Data Augmentation Approaches for Automatic Speech Recognition Using Text-to-Speech

Sei UENO

Abstract

Automatic speech recognition (ASR) systems are widely used as an aid and basis for speech-based human-human and human-robot communications. ASR systems need to achieve high accuracy with small latency, and they should be customized for the application domains and topics. Thanks to the development of deep neural networks (DNNs), *end-to-end ASR models* have been intensively investigated recently. The end-to-end ASR models convert speech into words faster with a simpler architecture than the conventional models. However, they require a large amount of paired data of speech and transcription for training.

To alleviate this problem, this thesis addresses data augmentation for the ASR model using a text-to-speech (TTS) system. In this framework, artificial speech data are generated from text-only data using a TTS system to prepare pseudo paired datasets. In the naive implementation, however, we observe that the improvement of ASR performance is limited compared to the case using real speech data. This is because there are serious mismatches between synthesized data and real data. In this study, we investigate three data augmentation approaches to solve the problem.

In Chapter 3, we adopt a waveform-based approach. In general, a TTS system is composed of two models: a text-to-mel network to generate log Mel-scale filterbank (lmfb) features and a vocoder network to convert the generated lmfb features into a waveform. We observe that the lmfb feature produced by the text-to-mel model is blurry, particularly on the time dimension. This problem is mitigated by introducing the vocoder to generate speech of better quality or spectrogram of better time-resolution. This makes it possible to train waveform-input end-to-end ASR. Here we use CNN filters and apply a masking method similar to SpecAugment. We compare the waveform-input model with two kinds of lmfb-input models: (1) lmfb features are directly generated by TTS, and (2) lmfb features are converted from the waveform generated by TTS. Experimental evaluations show the effectiveness of the combination of waveform-output TTS and the waveform-input end-to-end ASR model for improving the ASR performance.

In Chapter 4, we propose a data augmentation approach via a discrete speech representation. In general TTS, a text-to-mel network predicts continuous value (lmfb features), which is not an easy task. It is also not guaranteed that the generated lmfb features exist in the real world. In this work, we introduce a discrete speech representation, which TTS model predicts instead of lmfb features. We expect that the use of the discrete representation based on vq-wav2vec not only makes TTS training easier but also mitigates the mismatch with real data. The ASR model also uses the discrete representation as its input. Experimental evaluations show that the proposed method outperforms the data augmentation method using the conventional TTS. We found that it reduces speaker dependency, and the generated features are distributed more closely to the real features.

In Chapter 5, we propose a phone-informed post-processing network that refines lmfb features without using the vocoder. The widely-used procedure, as presented in Chapter 3, first generates an lmfb feature from text data, then converts it into a waveform, and converts it again to an lmfb feature. These conversions take a long time and are not necessary for data augmentation. In this work, we propose a mel-to-mel network that directly refines the lmfb features. The proposed network consumes not only lmfb features but also phone information for refinement. This approach takes less time than converting to the waveform domain. Experimental evaluations in domain adaptation show that the proposed network achieves better improvement of ASR performance than using the vocoder network with much faster processing time. It is also shown that the use of phone information is critical for the improvement.

Chapter 6 concludes this thesis with a comparison of the three works, investigation on mismatch of TTS, and a brief look at future work.

Acknowledgment

This work has been accomplished at Speech and Audio Processing Laboratory, Graduate School of Informatics, Kyoto University. I express my gratitude to all people who helped me and this work.

First of all, I would like to express my special thanks and appreciation to my supervisor Professor Tatsuya Kawahara. He gave me the opportunity to study in the Speech and Audio Processing Lab. His comments have been essential and insightful for advancing this work. I have been supervised by him for 5 years in my master and Ph.D. course. When I was an undergraduate student at Doshisha University, I met him for the first time and asked him about his laboratory. After entering the master course, he has patiently supervised me about research from various viewpoints. In particular, he supervised me on how to write a paper in much detail. This work would not have been completed without his continuing engagement and generous support.

I also express my special thanks and appreciation to Associate Professor Kazuyoshi Yoshii for a lot of insightful comments on my research. He gave me various advice from perspectives other than deep learning.

Furthermore, I express my special thanks and appreciation to the members of my dissertation committee, Professor Ko Nishino and Professor Sadao Kurohashi, for their time and valuable comments.

This study cannot be accomplished without Mr. Masato Mimura and Dr. Shinsuke Sakai. They gave me insightful advice from their deep knowledge of machine learning and speech recognition. They also gave much time to meaningful discussions.

I would like to thank Dr. Atsunori Ogawa and Dr. Shoko Araki, who are members of NTT Communication Science Laboratories, for their hosting my internship.

I wish to deeply thank Professor Hiroshi Saruwatari, Assistant Professor Shinnosuke Takamichi, and Project Research Associate Yuki Saito, who are members of The University of Tokyo, for their hosting my research stay and insightful advice about text-to-speech.

I also express my thanks to the members of Speech and Audio Processing Lab. I am grateful for comments and supports from Assistant Professor Koji Inoue, Ms. Mayumi Abe, Dr. Hirofumi Inaguma, Mr. Kak Soky, Mr. Kohei Matsuura, Mr. Hayato Futami, Mr. Jumon Nozaki, Mr. Han Feng, Mr. Trung V. Dang, Dr. Yoshiaki Bando, Dr. Ryo Nishikimi, Dr. Kohei Sekiguchi, and all the other members.

I am also grateful to Professor Emeritus Seiichi Yamamoto and Associate Professor Tsuneo Kato, who are members of Doshisha University, and Associate Professor Komei Sugiura, who is a member of Keio University. They introduced me the area of signal processing when I was at Doshisha University.

This work was supported by the Japan Society for the Promotion and Science (JSPS) with its financial support as Fellowship for Young Scientists (DC1).

I wish to thank my friends since I was at Doshisha University. I especially thank Mr. Yu Manabe.

Lastly, I sincerely thank my family for their support and encouragement for my long student life.

Contents

Chapter 1

Introduction

1.1 Background

Speech is one of the fundamental ways of human communication. The recent prevalence of social media and online meeting tools makes it easier to communicate with each other even in distance. In particular, under the pandemic, these tools become essential for work, education, and even daily lives. They include text-based communication and speech-based communication functions (*e.g.*, Zoom and WebEx). For work and education, speech is still essential for communicating smoothly. During meetings and lectures, people often need transcription or captions to understand easily. Transcribing these speeches by humans requires an abundant amount of effort and cost, and thus the automatic transcription is required. On the other hand, speech is used for communication not only with other people but also with machines. Voice assistants for a smartphone (*e.g.*, Siri and Cortana) and smart speakers (*e.g.*, Google Home and Amazon Alexa) have been widely used. People generally use their speech to make a simple command or query for these assistants. Moreover, social communication robots are being developed to conduct not only simple commands but also general conversations. These robots need to recognize natural human speech. In summary, to recognize or transcribe human speech is necessary for many applications in human-to-human and human-to-robot communication.

Automatic speech recognition (ASR) systems are increasingly becoming a vital module in satisfying these social needs. We need high-performance ASR since the ASR performance affects the downstream tasks such as natural language understanding, translation, and dialogue. In the several downstream tasks, recognizing domain-specific words is critical. For example, a dialogue system needs to recognize the name of the place for smooth communication when a speaker talks about travel. Therefore, we need to customize the ASR system for the target domain task (*e.g.*, change the training dataset and architecture). Moreover, low latency of the ASR is often required for interaction and real-time applications.

