule of changing random variables can be used to interpret the SAR. Suppose the target feature sequence is denoted $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$, where $\boldsymbol{o}_t \in \mathbb{R}^D$. In the

previous section, we introduce the random variable $c_{t,d} = o_{t,d} - \sum_{k=1}^{K} a_{k,d} o_{t-k,d}$ for $d \in [1, D]$. This relationship can be re-written in a vector form as

$$\boldsymbol{c}_t = \boldsymbol{o}_t - \sum_{k=1}^{K} \boldsymbol{a}_k \odot \boldsymbol{o}_{t-k}, \tag{5.27}$$

where $\boldsymbol{c}_k = [c_{k,1}, \cdots, c_{k,D}]^\top$, $\boldsymbol{o}_k = [o_{k,1}, \cdots, o_{k,D}]^\top$, and $\boldsymbol{a}_k = [a_{k,1}, \cdots, a_{k,D}]^\top$. If we format the vector sequences $\boldsymbol{o}_{1:T}$ and $\boldsymbol{c}_{1:T}$ as column vectors $\boldsymbol{o}_{1:T} = [\boldsymbol{o}_1^\top, \cdots, \boldsymbol{o}_T^\top]^\top$ and $\boldsymbol{c}_{1:T} = [\boldsymbol{c}_1^\top, \cdots, \boldsymbol{c}_T^\top]^\top$, we can define a function $f : \mathbb{R}^{DT} \to \mathbb{R}^{DT}$ as

$$\boldsymbol{c}_{1:T} = f(\boldsymbol{o}_{1:T}) = \boldsymbol{A}\boldsymbol{o}_{1:T}, \tag{5.28}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ -\boldsymbol{A}_1 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ -\boldsymbol{A}_2 & -\boldsymbol{A}_1 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ -\boldsymbol{A}_K & \cdots & -\boldsymbol{A}_2 & -\boldsymbol{A}_1 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{A}_K & \cdots & -\boldsymbol{A}_2 & -\boldsymbol{A}_1 & \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & -\boldsymbol{A}_K & \cdots & -\boldsymbol{A}_2 & -\boldsymbol{A}_1 & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\boldsymbol{A}_K & \cdots & -\boldsymbol{A}_2 & -\boldsymbol{A}_1 & \mathbf{1} \end{bmatrix}^{[3]}, \tag{5.29}$$

$\mathbf{1}$ is an identity matrix of size $D \times D$, $\mathbf{0}$ is a zero matrix, and $\boldsymbol{A}_k = \text{diag}(\boldsymbol{a}_k)$ is a diagonal matrix whose diagonal line is equal to the vector $\boldsymbol{a}_k$. Notice that this transformation is invertible and $\det(\boldsymbol{A}) = 1$.

With all the above facts, we can interpret the function $f(\cdot)$ in Equation (5.28) as the 'inverse function that recovers' the data $\boldsymbol{c}_{1:T}$ from the observed $\boldsymbol{o}_{1:T}$. Then,

[3] The matrix $\boldsymbol{A}$ can be formed by merging the matrices $\{\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(D)}\}$ defined in Equation (5.7).

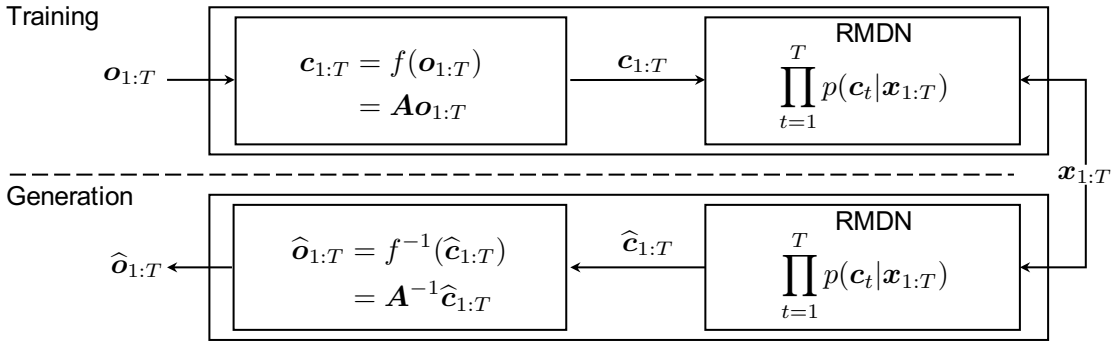


Figure 5.5: Interpretation of SAR using normalizing flow (feature transformation)

we apply the rule of changing random variable and show that

$$\begin{aligned} \boldsymbol{p}_{\boldsymbol{o}|\boldsymbol{x}}(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T}) &= \boldsymbol{p}_{\boldsymbol{c}|\boldsymbol{x}}(f(\boldsymbol{o}_{1:T})|\boldsymbol{x}_{1:T})\left|\det\frac{\partial f(\boldsymbol{o}_{1:T})}{\partial \boldsymbol{o}_{1:T}}\right| \\ &= \boldsymbol{p}_{\boldsymbol{c}|\boldsymbol{x}}(f(\boldsymbol{o}_{1:T})|\boldsymbol{x}_{1:T})\left|\det\boldsymbol{A}\right| \\ &= \boldsymbol{p}_{\boldsymbol{c}|\boldsymbol{x}}(\boldsymbol{c}_{1:T}|\boldsymbol{x}_{1:T}) \end{aligned} . \tag{5.30}$$

This equation explains the SAR from the perspective of normalizing flow and justifies the equality in Equation (5.11). Specifically, the SAR uses the function $f(\cdot)$ to transform the target feature $\boldsymbol{o}_{1:T}$ into $\boldsymbol{c}_{1:T}$ and uses the RMDN to evaluate $\boldsymbol{p}_{\boldsymbol{c}|\boldsymbol{x}}(\boldsymbol{c}_{1:T}|\boldsymbol{x}_{1:T})$. In the generation stage, the SAR uses the RMDN to generate a sequence $\widehat{\boldsymbol{c}}_{1:T}$ and recovers the target sequence $\widehat{\boldsymbol{o}}_{1:T} = f^{-1}(\widehat{\boldsymbol{c}}_{1:T})$. Because $\det(\boldsymbol{A}) = 1$, the SAR is a special case of volume-preserving normalizing flow [146] with just a single stage of transformation.

The above interpretation indicates the potential of SAR from a different perspective. Since a baseline RMDN cannot model the temporal correlation within $\boldsymbol{o}_{1:T}$, one strategy is to transform the original data $\boldsymbol{o}_{1:T}$ into another domain, hoping that the transformed data $\boldsymbol{c}_{1:T}$ contain less temporal correlation. In this way, the transformed data $\boldsymbol{c}_{1:T}$ may be more compatible with the model assumption of the RMDN. This interpretation can be illustrated by Figure 5.5, which is similar to that in Figure 5.4 except that the filters are replaced by a invertible function $f(\cdot)$.

5.4.3 Extended SAR with time-variant transformation

Theory

The interpretation in Figure 5.5 is quite general. In the case of the SAR, the function $f(\cdot)$ is determined by the AR linear transformation in Equation (5.27). However, we can use more complicated functions as long as they are invertible and allow the Jacobian matrix to be calculated easily.

For example, we can define a non-linear transformation as

$$\boldsymbol{c}_t = \boldsymbol{o}_t - f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{1:t-1}), \tag{5.31}$$

where $f(\boldsymbol{o}_{1:t-1}) : \mathbb{R}^{D\times(t-1)} \to \mathbb{R}^D$ is a function that merges all the previous frames $\boldsymbol{o}_{1:t-1}$ into a bias vector. Although this function looks simple, it generalizes the original SAR with a time-invariant AR dependency function (Equation 5.2) and an extended SAR with a time-variant function:

$$\begin{aligned} \text{time-invariant} : f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{1:t-k}) &= \sum_{k=1}^{K} \boldsymbol{a}_k \odot \boldsymbol{o}_{t-k}, \\ \text{time-variant} : f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{1:t-k}) &= \sum_{k=1}^{K} f^{(k)}_{\boldsymbol{\psi_k}}(\boldsymbol{o}_{1:t-k}) \odot \boldsymbol{o}_{t-k}, \end{aligned} \tag{5.32}$$

In the time-invariant case, $\boldsymbol{\Psi} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_K\}$ defines the time-invariant scaling vectors. In the time-variant case, $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_K\}$, and $f^{(k)}_{\boldsymbol{\psi}_k} : \mathbb{R}^{D\times(t-k)} \to \mathbb{R}^D$ calculates the scaling vectors based on $\boldsymbol{o}_{1:t-k}$. Compared with the two cases above, the general function in Equation (5.31) does not use a linear weighted sum to summarize the information only from $\{\boldsymbol{o}_{t-K}, \cdots, \boldsymbol{o}_{t-1}\}$. Therefore, it is potentially more general and powerful.

Given the transformation function above for each time step t, the transformation between the sequences $\boldsymbol{o}_{1:T}$ and $\boldsymbol{c}_{1:T}$ can be written as

$$\boldsymbol{c}_{1:T} = f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{1:T}). \tag{5.33}$$

In contrast to Equation (5.28) of the original SAR, this transformation function may not necessarily be a linear function. It is may be impractical to compute

the full Jacobian matrix $\frac{\partial f(\boldsymbol{o}_{1:T})}{\partial \boldsymbol{o}_{1:T}} = \frac{\partial \boldsymbol{c}_{1:T}}{\partial \boldsymbol{o}_{1:T}}$. However, the rule of changing random variables only needs $\det \frac{\partial f(\boldsymbol{o}_{1:T})}{\partial \boldsymbol{o}_{1:T}}$. If the Jacobian matrix has special structures, its determinant can be computed without calculating all the elements in the matrix.

Fortunately, $\frac{\partial \boldsymbol{c}_{1:T}}{\partial \boldsymbol{o}_{1:T}}$ based on Equation (5.31) is such a case. According to that equation, we know that

$$\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{o}_j} = \begin{cases} \mathbf{1}, & j = t \\ \mathbf{0}, & j > t \\ \dfrac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{o}_j}, & j < t \end{cases} . \tag{5.34}$$

Consequently, the Jacobian matrix $\frac{\partial \boldsymbol{c}_{1:T}}{\partial \boldsymbol{o}_{1:T}}$ is a low-triangular matrix whose the diagonal blocks are identity matrices, i.e.,

$$\frac{\partial \boldsymbol{c}_{1:T}}{\partial \boldsymbol{o}_{1:T}} = \begin{bmatrix} \frac{\partial \boldsymbol{c}_1}{\partial \boldsymbol{o}_1} & \frac{\partial \boldsymbol{c}_1}{\partial \boldsymbol{o}_2} & \cdots & \frac{\partial \boldsymbol{c}_1}{\partial \boldsymbol{o}_T} \\ \frac{\partial \boldsymbol{c}_2}{\partial \boldsymbol{o}_1} & \frac{\partial \boldsymbol{c}_2}{\partial \boldsymbol{o}_2} & \cdots & \frac{\partial \boldsymbol{c}_2}{\partial \boldsymbol{o}_T} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{c}_T}{\partial \boldsymbol{o}_1} & \frac{\partial \boldsymbol{c}_T}{\partial \boldsymbol{o}_2} & \cdots & \frac{\partial \boldsymbol{c}_T}{\partial \boldsymbol{o}_T} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \frac{\partial \boldsymbol{c}_2}{\partial \boldsymbol{o}_1} & \mathbf{1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{c}_T}{\partial \boldsymbol{o}_1} & \frac{\partial \boldsymbol{c}_T}{\partial \boldsymbol{o}_2} & \cdots & \mathbf{1} \end{bmatrix} . \tag{5.35}$$

Although the off-diagonal block matrices in the lower triangular part may be difficult to compute, the determinant of the Jacobian matrix does not depends on them and is simply equal to one, i.e., $\det \frac{\partial \boldsymbol{c}_{1:T}}{\partial \boldsymbol{o}_{1:T}} = 1$. Therefore, the rule of changing random variables can be easily applied to the generalized SAR.

The transformation can also defined in an anti-AR manner, for example as

$$\boldsymbol{c}_t = \boldsymbol{o}_t - f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{t+1:T}). \tag{5.36}$$

In this case, the Jacobian matrix will be an up-triangular matrix, and its determinant is still equal to 1. This anti-AR transformation can be applied after the AR transformation. There is no conflict as long as each transformation only contains uni-directional dependency.

In this thesis, we refer to the SAR with the non-linear AR/anti-AR dependency function as the extended SAR (eSAR). Although the transformation is now time-variant, the extended SAR still implements the AR dependency since $\boldsymbol{c}_t$ only depends on the current and the previous observations.

Implementation

In implementation, we use a uni-directional RNN layer and a linear transformation layer to calculate $f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{1:t-1})$. We refer to the combination of these two layers as a normalizing flow block. This uni-directional RNN layer should compute in a forward direction from $t = 1$ to $t = T$ in the case of AR dependency. For the anti-AR dependency, however, the uni-directional RNN should compute reversely from $t = T$ to $t = 1$.

In the training stage, we need to learn the parameter $\{\boldsymbol{\Psi}, \boldsymbol{\Theta}\}$ by maximizing the likelihood function $p(f_{\boldsymbol{\Psi}}(\boldsymbol{o}_{1:T})|\boldsymbol{x}_{1:T}; \boldsymbol{\Theta})$. Since both $\boldsymbol{\Psi}$ and $\boldsymbol{\Theta}$ are the weights of neural network, we may update them iteratively. A practical training recipe is:

1. Use a trained baseline RMDN to initialize $\boldsymbol{\Theta}$;
2. Fix $\boldsymbol{\Theta}$ and update $\boldsymbol{\Psi}$ by maximizing the likelihood;
3. Fix $\boldsymbol{\Psi}$ and update $\boldsymbol{\Theta}$ by maximizing the likelihood;
4. Repeat steps 2 and 3

If multiple transformations are used, we can update all the transformation networks together, i.e., treating $\boldsymbol{\Psi} = \{\boldsymbol{\Psi}_1, \cdots, \boldsymbol{\Psi}_N\}$ where $\boldsymbol{\Psi}_n$ is the network parameter set for n-th normalizing block.

In the generation stage, we can sequentially generate $\widehat{\boldsymbol{o}}_{1:T}$ after acquiring $\widehat{\boldsymbol{c}}_{1:T}$ from the RMDN part:

$$\begin{aligned}
\widehat{\boldsymbol{o}}_1 &= \widehat{\boldsymbol{c}}_1 \\
\widehat{\boldsymbol{o}}_2 &= \widehat{\boldsymbol{c}}_2 + f_{\boldsymbol{\Psi}}(\widehat{\boldsymbol{o}}_1) \\
&\cdots \\
\widehat{\boldsymbol{o}}_t &= \widehat{\boldsymbol{c}}_t + f_{\boldsymbol{\Psi}}(\widehat{\boldsymbol{o}}_{1:t-1}) \\
&\cdots \\
\widehat{\boldsymbol{o}}_T &= \widehat{\boldsymbol{c}}_T + f_{\boldsymbol{\Psi}}(\widehat{\boldsymbol{o}}_{1:T-1})
\end{aligned} \tag{5.37}$$

If multiple transformation blocks are used, we have to do the above process in each transformation block. If the transformation block is anti-AR, the generation process should be proceeded from $t = T$ to $t = 1$.

5.5 Evaluating SAR

5.5.1 Data and configuration

For experiments, pilot tests were first conducted to examine the stability issue and the filter order of the SAR. The main test was then conducted to compare the SAR and other neural F0 models. Details of the network configuration will be explained in each experiment. The English and the Japanese corpora were used for experiments, details of which can be found in Appendix A.

The input linguistic features are defined in Table A.1. The target F0 is the interpolated continuous-valued (Mel-scale) F0. Network structures will be explained in each experiment. Unless specifically explained, all the models were trained using stochastic gradient descent (learning rate 1e-5) with early stopping, after which the models were further tuned using the AdaGrad optimizer [147] (learning rate 0.001) with early stopping.

5.5.2 Pilot test I: effectiveness of SAR stability constraints

In Section 5.3.2, we proposed three methods to ensure the stability of the virtual synthesis filter $H(z)$ in the SAR. This pilot test examines the effectiveness of proposed methods. In this test, we compared four SARs listed in Table 5.1 on the English corpus. We set the SAR order $K = 6$ because a relatively high order can easily reveal the instability of a naive SAR[4].

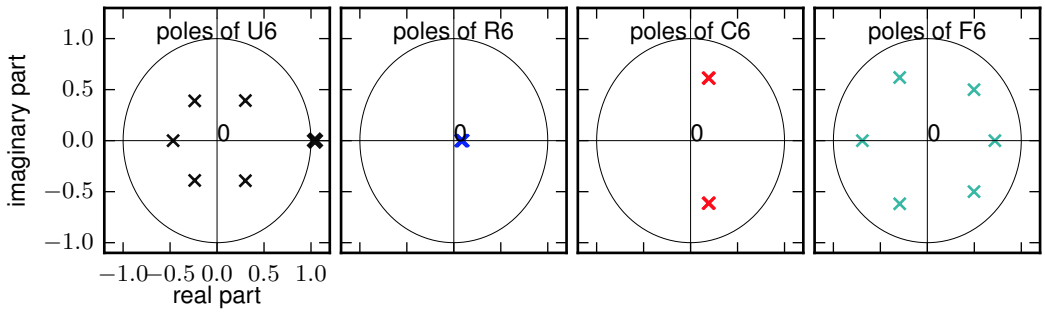
All the models used the same network structure: two feedforward layers of size 512 $\rightarrow$ two bi-directional LSTM recurrent layers of size 256 and 128, respectively $\rightarrow$ a linear layer to generate the parameters for the target data distribution. The target data distribution included a GMM of 2 mixture components for F0 and a binary distribution for U/V. The internal RNN of the SAR was initialized by a trained RNN (`RNN` in Section 5.5.4), and the SAR was fine tuned by using the stochastic gradient descent (learning rate 1e-09) for 5 epochs.

After training the models, we extracted the parameters of the virtual filter in each model and plot the poles of the synthesis filter $H(z)$ in Figure 5.6 (a).

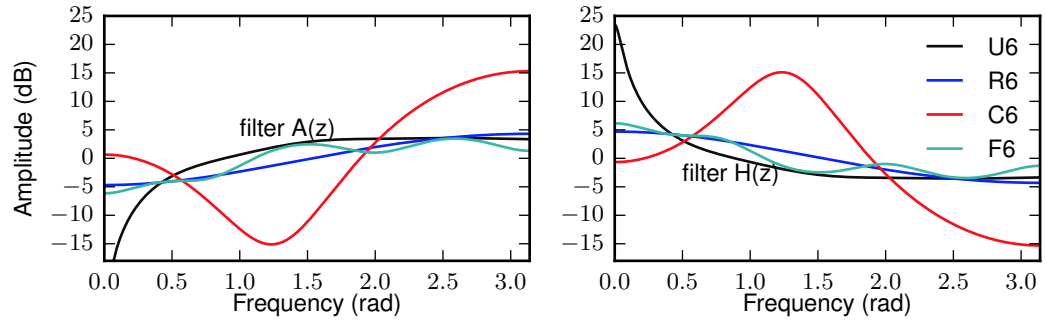
[4]For $K < 6$, the SAR trained on the experimental corpus happened to be stable even without any constraint on the filter stability. However, the stability is not guaranteed for other corpora.

Table 5.1: SAR models for pilot test on filter stability (K=6).

Name	Form of synthesis filter $H(z)$	Constraints on stability of $H(z)$
U6	$\frac{1}{1-\sum_{k=1}^{K} a_k z^{-k}}$	No constraints
R6	$\prod_{k=1}^{K} \frac{1}{1-\tanh(\tilde{\alpha}_k) z^{-1}}$	Method 1: Stable $H(z)$ with real-valued poles inside the unit circle
C6	$\prod_{k=1}^{K/2} \frac{1}{1-\alpha_k z^{-1}-\beta_k z^{-2}}$	Method 2: Stable $H(z)$ with conjugated complex poles inside the unit circle
F6	$\frac{1}{1-\sum_{k=1}^{K} a_k z^{-k}}$	Method 3: Stable $H(z)$ parameterized on the basis of log area ratio



(a) Poles of synthesis filter $H(z)$. The black circle denotes the unit circle.



(b) Frequency responses of analysis filter $A(z)$ and synthesis filter $H(z)$.

Figure 5.6: Poles' location and frequency responses of virtual filters in SARs.

The results show that the $H(z)$ of U6 acquired a real-valued pole outside the unit-circle, which means that the filter was unstable. Figure 5.6 (b), which shows the frequency response of the virtual filters, demonstrates that the $H(z)$ of U6 significantly magnified the low-frequency band due to its instability.

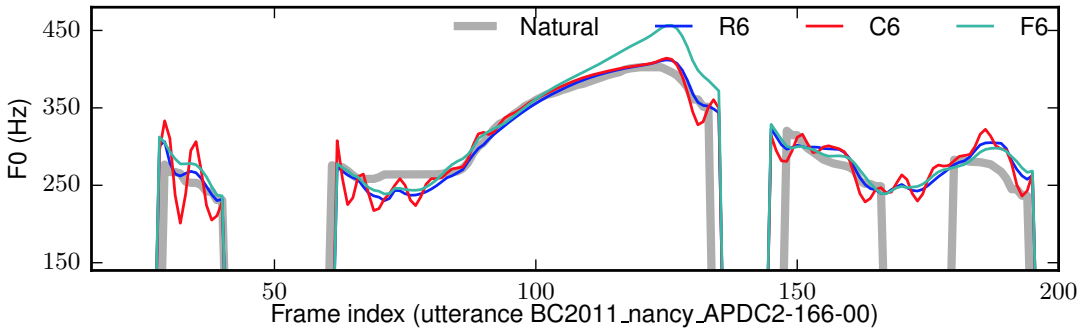


Figure 5.7: Generated F0 from SAR (mean-based generation, English corpus).

In contrast, the results showed that `R6`, `C6`, and `F6` ensured the stability of the filter. Specifically, `R6` constrained the synthesis filter's poles on the real-axis between (-1, 1) while `C6` constrained the poles to be three pairs of conjugated complex numbers within the unit circle. Compared with `R6` and `C6`, `F6` acquired both complex- and real-valued poles within the unit-circle.

Which type of constraints should be used for F0 modeling? According to Figure 5.7, the F0 contours generated by `R6` and `F6` were quite smooth while that from `C6` contained ripples. The ripple in the F0 contour is harmful to the perceived quality of reading speech because natural F0 contours never show such kind of rapid vibration. The ripple in the case of `C6` may be caused by the synthesis filter that learned to be a frequency selector as Figure 5.6 demonstrates. In general, `C6` is unsuitable for modeling F0 contours of speech data. However, it may be useful for other types data such as the F0 of singing voices that contain natural vibration.

5.5.3 Pilot test II: Selection of AR dependency order

Based on the first pilot test, we compared the SAR models listed in Table 5.2 in order to examine the impact of different values of the AR dependency order K. Note that `R6` and `F6` in Table 5.2 are identical to those in Table 5.1. The same training recipe from the pilot test I was used in this test.

Figure 5.8 plots the frequency response of the virtual filters for the experimental SAR models. Interestingly, all the `R*` models learned a 'high-pass' analysis filter and accordingly a 'low-pass' synthesis filter[5]. With the exception of `R1`, filters in

[5]The filter should be called 'high-emphasis' or 'low-emphasis' because it increases the power

Table 5.2: SAR models for pilot test on AR dependency order and its performance (mean-based generation) on the English corpus.

Name	AR order	Constraints on stability of $H(z)$	**RMSE**	**CORR**	**U/V**
R1	$K = 1$	Method 1: stable H(z) with real-valued poles inside unit circle	47.56	0.765	4.92%
R2	$K = 2$		47.51	0.766	4.91%
R4	$K = 4$		47.98	0.765	4.91%
R6	$K = 6$		48.16	0.764	4.91%
F2	$K = 2$	Method 3: stable H(z) using log area ratio	49.21	0.759	4.98%
F4	$K = 4$		48.80	0.759	4.98%
F6	$K = 6$		49.19	0.760	4.98%

other R* models showed similar frequency response curves. These models achieved similar objective performances and generated similar F0 contours as Table 5.2 and Figure 5.9 demonstrate.

The filters in R6 acquired a similar frequency response to R2 and R4 even if it had a higher order. One possible reason is that the virtual filter is constrained to be a cascade of 1st order filters with a real-valued pole (Equation (5.12)). Whatever the AR order K is, the filter's frequency response is equivalent to the product of the frequency response of each 1st order filter. Since each 1st order filter can either be 'low-pass' or 'high-pass', the cascade filter is highly possible to be a simple 'low-pass' or 'high-pass' filter.

In the case of F*, the frequency response of the analysis filter is generally 'high-pass'. The difference is that the filter frequency response curve in not monotonically increasing. This result is reasonable because a filter parameterized by the log-area ratio can acquire complex-valued poles. These complex-valued poles act as frequency selectors and lead to the bumps on the frequency response curve.

Although F* acquired virtual filters with more complicated frequency response curves, it does not mean that they are suitable for F0 modeling. In fact, F* performed slightly worse than R* in terms of objective performance. Comparison of the models with different K further suggests that it is unnecessary to use an SAR with a high AR order for F0 modeling. These suggestions may be reasonable

of the corresponding frequency band.

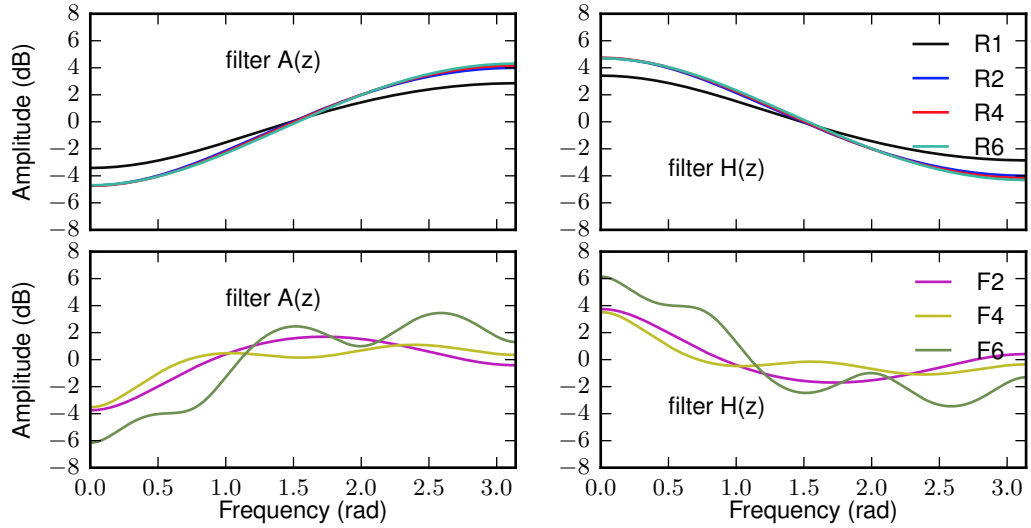


Figure 5.8: Frequency response of analysis (left column) and synthesis filters (right column) for SARs in Table 5.2.