1.2 Task Formulation

Before moving on to problems to be addressed in this thesis, we formulate two fundamental tasks for speech processing: automatic speech recognition (ASR) and text-to-speech (TTS).

1.2.1 Automatic Speech Recognition (ASR)

Automatic speech recognition (ASR) is a task to transcribe speech into text. Thus, it can be called speech-to-text. The ASR task is a many-to-one mapping problem. The input has various kinds of speech (*e.g.*, recording environments, speaker attributes, emotion) when a speaker utters the same text, and the ASR systems need to recognize them correctly. The ASR systems have been investigated for many decades [1–4]. The initial studies of the ASR systems were based on pattern matching such as dynamic programming (DP), and the effective acoustic features were investigated. However, these methods were not sufficient to model speech diversity such as gender, age, and so on. Then statistical models such as Hidden Markov model (HMM) model have been introduced. Gaussian mixture models (GMM) were used to model each state of acoustic patterns in the HMM. The acoustic model (AM) is combined with another statistical model called the language model (LM). The AM maps acoustic features into a phone sequence, while the LM learns a prior probability of a word sequence. Using the deep neural networks (DNNs) instead of GMM has drastically improved the performance of

AM. In addition to AM, introducing the DNN improves the ASR performance in LM. These hybrid systems are widely used for many applications. However, we must carefully design AM, LM, and a pronunciation dictionary module to achieve high performance, and each module is optimized independently.

Recently, end-to-end architecture has been investigated thanks to the progress of DNN and computing resources. The end-to-end model directly converts the acoustic feature into a word or subword sequence. While the hybrid model needs to train the AM and LM separately, end-to-end models unify these models in a single neural network architecture, which can decode the speech rapidly. It learns the ASR task efficiently as the whole model is optimized based on the unified criterion. Actually, the end-to-end models have realized better performance than the conventional hybrid systems when a large amount of training data is available.

1.2.2 Text-to-Speech (TTS)

Text-to-speech (TTS) is a task to generate natural-sounding artificial speech from a given text. The TTS systems also have a lot of applications (*e.g.*, readout of news articles and response for the robot). They have been investigated for many decades, as in the ASR. Since the 1990s, unit selection synthesis has been widely used [5, 6]. In inference, the unit selection synthesis searches for the speech segments stored in database that match a given text and produces speech by concatenating these segments together. While it can generate high intelligible speech, it requires huge recording database and the generated speech is natural but discontinuous.

A statistical parametric synthesis is investigated to address the drawbacks of the unit selection synthesis [7, 8]. It first generates the acoustic parameters that are necessary to produce speech and then synthesizes speech from the generated acoustic parameters. The statistical parametric synthesis needs to learn common characteristics of speech from various kinds of speech samples. It models the acoustic parameters using HMMs. Compared with the unit selection synthesis, the statistical parametric synthesis can generate speech with prosody and the model can be trained with a small amount of data. However, the generated speech has low intelligibility and we need highly technical knowledge to compose the model.

With the recent development of DNNs, TTS can also be realized with simple models referred to as end-to-end TTS without relying on complex feature engineering. They achieve speech quality close to natural human speech [9]. The end-to-end TTS systems generally have two components: text-to-mel network and vocoder (mel-to-waveform). The text-to-mel network generates log Mel-filter bank (lmfb) features as acoustic features from a given text. The vocoder converts the generated lmfb feature into a waveform that people can hear or evaluate.

1.3 Problems of End-to-End ASR Models

The end-to-end model realizes high performance ASR and faster inference. However, there are problems which are described below.

1.3.1 Need for a Large Amount of Paired Data of Speech and Transcription

First, we need to prepare pair data of speech and transcription, as the end-to-end ASR model is trained with acoustic features and its corresponding word or subword labels. In other words, it is necessary to transcribe the speeches for training data. Although the AM of the hybrid ASR model also needs the paired data, we do not need to prepare such a large amount of data compared to the end-to-end model. This is because the network size of AM is small, and the AM can be trained with a large amount of target label samples since each acoustic frame has one target label. On the other hand, the end-to-end model needs a rich profusion of the paired data since the network size is large. Moreover, in an end-to-end manner, certain lengths of acoustic features have one target label. As a result, a large amount of training data is required for the end-to-end model to achieve sufficient performance. For example, the end-to-end ASR model trained using the JNAS dataset [10] that has 40-hour speech and transcriptions yields much worse ASR performance than the hybrid DNN-HMM model. When using the CSJ dataset [11] with 600-hour paired data, the ASR performance of the end-to-end model is better than that of the DNN-HMM model. In practice, we must prepare over 100 hours of paired data at least. It is not easy to prepare such a large amount of data for spontaneous speech such as meetings since it costs a lot of manual works.

1.3.2 Data Sparseness and Uneven Distributions

Second, we suffer from data sparseness and uneven distributions in several aspects. In general, text data is available easier than speech data. The end-to-end model needs to learn acoustic and linguistic patterns, but it learns just the latter when using text-only data. It results in a high dependency on the LM function.

Moreover, the word or subwords have a large number of classes, and the distribution is biased toward common words, and other words are used infrequently. The critical words (*e.g.*, technical words) for the downstream task are generally uncommon words, and it is not easy to prepare the data including these words.

Preparing the paired data of the readout speech is readily available since the text is prepared in advance and a speaker speaks following the text. On the other hand, it is difficult to prepare the paired data for more natural speeches such as conversations or lectures because we must manually transcribe the speech after recording. As a result, the domain of the paired data is biased toward readout data.

1.4 Data Augmentation Using Speech-only or Text-only Data

To solve these problems of the end-to-end ASR model, the use of unpaired data has received much attention since developing the end-to-end models. To utilize speech-only data, unsupervised learning or semi-supervised approaches have been attractive. We can train an unsupervised model using contrastive representation learning [12] and variational autoencoder (VAE) [13] and extract an acoustic feature from raw waveform (only) data. Wav2vec 2.0 [14] achieves higher performance than a conventional ASR model using about 60,000 hours of speech data. In semi-supervised approaches, a teacher ASR model generates labels for the speech-only data and then a student ASR model can be trained by using them [15,16]. Noisy students [17] also improved the end-to-end ASR performance. However, we need a large amount of speech-only data and training these models is slow because there is no supervision labels.

On the other hand, we can easily train an external language model by using the text-only data. Given a previous label sequence, the language model predicts a target linguistic label at the next timestep. It can be trained using text-only data, and we have two approaches to use it for the ASR model. First, we apply the external language model with shallow fusion [18]. We calculate the final score based on not only the probability of the ASR but also that of the LM. Second, we apply the external language model for rescoring. In this approach, the ASR model generate several candidates from a given speech. The external LM then calculates the probabilities of these candidates and selects the best one. As the ASR model finishes initial decoding and the LM can consider a bidirectional context, we can apply the bidirectional LM such as BERT [19]. Although these methods do not have to change the ASR architecture, the performance gain is limited by the baseline ASR candidates. Moreover, for enhancing ASR models, it is not easy to recognize unknown words because the probabilities of these words tend to be low.