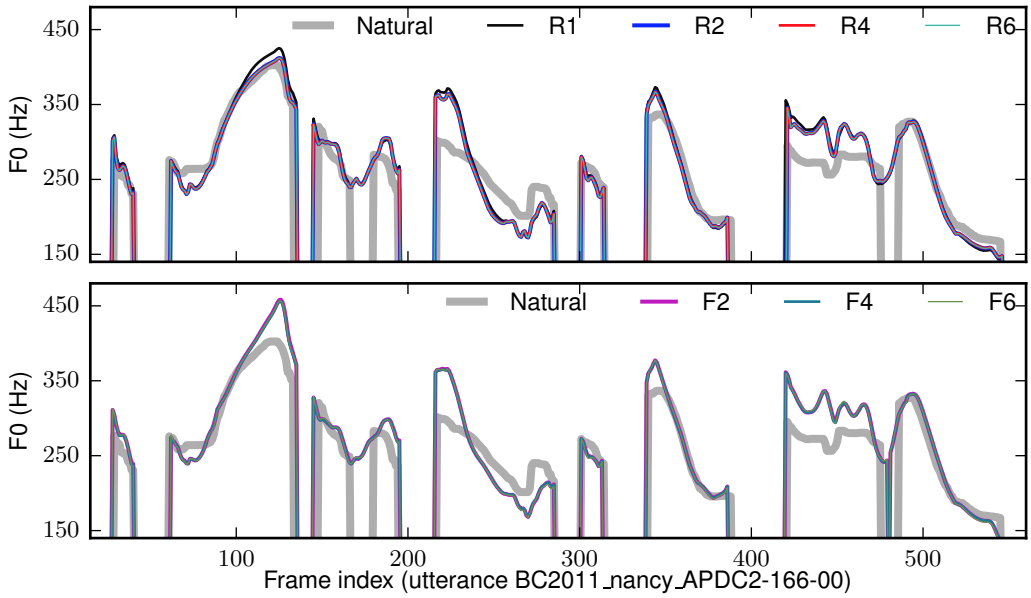


Figure 5.9: Generated F0 from SARs in Table 5.2 (mean-based generation, English corpus). Note that the F0 contours in each figure are quire similar.

because the time-invariant virtual filter is trained to match the general shape of all the F0 contours in the training set. Even though each individual F0 contour may contain varied shapes, the general shape of all F0 contours is smooth. Therefore, we set $K = 2$ for the SAR in the following experiments and use method 1 to ensure the stability of the virtual filter.

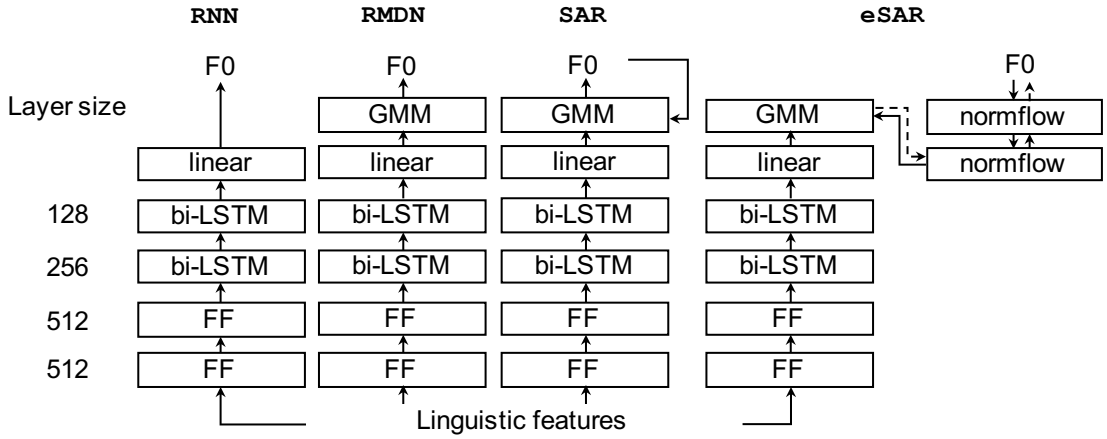


Figure 5.10: Network structures for models in Section 5.5.4. 'FF', 'bi-LSTM', and 'uni-LSTM' denotes feedforward layer, bi- and uni-directional LSTM layer, respectively. 'normflow' denotes a normalizing block.

5.5.4 Evaluating SAR against baseline models

Model configuration

Four models shown in Figure 5.10 were evaluated on the Japanese corpus. All of them had a similar network structure except the output layer. For `RNN`, the linear output layer had size 2, where 1 dimension was used for F0 and the other dimension for U/V. For `RMDN`, `SAR`, and `eSAR`, the output layer included a GMM of 2 mixture components for F0 and a binary distribution of U/V. Accordingly, the preceding linear layer had size 7 [6]. `SAR` used $K = 2$ and the method 1 to constrain the real-value poles. `eSAR` used two normalizing blocks, one for the AR and the other for anti-AR dependency. Each block used a uni-directional LSTM layer of size 64 and a linear layer of size 2 [7].

`RNN` was initialized using the layer-size-dependent uniform distribution [127]. `RMDN` was initialized with the trained `RNN` except the last linear layer. These two models were trained using the default training recipe, i.e., stochastic gradient descent plus AdaGrad tuning. `SAR` and `eSAR` were initialized by the trained `RMDN` and then further trained for 5 epochs using the stochastic gradient descent.

[6] $7 = 2$ mixtures $\times$ (1 Gaussian mean + 1 Gaussian variance + 1 mixture weight) + 1 for u/v

[7] 2 = 1 dimension for F0 value + 1 dimension for U/V. The dimension for U/V is a dummy in the implementation.

Table 5.3: Objective results on Japanese corpus for models defined in Figure 5.10. Mean-based generation method was used. GV of natural F0 is around 61. Log-likelihood was evaluated on the validation set.

	Log-likelihood	**RMSE**	**CORR**	**U/V**	**GV**
RNN	-	29.31	0.894	3.26%	51.4
RMDN	-3340.9	28.32	0.897	3.85%	54.2
SAR	-3278.6	34.57	0.898	3.85%	71.8
eSAR	-2660.5	29.74	0.897	3.70%	63.4

Objective results and analysis

Objective results of the models are listed in Table 5.3. First, the SAR and the eSAR achieved higher likelihood on the validation set. This suggests the better capability of the model to describe the F0 data distribution.

However, these two models did not outperform RMDN in terms of the RMSE. The reason may be that the RMSE is more consistent with the training criterion of RMDN. First, the maximum likelihood training criterion on RMDN is closely related to the square-error criterion as Section 3.2 explains. Second, the mean-based output is the 'best-output' in terms of mean-square-error as Equation (3.14) proves. Therefore, RMDN's performance on RMSE is reasonable. Although SAR and eSAR used the same training and generation methods as RMDN, they are trained by maximizing the likelihood on the transformed data, thus minimizing the error in the transformed domain. This above argument cannot be applied to CORR because none of the model was directly trained to increase CORR. Nevertheless, all the models achieved a similar CORR score, suggesting that they all worked properly.

The scores of CORR and RMSE are not the only objective metrics. Another important metric for F0 modeling is the GV score. For reference, Figure 5.11 shows the box-plot of GV on the test set, whereas Table 5.3 shows the average GV score. The results suggest that RMDN and RNN suffered from the over-smoothing effect while SAR and eSAR maintained the dynamic range of generated F0 contours. This difference can be observed from the generated F0 contours in Figure 5.12. Compared with SAR, eSAR acquired a GV score closer to that of the natural F0.

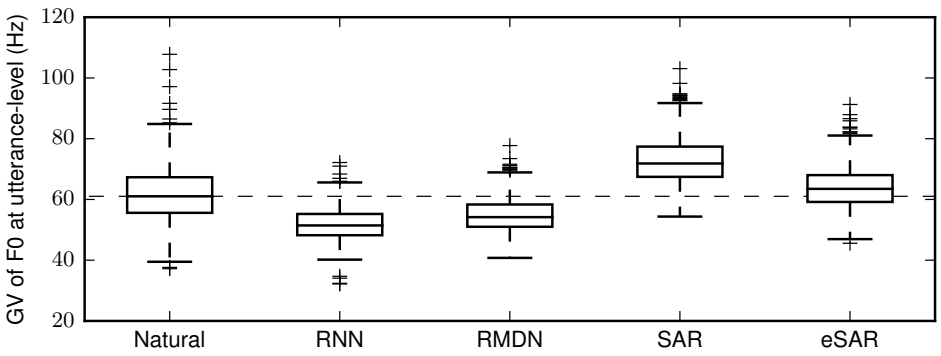


Figure 5.11: GV of generated F0 from models in Figure 5.10 (mean-based generation, Japanese corpus).

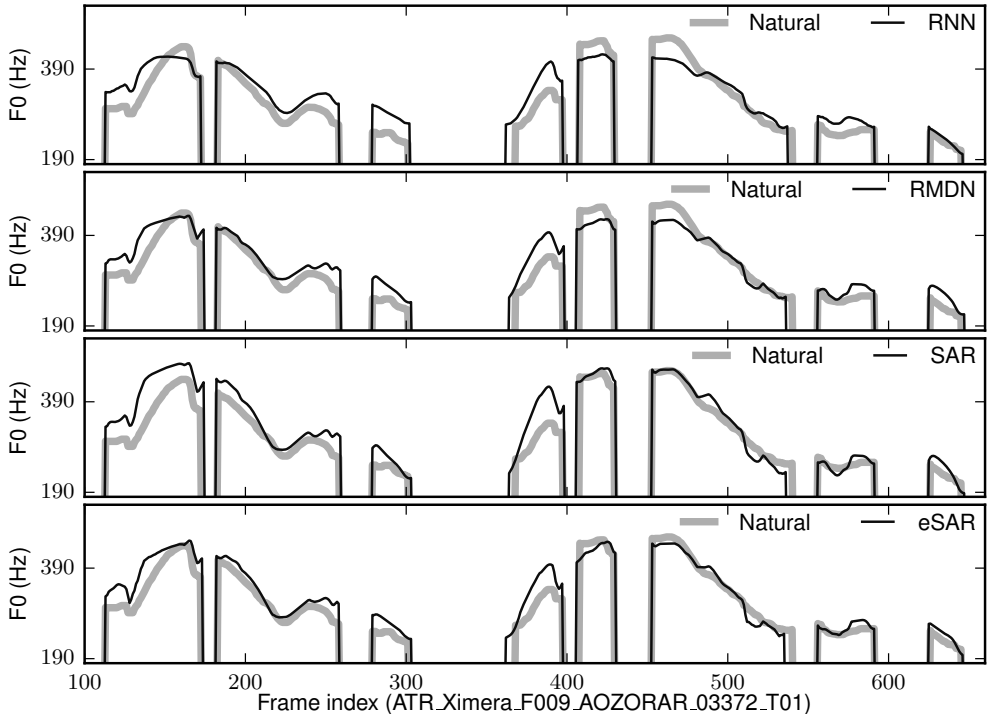


Figure 5.12: Generated F0 from models in Figure 5.10 (mean-based generation, Japanese corpus)

The performance of `SAR` can be explained using the interpretation of signal and filters. Similar to the frequency response in Figure 5.8, `SAR` acquired a virtual analysis filter that enhanced the high-frequency bands of the F0 contour while suppressing the low-frequency bands. Because the F0 contour usually evolves smoothly and has most of the energy in the low-frequency band, what the analysis

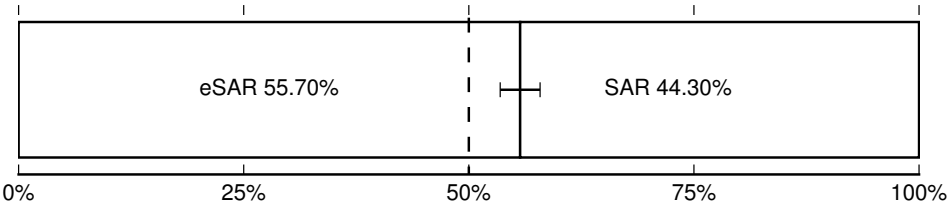


Figure 5.13: Preference test comparing eSAR and SAR (mean-based generation, Japanese corpus). The error bar shows the interval at the confidence level of 99%.

filter $A(z)$ did was similar to whitening the F0 contour. The virtual RMDN in `SAR` learned the distribution of the whitened F0 contours conditioned on the linguistic features. Therefore, the averaging effect of statistical modeling had less impact on the original F0 data. This may be the reason for the increased GV of `SAR`.

A similar argument may be used to explain `eSAR`, even though we cannot interpret it on the basis of linear filters. Nevertheless, `eSAR` generated F0 contours that were more similar to the natural ones than `SAR`, which can be observed in Figure 5.12.

Subjective results and analysis

For the subjective evaluation, a preference test was conducted to compare `eSAR` and `SAR`. For this evaluation, we summoned native Japanese speaker through paid online service. In each testing round, two test utterances were randomly selected from the test set. Let's name the corresponding synthetic samples from `eSAR` as eSAR1 and eSAR2, and those of `SAR` as SAR1 and SAR2. Four pairs of samples, i.e., (eSAR1, SAR1), (SAR1, eSAR1), (eSAR2, SAR2) and (SAR2, eSAR2), were then randomly shuffled and evaluated by the listener. Each listener can evaluate more than one testing round.

In total, 109 listeners conducted 851 testing rounds, where each listener conducted around 8 rounds on average. Accordingly, 3404 ($=851 \times 4$) preference scores were collected. The result is plotted in Figure 5.13. The percentage of selecting `eSAR` was around 56%. A two-sided binomial test showed that the preference towards `eSAR` was statistical significant (p-value < 0.01). Therefore, we conclude that `eSAR` achieved a better subjective result than `SAR` on the Japanese corpus.

In another listening test that compared the SAR models against the baseline RMDN and RNN, the results show that SAR performed worse than RMDN while similar to RNN. Meanwhile, eSAR was not significantly different from RMDN even though it is slightly better than RNN. In order to avoid reporting the results repeatedly, we put this test to the next chapter (Section 6.4.5). Although the results may be unsatisfactory, they can provide the initial answer to the issue asked at the beginning of this chapter.

5.6 Summary

This chapter focused on **Issue 2: Do the common neural models describe the temporal dependency in F0 contours?** The answer is simply 'No'.

This chapter used a simple random-sampling-based generation method and showed how a baseline RMDN ignored the temporal dependency of the target sequential data. Then, this chapter introduced the idea of AR dependency modeling and defined a model called SAR. A toy example showed the theoretical advantage of the SAR over the RMDN to model the temporal dependency in the target sequential data.

Although the model definition is simple, this chapter gave two interpretations of the SAR, one based on the signal and filter and the other one based on the framework of normalizing flow. Interestingly, the first interpretation revealed the issue of model stability and motivated three methods to ensure the stability of the SAR. On the other hand, the second interpretation allowed us to extend the original SAR into a more general AR model using an invertible and long-term AR dependency function.

Experimental results justified the effectiveness of proposed methods to ensure the stability of the SAR. Among the three methods, the one assuming real-valued poles (method 1) was found to be more suitable for F0 modeling than the other two methods. Other experiments showed that both the SAR and the extended SAR alleviated the over-smoothing effect. But the extended SAR was preferred over the SAR in a subjective evaluation test.

Despite the imperfect subjective performance, the SAR and its extended version are theoretically appealing. Besides, the theoretical advantage of using AR

dependency, the linking between feature transformation, signal filtering, and filter stability provides sound framework for sequential data modeling. Although they may not be the best choice for F0 modeling, our other works have demonstrated the better performance of the SAR when it was applied to model spectral feature sequences [148, 149] (See the part of results in Section 8.2).

Note we did not show the performance of the SAR using the random-sampling-based generation method. As the next chapter will explain, the result was negative because of the limitation of using linear or shallow functions to capture the AR dependency.

6

Deep Autoregressive Neural F0 model

This chapter also focuses on **Issue 2**, the issue of modeling the temporal correlation of the F0 contours. Although Chapter 5 introduces the SAR to model the temporal correlation, this chapter will show that the SAR failed to pass the test of random sampling. One important reason is that the SAR relies on linear or at least invertible functions to implement the AR dependency, which lacks the capability to fully describe the temporal correlation in F0 contours.

One strategy is to switch to deep AR dependency, a kind of statistical dependency that requires non-linear and non-invertible transformations upon the target data sequence. For this purpose, this chapter introduces the deep AR (DAR) model. Although the implementation is straightforward, a toy example can show that the DAR is more general than the SAR. With additional techniques such as quantized F0 modeling, a hierarchical softmax layer, and a data dropout strategy, the DAR can achieve better performance than the previous models. Additionally, it allows random-sampling-based F0 generation, which has not been achieved by the previous models.

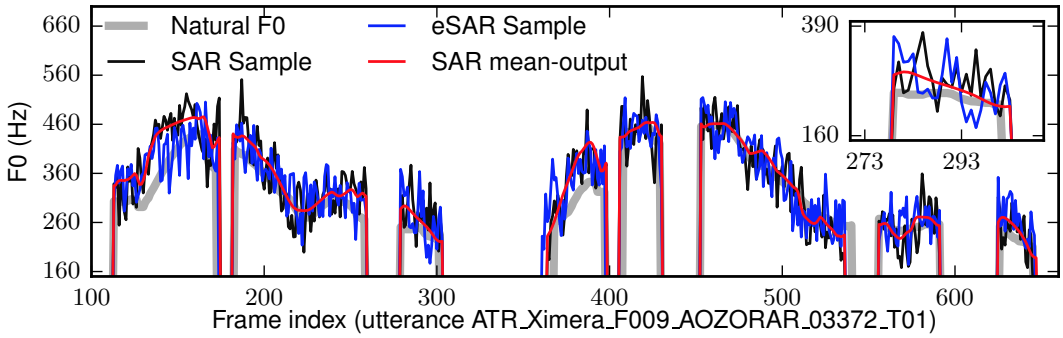


Figure 6.1: Generated F0 from SAR using mean-based and random sampling on Japanese corpus.

Section 6.1 shows the performance of the SAR in random-sampling-based F0 generation and explains the possible reasons. Section 6.2 introduces the idea of DAR and uses toy examples to explain its generality. Section 6.3 then introduces the aforementioned techniques for the DAR-based F0 model. Finally, Section 6.4 details the evaluation on DAR, including both the mean-based and random-sampling-based cases.

6.1 Weakness of SAR

6.1.1 Random sampling on SAR

Theory and experiment in the previous chapter indicate that the SAR is better than the RMDN in modeling temporal correlation of the target data sequence. However, the local and linear AR dependency modeled by the SAR may be still insufficient. This weakness can be revealed by using the random-sampling-based generation method. Notice that, since the SAR takes the AR dependency into consideration, an F0 contour must be sampled from the model sequentially. Specifically, $\widehat{\boldsymbol{o}}_1$ can be sampled from $p(\boldsymbol{o}_1|\boldsymbol{x}_{1:T})$, and $\widehat{\boldsymbol{o}}_2$ then can be drawn from $p(\boldsymbol{o}_2|\widehat{\boldsymbol{o}}_1, \boldsymbol{x}_{1:T})$. Repeating this ancestral sampling process can generate an F0 contour. This same sequential sampling process can be used on the extended SAR.

This sequential sampling process suggests that the distribution of $\widehat{\boldsymbol{o}}_t$ would be influenced by the samples drawn in the previous steps. Compared with the baseline RMDN where samples are independently drawn, the AR dependency in

the SAR may perform better. However, the sampled F0s from the SAR and its extended version (`SAR` and `eSAR` from Section 5.5.4) are still noisy as Figure 6.1 shows, which suggests that the SAR is still imperfect for F0 modeling.

6.1.2 Weakness of linear AR dependency in SAR

Although the SAR can generate smooth F0 contours using the mean-based generation method, it only indicates the smoothness of the mean but not the model's confidence on the generated data, i.e., variance of the data.

To illustrate, we use the SAR with a single GMM component for explanation and assume $o_t \in \mathbb{R}$. As explained in the previous chapter, the SAR that models $\boldsymbol{o}_{1:T} \in \mathbb{R}^T$ can be interpreted as the combination of an invertible linear transformation $\boldsymbol{c}_{1:T} = \boldsymbol{A}\boldsymbol{o}_{1:T}$ and an RMDN that calculates $p(\boldsymbol{c}_{1:T}|\boldsymbol{x}_{1:T})$ given input features $\boldsymbol{x}_{1:T}$. Since the RMDN part assumes the temporal independence, we can write $p(\boldsymbol{c}_{1:T}|\boldsymbol{x}_{1:T}) = \mathcal{N}(\boldsymbol{\mu}_{1:T}, \mathrm{diag}(\boldsymbol{\sigma}_{1:T}))$. The rule of changing random variable then tells us that $p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:T}) = \mathcal{N}(\boldsymbol{A}^{-1}\boldsymbol{\mu}_{1:T}, \boldsymbol{A}^{-1^\top}\mathrm{diag}(\boldsymbol{\sigma}_{1:T})\boldsymbol{A}^{-1})$.

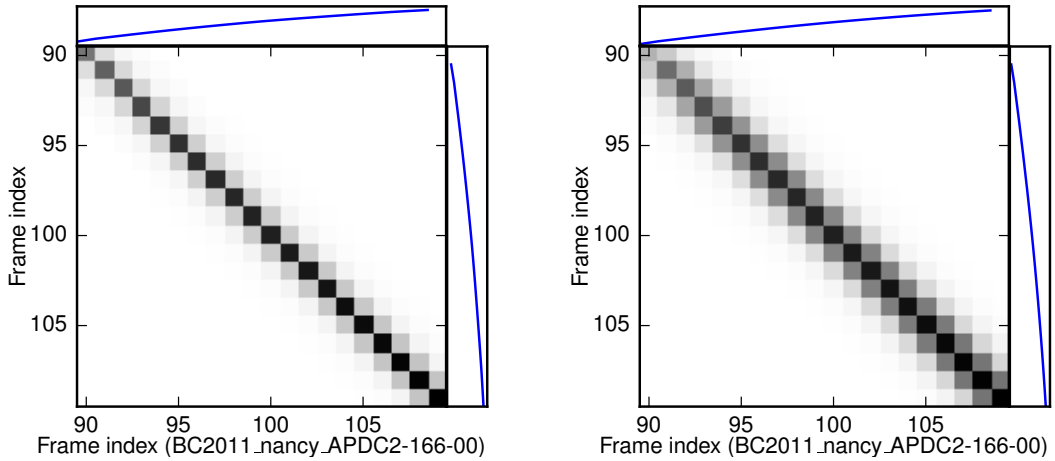
The performance of the SAR may be limited by the linear transformation. In the case of the normal SAR, the linear transformation matrix $\boldsymbol{A}$ is a lower-triangular Toeplitz matrix. Consider an extreme case where $K = T$. In this case, $\boldsymbol{A}$ and its inverse can be written as [150]:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -a_1 & 1 & 0 & 0 & 0 \\ -a_2 & -a_1 & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \\ -a_{T-1} & \cdots & -a_2 & -a_1 & 1 \end{bmatrix}, \boldsymbol{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ b_1 & 1 & 0 & 0 & 0 \\ b_2 & b_1 & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \\ b_{T-1} & \cdots & b_2 & b_1 & 1 \end{bmatrix}, \tag{6.1}$$

where

$$b_k = \begin{cases} a_1, & k = 1 \\ a_1 b_1 + a_2, & k = 2 \\ \sum_{j=1}^{k-2} a_{k-j} b_j + a_1 b_{k-1} + a_k, & k \in [2, T-1] \end{cases}, \tag{6.2}$$

As k increases, b_k may quickly decay if a_1 is smaller than 1 and a_k decays quickly



(a) SAR with $K = 2$ (b) SAR with $K = 6$

Figure 6.2: Covariance matrices given by SARs and the generated F0 contours (the blue lines) in a small segment around 20 frames.

too. This decaying would be more obvious in practice where K is smaller than T. For example, if we use $K = 1$, it is easy to show that $b_k = (a_1)^k$. In such a case, the off-diagonal elements of the covariance matrix ${\boldsymbol{A}^{-1}}^\top \mathrm{diag}(\boldsymbol{\sigma}_{1:T})\boldsymbol{A}^{-1}$ of $\boldsymbol{o}_{1:T}$ may also decay quickly.

The above behavior is actually observed from the experiments' results. We used the two SARs with $K = 2$ and $K = 6$ from Section 5.5.3 and plotted a small block of the covariance matrix ${\boldsymbol{A}^{-1}}^\top \mathrm{diag}(\boldsymbol{\sigma}_{1:T})\boldsymbol{A}^{-1}$ for each model in Figure 6.2. We can see that the off-diagonal elements decay quickly, which indicates that the temporal correlation described by the SAR is weak and local. Randomly sampled F0 contours from this model are thus noisy.

Can we design an AR transformation function so that the covariance matrix of $\boldsymbol{o}_{1:T}$ could have large off-diagonal values? The answer may be negative. As long as the linear AR dependency is applied, the transformation matrix $\boldsymbol{A}$ has a Toeplitz structure. Although the extended SAR could have been better than the normal SAR, experiments show that it cannot support random sampling either. Note that the extended SAR cannot be analyzed using the above method because the transformation is non-linear and the distribution of $\boldsymbol{o}_{1:T}$ cannot be written in an analytic formula.

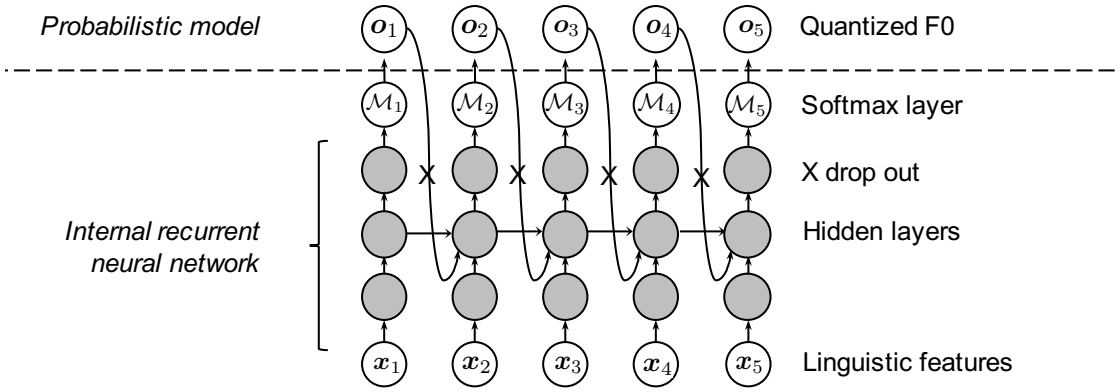


Figure 6.3: Example DAR. See quantized F0 and dropout in Section 6.3.

6.2 From SAR to DAR

6.2.1 Model definition

To further improve the AR model's capability, we may consider using non-linear and non-invertible transformations rather than the linear or invertible transformation in the SAR. One approach is to feed the target data of the previous frame back inside the neural network, for example, to a uni-directional recurrent layer at the current frame. The implementation is straightforward: concatenate $\boldsymbol{o}_{t-1}$ with the output of the previous hidden layer at the t-th frame and use the concatenated vector as an input to that recurrent layer. Because the feedback data are propagated by the recurrent layer, hidden features extracted from $\boldsymbol{o}_{1:t-1}$ can be propagated to the t-th frame. Therefore, the output of the internal RNN at the t-th frame, or consequently the parameter of $\boldsymbol{o}_t$'s distribution, can be computed as $\boldsymbol{\mathcal{M}}_t = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{o}_{1:t-1}, t)$. This means that the distribution of $\boldsymbol{o}_t$ depends on $\boldsymbol{o}_{1:t-1}$.

Since the AR dependency is non-linear and potentially beyond a local time window, this model is referred to as the *deep AR* model (DAR). The data-feedback path is referred to as the *feedback link*. The idea of the feedback link was first proposed in the Jordan network [151] and is now widely used for natural language or audio signal processing [152, 153, 154, 155]. One example of the DAR is plotted in Figure 6.3. Notice that the natural $\boldsymbol{o}_{t-1}$ is fed back during model training so that the network can learn the dependency of natural data. In the generation stage, the previously generated $\widehat{\boldsymbol{o}}_{t-1}$ or other statistics can be fed back.

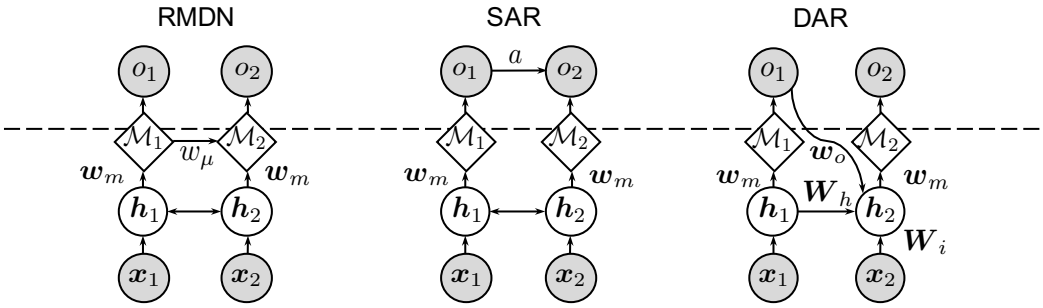


Figure 6.4: Toy DAR for comparison between RMDN, SAR, and DAR

6.2.2 Comparison between DAR and SAR

Although the network structure of DAR is simple, it implements the AR dependency in a more general manner. Let's use toy DAR in Figure 6.4 for explanation. We assume that all the layers use a linear activation function and set the bias to zero. Furthermore, the distribution of o_t is assumed to be a Gaussian distribution with a unit variance. Then, it is easy to know that in the DAR $\mathcal{M}_1 \triangleq \mu_1 = \boldsymbol{w}_m^\top \boldsymbol{h}_1$, $\mathcal{M}_2 \triangleq \mu_2 = \boldsymbol{w}_m^\top \boldsymbol{h}_2$, where $\boldsymbol{h}_2 = \boldsymbol{W}_h \boldsymbol{h}_1 + \boldsymbol{W}_i \boldsymbol{x}_2 + \boldsymbol{w}_o o_1$. If we define $\tilde{\mu}_2 \triangleq \boldsymbol{w}_m^\top (\boldsymbol{W}_h \boldsymbol{h}_1 + \boldsymbol{W}_i \boldsymbol{x}_2)$, then we can write down the distribution as

$$\begin{aligned} p(o_{1:2}|\boldsymbol{x}_{1:2}) =& \mathcal{N}(o_1; \mu_1, 1)\mathcal{N}(o_2; \tilde{\mu}_2 + \boldsymbol{w}_m^\top \boldsymbol{w}_o o_1, 1) \\ =& \frac{1}{2\pi} \exp(-\frac{(o_1 - \mu_1)^2}{2} - \frac{(o_2 - \tilde{\mu}_2 - \boldsymbol{w}_m^\top \boldsymbol{w}_o o_1)^2}{2}), \\ =& \frac{1}{2\pi} \exp(-\frac{1}{2}(\boldsymbol{o} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{o} - \boldsymbol{\mu})) \end{aligned} \tag{6.3}$$

where $\boldsymbol{o} = [o_1, o_2]^\top$, $\boldsymbol{\mu} = [\mu_1, \tilde{\mu}_2 + \boldsymbol{w}_m^\top \boldsymbol{w}_o \mu_1]^\top$, $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & \boldsymbol{w}_m^\top \boldsymbol{w}_o \\ \boldsymbol{w}_m^\top \boldsymbol{w}_o & 1 + (\boldsymbol{w}_m^\top \boldsymbol{w}_o)^2 \end{bmatrix}$. Table 6.1 compares the distribution calculated by the toy RMDN, the SAR, and the DAR. We can see that both SAR and DAR model the dependency between o_1 and o_2 because $\boldsymbol{\Sigma}$ is a full matrix.

In fact, the DAR is more general than the SAR. Remember that the above toy DAR uses linear activation functions. If the DAR uses a non-linear activation function $\sigma(\cdot)$, it computes $\boldsymbol{h}_2 = \sigma(\boldsymbol{W}_h \boldsymbol{h}_1 + \boldsymbol{W}_i \boldsymbol{x}_2 + \boldsymbol{w}_o o_1)$. Then, the mean of $p(o_2|\boldsymbol{x}_{1:2})$ becomes a non-linear function of o_1. Furthermore, if the data sequence has more than two frames, the distribution of o_t is also affected by $\boldsymbol{o}_{1:t-2}$ because hidden features extracted from $\boldsymbol{o}_{1:t-2}$ are propagated and stored in $\boldsymbol{h}_{t-1}$.

Table 6.1: Compare toy RMDN, SAR, and DAR in Figure 6.4. All models describe $p(o_{1:2}|\boldsymbol{x}_{1:2}) = 1/2\pi \exp(-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{o}-\boldsymbol{\mu}))$, where $\boldsymbol{o} = [o_1, o_2]$.

RMDN	SAR	DAR
$\boldsymbol{\mu} = [\mu_1, \tilde{\mu}_2 + \boldsymbol{w}_\mu \mu_1]$	$\boldsymbol{\mu} = [\mu_1, \mu_2 + a\mu_1]$	$\boldsymbol{\mu} = [\mu_1, \tilde{\mu}_2 + \boldsymbol{w}_m^\top \boldsymbol{w}_o \mu_1]$
$\tilde{\mu}_2 = \boldsymbol{h}_2^\top \boldsymbol{w}_m$	-	$\tilde{\mu}_2 = \boldsymbol{w}_m^\top (\boldsymbol{W}_h \boldsymbol{h}_1 + \boldsymbol{W}_i \boldsymbol{x}_2)$
$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & a \\ a & 1+a^2 \end{bmatrix}$	$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & \boldsymbol{w}_m^\top \boldsymbol{w}_o \\ \boldsymbol{w}_m^\top \boldsymbol{w}_o & 1 + (\boldsymbol{w}_m^\top \boldsymbol{w}_o)^2 \end{bmatrix}$

6.3 DAR for F0 modeling

The DAR theoretically shows better potential than the SAR. However, a few issues must be addressed before the DAR is used for F0 modeling.

6.3.1 Quantized F0 modeling

The first issue is the representation of the F0. Since the F0 value is unmeasurable in an unvoiced frame, one approach in the HMM-based SPSS is to define an F0 datum as $o_t \in \{\text{NULL}\} \cup \mathbb{R}$, where NULL is the symbol for the unvoiced frame. Then the MSD-HMM introduced in Section 2.3.3 can be used to model the [75]. Alternatively, artificial F0 values can be assigned to the unvoiced frames, after which the continuous F0 $o_t \in \mathbb{R}$ and unvoiced/voiced (U/V) state $v_t \in 0, 1$ can be modeled using a normal HMM [76]. Note that 'continuous' means that $o_t \in \mathbb{R}, \forall t \in \{1, \cdots, T\}$. In this thesis, we refer to $o_{1:T}$ as the interpolated continuous-valued F0 contour.

This continuous F0 modeling approach is widely used in neural F0 models, including the RNN [87] and the SAR. However, this approach performed poorly when the DAR is used as the model (see Section 6.4.2). The DAR may be affected by the artificial F0 values assigned in the unvoiced frames. Since these interpolated F0 values are artificial, they have a deterministic dependency to previous frames. However, the natural F0 values in the voiced frames have a somewhat stochastic dependency. The DAR may focus on the 'easy' temporal dependency in the unvoiced region, ignoring the temporal dependency in natural

F0 contours. Another reason could be the Gaussian distribution or GMM assumed by the model, which may be incompatible with the authentic F0 distribution conditioned on the F0 of previous steps.

Hence, the DAR requires an alternative method to represent both unvoiced and voiced frames without using F0 interpolation. We propose to quantize the F0 value of voiced frames and assign one additional symbol to the unvoiced frame. In this manner, both voiced and unvoiced frames can be represented as a set of categorical symbols. To quantize the F0 data, the first step is to map the original F0 onto a Mel scale, then the Mel-scale F0 is quantized into N levels. Finally, the quantized F0 of one frame can be encoded as a one-hot vector $\boldsymbol{o}_t = [o_{t,0}, o_{t,1}, \cdots, o_{t,N}]$, where $o_{t,j} \in \{0, 1\}$ and $||\boldsymbol{o}_t||_1 = 1$. If the frame is unvoiced, the 1st dimension of this one-hot vector is set to one, i.e., $\boldsymbol{o}_t = [1, 0, \cdots, 0]$.

The F0 quantization strategy is justifiable. First, quantized F0 may not necessarily influence the perceived speech quality because humans have difficulty differentiating between two sounds which have a small difference in frequency. This is known as the just-noticeable difference of pitch [21]. The second reason is that, given quantized F0 representation, the model can directly infer the categorical distribution of each F0 symbol without assuming a Gaussian distribution.

6.3.2 Hierarchical softmax for F0 modeling

The DAR may use a softmax output layer to calculate the probability for each of the quantized F0 and unvoiced symbol at each frame. However, a normal softmax layer is not the best choice because the quantity of unvoiced data in the corpus is much larger than that of any other quantized F0 symbol. We thus suggest using a hierarchical softmax layer [156].

Suppose that the hot dimension of $\boldsymbol{o}_t$ is indexed by j, then the PMF w.r.t. $\boldsymbol{o}_t$ can be defined as $P(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}, \boldsymbol{x}_{1:T}; \boldsymbol{\Theta}) \triangleq P_t(J = j; \boldsymbol{\Theta})$, where

$$P_t(J = j; \boldsymbol{\Theta}) = \begin{cases} \dfrac{e^{h_{t,0}}}{1 + e^{h_{t,0}}}, & j = 0 \\ \dfrac{1}{1 + e^{h_{t,0}}} \dfrac{e^{h_{t,j}}}{\sum_{k=1}^{N} e^{h_{t,k}}}, & j \in [1, N] \end{cases}. \tag{6.4}$$

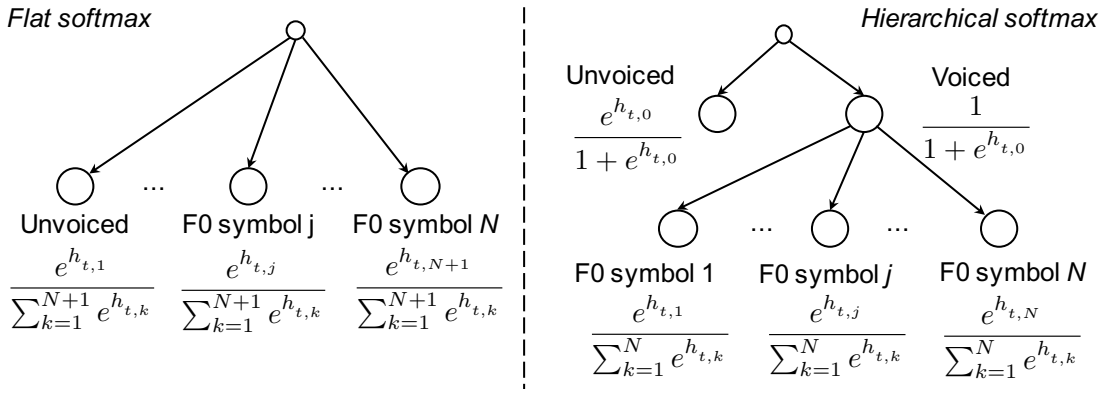


Figure 6.5: Flat (left) and hierarchical (right) softmax for quantized F0 modeling

Here, $h_{t,j}$ is the j-th dimension of the input vector $\boldsymbol{h}_t$ to the hierarchical softmax layer. This $\boldsymbol{h}_t$ is calculated from the network given feedback data $\boldsymbol{o}_{1:t-1}$ and linguistic features $\boldsymbol{x}_{1:T}$. Accordingly, it can be written that $P_t(J=j) = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{o}_{1:t-1}, t, j)$.

As Equation (6.4) shows, the first hierarchical level uses a sigmoid function to compute a probability of being unvoiced, i.e., $P_t(\text{unvoiced}) \triangleq P_t(J=0) = \frac{e^{h_{t,0}}}{1+e^{h_{t,0}}}$. The second level uses a normal softmax function to compute a conditional probability of each quantized F0 symbol given a voiced state, i.e., $P_t(J = j|\text{voiced}) = \frac{e^{h_{t,j}}}{\sum_{k=1}^{N} e^{h_{t,k}}}, j \in [1, N]$. Based on Equation (6.4), the network can be trained by maximizing the likelihood or equivalently minimizing the cross-entropy.

Once the training process is finished, the DAR can be used for F0 generation. First, if $P_t(J=0) > 0.5$, the t-th frame is classified as being unvoiced. Otherwise, this frame will be voiced and will be assigned an F0 value $\widehat{f}_t$ [1]. Although the DAR models quantized F0 data, it can generate continuous-valued F0 using a mean-based generation method. Suppose $\{v_1, \cdots, v_N\}$ denotes the F0 values of the N quantization levels, such as the center of each quantization interval. Since the F0 symbols for voiced frames are quantized F0 indexes, $\widehat{f}_t$ can be generated by taking the expected value as

$$\widehat{f}_t = \sum_{j=1}^{N} v_j P_t(J=j|\text{voiced}). \tag{6.5}$$

[1]To differentiate it from the quantized F0 one-hot vector $\boldsymbol{o}_t$, we use $\widehat{f}_t$ to denote the continuous-valued F0 generated by the DAR.

This method is the *mean-based generation* method for the quantized F0. After $\widehat{f}_t$ is generated, a vector of probability $[P_t(J=0), P_t(J=1), \cdots, P_t(J=N)]$ can be fed back to the next frame.

Remember that random-sampling can be used to test the model's capability. In this case, the F0 value $\widehat{f}_t$ can be sampled by

$$\widehat{f}_t = v_j, \quad \text{where} \quad j \sim P_t(J=j|\text{voiced}), \quad j \in \{1, \cdots, N\}. \tag{6.6}$$

Given the sampled value j, a one-hot vector $\widehat{\boldsymbol{o}}_t$ with the j-th dimension turned on is fed back to the next frame.

6.3.3 Exposure bias and data dropout

As aforementioned, the DAR feeds the natural F0 back to the network during model training. This is known as the teacher-forced training [157]. However, because the natural $\boldsymbol{o}_{t-1}$ usually has a similar value to $\boldsymbol{o}_t$, a trained DAR may only rely on the feedback F0 data while ignoring the input linguistic features $\boldsymbol{x}_{1:T}$. This behavior is unwanted because an F0 model should also use the linguistic features. Furthermore, while natural F0 data are propagated through the feedback links in the training stage, generated F0 data are fed back during generation. Because the distribution of generated F0 may not be identical to the natural one, the model may suffer from the problem called *exposure bias* [158]. Consequently, generation errors in the previous frames may be propagated to the next frame, which makes the entire generated F0 contour erratic.

We propose a *data dropout* strategy to alleviate the above problems. With a probability P_d, this strategy randomly sets the feedback F0 data to zero in both training and generation stages. It consequently forces the DAR to focus on the linguistic features. This strategy is similar to the idea of weakening the AR decoder for lossy variational auto-encoders [159, 155]. It will also alleviate the exposure bias as the model relies more on the linguistic features that are given with the same TTS front-end for both training and generation.

Some readers may suggest using a technique called schedule sampling. However, this technique may force the model to ignore the temporal dependency of the natural data sequence [160].

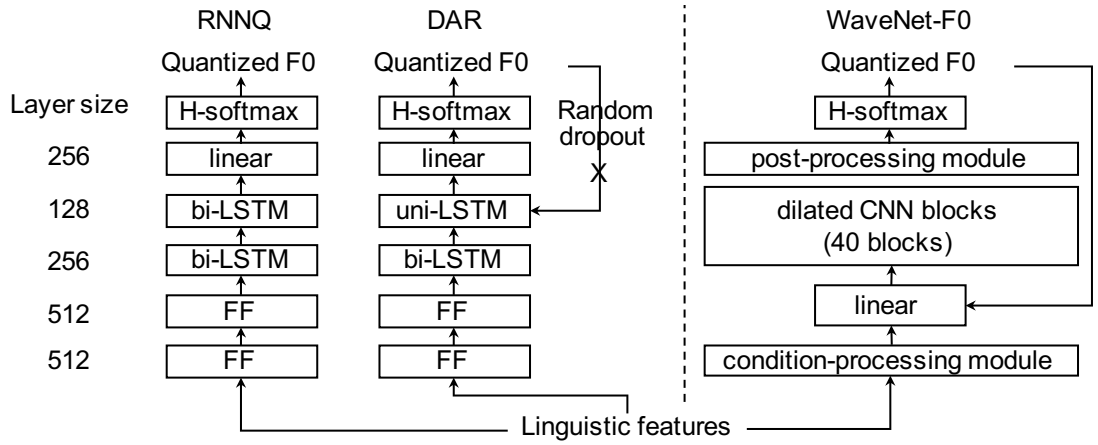


Figure 6.6: Network structures of quantized F0 models. 'FF', 'bi-LSTM', and 'uni-LSTM' denotes feedforward layer, bi- and uni-directional LSTM layer, respectively.

6.4 Experiments

6.4.1 Data and configuration

The experiment used the Japanese corpus listed in Appendix A.2. The input and output features were the same as those in the previous chapter. The first group of experimental models were the `RNN`, `RMDN`, `SAR`, and `eSAR` built in Section 5.5. These models were trained on interpolated continuous-valued F0 data. The second group included the `DAR` and a reference model `RNNQ`. Both `DAR` and `RNNQ` modeled quantized F0, but `RNNQ` did not use the feedback loop. Both models were trained using stochastic gradient descent (learning rate 1e-05) with early stopping. Then, they were further tuned using AdaGrad (learning rate 0.001) with early stopping.

WaveNet [59] was included as another reference model because it implements the deep AR dependency using dilated convolutional layers rather than recurrent ones. The structure and training recipe of the `WaveNet-F0` were similar to those of the WaveNet vocoder [149], with the dilation sizes of every 10 CNN blocks as $\{1, 2, 4, \cdots, 512\}$. However, the network used a hierarchical softmax output layer and modeled quantized F0. Additionally, the network operated frame by frame.

For quantized F0 models, the log-scale F0 were quantized into 255 levels between 66 and 529 on the Mel scale. The number 66 was equal to the minimum Mel-scale F0 value in the corpus. The number 529 was computed with $m + 3\sigma$,

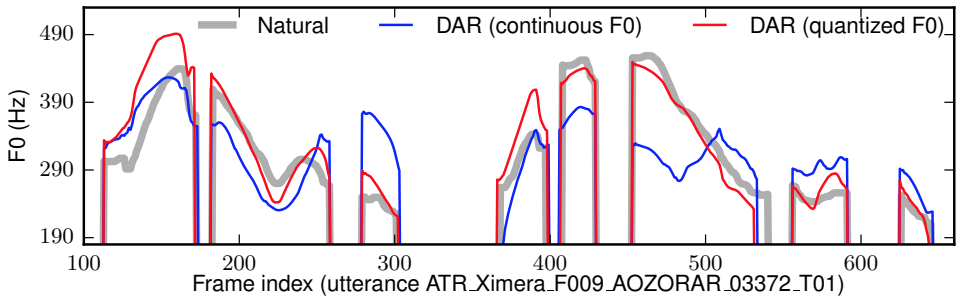


Figure 6.7: Generated F0 contours by DAR trained on continuous and quantized F0 data (mean-based generation), using the Japanese corpus.

Table 6.2: Objective evaluation of pilot test I (mean-based generation, Japanese corpus). The baseline RMDN is shown for reference.

	RMSE	CORR	U/V error
F0 quantization	01.19	0.999	0.00%
`RMDN` (continuous F0)	28.32	0.897	3.85%
`DAR` (quantized F0)	32.04	0.881	3.82%
`DAR` (continuous F0)	64.10	0.588	5.16%

where $m = 342.4$ and $\sigma = 62.2$ are the mean and standard deviation of Mel-scale F0 over the corpus. The number of quantization levels was decided from an analysis-by-synthesis test, which found that using 255 levels was sufficient to avoid perceptible F0 'quantization noise'. The quantized F0 and unvoiced symbol were encoded as a one-hot vector $\boldsymbol{o}_t \in \{0, 1\}^{256}$ for each frame. The F0 delta and delta-delta components were not used. For reference, after we recovered the F0 contours from the quantized F0 data, the error caused by F0 quantization was calculated and listed in the first row of Table 6.2.

6.4.2 Pilot test I: continuous versus quantized F0

The first test evaluates effectiveness of quantized F0 representation by comparing two DAR-based models: the first model `DAR` was trained on the quantized F0 without using dropout; the second model was similar to `DAR` but trained on interpolated continuous-valued F0 data. Accordingly, the second model uses a linear output layer rather than softmax.

Table 6.3: Objective results of DAR using different types of output softmax layer on the Japanese corpus (mean-based generation).

P_d	Softmax type	**U/V**	**V→U**	**U→V**	**RMSE**	**CORR**
0.75	Normal	5.31%	4.57%	0.74%	28.64	0.900
	Hierarchical	3.35%	1.66%	1.69%	26.52	0.909
0.50	Normal	4.87%	4.04%	0.83%	28.41	0.900
	Hierarchical	3.46%	1.86%	1.60%	28.30	0.903
0.25	Normal	4.55%	3.46%	1.09%	30.09	0.890
	Hierarchical	3.62%	1.86%	1.76%	29.70	0.896
0.00	Normal	4.43%	2.98%	1.45%	31.92	0.881
	Hierarchical	3.82%	1.73%	2.09%	32.04	0.881

The objective results are listed in Table 6.2. They suggest that DAR performed poorly when it was trained on the interpolated continuous-valued F0 data. This degradation can be further illustrated by Figure 6.7. This result may be due to the artificial F0 curves in the unvoiced frames. F0 values in the interpolated curves have deterministic dependency to their previous frames. However, the temporal dependency in natural F0 contours may be more complex. As the proportion of unvoiced frames (including silence) is about 50% in this Japanese corpus, a DAR may be well trained to maximize the likelihood over the simple artificial F0 curves rather than the highly varied natural F0 contours.

6.4.3 Pilot test II: hierarchical versus normal softmax

Since the first pilot test suggests that the DAR should use quantized F0, this test tries to answer which type of softmax should be used to model the quantized F0 data. For this test, we trained another DAR with the same structure as `DAR` except the output softmax layer type. Different dropout rates $P_d = \{0.0, 0.25, 0.5, 0.75\}$ were used for both DAR models.

The results are listed in Table 6.3. This table also lists the error rate of classifying unvoiced frames as voiced (U→V) and the error in the other way round (V→U). According to the results, both hierarchical and normal softmax achieved similar RMSE and CORR scores for each case of P_d. However, the U/V error shows interesting results. Using a normal softmax layer, the U→V is lower than

that of the hierarchical case while the V→U is much higher. Compared with the hierarchical case, the normal softmax acquired a worse overall U/V error.

The above results are related to the different modeling strategies used by the hierarchical and normal softmax output layers. Suppose there are N quantized F0 levels and 1 level for the unvoiced symbol. Given a sequence of quantized F0 data $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$, we can write down the log-likelihood $\mathcal{L}(\boldsymbol{\Theta})$ of the DAR with a normal softmax layer as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}) &= \sum_{t=1}^{T}\sum_{j=0}^{N} \delta_{\boldsymbol{o}_t,j} \log P_t(j;\boldsymbol{\Theta}) \\ &= \sum_{t=1}^{T}\left[\delta_{\boldsymbol{o}_t,0} \log P_t(\mathrm{U};\boldsymbol{\Theta}) + \sum_{j=1}^{N} \delta_{\boldsymbol{o}_t,j} \log P_t(j;\boldsymbol{\Theta})\right],\end{aligned} \tag{6.7}$$

where $P_t(\mathrm{U};\boldsymbol{\Theta})$ is the probability of being unvoiced, $P_t(j;\boldsymbol{\Theta})$ is the probability of the j-th quantized F0 level, $j \in [1, N]$, and $\delta_{\boldsymbol{o}_t,j}$ is the indictor function whose value is 1 only when $\boldsymbol{o}_t$ denotes the j-th level. Notice that in this case

$$P_t(\mathrm{U};\boldsymbol{\Theta}) + \sum_{j=1}^{N} P_t(j;\boldsymbol{\Theta}) = 1. \tag{6.8}$$

On the contrary, the log-likelihood of the DAR using a hierarchical softmax is:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}) &= \sum_{t=1}^{T}\left[\delta_{\boldsymbol{o}_t,0} \log P_t(\mathrm{U};\boldsymbol{\Theta}) + \sum_{j=1}^{N} \delta_{\boldsymbol{o}_t,j} \log[P_t(j|\mathrm{V};\boldsymbol{\Theta})P_t(\mathrm{V};\boldsymbol{\Theta})]\right] \\ &= \sum_{t=1}^{T}\left[\delta_{\boldsymbol{o}_t,0} \log P_t(\mathrm{U};\boldsymbol{\Theta}) + \sum_{j=1}^{N} \delta_{\boldsymbol{o}_t,j} \log P_t(j|\mathrm{V};\boldsymbol{\Theta})\right. \\ &\qquad \left. + \sum_{j=1}^{N} \delta_{\boldsymbol{o}_t,j} \log P_t(\mathrm{V};\boldsymbol{\Theta})\right]\end{aligned}, \tag{6.9}$$

where $P_t(j|\mathrm{V};\boldsymbol{\Theta}), j \in [1, N]$ is the probability of the j-th F0 level given that frame is voiced, $P_t(\mathrm{V};\boldsymbol{\Theta})$ is the probability of being voiced, and $P_t(\mathrm{U};\boldsymbol{\Theta})$ is the probability of being unvoiced. In this case

$$P_t(\mathrm{V};\boldsymbol{\Theta}) + P_t(\mathrm{U};\boldsymbol{\Theta}) = 1, \quad \sum_{j=1}^{N} P_t(j|\mathrm{V};\boldsymbol{\Theta}) = 1. \tag{6.10}$$

The constraints in Equation (6.8) and (6.10) may have caused the different results acquired by the flat and hierarchical softmax layers. In the training data, about 50% of the data frames is unvoiced (including 28% silent frames), but the ratio of a single quantized F0 level is less than 0.6%. For a normal softmax layer, this unbalanced data distribution means that most of the model likelihood is contributed by $P_t(\mathrm{U};\boldsymbol{\Theta})$. Thus, the model can be trained to achieve a high likelihood by simply over-estimating $P_t(\mathrm{U};\boldsymbol{\Theta})$. Consequently, the model had a low U→V error. However, the over-estimated $P_t(\mathrm{U};\boldsymbol{\Theta})$ led to a under-estimated probability for quantized F0 levels because of the constraint in Equation (6.8). This resulted in a high V→U error.