1.5 Data Augmentation for ASR Using TTS

Recently there are also studies on the use of the text-to-speech (TTS) model. One of them is to integrate the ASR and TTS models [20,21] referred to as speech chain. Given the unlabeled speech features, ASR transcribes the unlabeled input speech, while TTS reconstructs the original speech waveform based on the output text from ASR. Given only the text input, TTS generates speech waveform, while ASR also reconstructs the original text transcription given the synthesized speech.

This method improves the ASR performance when the speakers are specific and training data is small. However, it is not suitable for general ASR settings since the model must train both ASR and TTS. Moreover, it must be noted that TTS and ASR are not cyclic but should be rather adversarial in that TTS should generate a variety of speech data that the baseline ASR cannot cope with.

In this thesis, we focus on the data augmentation that generates speech data using a text-to-speech (TTS) model from the text-only data and train the ASR model with the generated data [22–32]. In this framework, we prepare not only the ASR system but also the TTS system. The TTS model generates speech data from the arbitrary unpaired text, which can be used as training data of ASR.

This framework has several advantages compared with other approaches.

- **Simple**: firstly, we do not have to consider the difference between the generated data format and the natural speech data format. We can assume both data as acoustic features, and we do not need the additional function to bridge the discrepancy between different modalities.
- **Direct**: secondly, the ASR models can directly learn the acoustic patterns and vocabulary of the target domain when the data are generated from the text of the target domain.
- **Independency**: finally, we can design the ASR and TTS system independently. It is often necessary to compose the ASR model for each target domain, and the ASR system will need to be tuned several times.

We observe that this framework yields the ASR performance improvement on several tasks. However, the improvement is limited, particularly when compared to using real speech. There are mismatches between the synthesized and real features. The synthesized feature is much less diverse than the real feature. For instance, while a TTS system generates the exact same speech given the same text, a human speaks differently even when he or she reads the same text. Moreover, the TTS system sometimes fails to generate speech at all, preventing the ASR model from correctly learning the acoustic patterns.

1.6 Approaches

In this thesis, we present three data augmentation approaches for ASR using TTS.

1.6.1 Synthesizing Waveform as Training Data for ASR

In Chapter 3, we present a waveform-based data augmentation approach which uses waveform domain for both ASR and TTS. We found the quality of speech data generated by TTS is not realistic; we observe the spectrum is blurry, particularly on the time dimension. It is likely to make the improvement of ASR limited. Actually, the time resolution of the artificial spectrum is not sufficient because the TTS model generates the spectrum of several contiguous frames at one decoding step for stable training and fast inference. While it is not easy to fix this problem on the text-to-mel network side, we can generate a high-quality speech waveform with a state-of-the-art vocoder. This waveform-based data augmentation scheme allows for another option of designing a complete end-to-end ASR model from waveform to a word sequence. Here, we introduce CNN-based feature extraction as a front-end. It is compared with the lmfb-based ASR systems, where (1) lmfb is generated from the waveform, and (2) lmfb is generated by the text-to-mel network. This comparison of the waveform-based model and the two kinds of the lmfb-based model is a major contribution of this Chapter. As data masking methods such as SpecAugment [17] significantly affect the lmfb-based ASR systems, we also design a masking method in the waveform-based end-to-end ASR.

1.6.2 Generating ASR Features via Discrete IDs

In Chapter 4, we present a data augmentation approach via a discrete speech representation. It has been found that the synthesized lmfb features have a serious mismatch with the real speech features. One reason is difficulty in predicting the continuous value of lmfb features. Moreover, it is not guaranteed that the generated lmfb features exist in real because the end-to-end TTS model aims to minimize the L1 loss between predicted and ground-truth lmfb features. In this work, the TTS model predicts discrete ID sequences instead of lmfb features, and the ASR also uses the ID sequences as training data. We expect that using a discrete representation based on vq-wav2vec makes TTS training easier and mitigates the mismatch with real data.

1.6.3 Mel-to-Mel Network to Refine Generated Spectral Features

In Chapter 5, we present a post-processing network that refines lmfb features without using the vocoder. In Chapter 3, we first generate an lmfb feature from text data, then converts it into a waveform, and converts it again to an lmfb feature. The vocoder is used to alleviate the difference between real and synthesized speech, but it requires a huge amount of runtime. Moreover, the waveform is not necessary for the data augmentation itself. We propose the mel-to-mel network that directly refines lmfb features, which takes much less time than converting to the waveform domain. General speech enhancement, which is also (not Mel) spectrogram-to-spectrogram model, can be improved given phone information of the speech which is available in TTS and data augmentation tasks. In this work, we also add text information of the speech for phone-informed refinement. We experimentally show that the proposed network achieves better WERs than the vocoder network in a domain adaptation task in a much smaller amount of data generation time.

1.7 Organization of this Thesis

The organization of this thesis is outlined in Fig. 1.1. Chapter 2 provides literature review about ASR and TTS methods. Chapter 3 (blue arrow) presents data augmentation for ASR using TTS and waveform-based data augmentation. We use a vocoder to convert the generated lmfb features into a waveform and also design a waveform-input ASR model. Chapter 4 (yellow arrow) presents a data augmentation via a discrete representation. The TTS model predicts a discrete

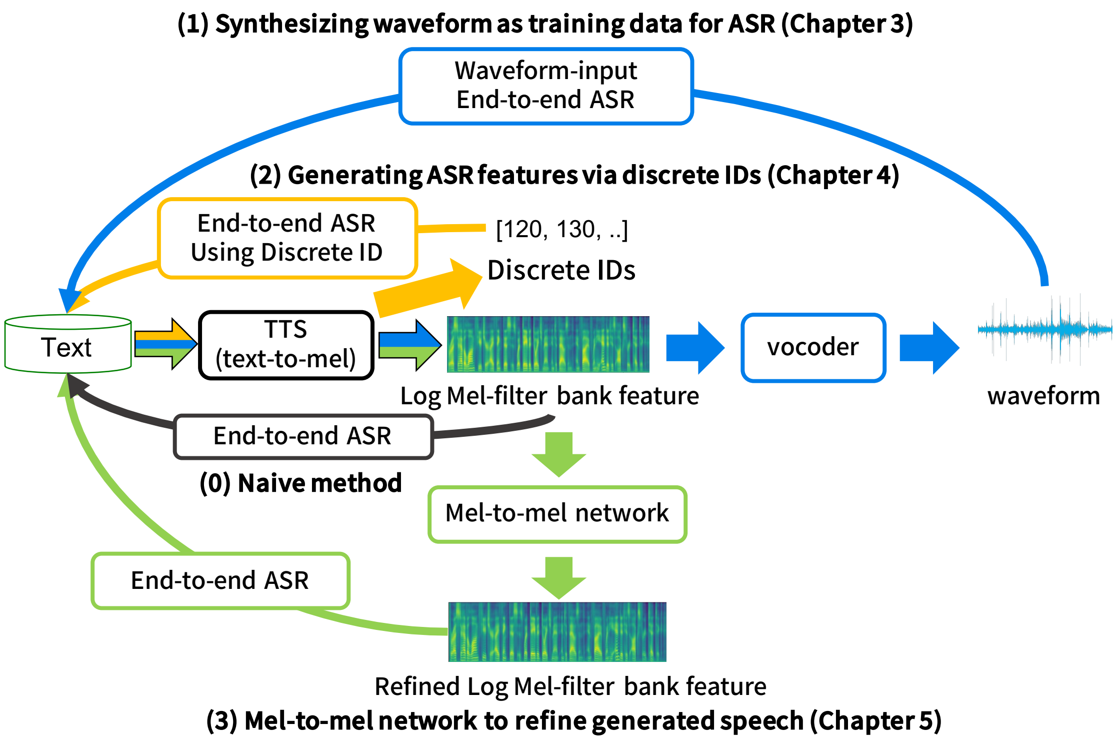

(1) Synthesizing waveform as training data for ASR (Chapter 3)
Waveform-input End-to-end ASR
(2) Generating ASR features via discrete IDs (Chapter 4)
End-to-end ASR Using Discrete ID
[120, 130, ..]
Discrete IDs
Text
TTS (text-to-mel)
vocoder
Log Mel-filter bank feature
waveform
End-to-end ASR
(0) Naive method
Mel-to-mel network
End-to-end ASR
Refined Log Mel-filter bank feature
(3) Mel-to-mel network to refine generated speech (Chapter 5)