In the case of the hierarchical softmax, the model will not over-estimate $P_t(\mathrm{U};\boldsymbol{\Theta})$. While $P_t(\mathrm{U};\boldsymbol{\Theta})$ was computed over the unvoiced data, another term $P_t(\mathrm{V};\boldsymbol{\Theta})$ was computed over the voiced data. Because $P_t(\mathrm{U};\boldsymbol{\Theta}) + P_t(\mathrm{V};\boldsymbol{\Theta}) = 1$, and the quantity of voiced and unvoiced data is similar, the model cannot achieve a higher likelihood by over-estimating $P_t(\mathrm{V};\boldsymbol{\Theta})$ or $P_t(\mathrm{U};\boldsymbol{\Theta})$. It can achieve the highest likelihood only when $P_t(\mathrm{V};\boldsymbol{\Theta})$ and $P_t(\mathrm{U};\boldsymbol{\Theta})$ are consistent with the training data distribution. Therefore, using a hierarchical softmax layer achieved balanced V→U and U→V scores and furthermore a lower overall U/V error.

6.4.4 Pilot test III: effectiveness of data dropout

This pilot test compares the performance of `DAR` when it was trained using dropout rates $P_d = \{0.0, 0.25, 0.5, 0.75\}$. `RNNQ` was also included in this test because it can be treated as `DAR` with $P_d = 1.0$. All the models used the mean-based generation method. The results listed in Table 6.4 indicate that `DAR` with $P_d > 0$ acquired better RMSE and CORR scores than `DAR` with $P_d = 0.0$, which shows the effectiveness of using dropout on `DAR`.

As Section 6.3.3 argues, a potential problem of using `DAR` is the exposure bias. This is supported by the results in Table 6.4. First, `DAR` with $P_d = 0.0$ achieved a higher likelihood than other cases on the validation set. Because the likelihood on the validation set was evaluated given the natural F0 data for feedback, it suggests that, when the natural F0 data were fed back, `DAR` better depicted the F0 data without using dropout. However, on the test set in which the model had to feed

Table 6.4: Objective results of DAR and RNNQ on the Japanese corpus (mean-based generation). For reference, GV of natural F0 is around 61. DAR's scores were copied from the 'Hierarchical' ones in Table 6.3.

	P_d	Log-likelihood (validation set)	**RMSE**	**CORR**	**U/V**	**GV**
RNNQ	-	-2443.8	26.77	0.907	5.74%	56.4
DAR	0.75	-1595.1	26.52	0.909	3.35%	57.8
	0.50	-1156.3	28.30	0.903	3.46%	61.5
	0.25	-943.6	29.70	0.896	3.62%	62.7
	0.00	-839.7	32.04	0.881	3.82%	64.4

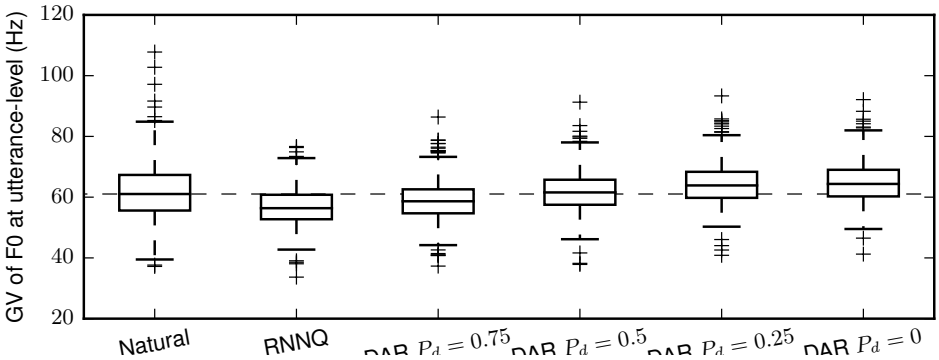


Figure 6.8: GV of generated F0 from DAR and RNNQ (mean-based generation). P_d denotes the probability to use dropout in DAR.

back the generated F0, DAR's performance degraded in terms of RMSE and CORR. In fact, DAR with $P_d = 0.0$ was very confident about the inferred F0 distribution. This confidence can be demonstrated by the small variance of the distribution in Figure 6.9. However, a small variance does not necessarily mean a small bias. When DAR used its output as feedback data on the test set, a slight difference between the generated and natural F0 may be propagated to the following frames and accumulated.

Dropout may provide DAR with the freedom to adjust the bias and variance. As Table 6.4 and Figure 6.9 show, a larger P_d led to a larger variance of the F0 distribution and consequently a smaller likelihood on the validation set. However, it improved the RMSE and CORR on the test set. Particularly, DAR with $P_d = 0.75$ achieved the best performance in terms of RMSE and CORR.

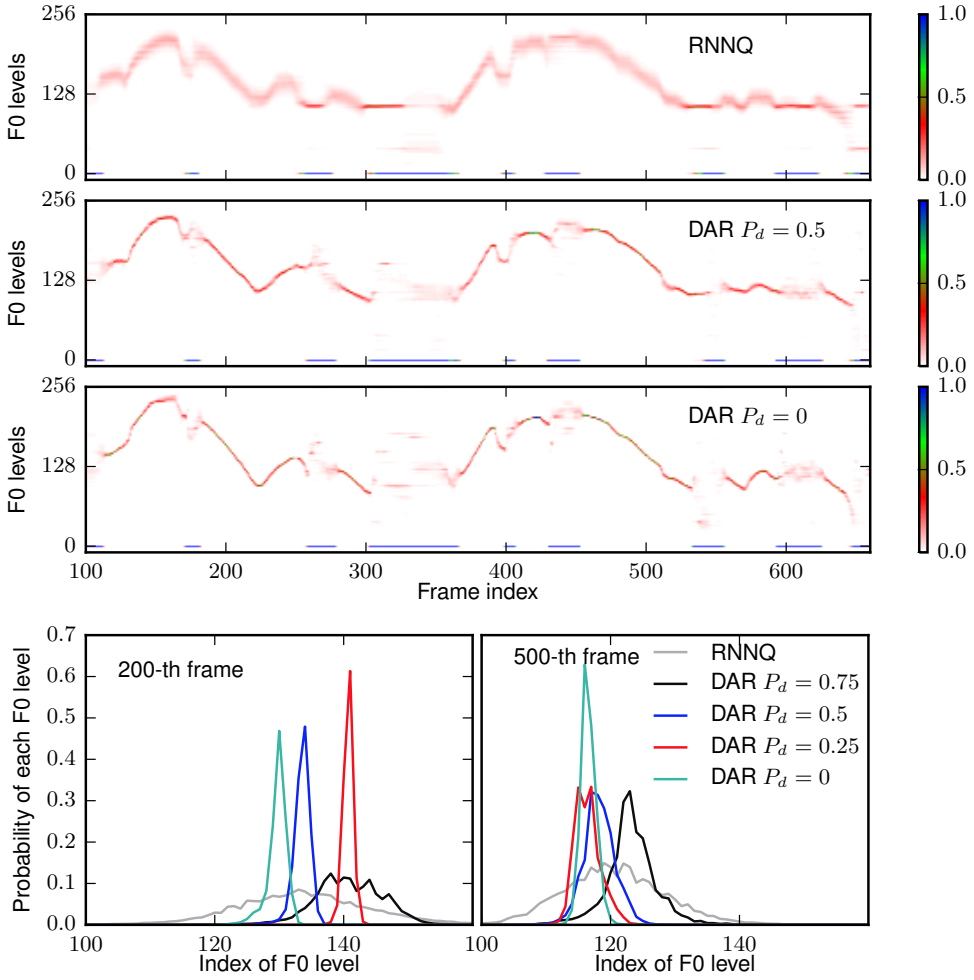


Figure 6.9: Top 3 rows: inferred F0 probability distribution ($P(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}, \boldsymbol{x}_{1:T}; \boldsymbol{\Theta})$ in Equation (6.4)) for one test utterance on the Japanese corpus. Bottom row: inferred F0 probability for two frames in the test utterance.

The rate of dropout should be selected carefully. An appropriate choice should strike a balance between the bias and variance. Another concern indicated in Figure 6.8 is that intensive dropout may reduce the GV of the generated F0 contours and make them over-smoothed. Because the over-smoothing effect may be more harmful to the perception of the pitch than the degraded value of RMSE or CORR, `DAR` with $P_d = 0.5$ was selected for the subjective evaluation discussed in the next section.

Table 6.5: Objective results on the Japanese data. DAR used $P_d = 0.5$. GV of natural F0 is around 61.

	RMSE	CORR	U/V	GV
RNN	29.31	0.894	3.26%	51.4
RMDN	28.32	0.897	3.85%	54.2
SAR	34.57	0.898	3.85%	71.8
eSAR	29.74	0.897	3.70%	63.4
WaveNet-F0	28.05	0.903	3.52%	60.4
DAR	28.30	0.903	3.46%	61.5

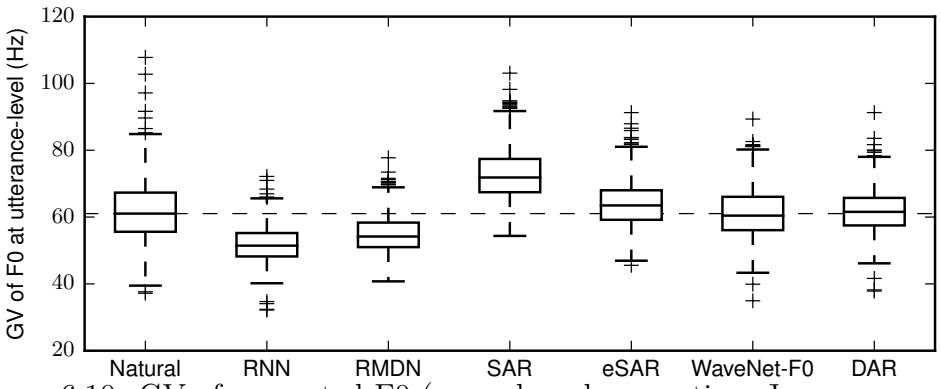


Figure 6.10: GV of generated F0 (mean-based generation, Japanese corpus)

6.4.5 Evaluation of DAR against other F0 models

This experiment compares the DAR against other F0 models. Based on the pilot tests, we decided to use the DAR with a hierarchical softmax output layer and trained it on quantized F0 with a dropout rate $P_d = 0.5$. Other F0 models, namely RNN, RMDN, SAR, and eSAR, were trained on continuous F0 data as the experiment in Section 5.5.4 explained. We just directly used the models trained in that Section.

The results are listed in Table 6.5. Compared with SAR and other baselines, DAR performed slightly better in terms of RMSE and CORR. As Figure 6.11 shows, the generated F0 contours from DAR were sufficiently smooth, even though the model was trained given quantized F0 data. Furthremore, these generated F0 contours are not over-smoothed. For example, Table 6.5 shows that DAR with $P_d = 0.5$ acquired a GV score that was closer to the natural F0. In particular, from Figure 6.11, the generated F0 contour from DAR with $P_d = 0.5$ had a larger dynamic range than those from the baseline RNN and RMDN.

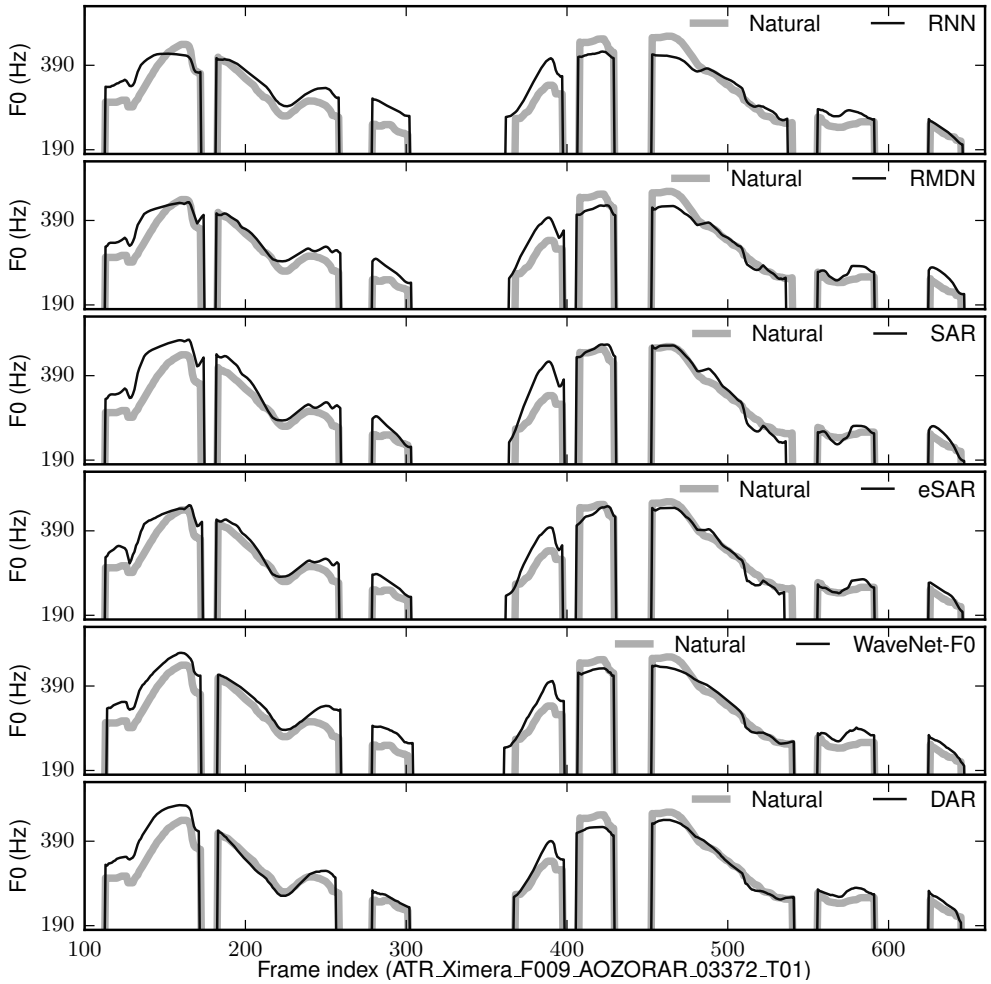
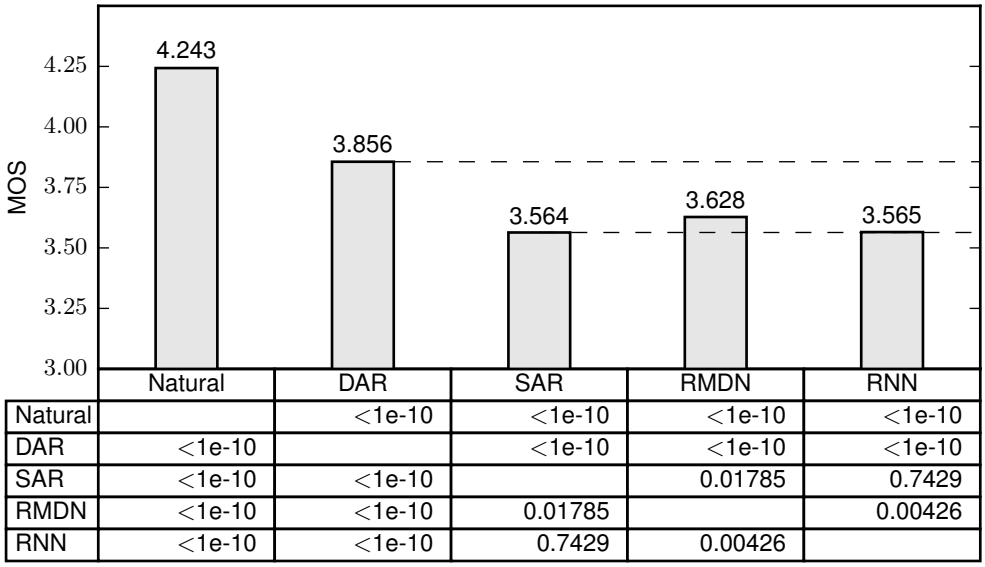


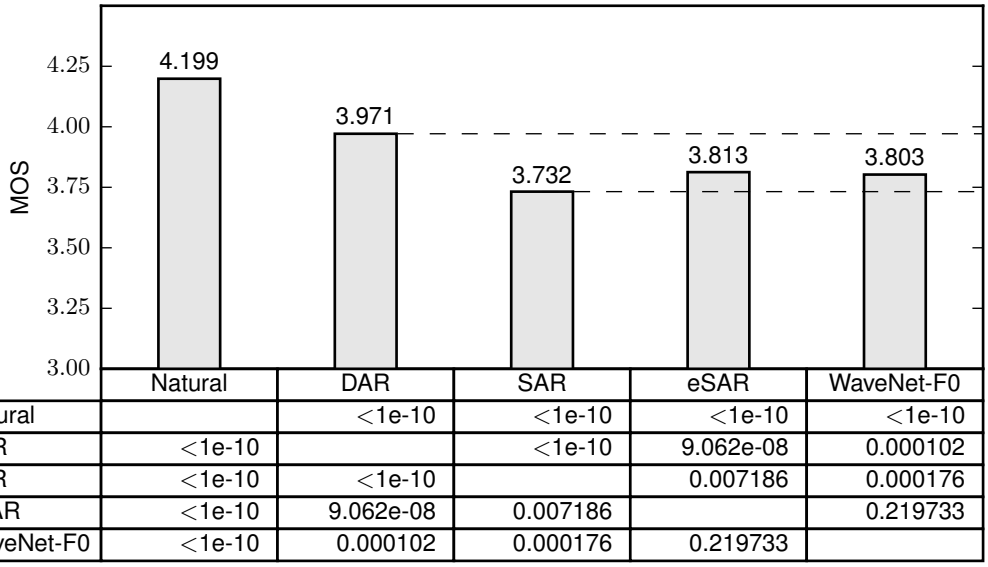
Figure 6.11: Generated F0 (mean-based generation, Japanese corpus)

Due to the historical reason, two mean-opinion-score (MOS) tests were conducted to compare the experimental models. For both tests, native Japanese listeners were paid to conduct the test, in which each listener listened to the samples and evaluated the sample quality in terms of intonation using a scale from 1 (unnatural) to 5 (natural). In each testing round, one vocoded speech sample and the synthetic samples from each model were played in a randomly shuffled order. All the samples were created by the WORLD vocoder [18] using natural spectral features. In total, 3000 and 1702 testing rounds were conducted in the first and the second test, respectively.

The results are shown in Figure 6.12, where the figure above corresponds to



	Natural	DAR	SAR	RMDN	RNN
Natural		<1e-10	<1e-10	<1e-10	<1e-10
DAR	<1e-10		<1e-10	<1e-10	<1e-10
SAR	<1e-10	<1e-10		0.01785	0.7429
RMDN	<1e-10	<1e-10	0.01785		0.00426
RNN	<1e-10	<1e-10	0.7429	0.00426	



	Natural	DAR	SAR	eSAR	WaveNet-F0
Natural		<1e-10	<1e-10	<1e-10	<1e-10
DAR	<1e-10		<1e-10	9.062e-08	0.000102
SAR	<1e-10	<1e-10		0.007186	0.000176
eSAR	<1e-10	9.062e-08	0.007186		0.219733
WaveNet-F0	<1e-10	0.000102	0.000176	0.219733	

Figure 6.12: MOS score (mean-based generation, Japanese corpus) and p-value of Mann-Whitney U test between each pair of experimental models

the first MOS test and the bottom one to the second test. Although DAR was still worse than the natural F0 (NAT), it outperformed other models. Results of two-sided Mann-Whitney U tests demonstrated that the difference between DAR and other F0 models was statistically significant ($p < 0.01$). One main reason for DAR's performance may be that the generated F0 contours were sufficiently smooth but not over-smoothed. Compared with SAR, DAR's output did not contain the under-smoothed curves that turned out to be perceptually harmful to the synthetic

speech. Meanwhile, generated F0 contours from DAR had a proper dynamic range and sounded less boring.

Although we cannot directly compare the results of the two tests, we can infer that RMDN, eSAR, and WaveNet-F0 achieved similar performances while SAR and RNN performed worse. Although WaveNet-F0 achieved similar objective results to DAR, it lagged behind DAR in the MOS test. It is possible that the network structure and configuration of dilation borrowed from the WaveNet-vocoder may be suboptimal for F0 modeling. However, it is not easy to find the best configuration of WaveNet-F0 because the number of hyper-parameters is quite large.

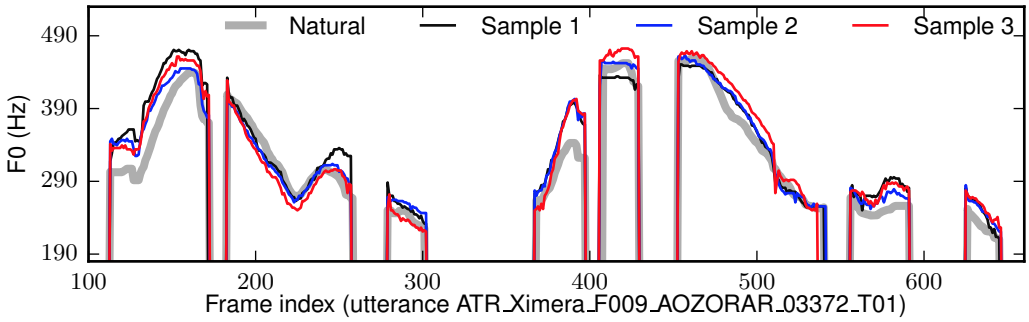
6.5 Random sampling from DAR

The results in the previous section have shown DAR's capability for F0 modeling when the mean-based generation method was used. This section tests DAR (with $P_d = 0.5$) using the random-sampling-based generation method.

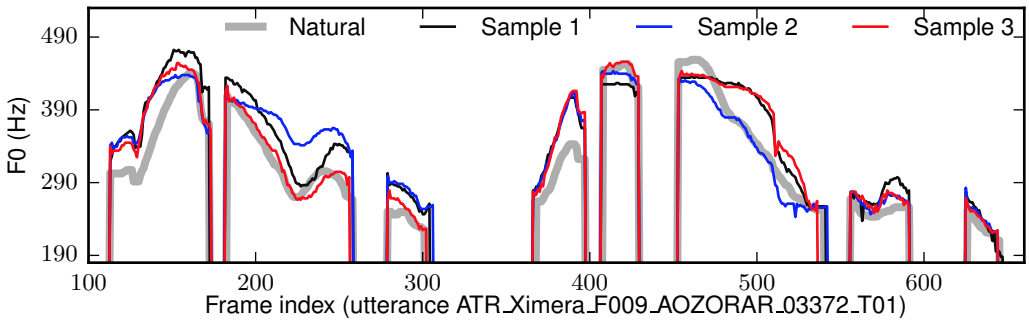
Random sampling results

First, F0 contours were randomly sampled from DAR for three rounds, where each round used a different random seed for sampling. Figure 6.13 (a) plots the three randomly sampled F0 contours for one test-set sentence. Interestingly, the randomly sampled F0 contours were smooth and much better than the sampled output of RMDN and SAR (Figure 6.1 and 5.1). Furthermore, these sampled F0 contours were quite close to the natural one. We found that the CORR on the test set was $0.887, 0.890, 0.889$ for the three sampling rounds, which is quite close to the mean-based generation results. Note that the sampled F0 contours in Figure 6.13 (a) contained small spikes due to the random sampling process. However, they are barely perceptible. Generally, these results indicate that DAR is better at F0 modeling than SAR and RMDN.

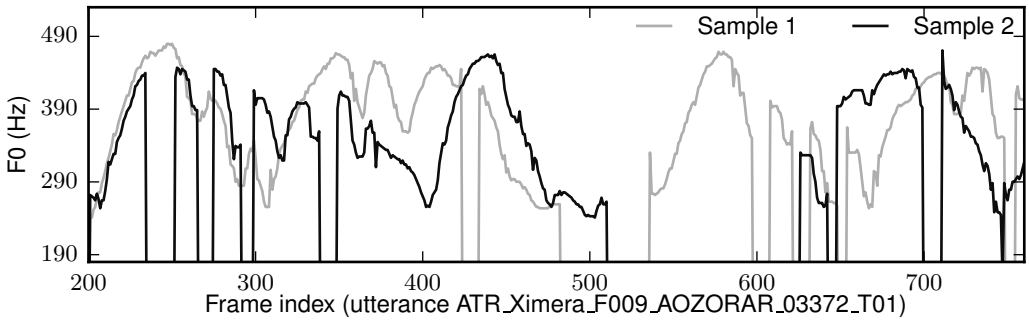
It is not surprising to see that smooth F0 contours can be sampled from DAR. As Figure 6.9 has shown, the F0 distribution inferred by DAR had a sharp mode, and this mode moved slowly across frames. Thus, it is highly possible to sample a smooth F0 contour. However, it is surprising that the sampled F0 contours were



(a) DAR trained with full input linguistic features



(b) DAR trained without pitch accent linguistic features



(c) DAR trained without any linguistic feature

Figure 6.13: Results of random sampling on DAR on Japanese corpus.

quite similar to the natural one. Despite the detailed differences, the sampled and natural F0 contours were perceived to be quite similar in terms of intonation.

Impact of linguistic features on random sampling

However, when we trained the DAR on the English corpus, the randomly sampled F0 contours were perceived to have different pitch accents. Some sampled English

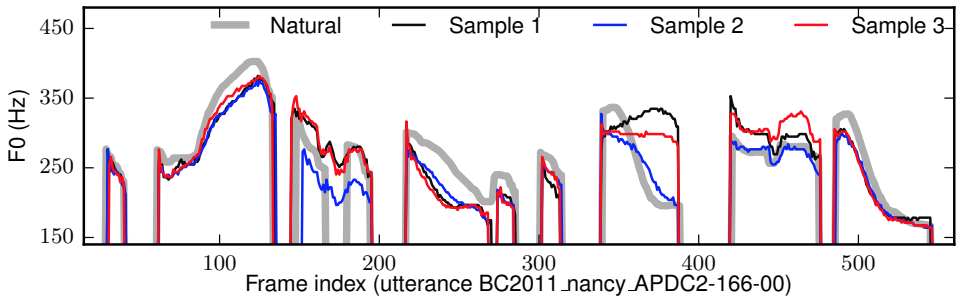


Figure 6.14: Randomly sampled F0 from DAR with full input linguistic features on English corpus.

F0 contours are plotted in Figure 6.14. We hypothesized that this is due to the characteristics of Japanese. In the case of reading speech, the F0 contour of an utterance may be sufficiently specified by the Japanese pitch accents. Although the Japanese pitch accents interact with each other in an utterance [161], they can be somewhat determined by a lexicon. Therefore, in the TTS system, the input linguistic features given by the front-end may be sufficiently informative for DAR to determine the shape of generated F0 contours; thus leaving less space for sampling F0 contours with varied shapes.

To verify the hypothesis, another DAR was trained after linguistic features related to the Japanese pitch accent were removed. The sampled F0 contours are shown in Figure 6.13 (b). Interestingly, Figure 6.13 (b) shows that sampled contours occasionally deviated from the natural F0 contour, e.g., after the 200th and 500th frames. According to the native Japanese speakers, those F0 curves were perceived as either unnatural segments or different accents from the natural ones. These results indicate that DAR's performance in this experiment benefited from the relatively accurate input linguistic features.

We believe that these results are somewhat consistent with the results on the English corpus. In English TTS systems, what we can obtain from a lexicon is the lexical stress. However, it does not sufficiently explain an F0 contour. For specifying the general shape of the F0 contour in English, we need to have accurate English pitch-accent information; however, the English pitch accents cannot be perfectly inferred from the text [43]. Therefore, it is thought that the degrees of informativeness of the English input linguistic features allowed the English DAR to

generate varied F0 contours through sampling.

Finally, it is interesting to see what would happen if no linguistic feature is provided for DAR. This was implemented by setting the input sequence $\boldsymbol{x}_{1:T}$ to zero during both training and generation. Two samples randomly drawn from such a DAR are shown in Figure 6.13 (c). Although these two F0 contours were nonsense, they were smooth and resembled the high-low movement of natural F0 contours. As RMDN and SAR could not generate smooth random samples under the same condition, this result provides further evidence of DAR's ability to model the temporal correlation of F0 contours.

6.6 Summary

This chapter follows the topic of the previous chapter to address **Issue 2** about the temporal dependency modeling in neural F0 models. At the beginning, this chapter used both theoretical and empirical evidence to show that the SAR is limited by the linear or shallow AR dependency function. Accordingly, this chapter proposed the DAR that uses the non-linear non-invertible transformation in the neural network to model the AR dependency. The basic idea is to feed back the previous F0 observation as the input to a uni-directional recurrent layer. To make the model practical, we also proposed quantized F0 representation, a hierarchical softmax output layer, and a data dropout strategy to train the DAR.

A few experiments were conducted to show the effectiveness of the proposed model and techniques. Particularly, experiments showed that the DAR generated smooth and quite natural F0 contours even if random sampling was used. This result has not been achieved with other F0 models. Furthermore, when the standard mean-based generation method was used, the DAR generated F0 contours with an appropriate dynamic range and high accuracy. It also outperformed the previous F0 models in a subjective evaluation test. Thus, the proposed DAR is a good solution to address model the temporal dependency of F0 contours for TTS.

7

Variational-auto-encoder-based F0 model

Given the results of the DAR in the previous chapter, this chapter switches to the third issue of neural F0 modeling, i.e., **Issue 3: Can linguistic features be processed more efficiently?** In Section 3.6, we mentioned that a baseline neural F0 model (and the DAR) has to duplicate the linguistic features to the frame level and process the features frame-by-frame. This processing strategy influences the efficiency of the model and may further affect its performance.

This chapter proposes the variational auto-encoder (VAE)-based F0 modeling framework to alleviate the above issue. In Section 7.1, the above issue is explained in more technical detail. Section 7.2 then explains the framework of the VAE-based F0 modeling and the practical implementation based on a special VAE called vector-quantization VAE (VQVAE). The DAR is also merged into the framework. After that, Section 7.3 conducts empirical evaluations on the VAE-based F0 models.

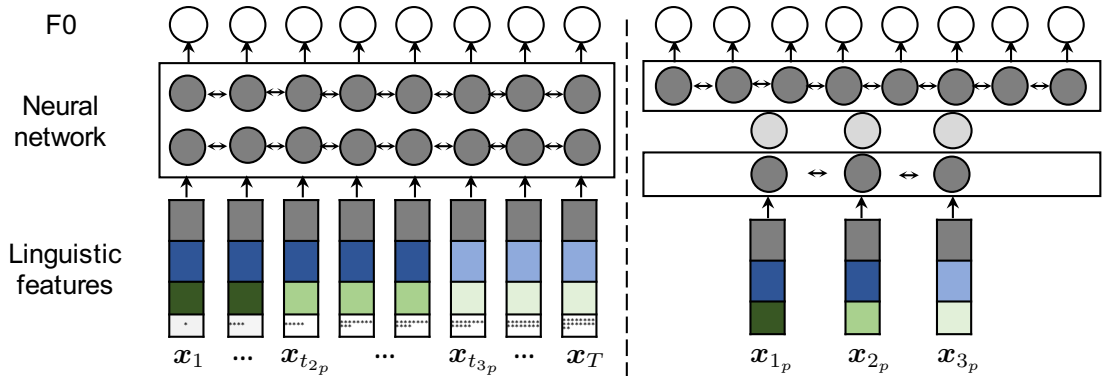


Figure 7.1: Linguistic feature processing in a frame-by-frame (left) or phone-by-phone (right) manner. T denotes T frames while 3_p denotes 3 phones. Linguistic features of different levels (from top to bottom: phrase, mora, phone, frame) are highlighted with different colors.

7.1 Motivation

In all the previous chapters, F0 modeling is defined to be a task that converts the input linguistic feature sequence $\boldsymbol{x}_{1:T}$ into an F0 contour $\boldsymbol{o}_{1:T}$ frame by frame for T frames. This process is shown on the left part of Figure 7.1. As Appendix A shows, the linguistic features are mainly the properties of phone, mora[1], phrase, and other segments in the linguistic hierarchy. Since a basic linguistic unit such as phone spans multiple frames, the linguistic features of the n-th phone should be duplicated to each frame in $\{\boldsymbol{x}_{t_n}, \boldsymbol{x}_{t_n+1}, \cdots, \boldsymbol{x}_{t_{(n+1)}-1}\}$, where t_n and $t_{(n+1)}$ denote the first frame of the n-th and $(n+1)$-th phone, respectively. Although some frame counters are attached to each $\boldsymbol{x}_t$, most of the linguistic features remain the same in the sequence of $\{\boldsymbol{x}_{t_n}, \boldsymbol{x}_{t_n+1}, \cdots, \boldsymbol{x}_{t_{(n+1)}-1}\}$.

Processing the linguistic features frame-by-frame is inefficient. When an RNN is used, the recurrent layer has to process many frames that contain almost the same linguistic features before it can retrieve the features of other linguistic units. A potentially more efficient framework is plotted on the right side of Figure 7.1. It divides the F0 model into two components: one maps the linguistic features of one linguistic unit into an 'excitation vector' (light grey circle), and the other generates the F0 contour given the 'excitation vector' and duration of each linguistic unit.

[1]The meaning of mora can be found in Table A.2.

7.2 VAE-based F0 model

We propose to cast the two-stage F0 model into a statistical framework. Suppose the phone is the basic linguistic unit and the utterance with T frames has N_p phones in total. This statistical framework can be written as:

$$p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:N_p};\boldsymbol{\Theta}) = \int_{\boldsymbol{e}_{1:N_p}} p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N_p};\boldsymbol{\Phi})p(\boldsymbol{e}_{1:N_p}|\boldsymbol{x}_{1:N_p};\boldsymbol{\Omega})d\boldsymbol{e}_{1:N_p}, \tag{7.1}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\Phi}, \boldsymbol{\Omega}\}$, and $\boldsymbol{e}_{1:N_p}$ and $\boldsymbol{x}_{1:N_p}$ denote the excitation vectors and the linguistic feature vectors of the N_p phones, respectively. For explanation, let's use 'F0 contour generator' and 'linguistic linker' to name the component $p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N_p};\boldsymbol{\Phi})$ and $p(\boldsymbol{e}_{1:N_p}|\boldsymbol{x}_{1:N_p};\boldsymbol{\Omega})$, respectively. Since the linguistic linker directly maps $\boldsymbol{x}_{1:N_p} = \{\boldsymbol{x}_{1_p}, \cdots, \boldsymbol{x}_{n_p}, \cdots, \boldsymbol{x}_{N_p}\}$ into the excitation vectors $\boldsymbol{e}_{1:N_p}$, it only needs to operate for N_p steps rather than T frames. Furthermore, if it uses a recurrent layer, the linguistic linker can easily retrieve the linguistic features from the neighboring phones.

Unfortunately, the above framework with latent $\boldsymbol{e}_{1:N_p}$ is impractical to use due to the integration over the space of $\boldsymbol{e}_{1:N_p}$. However, the F0 contour generator and the linguistic linker can be trained separately in an engineering way. Accordingly, we propose a three-step strategy:

1. Train a vector-quantization variational auto-encoder (VQVAE) [162] as $p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N_p};\boldsymbol{\Phi})$ and extract $\boldsymbol{e}_{1:N_p}$ from the training data $\boldsymbol{o}_{1:T}$;

2. Learn a linguistic linker $p(\boldsymbol{e}_{1:N_p}|\boldsymbol{x}_{1:N_p};\boldsymbol{\Omega})$ that maps $\boldsymbol{x}_{1:N_p}$ into $\boldsymbol{e}_{1:N_p}$ (or the code indices $l_{1:N_p}$ in the codebook of VQVAE for each vector in $\boldsymbol{e}_{1:N_p}$);

3. Use the linguistic linker and the codebook of VQVAE to generate $\widehat{\boldsymbol{e}}_{1:N_p}$ from $\boldsymbol{x}_{1:N_p}$ for a new sentence, and use the VQVAE decoder to generate the F0 contour $\widehat{\boldsymbol{o}}_{1:T}$ given $\widehat{\boldsymbol{e}}_{1:N_p}$[2].

The first two steps describe the training method while the third step is on the F0 generation method. These three steps are plotted in Figure 7.2 and will be explained with mode details in the following sections.

[2]A separate duration model is used to predict the duration of each linguistic unit. In this thesis, we use the natural duration acquired from force-alignment on the test utterance.

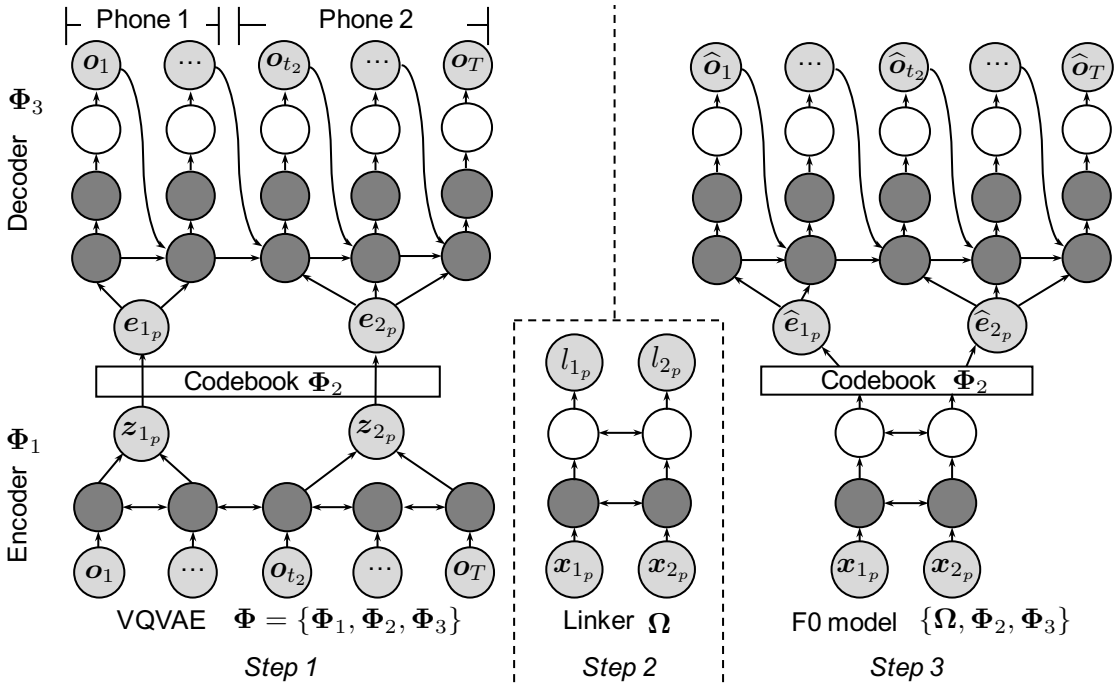


Figure 7.2: Framework of VQVAE-based F0 model. The utterance has T frames and 2 phones. Note l_{n_p} is the code index of the n-th phone.

7.2.1 VQVAE-based F0 encoder and decoder

Theory

A classical method to train a conditional generative model with latent variables $\boldsymbol{e}_{1:N}$ is the conditional VAE [163, 164], a variational Bayesian framework that maximizes an evidence lower bound (ELBO):

$$\begin{aligned}\log p(\boldsymbol{o}_{1:T}|\boldsymbol{x}_{1:N}) \geq & \mathbb{E}_{q(\boldsymbol{e}_{1:N}|\boldsymbol{o}_{1:T},\boldsymbol{x}_{1:N};\mathbf{\Omega}_1)} \log p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N},\boldsymbol{x}_{1:N};\mathbf{\Phi}) \\ & - \mathrm{KL}\Big[q(\boldsymbol{e}_{1:N}|\boldsymbol{o}_{1:T},\boldsymbol{x}_{1:N};\mathbf{\Omega}_1)||p(\boldsymbol{e}_{1:N}|\boldsymbol{x}_{1:N};\mathbf{\Omega}_2)\Big],\end{aligned} \tag{7.2}$$

where $q(\cdot;\mathbf{\Omega}_1)$ is the parametric posterior distribution, $p(\cdot;\mathbf{\Omega}_2)$ is the true parametric prior distribution, and KL is the Kullback-Leibler divergence.

However, when the conditional VAE is used on the sequential generative model, it suffers from 'posterior collapse' [162, 159, 155], a problem where the assumed posterior distribution $q(\cdot;\mathbf{\Omega}_1)$ becomes identical to the prior distribution $p(\cdot;\mathbf{\Omega}_2)$.

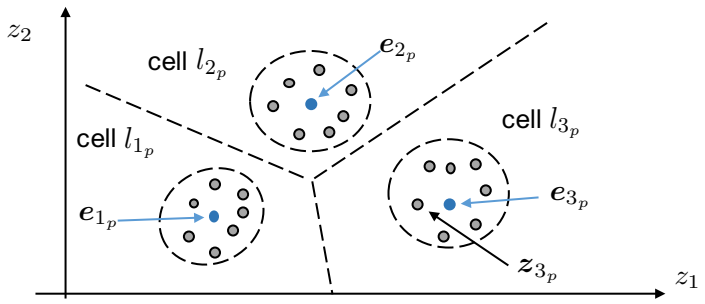


Figure 7.3: Example of vector quantization in VQVAE. The input vector $\boldsymbol{z}_{3_p} \in \mathbb{R}^2$ found the nearest codeword $\boldsymbol{e}_{3_p}$ in the cell indexed by l_{3_p}.

In such a case, $\boldsymbol{e}_{1:N}$ does not contain useful information for generating $\boldsymbol{o}_{1:N}$. Meanwhile, the decoder $p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N}, \boldsymbol{x}_{1:N}; \boldsymbol{\Phi})$ may ignore the conditional feature $\boldsymbol{x}_{1:N}$. This problem is even more obvious when the decoder $p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N}, \boldsymbol{x}_{1:N}; \boldsymbol{\Phi})$ is a strong sequential model such as the AR model [159]. Another disadvantage is that $q_{\boldsymbol{\Omega_1}}$ and $p_{\boldsymbol{\Omega_2}}$ must be Gaussian or simple parametric distributions so that the KL divergence can be calculated in a closed form. These assumed distributions, however, may be incompatible with the authentic data distributions.

To avoid the disadvantages above, we decide to use unconditioned VQVAE [162], an engineering yet effective framework that learns a compact latent space via minimizing the model's negative likelihood plus a penalty term. The framework of VQVAE is plotted on the left side of Figure 7.2. The latent variables $\boldsymbol{z}_{1:N} = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N\}$ are extracted from $\boldsymbol{o}_{1:T}$ by the encoder, after which each $\boldsymbol{z}_n$ is 'quantized' into $\boldsymbol{e}_n$ by retrieving the nearest neighbor in the codebook as Figure 7.3 shows. $\boldsymbol{e}_{1:N}$ is then used by the decoder to reconstruct $\boldsymbol{o}_{1:T}$.

The whole model is trained by minimizing

$$\mathcal{E}(\boldsymbol{\Phi}) = -\log p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N}; \boldsymbol{\Phi}_3) + (1+\beta)||\boldsymbol{e}_{1:N} - \boldsymbol{z}_{1:N}||^2, \tag{7.3}$$

where

$$\boldsymbol{z}_{1:N} = \text{Encoder}_{\boldsymbol{\Phi}_1}(\boldsymbol{o}_{1:T}), \quad \boldsymbol{e}_{1:N} = \text{Code\_search}_{\boldsymbol{\Phi}_2}(\boldsymbol{z}_{1:N}), \tag{7.4}$$

and $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \boldsymbol{\Phi}_3\}$. Here $\boldsymbol{\Phi}_3$, $\boldsymbol{\Phi}_2$, and $\boldsymbol{\Phi}_1$ denote the parameter set of the decoder, encoder, and codebook, respectively. β is a hyper-parameter that scales the loss caused by vector quantization. We use $\beta = 0.25$ as the original paper does.

In practice [162], the objective function is written as

$$\mathcal{E}(\mathbf{\Phi}) = -\log p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N};\mathbf{\Phi}_3) + ||\boldsymbol{e}_{1:N} - \text{sg}[\boldsymbol{z}_{1:N}]||^2 + \beta||\boldsymbol{z}_{1:N} - \text{sg}[\boldsymbol{e}_{1:N}]||^2, \quad (7.5)$$

where the sg operator sets the gradients to zero. For example, the second term of Equation (7.5) contributes to the gradients of $\boldsymbol{e}_{1:N}$ but not $\boldsymbol{z}_{1:N}$. Another trick to train the model is to move the gradients of $\boldsymbol{e}_{1:N}$ calculated from $\log p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N};\mathbf{\Phi}_3)$ to $\boldsymbol{z}_{1:N}$. Therefore, the encoder $\mathbf{\Phi}_1$ is trained by the minimizing the first and third terms of Equation (7.5), while the codebook $\mathbf{\Phi}_2$ and the decoder $\mathbf{\Phi}_3$ are updated based on the second and first term, respectively.

Implementation

The implementation of the VQVAE encoder is illustrated in Figure 7.4. The $\text{Encoder}_{\mathbf{\Phi}_1}(\cdot)$ is a neural network that extracts latent variable $\boldsymbol{z}_{1:N}$ from the observed data $\boldsymbol{o}_{1:T}$. After the linear transformation layer reduces the dimension of $\boldsymbol{o}_t$, the hidden features are processed by two bi-direction LSTM-RNN layers. Notice that, while the output of the hidden layer has T frames, $\boldsymbol{z}_{1:N}$ should contain N vectors where N is the number of linguistic units in the utterance. This time-resolution reduction is implemented by simply concatenating the hidden vectors at the first and last frames of the linguistic unit.The boundary of the linguistic unit is acquired through force-alignment at the frame-level.

After 'quantizing' $\boldsymbol{z}_{1:N}$ into $\boldsymbol{e}_{1:N}$ by retrieving the nearest neighbor in the codebook, $\boldsymbol{e}_{1:N}$ can be used by the decoder to reconstruct the F0 contour $\boldsymbol{o}_{1:T}$. Specifically, given the duration of each linguistic unit, each $\boldsymbol{e}_n$ is duplicated to the frame-level as $\{\boldsymbol{e}_{t_n}, \boldsymbol{e}_{t_n+1}, \cdots, \boldsymbol{e}_{t_{(n+1)}-1}\}$, where t_n and $t_{(n+1)} - 1$ denote the first and last frame index of the n-th phone, respectively. The decoder then converts $\boldsymbol{e}_{1:T}$ into $\boldsymbol{o}_{1:T}$. Although the decoder still works frame-by-frame, it can be simpler than a full neural F0 model. For example, as the decoder in Figure 7.2 shows, we use the idea of DAR and implement the decoder using a uni-directional LSTM layer that receives the feedback data and then a linear layer to change the feature dimension. The output layer is a softmax layer that generates the probability for quantized F0 symbol.

It is straightforward to take into account linguistic units of different levels. In

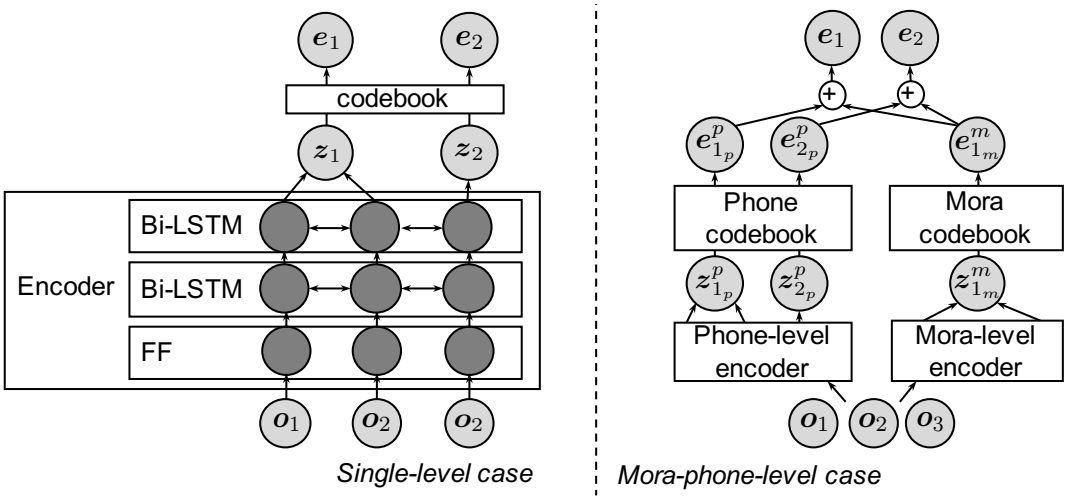


Figure 7.4: VQVAE encoder of single-level case (left) and multi-level (right) case. The utterance has 3 frames $\{\boldsymbol{o}_1, \boldsymbol{o}_2, \boldsymbol{o}_3\}$, where $\{\boldsymbol{o}_1, \boldsymbol{o}_2\}$ forms the first phone and $\boldsymbol{o}_3$ forms the second phone. These two phones further form one mora.

this case, the VQVAE contains multiple encoders but one decoder. The right side of Figure 7.4 plots the encoder that covers the mora and phone levels. In such an encoder, we can extract the mora-level latent vectors $\boldsymbol{z}^{\mathrm{m}}_{1:N_m}$ given the mora boundary. Then, a mora codebook can be used to generate $\boldsymbol{e}^{\mathrm{m}}_{1:N_m}$. Meanwhile, we can use a sub-encoder at the phone-level to extract the phone-level code $\boldsymbol{e}^{\mathrm{p}}_{1:N_p}$. Finally, the mora-level code is duplicated to each phone in the mora, and the sum of the mora- and phone-level codes is used as the final latent code $\boldsymbol{e}_{1:N_p}$. In such a multi-level encoder, the length of the final latent code sequence is determined by the number of linguistic units at the lowest level. Note that the latent codes of different linguistic layers have the same dimension.

In a multiple-level encoder, the sub-encoder and codebook of each level should be trained in a top-down manner in order to prevent the VQVAE from ignoring the latent codes provided by a high-level encoder. For the case in Figure 7.4, the mora-level sub-encoder and codebook are first trained and fixed, after which the phone-level part is added and trained. In this step, the mora-level part still outputs the latent code for each mora, and the moral-level codes are summed with phone-level codes as Figure 7.4 shows. Note that the single decoder is updated at every level.

7.2.2 Linguistic linker

The VQVAE only learns to encode and decode F0 contours in an unsupervised manner. For F0 modeling in TTS, another component should learn the mapping from the linguistic features to the F0 codes. This is done by a **linguistic linker**. Since each latent code $\boldsymbol{e}_n$ in the sequence $\boldsymbol{e}_{1:N}$ is a codeword from a finite-size codebook, the task of the linguistic linker is equivalent to a sequential classification task, where the target is the sequence of code index $\boldsymbol{l}_{1:N} = \{l_1, \cdots, l_N\}$, and the input is the sequence of linguistic features $\boldsymbol{x}_{1:N} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$. A baseline RMDN with a softmax output layer can be used for this task (see Section 3.4).

Naive implementation

When multiple linguistic levels are used, the linker needs to predict multiple code index sequences given $\boldsymbol{x}_{1:N}$, where N denotes the number of the finest linguistic unit. However, the index sequence of a higher linguistic level may be shorted than N in length. In this case, a naive method is to duplicate the higher level code index. Example of such a linker is plotted in the upper part of Figure 7.5. Here, the mora level indices $\{l^m_{1_m}, l^m_{2_m}\}$ are duplicated to the phones, and the linker learns to predict one mora and one phone code index for every phone.

Multi-time-resolution linker

To model multiple code index sequences, a better approach is to use a multi-time-resolution network. Suppose the mora and phone levels are used and their code indices are denoted as $l^m_{1:N_m}$ and $l^p_{1:N_p}$, respectively. The basic idea is to emit a hidden state and output value for a mora only when the current time step hits the first phone in the mora, a strategy similar to the clockwork RNN [121]. One example linker based on this strategy is plotted at the bottom of Figure 7.5. Note that the alignment between different linguistic levels can be easily retrieved from the linguistic features, without using the alignment at the frame-level.

Compared with the naive approach, the multi-time-resolution one does not need to model the duplicated code indices. Furthermore, it avoids possible inconsistency during F0 generation. For example, since a naive linker predicts a mora code index for each phone step, a mora with multiple phones will acquire multiple

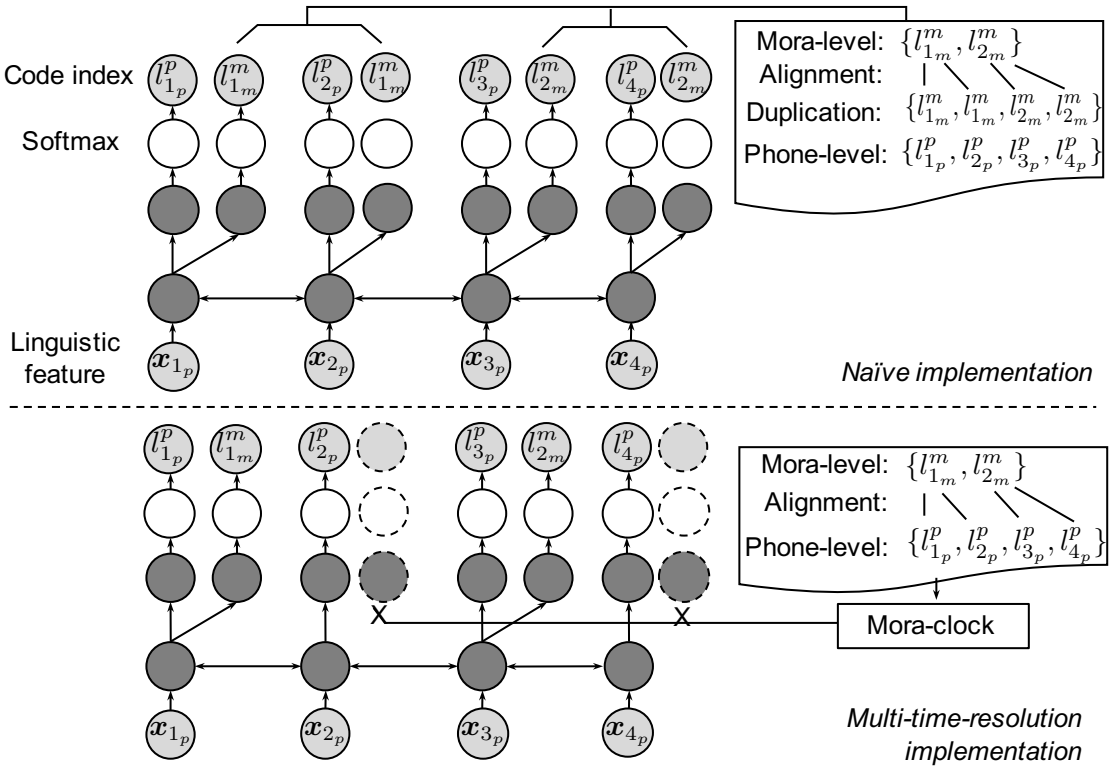


Figure 7.5: Naive linker (above) and multi-time-resolution linker (bottom) that predicts mora-level code indices $\{l^m_{1_m}, l^m_{2_m}\}$ and phone-level code indices $\{l^p_{1_p}, \cdots, l^p_{4_p}\}$

mora codes that may be unequal. It is then awkward to decide which code to use. From another perspective, the naive linker models the duplicated code indices and assumes that they are independent samples. However, this assumption is against the fact that duplicated indices come from to a single sample. The multi-time-resolution linker solves this inconsistency problem.