Figure 1.1: Organization of this thesis. In Chapter 3 (blue arrow), we generate the lmfb feature and then convert the lmfb feature into a waveform. We also design a waveform-input ASR model. In Chapter 4 (yellow arrow), we generate a discrete representation instead of a continuous value of lmfb features. We also design the end-to-end ASR model, which consumes the discrete ID-based acoustic feature. In Chapter 5 (green arrow), we compose a mel-to-mel network to refine the generated lmfb features without relying on a vocoder.

representation instead of lmfb features, and the ASR model also uses the discrete representation-based feature. Chapter 5 (green arrow) presents a mel-to-mel network to refine generated speech. We comprise a mel-to-mel network to refine the lmfb features generated by the text-to-mel network.

Chapter 2

Review of Automatic Speech Recognition and Text-to-Speech

This chapter reviews the literature related to ASR and TTS systems. Section 2.1 describes DNN-HMM hybrid ASR systems and Section 2.2 describes end-to-end ASR systems. Let $\mathbf{X} = (x_1, ..., x_T)$ denote an acoustic feature of lengths T, and let $\mathbf{y} = (y_1, ..., y_L)$ denote a target label sequence of lengths L, where $y_l \in \{1, ..., K\}$ and K is the number of target labels.

2.1 DNN-HMM Hybrid ASR Systems

DNN-HMM hybrid ASR systems have been used for many practical applications. They have two separate modules to recognize speech. One is an acoustic model (AM). The other is a language model (LM). In the conventional ASR systems, a word sequence $\mathbf{Y}_{\text{ASR}}$ is decoded for a given acoustic feature sequence $\mathbf{X}$ with the maximum a posterior decision and the Bayes' theorem as follows:

$$\text{argmax}\, p(\mathbf{Y}_{\text{ASR}}|\mathbf{X}) = \text{argmax}\, p(\mathbf{Y}_{\text{ASR}})p(\mathbf{X}|\mathbf{Y}_{\text{ASR}}) \tag{2.1}$$

Moreover, a word is divided into phones $\mathbf{Y}_{\text{phone}}$ and we can rewrite equation (2.1) as follows:

$$p(\mathbf{Y}_{\text{ASR}})p(\mathbf{X}|\mathbf{Y}_{\text{ASR}}) \approx \max_{\mathbf{Y}_{\text{phone}}} p(\mathbf{Y}_{\text{ASR}})p(\mathbf{X}|\mathbf{Y}_{\text{phone}}) \tag{2.2}$$

where $p(X|\mathbf{Y}_{\text{phone}})$ and $p(\mathbf{Y}_{\text{ASR}})$ are the probability of acoustic model and language model respectively. The acoustic model learns the distribution of acoustic features by DNN and the latent state transition by HMM. The language model estimates the probability of a word occurrence given a context. It conventionally uses an N-gram model, and the integration of N-gram and RNN achieves the improvement [33]. In this manner, this hybrid ASR system is split into two disjoint modules and trained separately with different criteria. In addition, pronunciation dictionary that defines mapping from a word to a phone sequence is needed, and it often requires phonological knowledge.

2.2 End-to-End Architecture for ASR

The hybrid ASR systems train the AM and LM separately. Integrating the functions of AM and LM has been investigated thanks to the recent development of DNN, referred to as end-to-end speech recognition. We directly compute $p(\mathbf{Y}_{\text{ASR}}|\mathbf{X})$ in the end-to-end model. In practice, we split $\mathbf{Y}_{\text{ASR}}$ into subwords to reduce the vocabulary size of the end-to-end model and recognize the unknown words. To make subwords, we use sentencepiece [34] or byte-pair-encoding (BPE) [35]. End-to-end speech recognition has four types of architecture: connectionist temporal classification (CTC) [4, 36], transducer model, attention-based encoder decoder model, and Transformer-based model.

2.2.1 Connectionist Temporal Classification

Connectionist temporal classification (CTC) is a kind of objective function for labeling a sequence problem. Generally, the time length of X is much longer than that of $\mathbf{Y}_{\text{ASR}}$. The CTC-based model introduces a special label called "blank" (ϕ) in addition to subword labels. In this model, these outputs define the probabilities of all possible ways of aligning all possible label sequences with the input sequence. The total probability of one label can be calculated by summing the probabilities of its different alignment.

Training

The CTC loss is defined based on the minimum log-likelihood criterion.

$$L_{CTC}(\mathbf{X}, \mathbf{Y}_{\text{ASR}}) = -\log P(\mathbf{Y}_{\text{ASR}}|\mathbf{X}) \tag{2.3}$$

The key to compute $P(\mathbf{Y}_{\text{ASR}}|\mathbf{X})$ is allow the model to output blank (ϕ) labels. $P(\mathbf{Y}_{\text{ASR}}|\mathbf{X})$ is marginalized using the probabilities of all possible alignment in $\Omega(\mathbf{Y}_{\text{ASR}})$ as:

$$p(\mathbf{Y}_{\text{ASR}}|\mathbf{X}) = \sum_{\boldsymbol{\pi}\in\Omega(\mathbf{Y}_{\text{ASR}})} p(\pi|\mathbf{x}) = \sum_{\boldsymbol{\pi}\in\Omega(\mathbf{Y}_{\text{ASR}})} \prod_{t=1}^{T} p(\pi_t|\mathbf{x}_t) \tag{2.4}$$

where $\boldsymbol{\pi} = (\pi_1, ..., \pi_T)$ is an output sequences over the target label $\pi_t \in \{1, ..., K\} \cup \{\phi\}$ and the posterior probabilities $p\left(\pi_t|\mathbf{x}_t\right)$ are modeled with a recurrent neural network $N_w : \mathbb{R}^{m\times T} \mapsto \mathbb{R}^{n\times T}$ such as LSTM which maps an input acoustic sequence $\mathbf{X}$ into a m-dimensional continuous value. The CTC loss and its gradient with respect to the network parameters are efficiently computed with the forward-backward algorithm. Usually CTC-based model learns a monotonic alignment. It is advantageous for speech recognition because the output label sequence is monotonic in speech recognition. However, they do not explicitly learn the internal relationship at different times since they assume that the probability of each label is independent of others as in equation (2.4).

Inference and Forced Alignment

In inference, we remove all repeating labels and blank labels from the paths in $\Omega^{-1}(\boldsymbol{\pi}) = \mathbf{y}$. For example, we can recognize $\Omega^{-1}(\phi aa\phi\phi a\phi bb) = aab$.