Other tricks on network structure and training methods

Since the linker is a sequential classifier, we can use any trick that applies to a classification model. Here are some tricks that we found to be effective in the experiments: deep highway layers near the output side [165], dropout [166] in all the hidden layers with a small probability (5% in our case), and the AR feedback loop (see the feedback loop of DAR in Section 6.2). Details of the tricks can be found in Section 7.3.4 where experiments were conducted on the linguistic linker.

Soft code vectors for F0 generation

After the linker is trained, a full-fledged neural F0 model for TTS can be built by concatenating the linker, the codebook, and the VQVAE decoder. During F0 generation, given the linguistic feature sequence $\tilde{\boldsymbol{x}}_{1:N}$ provided by the TTS front-end, the linker can predict a sequence of most probable code index $\widehat{\boldsymbol{l}}_{1:N} = \{l_1, \cdots, l_N\}$. It then can search for the corresponding code vectors from the codebook and ask the VQVAE decoder to generate the F0 contour based on the code vectors.

This above strategy is feasible, but it may not be optimal. Since Section 3.4 has explained the probabilistic side of the neural classification model, the statistics calculated by the model could have been used. Therefore, rather than predicting the code index directly, the linker can generate a sequence of probability vectors $\boldsymbol{P}_{1:N} = \{\boldsymbol{P}_1, \cdots, \boldsymbol{P}_N\}$, where each $\boldsymbol{P}_n = [P(l_n = 1|\boldsymbol{x}_{1:N}), \cdots, P(l_n = M|\boldsymbol{x}_{1:N})]$ describes the probability of the code indices at the n-th step and M is the size of the codebook. Then, for the n-the step, a soft code vector $\widehat{\boldsymbol{e}}_n$ can be computed as $\widehat{\boldsymbol{e}}_n = \sum_{m=1}^{M} P(l_n = m|\boldsymbol{x}_{1:N})\boldsymbol{e}_m$, where $\boldsymbol{e}_m$ is the m-th code vector in the codebook. Finally, the soft codes can be fed to the decoder for F0 generation.

The full-fledged neural F0 model based on the above strategy is plotted on the right-hand side of Figure 7.2. The same strategy can be used to generate the soft code vector for each linguistic level. Using the soft code vector $\widehat{\boldsymbol{e}}_n$ for F0 generation approximates the integration in Equation (7.1).

7.3 Experiments

7.3.1 Data and configuration

We used the Japanese corpus and feature configuration listed in Appendix A.2. Based on the results from the DAR, we only used quantized F0 for experiments. The F0 quantization was conducted in the same way as that in the last chapter. Quantized F0 of each frame was encoded as one-hot-vector of 256.

For the VQVAE part, no matter whether it had single or multiple linguistic levels, we configured the encoder for one linguistic level with one feedforward layer of size 64 and two Bi-LSTM layers of size 64. The codebook for each level had the

size of 128, and each code had the dimension of 64. The decoder contained one uni-directional LSTM layer of size 128, a linear layer of size 256, and a hierarchical softmax output layer, which was identical to the upper part of DAR (Figure 6.6). The VQVAE was trained using the Adam with default configuration [167] and early stopping.

The structures of the linguistic linkers were explained in the corresponding section (Section 7.3.4). All the linkers were trained using stochastic gradient descent with early stopping (learning rate 1e-05) and further tuned using AdaGrad (learning rate 0.001) with early stopping.

7.3.2 Part I: F0 encoding and decoding using VQVAE

The first experiment focuses on the VQVAE part and examines its performance through F0 encoding and decoding. The task of the VQVAE is to encode the F0 contour as a few latent codes, one for each linguistic unit, and reconstruct the F0 contour using the latent code and natural duration of each linguistic unit. Note that each VQVAE was trained on the training set and tested on the test set. The duration of linguistic units were acquired through force-alignment for both training and test sets.

We considered four levels of linguistic units for the experiment: phone, mora, word, and phrase. By counting the duration of linguistic units at each level in the training data, we computed the median duration and list the data in Table 7.1. These statistics could tell us the 'bit rate' of the VQVAE. For example, if the VQVAE extracts one code per phone, equivalently it extracts 1/13 code per frame, where 13 is the median of the phone duration in frames. Since the codebook size is $128 = 2^7$, the bit rate of this VQVAE is equal to $7/13 = 0.538$ bit per frame. For reference, since the natural F0 of each frame is quantized into $256 = 2^8$ bins, the bit-rate of quantized F0 is 8 bits per frame.

Encoding and decoding using a single linguistic level

Let's first compare the VQVAE's performance on each linguistic level separately. Reconstruction errors are listed as Phrase (qF0), Word (qF0), Mora (qF0), and Phone (qF0) in Table 7.2, and reconstructed F0 contours are plotted in Figure 7.6.

Table 7.1: Median duration of linguistic unit

	Phrase	Word	Mora	Phone
Median duration (frames)	95	51	25	13

Table 7.2: Objective results of F0 encoding-coding using VQVAE

Linguistic levels	Bit rate (bit/frame)	**RMSE**	**CORR**	**U/V**
F0 quantization	8	01.19	0.999	0.00%
Phrase (qF0)	7/95	61.87	0.425	28.78%
Word (qF0)	7/51	41.81	0.739	22.94%
Mora (qF0)	7/25	22.27	0.919	14.55%
Phone (qF0)	7/13	13.60	0.972	6.88%
Phrase (i-qF0)	7/95	58.83	0.469	-
Word (i-qF0)	7/51	37.05	0.796	-
Mora (i-qF0)	7/25	21.57	0.934	-
HWMP	$\frac{7}{95}+\frac{7}{51}+\frac{7}{25}+\frac{7}{13}$	11.46	0.982	4.58%
WMP	$\frac{7}{51}+\frac{7}{25}+\frac{7}{13}$	11.54	0.982	4.27%
MP	$\frac{7}{25}+\frac{7}{13}$	12.11	0.981	4.60%

Note: 'qF0' denotes quantized F0; 'i-qF0' denotes quantized F0 after interpolation in unvoiced frames. 'HWMP', 'WMP', and 'MP' denote phrase+word+mora+phone, word+mora+phone, and mora+phone, respectively. H, W, M used i-qF0 in these three encoders.

As expected, encoding-decoding at the phrase level is insufficient for F0 reconstruction. The codebook may be too small to encode the F0 contours of all the possible phrases. Another reason may be that the VQVAE is incapable of encoding or decoding F0 curves of roughly 95 frames in length even if the VQVAE used a DAR-based decoder. As long as the VQVAE was trained at the mora level, the F0 was better encoded and reconstructed despite the high U/V error. At the phone-level, the reconstructed F0 was quite close to the natural F0 as Figure 7.6 shows.

Of course, even the reconstructed F0 from the phone-level VQVAE was imperfect because the bit rate of the VQVAE was only 7/13, which is much lower than 8 bits-per-frame of the quantized F0 data. What's more, the VQVAE was trained on the training set while tested on the test set.

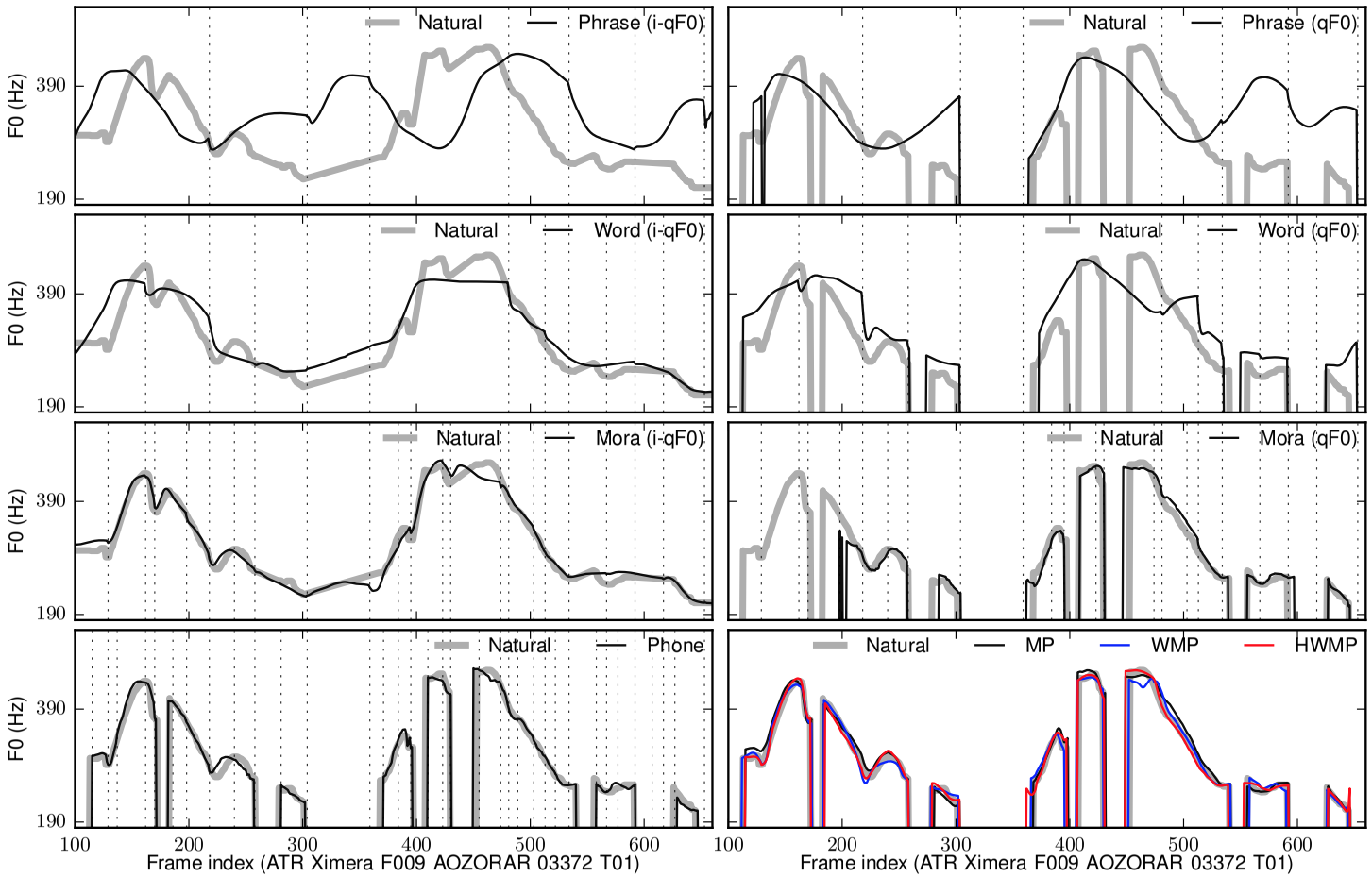


Figure 7.6: Reconstructed F0 from VQVAE through encoding-decoding. 'qF0' denotes quantized F0 that includes the unvoiced symbols; 'i-qF0' denotes quantized F0 after interpolation in unvoiced frames, which does not contain unvoiced symbols. 'HWMP', 'WMP', and 'MP' denote phrase+word+mora+phone, word+mora+phone, and mora+phone, respectively. High level encoders in the last figure except phone used i-qF0.

Issue with voicing status

In the previous experiment, we directly used the VQVAE to encode and decode the quantized F0 data. Because the quantized F0 data contain unvoiced symbols, the VQVAE needs to model not only the F0 contour but also the voicing status for each linguistic unit. However, since the voicing status is mainly the property of low-level linguistic units such as phones, it may be inappropriate to model the voicing status at the level of mora, word, or phrase.

As a pilot test, we quantized the interpolated F0, which were originally used for the RNN, and trained the VQVAE used the interpolated quantized F0 (i-qF0). The results are listed as Phrase (i-qF0), Word (i-qF0), and Mora (i-qF0) in Table 7.2. Although the VQVAE at the phrase level still failed to work, the VQVAE at mora level performed quite well. As Figure 7.6 shows, the reconstructed F0 was consistent with the shape of the natural F0 contour. Note that, since the target of the decoder was i-qF0, the VQVAE cannot model the voicing status [3].

Encoding and decoding at multiple linguistic levels

Based on the results above, we tried to encode and decode the F0 contours using multiple linguistic levels. In this experiment, we compared phrase + word + mora + phone (HWMP), word + mora + phone (WMP), and mora + phone (MP). As explained in Section 7.2.1, encoders at different linguistic levels should be trained in a top-down manner. In the case of WMP, we first used the i-qF0 to train the word-level VQVAE. We then fixed the word-level encoder, added the mora-level encoder, and trained it with the decoder using the i-qF0. Finally, we fixed the word- and mora-level encoders, added the phone-level encoder, and trained it with the decoder using qF0. Since the decoder was trained using the qF0 at the last stage, it can generate both U/V status and F0 contours.

The results are listed in the last three rows of Table 7.2, and reconstructed F0 contours are plotted in Figure 7.6. The results indicate that using multiple levels improved the F0 encoding-decoding performance.

[3] Although the experiments in Section 6.4.2 suggested that interpolated F0 cannot be well modeled by the DAR, the influence of the interpolated curves may be less severe than the unvoiced segments for the high level encoders.

7.3.3 Visualization of code space

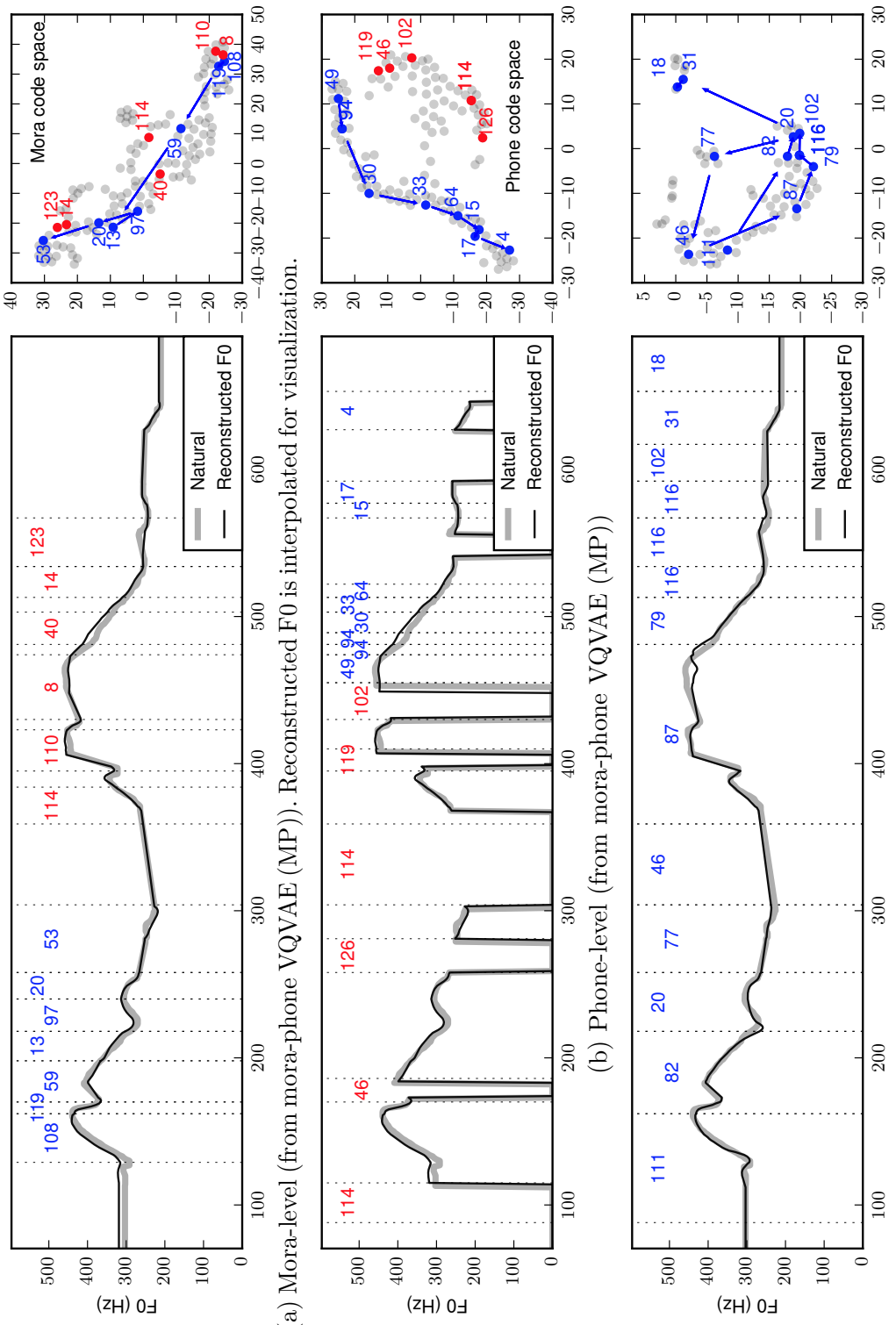
Before further experiments, we'd like to visualize the latent code spaces learned using the mora-phone VQVAE (MP). We used t-SNE [168] to compress the dimension of code vectors and plotted them in a two-dimensional space. The results are plotted in Figure 7.7, where (a) shows the code space learned at the mora level and (b) shows the phone level. For reference, the left column of the figure shows the code indices assigned to the moras or phones in the utterance, and the right column highlights these codes in the code space.

Interestingly, both the mora- and phone-level code spaces turned out to be quite regular. At the mora level in Figure 7.7 (a), the code indices assigned to the first segment of the example F0 contour were $\{108, 119, 59, 13, 97, 20, 53\}$. It seems that the location of the code generally represented the average F0 height of a linguistic unit. For example, the mora with the 108-th code contained an F0 peak, and the following moras had decreasing F0 curves. Accordingly, the line connecting the code sequence $[108, 119, 59, 13, 97, 20, 53]$ started from the right-bottom corner, moved along the code manifold, and ended at the left-top corner, which indicated the decreasing average F0 value of moras.

The encoder at the phone-level seemed to encode both the voicing status and the F0 values. For example, the code indices $\{114, 45, 126, 102\}$ in Figure 7.7 (b) were assigned to unvoiced phones and located on the right-hand side of the code space. In contrast, the code indices for voiced phones were found on the left-hand side and encoded the average F0 value in a similar manner to the mora-level case.

These results may help us interpret the 'meaning' of latent codes and the role of the decoder in the VQVAE-based F0 model: the latent code mainly encode the skeleton of the F0 contour (e.g., average F0 height in one unit) while the decoder fills in detailed F0 curves and reconstructs the F0 contour. The average F0 height may be general and safe skeleton representation for F0 contours because it does not rely on specific linguistic or physiological theories on the F0 shapes.

We also plotted the manifold learned by the word (bottom figure in Figure 7.7) and phrase encoders but did not find regular patterns. The reason may be the difficulty of learning meaningful long-time F0 patterns using models trained by a frame-level maximum-likelihood criterion.



(a) Mora-level (from mora-phone VQVAE (MP)). Reconstructed F0 is interpolated for visualization.

(b) Phone-level (from mora-phone VQVAE (MP))

(c) Word-level (from Word-mora-phone VQVAE (WMP))

Figure 7.7: Explore latent code space. Left: F0 contours with the linguistic unit boundaries (dot line) and code indices (colored number). Right: colored dots are the codes used in the left figure; grey dots denote other codes.

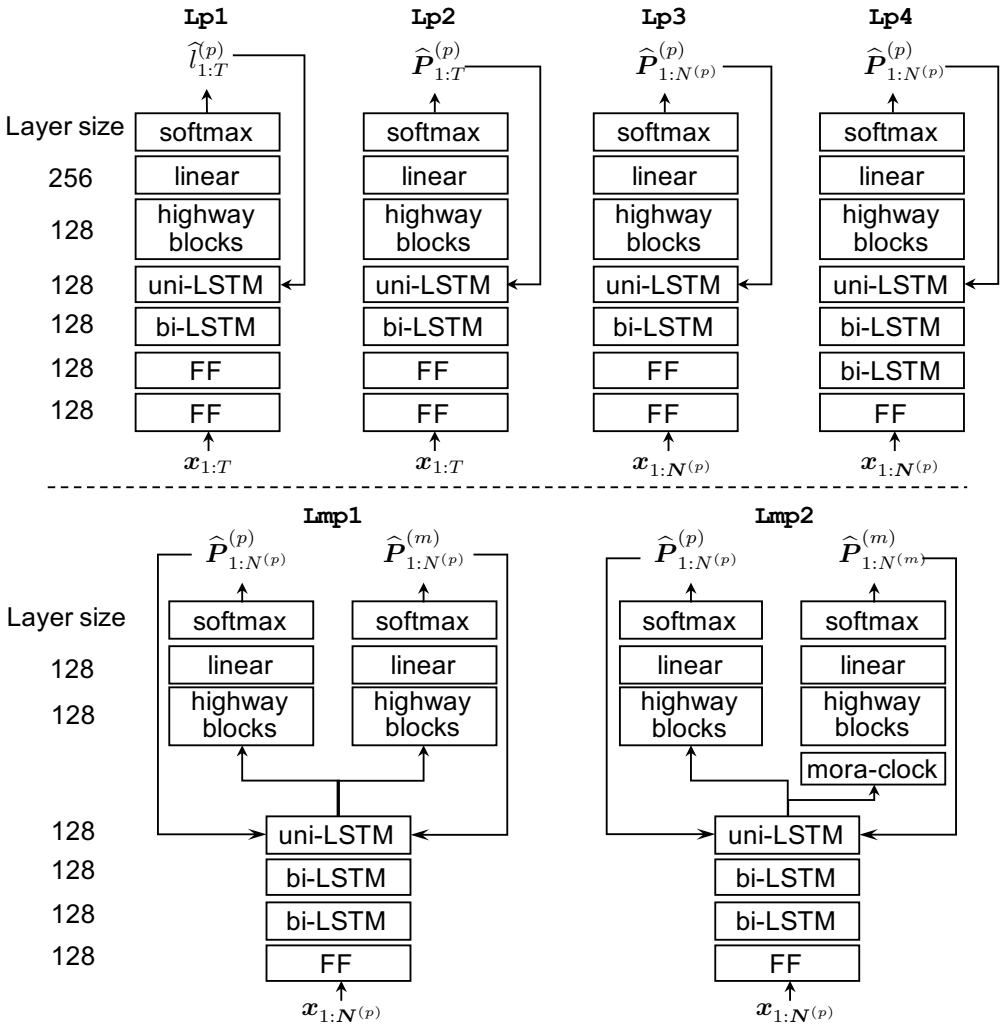


Figure 7.8: Structure of linguistic linker listed in Table 7.3. `Lmp3` is the same as `Lmp2` except using dropout (dropout rate = 5%) in every hidden layer. Other models didn't use dropout in hidden layers. $\widehat{l}$ and $\widehat{\boldsymbol{p}}$ denote generated code index and code probabilities, respectively. T, $N^{(p)}$, and $N^{(m)}$ denote the number of frames, phones, and moras, respectively.

7.3.4 Part II: Text-to-code using linguistic linker

This section conducted experiments on the linguistic linker. Based on the VQVAEs learned in the previous section, we extracted the latent codes from the F0 contours and trained linguistic linkers using the same training and validation set as those in the VQVAE experiment. After training the linguistic linkers, we test the

Table 7.3: Objective results on F0 generation given linguistic features

Model	VQVAE	Linker	Linker training time		**RMSE**	**CORR**	**U/V**
			s/epoch	ms/step			
P1	Phone	Lp1	1300	0.036	34.33	0.839	7.96%
P2		Lp2	1300	0.036	32.79	0.856	7.61%
P3		Lp3	54	0.031	27.11	0.906	6.36%
P4		Lp4	61	0.035	26.74	0.908	6.36%
MP1	MP	Lmp1	59	0.034	27.35	0.907	6.22%
MP2		Lmp2	63	0.036	26.46	0.912	6.24%
MP3		Lmp3	65	0.037	25.55	0.916	4.87%
WMP1	WMP	-	70	0.040	26.72	0.909	5.04%
HWMP1	HWMP	-	76	0.044	26.18	0.909	4.81%

Note: the 'step' means 'frame' for Lp1 and Lp2 while 'phone' for the other linkers. 'HWMP', 'WMP', and 'MP' denote phrase+word+mora+phone, word+mora+phone, and mora+phone encoders (Table 7.2), respectively. These encoders used i-qF0 for the phrase, word, and mora levels. Linkers for WMP1 and HWMP1 were similar to lMp3 except additional branches and clocks were added to predict the word and phrase level codes.

performance by concatenating the linkers with the VQVAE decoder and generating F0 contours from linguistic features on the test set. Therefore, objective results in this section denote the performance of full-fledged VAE-based F0 models for TTS.

Phone-level models

We first compared four linguistic linkers Lp1, Lp2, Lp3, and Lp4 in Figure 7.8 for the phone-level VQVAE. These linkers only modeled the phone-level codes. Notice that Lp1 and Lp2 converted the linguistic features to the target in a similar frame-by-frame manner to a normal neural F0 model. The only difference was the generation method: while Lp1 predicted the best code indices $\widehat{\boldsymbol{l}}^{(p)}_{1:T}$, Lp2 generated the index probabilities $\widehat{\boldsymbol{P}}^{(p)}_{1:T}$ for soft code vector generation (Section 7.2.2). To train Lp1 and Lp2, the code indices in the training set were duplicated to the frame-level. From Lp2 to Lp3, the linker switched from frame-level modeling to phone-level modeling, i.e., generating the probability vectors phone-by-phone. Lp4 further replaced one feedforward layer of Lp3 into a bi-directional LSTM.

After model training, the linkers were combined with the VQVAE decoder and

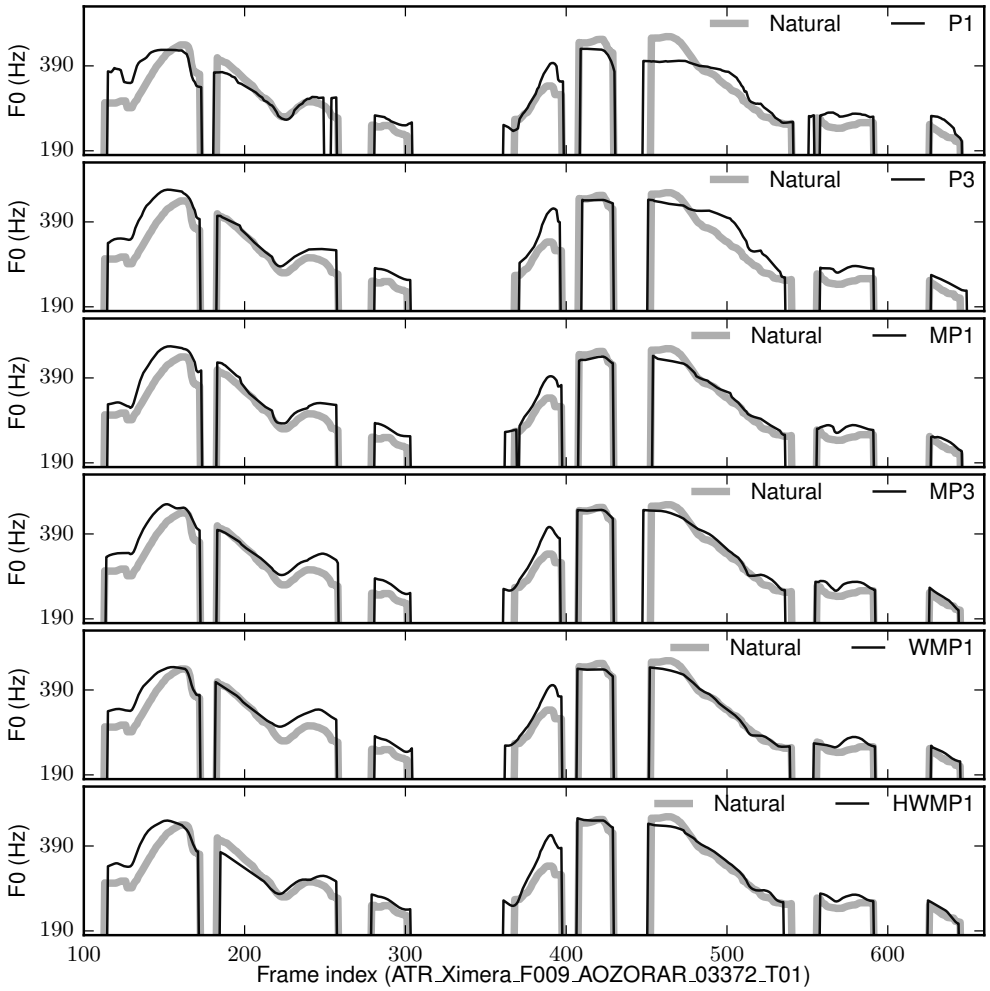


Figure 7.9: Generated F0 from VAE-based F0 model given linguistic features. Definition of each model is listed in Table 7.3.

the codebook (from Phone(qF0) in Table 7.2) in order to generate F0 contours given linguistic features on the test set. The objective results are listed in the first four rows of Table 7.3. Notice that `Lp1` and `Lp1` shared the same set of network weights. They only differed in the way of generating the latent code for the decoder. The comparison between `Lp1` and `Lp2` suggests that using the soft code vector generation improved the performance.

Compared with `Lp2`, the performance of `Lp3` improved a lot. In the case of `Lp2`, processing the linguistic features frame by frame is inefficient because the

same linguistic feature vector is duplicated to all the frames in the phone and then processed. Another reason specific to this task may be that the same phone code index is also duplicated for all the frames in the phone. Therefore, the target sequence for the linker `Lp2` is stepwise and may not be well modeled. Furthermore, there is no guarantee that `Lp2` would predict the same code index for all the frames inside the phone.

In addition to the improved objective results, `Lp3` cost less training time. Although both `Lp3` and `Lp2` require around 0.03 ms to process one step, the total number of steps is reduced in the case of `Lp3`. Specifically, the total number of steps for `Lp2` is equal to the frame number while it is the number of phones for `Lp3`.

Since the length of the data sequence is reduced, adding an additional recurrent layer would not increase training or generation time too much. For example, `Lp4` replaced a feedforward layer with a recurrent layer, but the training speed was still fast[4]. With a slight increase in computation load, `Lp4` further improved the objective performance.

Multi-linguistic-levels models

This experiment focuses on the linkers generating the mora- and phone-level latent codes. The VQVAE model was the one trained at the mora and phone levels (MP in Table 7.2). Based on this VQVAE model, three linkers `Lmp1`, `Lmp2`, and `Lmp3` were compared. Both `Lmp1` and `Lmp2` operated phone-by-phone and learned to generate the code index probabilities for the phone and mora levels. However, `Lmp2` used the multi-time-resolution architecture based on the mora-clock in Figure 7.5 while `Lmp1` used the naive implementation. `Lmp3` was the same to `Lmp2` except using dropout with a probability of 5% in all the hidden layers.

The results are listed in the middle of Table 7.3. First, `Lmp2` achieved better performance than `Lmp1` in terms of RMSE and CORR. Note that the improvement brought by the multi-time-resolution may be small because one mora in Japanese usually contains 1 or 2 phones. Compared with `Lmp2`, `Lmp3` further improved the performance, especially the U/V. This result shows the importance of network regularization when the training data amount is limited. This is reasonable because

[4]The listed training time may be influenced by the load of the GPU server.

the number of data samples in the training set is equal to the number of phones rather than frames.

In addition to the mora-phone-level linkers, we also evaluated the linkers with more linguistic levels. These linkers were similar to `Lmp3` except that they included additional network branches to predict the word and phrase level codes. However, the results listed in the last two rows of Table 7.3 suggest that using more linguistic levels for F0 modeling was worse than the mora-phone-level case. One possible reason was that the VQVAE at the high linguistic levels did not learn useful abstract F0 representations, which has been indicated by the unsatisfied results from manifold interpretation (Figure 7.7 (c)).

7.3.5 Compare VAE-based F0 model with DAR

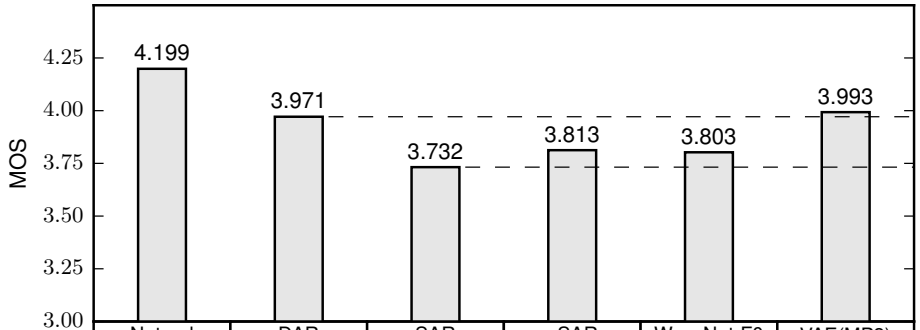
Based on the above results, let us briefly compare the best VAE-based F0 model (`MP3` in Table 7.3) and `DAR` in the previous chapter. As Table 7.4 suggests, `MP3` achieved better RMSE and CORR scores even though the U/V error is increased. The worse U/V performance is reasonable because the latent code per phone may be insufficient to encode the U/V of all the frames in one phone.

Figure 7.10 shows the results of the subjective MOS test. The test configuration was identical to the second MOS test in Section 6.4.5. The results demonstrated that `MP3` slightly outperformed `DAR`, even though the difference is not statistically significant. However, `MP3` achieved this performance more efficiently than `DAR`. Specifically, `MP3` had a smaller number of model parameters (1.11 = VQVAE 0.44 + linker 0.67) and used less training time (1565 = VQVAE 1500 + linker 65) than `DAR`. Furthremore, since the VQVAE encoder is unnecessary during F0 generation, the size of `MP3` can be smaller when it is used for F0 generation (0.93 = VQVAE decoder 0.26 + linker 0.67).

Although the linker for `MP3` can be trained very fast, the mora-phone-level VQVAE part requires more training time because of the top-down training approach and the frame-by-frame operation manner. In the generation time, iterating and searching the codebook also increases additional time cost.

Table 7.4: Objective results on the Japanese data. GV of natural F0 is around 61.

	#. Para. (million)	Time consumption Train. (s/epoch)	Time consumption Gen. (ms/frame)	**RMSE**	**CORR**	**U/V**	**GV**
DAR	1.48	2000	0.205	28.30	0.903	3.46%	61.5
VAE(MP3)	1.11	1565	0.147	25.55	0.916	4.87%	57.9



	Natural	DAR	SAR	eSAR	WaveNet-F0	VAE(MP3)
Natural		<1e-10	<1e-10	<1e-10	<1e-10	<1e-10
DAR	<1e-10		<1e-10	9.062e-08	0.000102	0.392125
SAR	<1e-10	<1e-10		0.007186	0.000176	<1e-10
eSAR	<1e-10	9.062e-08	0.007186		0.219733	5.946e-10
WaveNet-F0	<1e-10	0.000102	0.000176	0.219733		2.635e-06
VAE(MP3)	<1e-10	0.392125	<1e-10	5.946e-10	2.635e-06	

Figure 7.10: MOS score (mean-based generation, Japanese corpus) and p-value of Mann-Whitney U test between each pair of experimental models

7.4 Summary

This chapter looks into **Issue 3: Can linguistic features be processed more efficiently?** The answer is yes, and one more efficient model is the VAE-based neural F0 model proposed in this chapter.

The basic idea of this model is to separate F0 contour modeling from the linguistic feature processing. The F0 contour modeling part can be implemented based on a VQVAE model, which learns a compact code space in an unsupervised

manner. The second part uses a linker to learn the mapping from linguistic feature to the latent code. This VAE-based F0 model is more efficient mainly because the linker processes the linguistic features at the phone or higher linguistic level rather than transforming the highly redundant linguistic features frame by frame.

Experiments demonstrated that the VQVAE part was able to encode the F0 contour of a linguistic unit, e.g., phone or mora, into a single vector and learn meaningful code spaces. Meanwhile, the linkers predicted latent codes from linguistic features quite well. By using a mora-phone-level VQVAE and a corresponding linguistic linker, the combined VAE-based F0 model performed no worse than the DAR. Meanwhile, the VAE-based F0 model was smaller and faster.

The VAE-based F0 model resembles the classical two-step F0 modeling approaches [71, 72, 73], where the first step uses an F0 parametric model such as Fujisaki model to extract F0 parameters for each linguistic unit, and the second step learns the mapping from linguistic features to the F0 parameters of each linguistic unit. However, those classical approaches rely on deterministic and expert-designed F0 parametric models. In contrast, the proposed VAE-based model uses trainable neural networks and avoids any assumption about the F0 shape. Thus, the proposed model is hopefully applicable to other languages.

Recently, there are new works using the VAE [169] or auto-encoder [170] for F0 modeling. However, that VAE-based work still relies on the contraints of Fujisaki-model. Additionally, how it performs for TTS is not yet reported. The F0 auto-encoder only handles fixed-length F0 contours and requires F0 interpolation before modeling. What's more, its performance for TTS tasks is not reported either. Compared with these related works, the VAE-based model proposed in this chapter is more flexible and has shown good performance for TTS tasks.

8 Conclusion

In the previous chapters, we have explored the three issues for neural-network-based F0 modeling. In this chapter, we summarize our answers to the three issues in Section 8.1. We then use the proposed F0 model in our TTS system with other advanced modules and evaluate its performance in Section 8.2. In 8.3, we discuss remaining issues for neural F0 modeling. Finally, we summarize with a final remark in Section 8.4.

8.1 Replies to the three issues of neural F0 modeling

Issue 1: Is it appropriate to model F0 together with other acoustic features in a neural network? Probably not.

We investigated the validity of the default strategy in common neural-network-based SPSS framework, where the F0 and other acoustic features are jointly modeled.

By conducting experiments on single- and multi-stream highway networks, we found that a single-stream highway network prioritized the modeling of spectral features rather than the F0. Specifically, a single-stream network that jointly modeled the F0 and spectral features must be sufficiently wide or deep before it reached the F0 modeling performance similar to that of a multi-stream network that separated the F0 and spectral features. Similar results were observed on both the English and Japanese data.

More interestingly, results based on the histogram and sensitivity analyses on the multi-stream highway network further suggested that a network would use different input linguistic features and derive different hidden features for modeling the F0 and spectral features. This is against the belief that F0 and spectral features can share hidden features and benefit from multi-task learning. In all, the experimental results suggest that it is worthy of a trial to separate the F0 from the spectral features for at least a better result of F0 modeling.

Issue 2: Do the common neural models describe the temporal correlation in F0 contours? No. But better models can be defined.

Based on the probabilistic interpretation of neural networks, we found that a normal neural network cannot model the temporal correlation of the target feature sequence (e.g., the F0 contour). We further used a random sampling method to reveal the noisy output samples caused by the missing temporal correlation.

To better model the temporal correlation, we first proposed the SAR that used a linear transformation to model the AR temporal dependency of the F0 contour. It turned out the SAR can be further interpreted as the combination of filters and a normal RMDN, which allowed us to use sufficient but unnecessary techniques to ensure the stability of the SAR. Experiments showed that the SAR remained stable and alleviated the over-smoothing problem, which was partially caused by the missing temporal correlation. We further interpreted the SAR from the perspective of feature transformation and proposed the extended SAR using non-linear yet invertible normalizing flow. This extended SAR improved the performance of the original SAR.

Although the SAR is theoretically appealing, it didn't pass the test of random

sampling. Therefore, we proposed the DAR to leverage non-linear and non-invertible AR transformation. This idea is to feed the data of the previous time step back into a recurrent layer for the current time step. Although the idea is simple, we used toy examples to show the flexibility and generality of the DAR. Based on additional techniques such as quantized F0 representation and dropout, the DAR outperformed other neural F0 models in terms of perceptual quality. What's more, the DAR generated good samples by random sampling, which has never been achieved by other F0 models.

In all, to better model the temporal correlation of the F0 contours, the proposed DAR model is a good choice for neural F0 modeling.

Issue 3: Frame-by-frame processing even for linguistic features? Not recommended. The model can be more efficient.

Based on the DAR, we further explored whether the frame-by-frame F0 modeling architecture can be more efficient, especially in the part that processes linguistic features. For this purpose, we introduced the idea of VAE, based on which the F0 modeling task can be decomposed as an F0 contour modeling part and a linguistic linking part. This strategy allows us to encode the F0 contour and extract meaningful F0 representation for linguistic units with varied length. It further allows the mapping from the linguistic features to the latent F0 representation above the frame level, which is more efficient.

Experiments showed that this VAE-based framework worked quite well on the phone and mora levels. On the F0 encoding-decoding part, the VQVAE learned to encode the F0 contour at the phone and mora levels and reconstruct it with near-perfect accuracy. On the other part, it was justified that a linguistic linker could map the linguistic features to the F0 codes phone-by-phone. More importantly, even with fewer model parameters and less processing time, the VAE-based F0 model based on the VQVAE decoder and the linguistic linker achieved a slightly better performance than the DAR.

It is unnecessary to process the linguistic features as we would do in a normal neural F0 model. To reduce the processing time and improve the performance, the VAE-based F0 model with multiple linguistic levels is recommended.

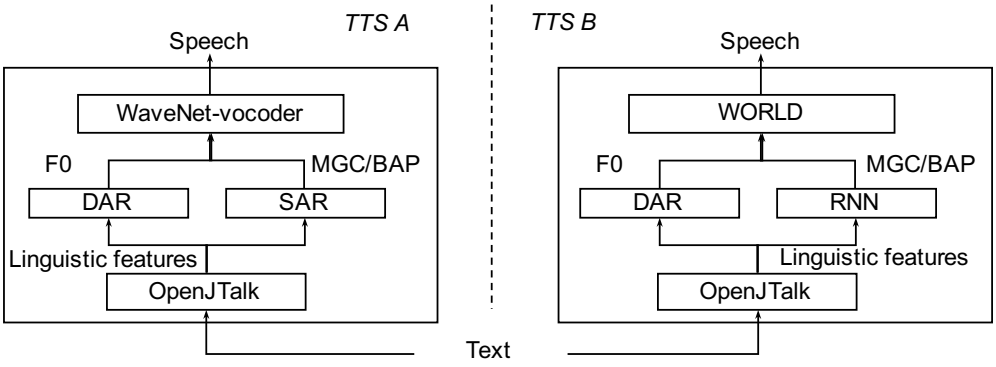


Figure 8.1: Use DAR and SAR for TTS systems

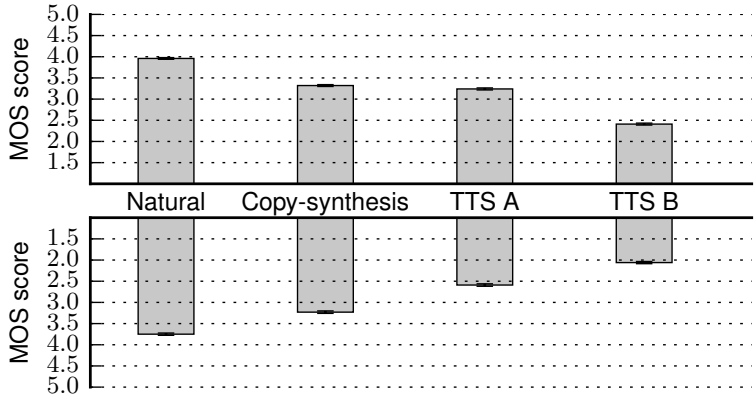


Figure 8.2: Results of subjective evaluation

8.2 Apply proposed F0 model in TTS systems

Given the results obtained so far, we think it is interesting to see whether the proposed F0 model could help the TTS system to reach the performance of natural speech. For this purpose, we conducted experiments on the Japanese corpus using the DAR and other acoustic models. The configuration of the data corpus and acoustic features were the same as the previous chapters (see Appendix A.2 for reference). The experimental TTS systems were plotted in Figure 8.1. To achieve the best performance, we used the WaveNet-based vocoder [149] for waveform generation, the SAR for MGC and BAP modeling, and the DAR for F0 modeling. As the reference system, we used one system with the RNN for MGC and BAP modeling and the WORLD vocoder for waveform generation. These two TTS systems used the OpenJTalk [171] front-end. Configurations of the SAR and the WaveNet can be found in [149].

For the system using the WaveNet, the speech sampling rate was equal to 16kHz. For other systems, the synthetic speech was originally 48kHz but downsampled to 16kHz. The evaluation was crowdsourced online. Each evaluator must answer two questions on each screen. First, they listened to a sample of the system under evaluation and rated the naturalness on a 1-to-5 mean-opinion-score (MOS). They then rated the similarity of that test sample to the natural 48 kHz sample on a 1-to-5 MOS scale. We collected a total of 1500 evaluation sets from 235 native Japanese listeners. The statistical analysis was based on unpaired t-tests with a 95% confidence margin and Holm-Bonferroni compensation.

The results are plotted in Figure 8.2. Interestingly, the system using the WaveNet-vocoder, the SAR, and the DAR could achieve a quality roughly similar to that of the vocoded speech. However, the similarity score is quite limited. This may be due to the varied quality of the natural speech, which was recorded over 1 year. Of course, each component should be further improved, especially the front-end and the vocoder that extracted the acoustic features on the training set. Nevertheless, the result shows that the SPSS-based TTS framework is quite promising if good acoustic models can be used. Note that the SAR was quite effective when it was used for MGC modeling.

8.3 Remaining issues

Both results from the previous section and Chapter 6 showed that there is still a gap between the synthetic and natural speech. Although recent work on Tactron 2 [4] has demonstrated synthetic speech with a natural quality, its framework and feature design are different from the SPSS-based framework discussed in this thesis. Certain aspects of the conventional SPSS framework may be further improved. Here, we mainly consider the issue related to F0 modeling.

Remaining Issue I: incorrect linguistic features

The first factor may be the noise in the linguistic features used for the experiment. In this thesis, all the linguistic features were derived using a TTS front-end for both training and test data. It is well known that not every prosodic tag can be

Table 8.1: Impact of noisy linguistic features on the DAR.

Linguistic features	RMSE	CORR	V/U
OpenJTalk	28.30	0.903	3.46%
Manual annotation	23.31	0.940	3.25%

accurately predicted from the text [43]. In addition, incorrect word pronunciation, stress, phrase boundary, and other linguistic features may have a huge impact on F0 modeling. On the Japanese data, the mismatch between the characters in text and the automatically derived phonemes is more serious.

In fact, we did experiments using the DAR and manually corrected linguistic features, and the results are listed in Table 8.1. Note that the training and test sets are slightly smaller than those used in previous chapters because of the missing annotation on some utterances. Obviously, using correct linguistic features in the training and test set could improve the performance by a large margin.

However, linguistic features cannot be perfectly predicted from a text. As Halliday pointed out [3], a text is a lossy media to convey sufficient information for speaking. Another pessimistic view is the famous one: accent is predictable (if you're a mind-reader)] [172][1]. It is difficult to predict linguistic features perfectly. Note that noise in the input data may act as a regularizer for a statistical model [173][174]. However, the 'noise' caused by incorrect linguistic features may be too large.

Remaining Issue II: incomplete linguistic features

It is not only the noise in the existing linguistic features but also missing linguistic features that influence F0 modeling. Intonation is a half-tamed savage [22]. It has not only the half side with a linguistic structure but also the other side that is more close to animal communication. For the untamed-half, it is really difficult to figure what triggers the change of F0 contour. It may be the current health condition of the speaker, psychological status, or other factors that cannot be easily described by categorical tags. In a word, we lack the necessary information.

More importantly, we even lack the complete linguistic features to fully explain

[1]The accent here means the prosodic tag

the F0 movement. For example, the ToBI protocol does not try to explain all the detailed F0 contours [175]. Although perceptually it may be unnecessary to describe the detailed F0 contour, we agree with the opinion that any detail should not be ignored by assuming they are not important [176]. Some researchers have designed alternative linguistic features for SPSS-based TTS [177], which may be worthy of trial.

Remaining Issue III: evaluation strategy

The last point is about the evaluation strategy, both objective and subjective. First, if we could have a perfect objective metric that measures the perceptual quality of a synthetic F0 contour, it would save the effort of subjective evaluation. Although there is research work towards this direction [178], it is too early to claim that such an objective metric could be accurate enough, F0-oriented, and feasible for off-line evaluation. To approximate the perceptual quality of the F0, there has been some work proposing different metrics [179]. However, it seems to be difficult to define a proper metric.

If we want to evaluate the synthetic F0 model subjectively, the strategy of evaluation also matters. Currently, all the subjective tests were conducted by measuring the quality of isolated utterances. However, evaluating F0 contours in a certain context would be more beneficial. This was not conducted in this thesis because the corpora used are not suitable for dialogue-style test evaluation. A new evaluation strategy may also be interesting to test the randomly sampled F0 contours from the DAR.

8.4 Final remark

- This thesis has used experiments to show the inappropriateness of joint F0 and spectral feature modeling in neural-network-based SPSS. This is the first trial of the use of the highway network for investigation in the speech synthesis community.

  The reader should be careful: the conclusion we claimed may not apply to every scenario. The decently uttered neural style reading speech used in this

thesis may be well handled by the vocoders so that the F0 and spectral features are well separated from the waveform. Indeed, this is the goal of high-quality vocoders. However, for other speech data such as singing voice or spontaneous speech, the extracted F0 and spectral features may not be well separated. In such a case, joint modeling may be tried.

- This thesis proposed the SAR and DAR. Although the idea of AR dependency is not new, the interpretation and analysis of the SAR and the DAR are novel. Quantized F0 is also used in F0 generation for the first time. The DAR is also the first model to support random sampling at the frame level.

  The idea of DAR and SAR are general. As the toy network examples have shown, DAR and SAR are theoretically better than the normal RNN. They can be used for other data and different types of F0 sequential features. Especially, our result in this chapter has shown that the SAR worked better than RNN on spectral feature modeling.

- This thesis proposed the VAE-based F0 model. It achieved a high objective performance (F0 correlation over 0.91) and is more efficient. Although the VAE-based F0 model resembles many conventional F0 approaches that use expert-knowledge based models, it does not assume any linguistic theory on F0 but just uses a data-driven method. This may allow the VAE-based F0 model to be used for other data types or even different languages.

F0 modeling can be approached from various perspectives This thesis displayed our trials on interpreting and improving the statistical models of the F0. Although we are still the followers of pragmatism that tries to find engineering solutions to a linguistic/speech-processing task, we hope we are not the 'Pendulum Swung Too Far' [180].

A Appendix

A.1 Linguistic features for neural-network-based SPSS

Linguistic features are automatically derived using the TTS front-ends listed in Table A.1. Tables A.2 and A.3 show the linguistic features used for the neural-network-based SPSS. These features are encoded as features vectors $\boldsymbol{x}_t$ and used as the input of neural networks.

Table A.1: Configuration of linguistic features

Feature name	Dimension	Front-end
Japanese linguistic feature vector	389	OpenJTalk [171]
English linguistic feature vector	382	Flite [181]

Table A.2: Japanese linguistic features

Level	Linguistic feature class
Phoneme	previous previous phoneme identity
	previous phoneme identity
	current phoneme identity
	next phoneme identity
	next-next phoneme identity
Mora	difference between accent location and position of current mora
	position of current mora in AP (forward and backward)
Word	part-of-speech (POS) of previous, current, and next word
	inflected forms of previous, current, and next word
	conjugation type of previous, current, and next word
Accent phrase (AP)	number of moras in previous, current, and next AP
	accent type previous, current, and next AP
	is previous AP interrogative or not
	is current AP interrogative or not
	is next AP interrogative or not
	is there pause between previous and current AP
	is there pause between next and current AP
	position of current AP in BG by AP (forward and backward)
	position of current AP in BG by mora (forward and backward)
Breath group (BG)	number of APs in previous, current, and next BG
	number of moras in previous, current, and next BG
	position of current BG by BG (forward and backward)
	position of current BG by AP (forward and backward)
	position of current BG by mora (forward and backward)
Utterance	number of BGs in this utterance
	number of APs in this utterance
	number of moras in this utterance
Frame	position of current frame (forward and backward)
	number of frames

Note: a mora is a phonological unit that consists of phones and determines the timing of Japanese. For example, the word 'Japan' in Japanese has two pronunciations: Ni-ho-n (3 moras) and Ni-p-po-n (4 moras).

The 'accent type' denotes the location where the pitch falls from high to low, i.e., accent nucleus. For example, ha-shi (chopsticks, accent-type=1), a-na-ta (you, accent-type=2). Default low-high accent-type is 0, e.g., ha-shi (bridge).