The time indices of non-blank tokens in $\boldsymbol{\pi}$ are used as the reference token boundaries. When repeated non-blank labels exist, the leftmost index corresponding to the same non-blank token is used as a reference token boundary. For instance, given a CTC path $\boldsymbol{\pi} = (\phi aa\phi\phi a\phi bb)$ corresponding to a reference transcription "a a b", we convert it to $(\phi, a, \phi, \phi, a, \phi, b, \phi)$ and then extract the time indices of the non-blank tokens alignment $= (2, 5, 7)$. In this thesis, we used the alignment to train FastSpeech 2 model (Section 2.3.2).

2.2.2 Attention-Based Encoder Decoder Model

The other way to fill the lengths difference is to use sequence-to-sequence (seq2seq) modeling. The attention-based encoder-decoder model is a kind of seq2seq model. An encoder network maps an acoustic feature sequence to a distributed representation of the same lengths T. The decoder network predicts a target sequence whose length is L using the encoded intermediate information. The decoder network uses only a relevant portion of the encoded sequential representation to predict a symbol at each step using the attention mechanism. The encoder is implemented with a multi-layer bidirectional LSTM, and the decoder usually consists of a 1-layer of unidirectional LSTM followed by a softmax layer.

The attention-based model is formulated as follows. The encoder transforms an acoustic feature sequence X to an intermediate representation $\mathbf{h}_{ASR} = (\mathbf{h}_1, ..., \mathbf{h}_T)$. In the decoder network, the hidden state activation of the RNN-based decoder at the l-th time step is computed as:

$$\mathrm{s}_l = \text{Recurrency}\left(\mathrm{s}_{l-1}, \mathbf{g}_l, y_l\right) \tag{2.5}$$

where $\mathbf{g}_l$ and y_{l-1} denote the "glimpse" at the l-th target label and the predicted symbol at the previous step. The glimpse $\mathbf{g}_l$ is a weighted sum of the encoder output sequence as:

$$\mathbf{g}_l = \sum_t \alpha_{l,t} \mathbf{h}_t \tag{2.6}$$

where $\alpha_{l,t}$ is an attention weight of $\mathbf{h}_t$. In this work, we use a content-based attention mechanism formulated as follows:

$$\begin{aligned} e_{l,t} &= \mathbf{w}^T \text{tanh}(\mathbf{W}\mathrm{s}_{l-1} + \mathbf{V}\mathbf{h}_t + \mathbf{U} f_{l,t} + \mathbf{b}) & (2.7) \\ \mathbf{f}_l &= \mathbf{F} * \boldsymbol{\alpha}_{l-1} & (2.8) \\ \alpha_{l,t} &= \exp(e_{l,t}) / \sum_{t'=1}^{T} \exp(e_{l,t'}) & (2.9) \end{aligned}$$

where $*$ denotes a 1-dimensional convolution. Using $\mathbf{g}_l$ and s_{l-1}, the decoder predicts the next symbol $\mathbf{y}_l$ as:

$$\mathbf{y}_l \sim \text{Generate}\left(\mathrm{s}_{l-1}, \mathbf{g}_l\right) \tag{2.10}$$

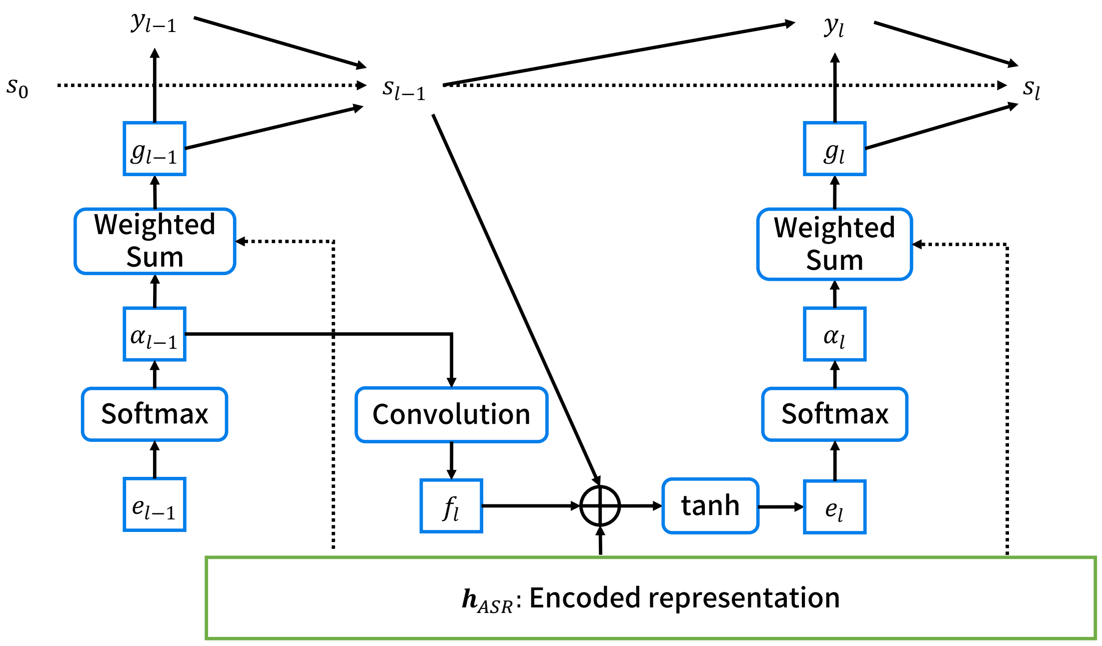


Figure 2.1: The flow of the decoder network. The vector s_0 has all zero elements by initialization.

where the Generate function is implemented as:

$$\mathbf{R}\tanh\left(\mathbf{P}\mathbf{s}_{l-1} + \mathbf{Q}\mathbf{g}_l\right) \tag{2.11}$$

Figure 2.1 shows the flow of the decoder network.

The objective function for training the attention models is cross entropy. The loss is reduced to the negative log-likelihood between the predicted symbol sequences and the target oracle symbol sequences.

$$\mathcal{L}_{att} = -\log P_{att}(\mathbf{Y}_{\text{ASR}}|\mathbf{X}) \tag{2.12}$$

For efficient training, we apply the three training methods [17,37,38].

Joint CTC/Attention training

When training the attention-based model, we use the cross entropy between the ground-truth labels and the predicted labels ($\mathcal{L}_{att}$). In the ASR task, the attention between the acoustic features and the target label has monotonicity (left-to-right), but a structure of attention itself does not have the constraint, which sometimes causes the label repetition. To ensure the monotonicity, we introduce the multi-task learning with CTC loss (λ_{ctc}) [37] as follows:

$$\mathcal{L}_{ASR} = (1 - \lambda_{ctc})\mathcal{L}_{att} + \lambda_{ctc}\mathcal{L}_{CTC} \tag{2.13}$$

where λ_{ctc} is a hyperparameter ($0 \leq \lambda_{ctc} \leq 1$) for the CTC loss weight.

Label Smoothing

Label smoothing [38] is a regularization method for preventing a model from over-fitting. When calculating the cross-entropy, we do not simply use a grand-truth label of 1.0 but discount it and assign a small probabilities to all other symbols with a uniform distribution. In this paper, we set the probability of the ground-truth label to 0.9 and other labels to uniform $0.1/K$.

SpecAugment

SpecAugment [17] is a data augmentation method by masking both frequency and time domains randomly. Before input to the ASR model, the mask is applied to lmfb features. We randomly choose the value of f, and then $f0$ consecutive mel frequency channels $[f0, f0 + f)$ are masked, where f is first chosen from a uniform distribution from 0 to the frequency mask parameter F, and $f0$ is chosen from $[0, \nu - f)$, where ν is the number of mel frequency channels. Time masking is applied so that t consecutive time steps $[t_0, t_0 + t)$ are masked, where t is first chosen from a uniform distribution from 0 to the time mask parameter T, and then t_0 is chosen from $[0, \tau - t)$. The lmfb values in the range of masked areas are replaced with 0. We do not use time-warping and use two masks in the time domain and one mask in the frequency domain.

Beam Search and Shallow Fusion

In inference, greedy search simply takes the label with the highest probability at each position and computes fast. Once we had identified the best label for that position, we did not consider what came before it. In the attention-based model, we conduct the beam search to consider the probabilities of the combination of the preceding labels along with the label in the current position. Beam search picks the N best sequences and predicts the probability of the label in the current position for all N candidates. We then pick the N best sequences based on the combined probability and repeat this procedure until the end. We prepare special symbols for denoting the start-of-sentence ($\langle$sos$\rangle$) and end-of-sentence ($\langle$eos$\rangle$). The decoder completes the process when an $\langle$eos$\rangle$ symbol is recognized. In the prediction of each position, we memorize s_l, $\boldsymbol{\alpha}_l$, and $\mathbf{Y}_{\text{ASR}}$ of each candidate. When calculating the scores of beam search, shallow fusion [18] is widely applied. In shallow fusion, the external LM $y_{\text{LM}}(\mathbf{Y}_{\text{ASR}})$ is incorporated via log-linear interpolation at inference time only. In addition to shallow fusion, we also introduce length reward (penalty) terms. As a result, we calculate a score for beam search as follows:

$$\hat{\mathbf{Y}_{\text{ASR}}} = \underset{\mathbf{Y}_{\text{ASR}}}{\text{argmax}}(\log p_{\text{ASR}}(\mathbf{Y}_{\text{ASR}}|\mathbf{X}) + \beta \ \log p_{\text{LM}}(\mathbf{Y}_{\text{ASR}}) + \gamma \ |\mathbf{Y}_{\text{ASR}}|) \tag{2.14}$$

where β is a weight for language model and γ is a length reward ($\gamma > 0$) or penalty ($\gamma < 0$).

2.2.3 Transformer

Transformer [39] is a DNN model that adopts an self-attention mechanism without a recurrent neural network (RNN). The transformer block has two modules of multi-head attention and feed-forward network (FFN), and we stack several transformer blocks to compose the transformer-based model. We add "positional encodings" to the input embeddings at the bottom of the encoder and decoder stacks to make use of the order of the sequence. In this study, we use sine and cosine functions of different frequencies: $PE(pos, 2i) = sin(pos/10000^{2i/d_{model}})$ and $PE(pos, 2i + 1) = cos(pos/10000^{2i/d_{model}})$ where pos is the position, i is the dimension, and d_{model} is dimension of the transformer block.

The multi-head attention of the transformer is based on scaled dot-product attention. The scaled dot-product attention learns three weight matrices to calculate the attention; the $d_{q,k}$-dimensional query weights $W_Q \in \mathbb{R}^{d_{model} \times d_{q,k}}$, the

$d_{q,k}$-dimensional key weights $W_K \in \mathbb{R}^{d_{model} \times d_{q,k}}$, and the d_v-dimensional value weights $W_V \in \mathbb{R}^{d_{model} \times d_v}$. We produce the query vector $Q = W_Q X_Q$, the key vector $K = W_K X_{K,V}$, and the value vector $V = W_V X_{K,V}$ using the input $X_{K,V}$ of the key and value, and X_Q of the query. We then calculate the attention results as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.15}$$

where $\frac{1}{\sqrt{d_k}}$ is a scaled factor. In particular, Eq. (2.15) is referred as self-attention when $X_{K,V} = X_Q$. The transformer model uses multi-head attention, which calculates several attentions on one stack. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. We write h-head attention as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O \tag{2.16}$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.17}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_{q,k}}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_{q,k}}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. In the multi-head attention, we set $d_{q,k} = d_v = d_{model}/h$. Figure 2.2 shows flows of the scaled dot-product attention and multi-head attention.

In addition to attention sub-layers, each layer contains a fully connected FFN, which is applied to each position separately and identically. This FFN module has several variants. FFN has two linear transformations with a ReLU activation in the original work. Another variant has two CNN layers with a ReLU activation.

We can use the transformer as an encoder and a decoder, but we use it as an encoder only. In this thesis, we use the Transformer architecture for not only the ASR model but also the TTS model (Section 2.3.2).

2.2.4 Conformer

Conformer [40], which is a variant model of transformer encoder, achieves high performance on the ASR tasks. Fig. 2.3 shows the comparison of transformer block

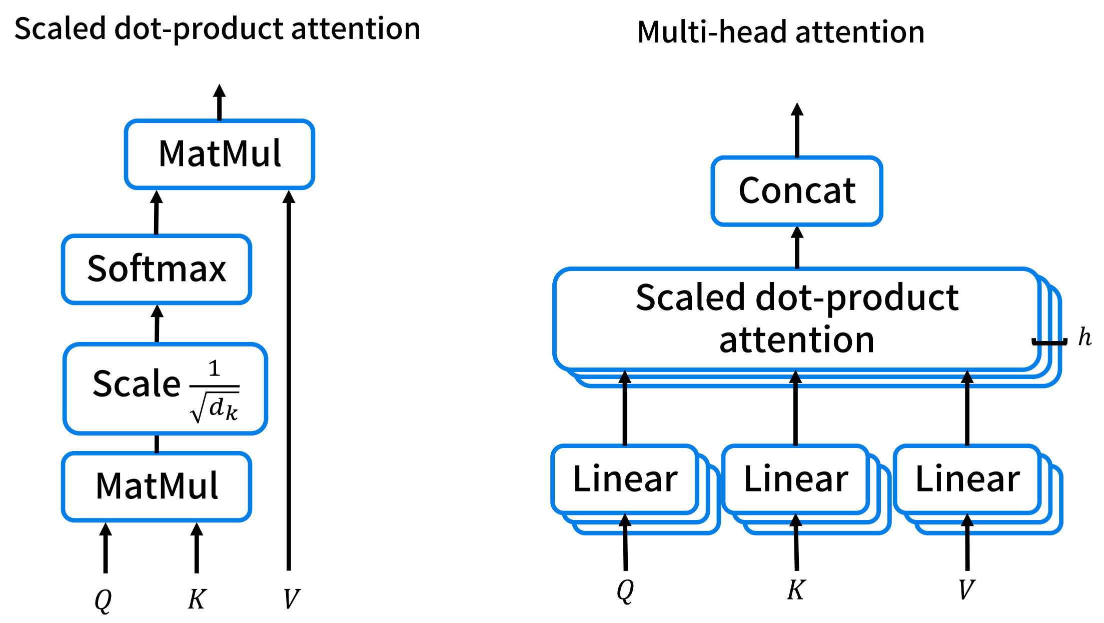

Scaled dot-product attention
MatMul
Softmax
Scale $\frac{1}{\sqrt{d_k}}$
MatMul
Q
K
V
Multi-head attention
Concat
Scaled dot-product attention
h
Linear
Linear
Linear
Q
K
V

Figure 2.2: The flows of the scaled dot-product attention and multi-head attention. d_k means dimension of K.

and conformer block. The differences are the positions of FFN, a convolution module, and relative positional encoding. We use two FFN blocks. Each FFN block has two linear transformations with a Swish activation. The convolution module is composed of a pointwise convolution, gated linear unit (GLU), 1-D convolutional depthwise convolutional layer, batchnorm with Swish activation, and a pointwise convolution. While the original transformer block uses absolute positional encoding, the conformer block uses relative positional encoding. The relative positional encoding is employed in the multi-head attention module. The relative positional encoding allows the self-attention module to generalize better on different input lengths, and the encoder is more robust to the variations of the utterance length.