Table A.3: English linguistic features

Level	Linguistic feature class
Phoneme	previous previous phoneme identity
	previous phoneme identity
	current phoneme identity
	next phoneme identity
	next-next phoneme identity
	position of current phoneme in syllable (forward and backward)
Syllable	number of phonemes in previous, current, and next syllable
	is previous syllable bearing lexical stress (stressed)?
	is previous syllable bearing an English pitch-accent (accented)?
	is current syllable bearing lexical stress (stressed)?
	is current syllable bearing an English pitch-accent (accented)?
	is next syllable bearing lexical stress (stressed)?
	is next syllable bearing an English pitch-accent (accented)?
	position of current syllable in word (forward and backward)
	position of current syllable in phrase (forward and backward)
	number of stressed syllables preceding current syllable in phrase
	number of stressed syllables following current syllable in phrase
	number of accented syllables preceding current syllable in phrase
	number of accented syllables following current syllable in phrase
	distance from previous stressed syllable, accented syllable
	distance to next stressed syllable, accented syllable
Word	POS of previous, current, and next word
	number of syllables in previous, current, and next word
	position of current word in phrase (forward and backward)
	how many content words follow current word in phrase
	how many content words precede current word in phrase
	distance from previous content word
	distance to next content word
Phrase	number of syllables in previous, current, and next phrase
	number of words in previous, current, and next phrase
	position of current phrase in utterance (forward and backward)
	ToBI boundary tone
Utterance	number of syllables, words, phrases
Frame	position of current frame (forward)
	position of current frame (backward)
	number of frames

A.2 Data corpora and acoustic features

This section lists the speech data corpora and the acoustic features used in this thesis. Note that delta components are not used unless specified in the experiment.

Table A.4: Speech corpora used in this thesis

Blizzard Challenge 2011 Nancy voice [182] (English)	Size:	12019 utterances, 16 hours
	Format:	48kHz, 16bit, mono-channel waveform
	Style:	Neural reading style, female
	Division:	training 11019 utterances validation 500 utterances test 500 utterances
ATR Ximera corpora F009 voice [183] (Japanese)	Size:	30016 utterances, 48 hours
	Format:	48kHz, 16bit, mono-channel waveform
	Style:	Neural reading style, female
	Division:	training 29016 utterances validation 500 utterances test 500 utterances

Table A.5: Configuration of acoustic features

Feature name	Dimension	Extractor
Mel-generalized cepstral (MGC)	60	STRAIGHT [15]
Band-aperiodicity (BAP)	25	STRAIGHT
F0	1	Multiple pitch trackers [184]

Note that the raw F0 is converted to the Mel-scale by $1127 \log(1 + \mathrm{F0}/700)$ [185].

Bibliography

[1] Steve Young and Fallside Frank. Speech synthesis from concept: A method for speech output from information systems. *The Journal of the Acoustical Society of America*, 66(3):685–695, 1979.

[2] Paul Taylor. *Text-to-Speech Synthesis*. Cambridge University Press, 2009.

[3] Michael Alexander Kirkwood Halliday, Christian Matthiessen, and Michael Halliday. *An introduction to functional grammar*. Routledge, 2014.

[4] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, et al. Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. In *Proc. ICASSP*, pages 4779–4783, 2017.

[5] Julia Hirschberg. Using discourse context to guide pitch accent decisions in synthetic speech. In *Proc. ESCA Workshop on Speech Synthesis*, pages 367–376, 1991.

[6] Manuel Sam Ribeiro. *Suprasegmental representations for the modeling of fundamental frequency in statistical parametric speech synthesis*. PhD thesis, The University of Edinburgh, 2018.

[7] Paul Taylor. Analysis and synthesis of intonation using the Tilt model. *JASA*, 107(3):1697–1714, 2000.

[8] Santitham Prom-On, Yi Xu, and Bundit Thipakorn. Modeling tone and intonation in Mandarin and English as a process of target approximation. *JASA*, 125(1):405–424, 2009.

[9] Xin Wang, Shinji Takaki, and Junichi Yamagishi. A comparative study of the performance of HMM, DNN, and RNN based speech synthesis systems trained on very large speaker-dependent corpora. In *Proc. SSW9*, pages 125–128, 2016.

[10] Michael S. Scordilis and John N. Gowdy. Neural network based generation of fundamental frequency contours. In *Proc. ICASSP*, pages 219–222 vol.1, 1989.

[11] Yoshinori Sagisaka. On the prediction of global F0 shape for Japanese text-to-speech. In *Proc. ICASSP*, pages 325–328, 1990.

[12] Christof Traber. F0 generation with a data base of natural F0 patterns and with a neural network. In *Proc. ESCA Workshop on Speech Synthesis*, pages 141–144, 1991.

[13] Henning Reetz and Allard Jongman. *Phonetics: Transcription, production, acoustics, and perception*, volume 34. John Wiley & Sons, 2011.

[14] David Talkin. A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518, 1995.

[15] Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain de Cheveigne. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds. *Speech Communication*, 27:187–207, 1999.

[16] Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[17] Arturo Camacho. *SWIPE: A sawtooth waveform inspired pitch estimator for speech and music*. PhD thesis, University of Florida Gainesville, 2007.

[18] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. WORLD: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Trans. on Information and Systems*, 99(7):1877–1884, 2016.

[19] Ingo R. Titze. *Principles of Voice Production*. Prentice Hall, 1994.

[20] Philip Lieberman and Sheila E Blumstein. *Speech physiology, speech perception, and acoustic phonetics*. Cambridge University Press, 1988.

[21] Brian C J Moore. *An Introduction to the Psychology of Hearing*. Brill, 6th edition, 2012.

[22] Carlos Gussenhoven. *The phonology of tone and intonation*. Cambridge University Press, 2004.

[23] Jennifer J Venditti. The J_ToBI model of Japanese intonation. *Prosodic typology: The phonology of intonation and phrasing*, pages 172–200, 2005.

[24] Alejna Brugos, Stefanie Shattuck-Hufnagel, and Nanette Veilleux. Transcribing prosodic structure of spoken utterances with ToBI. *MIT Open Course Ware*, 2006.

[25] Jennifer Cole. Prosody in context: a review. *Language, Cognition and Neuroscience*, 30(1-2):1–31, 2015.

[26] Janet Breckenridge Pierrehumbert. *The phonology and phonetics of English intonation*. PhD thesis, Massachusetts Institute of Technology, 1980.

[27] Janet Pierrehumbert and Julia Bell Hirschberg. The meaning of intonational contours in the interpretation of discourse. *Intentions in communication*, pages 271–311, 1990.

[28] Julia Hirschberg. Studies of intonation and discourse. In *Proc. ESCA Workshop on Prosody*, pages 90–95, 1993.

[29] Mara Breen, Evelina Fedorenko, Michael Wagner, and Edward Gibson. Acoustic correlates of information structure. *Language and cognitive processes*, 25(7-9):1044–1098, 2010.

[30] Mattias Heldner, Jens Edlund, and Julia Hirschberg. Pitch similarity in the vicinity of backchannels. In *Proc. Interspeech*, pages 3054–3057, 2010.

[31] Geoffrey N Leech. *The pragmatics of politeness*. Oxford Studies in Sociolinguis, 2014.

[32] Jonathan Culpeper. It's not what you said, it's how you said it!": Prosody and impoliteness. *Discursive approaches to politeness*, pages 57–83, 2011.

[33] David Crystal. Prosodic and paralinguistic correlates of social categories. *Social anthropology and language*, pages 185–206, 1971.

[34] Haver C Cuerie. A projection of socio-linguistics: The relationship of speech to social status. *Southern Journal of Communication*, 18(1):28–37, 1952.

[35] Robert W Frick. Communicating emotion: The role of prosodic features. *Psychological Bulletin*, 97(3):412, 1985.

[36] Kim E. A. Silverman, Mary E. Beckman, John F. Pitrelli, Mari Ostendorf, Colin W. Wightman, Patti Price, Janet B. Pierrehumbert, and Julia Hirschberg. ToBI: a standard for labeling English prosody. In *Proc. ICSLP*, pages 867–870, 1992.

[37] Thierry Dutoit. *An Introduction to Text-to-speech Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[38] Mark Y Liberman and Kenneth W Church. Text analysis and word pronunciation in text-to-speech synthesis. *Advances in speech signal processing*, pages 791–831, 1992.

[39] Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. Normalization of non-standard words. *Computer speech & language*, 15(3):287–333, 2001.

[40] Alan W Black, Kevin Lenzo, and Vincent Pagel. Issues in building general letter to sound rules. In *Proc. SSW3*, 1998.

[41] Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 171–189. Springer, 2011.

[42] Julia Hirschberg and Pilar Prieto. Training intonational phrasing rules automatically for English and Spanish text-to-speech. *Speech Communication*, 18(3):281–290, 1996.

[43] Julia Hirschberg. Pitch accent in context predicting intonational prominence from text. *Artificial Intelligence*, 63(1):305–340, 1993.

[44] Jan P.H. Van Santen. Assignment of segmental duration in text-to-speech synthesis. *Computer Speech & Language*, 8(2):95–128, 1994.

[45] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proc. ICASSP*, pages 373–376, 1996.

[46] Dennis H Klatt. Review of text-to-speech conversion for English. *The Journal of the Acoustical Society of America*, 82(3):737–793, 1987.

[47] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech synthesis based on hidden Markov models. *Proceedings of the IEEE*, 101(5):1234–1252, 2013.

[48] Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *Speech Communication*, 51:1039–1064, 2009.

[49] Keiichi Tokuda, Takao Kobayashi, Takashi Masuko, and Satoshi Imai. Mel-generalized cepstral analysis a unified approach. In *Proc. ICSLP*, pages 1043–1046, 1994.

[50] Fumitada Itakura. Line spectrum representation of linear predictor coefficients of speech signals. *The Journal of the Acoustical Society of America*, 57(S1):S35–S35, 1975.

[51] Gilles Degottex, Pierre Lanchantin, and Mark Gales. A log domain pulse model for parametric speech synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2017.

[52] Tuomo Raitio, Antti Suni, Junichi Yamagishi, Hannu Pulakka, Jani Nurminen, Martti Vainio, and Paavo Alku. HMM-based speech synthesis utilizing glottal inverse filtering. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1):153–165, 2011.

[53] Lauri Juvela, Bajibabu Bollepalli, Manu Airaksinen, and Paavo Alku. High-pitched excitation generation for glottal vocoding in statistical parametric speech synthesis using a deep neural network. In *Proc. ICASSP*, pages 5120–5124, 2016.

[54] Thomas Drugman and Tuomo Raitio. Excitation modeling for HMM-based speech synthesis: breaking down the impact of periodic and aperiodic components. In *Proc. ICASSP*, pages 260–264, 2014.

[55] Takayoshi Yoshimura, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Mixed excitation for HMM-based speech synthesis. In *Proc. Eurospeech*, pages 2263–2266, 2001.

[56] Ranniery Maia, Tomoki Toda, Heiga Zen, Yoshihiko Nankaku, and Keiichi Tokuda. An excitation model for HMM-based speech synthesis based on residual modeling. In *Proc. SSW6*, pages 131–136, 2007.

[57] Daniel Erro, Inaki Sainz, Eva Navas, and Inma Hernaez. Harmonics plus noise model based vocoder for statistical parametric speech synthesis. *IEEE Journal of Selected Topics in Signal Processing*, 8(2):184–194, 2014.

[58] Qiong Hu, Korin Richmond, Junichi Yamagishi, and Javier Latorre. An experimental comparison of multiple vocoder types. In *Proc. SSW8*, pages 135–140, 2013.

[59] Akira Tamamori, Tomoki Hayashi, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda. Speaker-dependent WaveNet vocoder. In *Proc. Interspeech*, pages 1118–1122, 2017.

[60] Zhen Hua Ling, Shi Yin Kang, Heiga Zen, Andrew Senior, Mike Schuster, Xiao Jun Qian, Helen M Meng, and Li Deng. Deep learning for acoustic

modeling in parametric speech generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine*, 32(3):35–52, 2015.

[61] Heiga Zen, Alan Senior, and Martin Schuster. Statistical parametric speech synthesis using deep neural networks. In *Proc. ICASSP*, pages 7962–7966, 2013.

[62] Yuchen Fan, Yap Qian, Feilong Xie, and Frank K. Soong. TTS synthesis with bidirectional LSTM based recurrent neural networks. In *Proc. Interspeech*, pages 1964–1968, 2014.

[63] Oliver Watts, Gustav Eje Henter, Thomas Merritt, Zhizheng Wu, and Simon King. From HMMs to DNNs: where do the improvements come from? In *Proc. ICASSP*, pages 5505–5509, 2016.

[64] Yuxuan Wang, R.J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech*, pages 4006–4010, 2017.

[65] Hiroya Fujisaki and Keikichi Hirose. Analysis of voice fundamental frequency contours for declarative sentences of Japanese. *Journal of the Acoustical Society of Japan (E)*, 5(4):233–242, 1984.

[66] Hansjörg Mixdorff. *Intonation patterns of German-model-based quantitative analysis and synthesis of F0 contours*. PhD thesis, TU Dresden, 1998.

[67] Alan Black and Andnzw Hunt. Generating F0 contours from ToBI labels using linear regression. In *Proc. ICSLP*, volume 3, pages 1385–1388, 1996.

[68] Manuel Sam Ribeiro, Junichi Yamagishi, and Robert AJ Clark. A perceptual investigation of wavelet-based decomposition of f0 for text-to-speech synthesis. In *Proc. Interspeech*, 2015.

[69] Manuel Sam Ribeiro and Robert A J Clark. A multi-level representation of F0 using the continuous wavelet transform and the discrete cosine transform. In *Proc. ICASSP*, pages 4909–4913, 2015.

[70] Christophe d'Alessandro and Piet Mertens. Automatic pitch contour stylization using a model of tonal perception. *Computer Speech & Language*, 9(3):257 – 288, 1995.

[71] Kurt E Dusterhoff, Alan W Black, and Paul A Taylor. Using decision trees within the Tilt intonation model to predict F0 contours. *Proc. Eurospeech*, pages 1627–1630, 1999.

[72] Keikichi Hirose, Kentaro Sato, Yasufumi Asano, and Nobuaki Minematsu. Synthesis of F0 contours using generation process model parameters predicted from unlabeled corpora: Application to emotional speech synthesis. *Speech communication*, 46(3):385–404, 2005.

[73] Hao Liu. *Fundamental frequency modelling: an articulatory perspective with target approximation and deep learning*. PhD thesis, UCL (University College London), 2017.

[74] Janet Pierrehumbert. Synthesizing intonation. *The Journal of the Acoustical Society of America*, 70(4):985–995, 1981.

[75] Keiichi Tokuda, Takashi Masuko, Noboru Miyazaki, and Takao Kobayashi. Multi-space probability distribution HMM. *IEICE Trans. on Information and Systems*, 85(3):455–464, 2002.

[76] Kai Yu and Steve Young. Continuous F0 modeling for HMM based statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(5):1071–1079, 2011.

[77] Takayoshi Yoshimura, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadasgu Kitamura. Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis. In *Proc. Eurospeech*, pages 2347–2350, 1999.

[78] Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proc. ICASSP*, pages 936–939, 2000.

[79] Xin Wang, Minghui Dong, and Zhenhua Ling. A full training framework of cross-stream dependence modelling for HMM-based singing voice synthesis. In *Proc. ICASSP*, pages 5165–5169, March 2016.

[80] Chengcheng Wang, Zhenhua Ling, Bufan Zhang, and Lirong Dai. Multi-layer F0 modeling for HMM-based speech synthesis. In *Proc. ISCSLP*, pages 1–4. IEEE, 2008.

[81] Yao Qian, Zhizheng Wu, Boyang Gao, and Frank K Soong. Improved prosody generation by maximizing joint probability of state and longer units. *IEEE Trans. on Audio, Speech, and Language Processing*, 19(6):1702–1710, 2011.

[82] Ming Lei, Yijian Wu, Frank K Soong, Zhen Hua Ling, and Lirong Dai. A hierarchical F0 modeling method for HMM-based speech synthesis. In *Proc. Interspeech*, pages 2170–2173, 2010.

[83] Xiaojun Zou, Xiao Bao, and Lidong Luo. Integration of intonation in F0 trajectory prediction using MSD-HMMs. In *Proc. Speech Prosody*, page 952, 2010.

[84] Shinji Takaki, SangJin Kim, Junichi Yamagishi, and Jongjin Kim. Multiple feed-forward deep neural networks for statistical parametric speech synthesis. *Proc. Interspeech*, pages 2242–2246, 2015.

[85] Heiga Zen and Haşim Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Proc. ICASSP*, pages 4470–4474, 2015.

[86] Heiga Zen and Andrew Senior. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *Proc. ICASSP*, pages 3844–3848, 2014.

[87] Raul Fernandez, Asaf Rendel, Bhuvana Ramabhadran, and Ron Hoory. Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks. In *Proc. Interspeech*, pages 2268–2272, 2014.

[88] Sin Horng Chen, Shaw Hwa Hwang, and Yih Ru Wang. An RNN-based prosodic information synthesizer for Mandarin text-to-speech. *IEEE Transactions on Speech and Audio Processing*, 6(3):226–239, 1998.

[89] ITU-T. Methods for objective and subjective assessment of quality, 1996.

[90] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[91] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[92] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[93] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[94] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. The MIT Press, 1986.

[95] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[96] Terrence J Sejnowski and Charles R Rosenberg. Parallel networks that learn to pronounce English text. *Complex systems*, 1(1):145–168, 1987.

[97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[98] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[99] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749, 2005.

[100] Hiroshi Sawada, Hirokazu Kameoka, Shoko Araki, and Naonori Ueda. Multichannel extensions of non-negative matrix factorization with complex-valued data. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):971–982, 2013.

[101] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *Proc. ICLR*, Apr 2016.

[102] Christopher M. Bishop. Mixture Density Networks. Technical report, Aston University, 2004.

[103] Mike Schuster. Better generative models for sequential data problems: Bidirectional recurrent mixture density networks. In *Proc. NIPS*, pages 589–595, 1999.

[104] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[105] Eric B Baum and Frank Wilczek. Supervised learning of probability distributions by neural networks. In *Neural information processing systems*, pages 52–61, 1988.

[106] Michael D Richard and Richard P Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483, 1991.

[107] Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. Finding function in form: Compositional character models for open vocabulary word representation. In *Proc. EMNLP*, pages 1520–1530, 2015.

[108] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. ACL (Volume 1: Long Papers)*, volume 1, pages 1064–1074, 2016.

[109] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Proc. ICASSP*, pages 4460–4464, 2015.

[110] Zhizheng Wu and Simon King. Investigating gated recurrent networks for speech synthesis. In *Proc. ICASSP*, pages 5140–5144. IEEE, 2016.

[111] Srikanth Ronanki, Oliver Watts, and Simon King. A hierarchical encoder-decoder model for statistical parametric speech synthesis. In *Proc. Interspeech*, pages 1133–1137, 2017.

[112] Shiyin Kang and Helen Meng. Statistical parametric speech synthesis using weighted multi-distribution deep belief network. In *Proc. Interspeech*, pages 1959–1963, 2014.

[113] Manuel Sam Ribeiro, Oliver Watts, Junichi Yamagishi, and Robert AJ Clark. Wavelet-based decomposition of f0 as a secondary task for DNN-based speech synthesis with multi-task learning. In *Proc. ICASSP*, pages 5525–5529. IEEE, 2016.

[114] Xiang Yin, Ming Lei, Yao Qian, Frank K Soong, Lei He, Zhen-Hua Ling, and Li-Rong Dai. Modeling F0 trajectories in hierarchically structured deep neural networks. *Speech Communication*, 76:82–92, 2016.

[115] Srikanth Ronanki, Gustav Eje Henter, Zhizheng Wu, and Simon King. A template-based approach for speech synthesis intonation generation using LSTMs. In *Proc. Interspeech*, pages 2463–2467, 2016.

[116] Geoffrey E Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[117] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. EMNLP*, pages 1724–1734, 2014.

[118] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[119] Felix Gers. *Long Short-Term Memory in Recurrent Neural Networks*. PhD thesis, Universität Hannover, 2001.

[120] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[121] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A Clockwork RNN. In *Proc. ICML*, pages 1863–1871, 2014.

[122] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. In *Proc. Deep Learning Workshop*, 2015.

[123] Bo Chen, Zhehuai Chen, Jiachen Xu, and Kai Yu. An investigation of context clustering for statistical speech synthesis with deep neural network. In *Proc. Interspeech*, pages 2212–2216, 2015.

[124] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If ResNets are the answer, then what is the question? In *Proc. ICML*, pages 342–350, 2017.

[125] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

[126] Khe Chai Sim. On constructing and analysing an interpretable brain model for the DNN based on hidden activity patterns. In *Proc. ASRU*, pages 22–29, 2015.

[127] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, pages 249–256, 2010.

[128] Oliver Watts, Junichi Yamagishi, and Simon King. The role of higher-level linguistic features in HMM-based speech synthesis. In *Proc. Interspeech*, pages 841–844, 2010.

[129] Gustav E. Henter, Thomas Merritt, Matt Shannon, Catherine Mayo, and Simon King. Measuring the perceptual effects of modelling assumptions in speech synthesis using stimuli constructed from repeated natural speech. In *Proc. Interspeech*, pages 1504–1508, 2014.

[130] M Shannon. *Probabilistic acoustic modelling for parametric speech synthesis*. PhD thesis, University of Cambridge, 2014.

[131] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[132] Heiga Zen, Mark JF Gales, Yoshihiko Nankaku, and Keiichi Tokuda. Product of experts for statistical parametric speech synthesis. *IEEE Trans. on Audio, Speech, and Language Processing*, 20(3):794–805, 2012.

[133] Heiga Zen, Keiichi Tokuda, and Tadashi Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Computer Speech & Language*, 21(1):153–173, 2007.

[134] Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. Trajectory training considering global variance for speech synthesis based on neural networks. In *Proc. ICASSP*, pages 5600–5604, 2016.

[135] Brendan J. Frey. *Graphical Models for Machine Learning and Digital Communication*. A Bradford book. Bradford book, 1998.

[136] Sanjit Kumar Mitra and Yonghong Kuo. *Digital signal processing: a computer-based approach*, volume 2. McGraw-Hill Higher Education New York, 2006.

[137] R Viswanathan and John Makhoul. Quantization properties of transmission parameters in linear predictive systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(3):309–321, 1975.

[138] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[139] John G Proakis and Dimitris G Manolakis. *Digital signal processing: principles, algorithms, and applications*. Upper Saddle River, NJ: Prentice Hall, 1996.

[140] Sophocles J. Orfanidis. *Optimum Signal Processing: An Introduction*. Macmillan publishing company, 1988.

[141] Frank K. Soong and Bling-Hwang Juang. Line spectrum pair (LSP) and speech data compression. In *Proc. ICASSP*, volume 9, pages 37–40. IEEE, 1984.

[142] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proc. ICML*, pages 1530–1538, 2015.

[143] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Proc. NIPS*, pages 4743–4751, 2016.

[144] Mark J.F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

[145] Junichi Yamagishi, Takao Kobayashi, Yuji Nakano, Katsumi Ogata, and Juri Isogai. Analysis of speaker adaptation algorithms for HMM-based speech synthesis and a constrained SMAPLR adaptation algorithm. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(1):66–83, 2009.

[146] Jakub M Tomczak and Max Welling. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

[147] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[148] Xin Wang, Shinji Takaki, and Junichi Yamagishi. An autoregressive recurrent mixture density network for parametric speech synthesis. In *Proc. ICASSP*, pages 4895–4899, 2017.

[149] Xin Wang, Jaime Lorenzo-Trueba, Shinji Takaki, Lauri Juvela, and Junichi Yamagishi. A comparison of recent waveform generation and acoustic modeling methods for neural-network-based speech synthesis. In *Proc. ICASSP*, pages 4804–4808, 2018.

[150] Neville J Ford, Dmitry V Savostyanov, and Nickolai L Zamarashkin. On the decay of the elements of inverse triangular toeplitz matrices. *SIAM Journal on Matrix Analysis and Applications*, 35(4):1288–1302, 2014.

[151] Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

[152] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112, 2014.

[153] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE, 2015.

[154] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Proc. NIPS*, pages 2199–2207, 2016.

[155] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proc. CoNLL*, pages 10–21, 2016.

[156] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proc. AISTATS*, volume 5, pages 246–252, 2005.

[157] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[158] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *Proc. ICLR*, 2016.

[159] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *Proc. ICLR*, pages –, 2017.

[160] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[161] Nobuaki Minematsu, Ryo Kuroiwa, Keikichi Hirose, and Michiko Watanabe. CRF-based statistical learning of Japanese accent sandhi for developing Japanese text-to-speech synthesis systems. In *Proc. SSW6*, 2007.

[162] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proc. NIPS*, page to appear, 2017.

[163] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Proc. NIPS*, pages 3483–3491, 2015.

[164] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[165] Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. In *Proc. ICLR*, 2017.

[166] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[167] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, page unknown, 2014.

[168] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[169] Kou Tanaka, Hirokazu Kameoka, and Kazuho Morikawa. VAE-SPACE: Deep generative model for voice fundamental frequency contours. In *Proc. ICASSP*, pages 5779–5793, 2018.

[170] Nicolas Obin and Julie Beliao. Sparse coding of pitch contours with deep auto-encoders. In *Proc. Speech Prosody*, page To appear, 2018.

[171] HTS Working Group. The Japanese TTS System 'Open JTalk', 2015.

[172] Dwight Bolinger. Accent is predictable (if you're a mind-reader). *Language*, pages 633–644, 1972.

[173] Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[174] Christopher M Bishop. Regularization and complexity control in feed-forward networks. Technical Report NCRG/95/22, Aston University, 1995.

[175] Mary E. Beckman and Gayle Ayers. Guidelines for ToBI labelling. *The OSU Research Foundation*, 3, 1997.

[176] Jan van Santen and Bernd Möbius. Modeling pitch accent curves. In *Intonation: Theory, Models and Applications*, 1997.

[177] Rasmus Dall, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. Redefining the linguistic context feature set for HMM and DNN TTS through position and parsing. In *Proc. Interspeech*, pages 2851–2855, 2016.

[178] Brian Patton, Yannis Agiomyrgiannakis, Michael Terry, Kevin Wilson, Rif A Saurous, and D Sculley. Automos: Learning a non-intrusive assessor of naturalness-of-speech. *arXiv preprint arXiv:1611.09207*, 2016.

[179] Robert AJ Clark and Kurt E Dusterhoff. Objective methods for evaluating synthetic intonation. In *Proc. Eurospeech*, 1999.

[180] Kenneth Church. A pendulum swung too far. *Linguistic Issues in Language Technology*, 6(5):1–27, 2011.

[181] HTS Working Group. The English TTS system Flite+HTS_engine, 2014.

[182] Simon King and Vasilis Karaiskos. The Blizzard Challenge 2011. In *Proc. Blizzard Challenge Workshop*, pages 1–10, 2011.

[183] Hisashi Kawai, Tomoki Toda, Jinfu Ni, Minoru Tsuzaki, and Keiichi Tokuda. XIMERA: A new TTS from ATR based on corpus-based technologies. In *Proc. SSW5*, pages 179–184, 2004.

[184] Lauri Juvela, Xin Wang, Shinji Takaki, SangJin Kim, Manu Airaksinen, and Junichi Yamagishi. The NII speech synthesis entry for Blizzard Challenge 2016. In *Proc. Blizzard Challenge Workshop*, 2016.

[185] Douglas O'Shaughnessy. *Speech Communications: Human and Machine*. Institute of Electrical and Electronics Engineers, 2000.