2.3 End-to-End Architecture for TTS

In the TTS task, we conventionally use the HMM-based model to align between a target speech $\mathbf{X}$ and a length-L input text sequence $\mathbf{Y}_{\text{TTS}}$. These models have

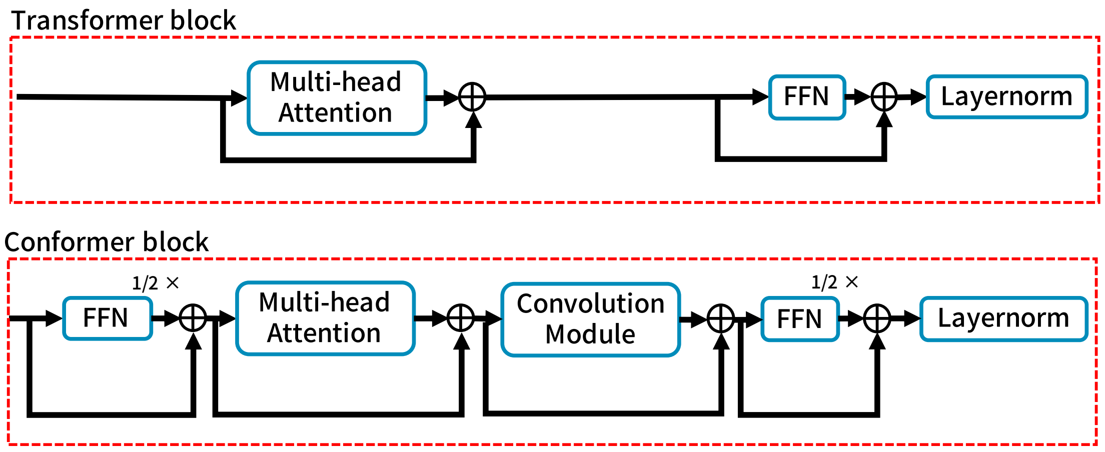


Figure 2.3: Transformer block (top) and conformer block (bottom).

complicated modules, and we need to design them carefully similarly to the ASR models. Thanks to the development of DNNs, we realize the TTS system with a much simpler architecture, which is called an end-to-end TTS model. In contrast to the ASR model, the end-to-end based TTS models predict the target speech $\mathbf{X}$ for a given input text sequence $\mathbf{Y}_{\text{TTS}}$ as:

$$\mathbf{X} = \text{argmax}\ p_{\text{TTS}}(\mathbf{X}|\mathbf{Y}_{\text{TTS}}) \tag{2.18}$$

$\mathbf{Y}_{\text{TTS}}$ is a phone sequence converted from a word sequence. The goal of the TTS is to synthesize a waveform that the human can hear and evaluate. In the realm of the TTS, $\mathbf{X}$ is the waveform. We compose two pipelines in general: (1) mel-to-mel network for lmfb features generation and (2) vocoder (mel-to-waveform network) for waveform generation from the generated lmfb features. We rewrite (2.18) as follows:

$$\mathbf{X} = \text{argmax}\ P_{\text{vocoder}}(\mathbf{X}|\mathbf{X}_{\text{lmfb}})P_{\text{text-to-mel}}(\mathbf{X}_{\text{lmfb}}|\mathbf{Y}_{\text{TTS}}) \tag{2.19}$$

where $\mathbf{X}_{\text{lmfb}}$ is a length-T lmfb feature. In practice, we train two models separately. The text-to-mel networks, similar to speech recognition, require solving problems with different lengths of input labels and output speech. We use Tacotron 2 [9] and FastSpeech 2 [41] as the mel-to-mel network. The vocoder network also needs to solve the problem with different length of input lmfb features and output waveform. However, it is readily conditioned since lmfb features is computed

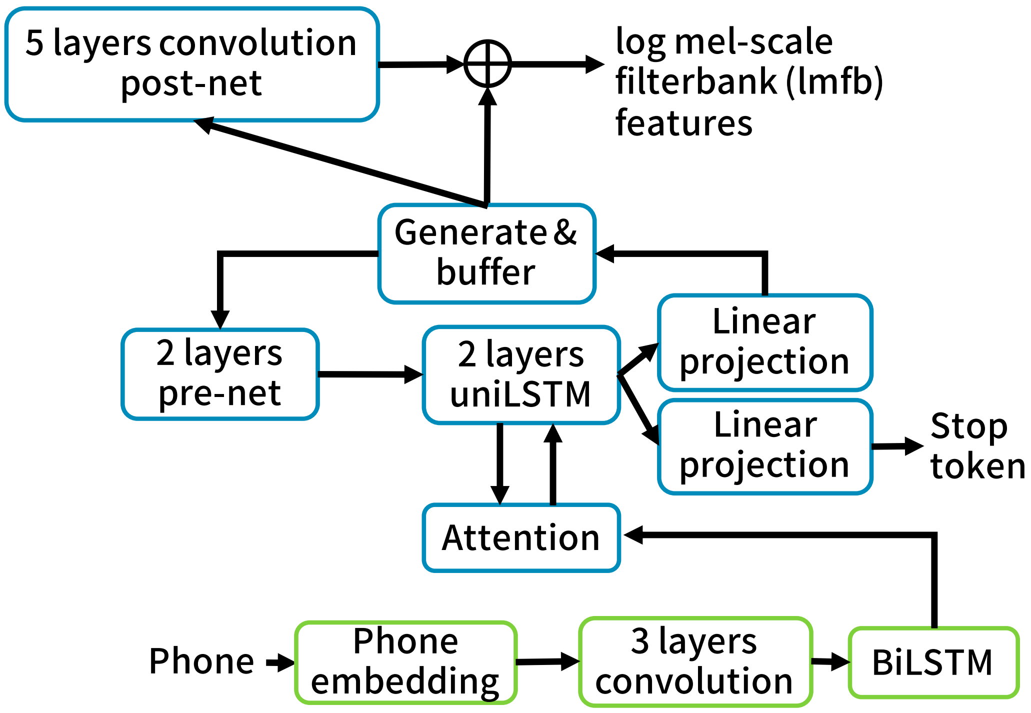


Figure 2.4: Tacotron 2 architecture.

from a constant speech waveform interval. We utilize the WaveNet-based model and GAN-based models as the vocoder networks.

2.3.1 Tacotron 2

Tacotron 2 [9] is a kind of the text-to-mel network and a seq2seq modeling with an attention mechanism, which is composed of an encoder decoder network. Figure. 2.4 shows the Tacotron 2 architecture. The encoder network transforms an input text sequence $\mathbf{Y}_{\mathrm{TTS}}$ to an intermediate representation following an embedding layer, CNN layers, and BiLSTM layers. The decoder comprises the 2-layer unidirectional LSTM. To learn a relevant portion of the encoded feature, we add attention mechanisms similar to the ASR model. We use a linear layer to predict the lmfb features from the outputs of the unidirectional LSTM layer. Finally, the predicted lmfb feature is passed through a 5-layer convolutional post-net which predicts a residual to add to the prediction to improve the overall reconstruction. We minimize the summed mean squared error (MSE) or mean absolute error (MAE) between ground-truth and predicted lmfb features before and after the post-net.

While the ASR systems readily stop the prediction when $\langle$eos$\rangle$ is predicted, the lmfb features do not have $\langle$eos$\rangle$ and the TTS system needs to predict the end of the prediction. Therefore, we prepare a layer to predict a stop token instead of an $\langle$eos$\rangle$ label. In training the stop token, we use binary cross entropy between ground-truth and predicted stop token flag (0 means "not finished" and 1 means 'finished'). In inference, the generation is stopped when the probability of a stop token is over 0.9.

The original Tacotron 2 model is designed for a single speaker TTS model. When it comes to data augmentation for ASR, the generated speech should have various speech. In this study, we compose a multi-speaker TTS model which generates various speech given the same text. We add a speaker ID to the input feature to generate multi-speaker speech.

2.3.2 FastSpeech 2

Recently, the non-autoregressive networks have been investigated such as FastSpeech [41, 42] and Parallel Tacotron [43, 44]. These networks generate Mel spectrogram faster than the autoregressive network because they do not need to wait for the generation at the previous timestep. The non-autoregressive model is appropriate for the data augmentation since we need to generate a larger amount of speech than the standard TTS task. In this work, we use FastSpeech 2-based model [41]. The FastSpeech 2 model is composed of a Transformer-based encoder decoder network. The main characteristic of the FastSpeech 2 model is adding a network called variance adaptor to predict the duration of lmfb features corresponding to each input phone. In addition to the network to predict the duration, the variance adaptor optionally consists of some predictors to other kinds of acoustic information. Figure 2.5 shows the architecture of variance adaptor. In this work, we implement three predictors: duration predictor, pitch predictor, and energy predictor. Each predictor has a 1-D CNN block + ReLU

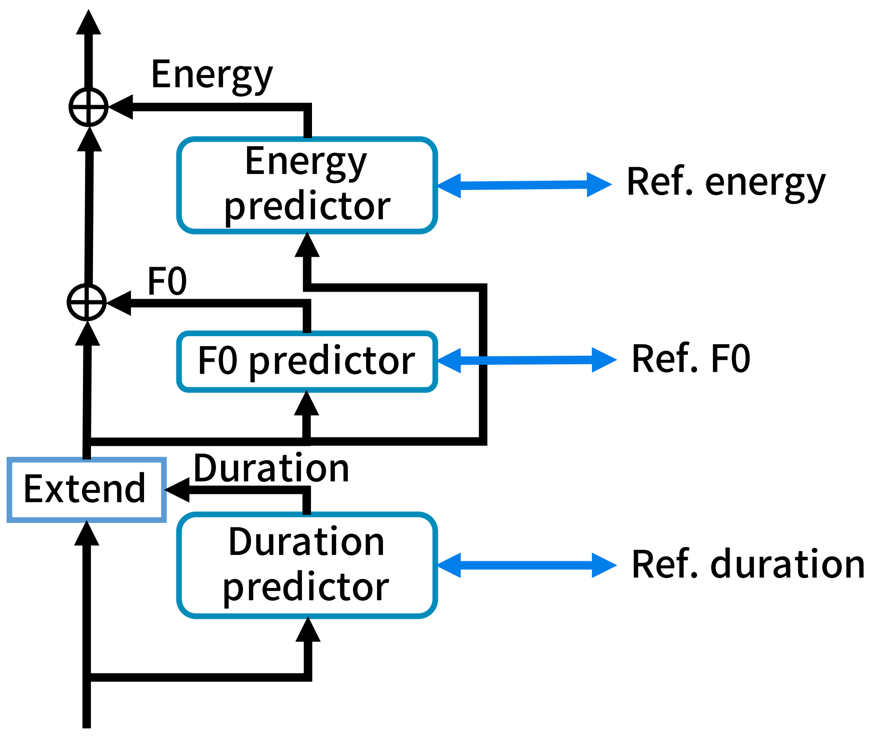


Figure 2.5: Variance adaptor of FastSpeech 2.

activation, a layer normalization block, a 1-D CNN block + ReLU activation, a layer normalization block, and a linear layer. For training of duration prediction, we prepare the alignment before training the FastSpeech 2-based model.

More formally, the encoder transforms an input text sequence $\mathbf{Y}_{\text{TTS}}$ into an intermediate representation $\mathbf{H} = (\mathbf{h}_1, ..., \mathbf{h}_L)$. The duration predictor predicts a duration of each input text $\mathbf{D} = (d_1, ..., d_L)$, where $d_1 + ... + d_L = T$. We then extend the intermediate features according to $\mathbf{D}$ as follows:

$$\hat{\mathbf{H}} = (h_1, h_1, h_1, ..., h_L, h_L) \tag{2.20}$$

The pitch and energy predictor predicts pitch and energy, respectively and their predicted acoustic information are embedded. The embedded features are added to the $\hat{\mathbf{H}}$ via residual blocks. Finally, the decoder predicts $\mathbf{X}_{\text{lmfb}}$ using $\hat{\mathbf{H}}$ in parallel. In this thesis, we use a 5-layer convolutional post-net [9] which predicts a residual to add to the prediction to improve the overall reconstruction. In training, we minimize five mean absolute error (MAE) losses for lmfb features of the FastSpeech 2 output, those of the post-net output, duration, pitch, and energy. We need to prepare the alignment, pitch, and energy in advance. In this thesis, we use a CTC-based model to get the alignment and WORLD [45] to obtain the pitch.

2.3.3 WaveNet-Based Vocoder

We need to comprise an additional network referred to as a vocoder to generate a waveform. Similar to the mel-to-mel network, the vocoder networks are categorized into an autoregressive and a non-autoregressive model. The WaveNet [46] vocoder is a kind of autoregressive model. In the vocoder task, the timesteps of a waveform are much larger than that of an lmfb feature (*e.g.*, 16kHz sampling waveform has 16,000 timesteps per second, and a 12.5ms-shifted lmfb feature has 80 timesteps per second). It constructs the multi-layer CNNs, where the convolutional layers have various dilation factors that allow the receptive field to grow exponentially with depth and cover thousands of timesteps. It also uses the generated lmfb features provided by a text-to-mel network to condition acoustic information. Although the WaveNet generates a high quality waveform compared to the parametric TTS model, the generation takes a long time because it must wait for the waveform samples on the previous timesteps, and the waveform has a much long form. For a faster generation, WaveRNN [47] based on simple single-layer RNN has been proposed. However, the autoregressive models are not suitable for real-time applications.

2.3.4 GAN-Based Vocoder

Recently, many attempts have been made with non-autoregressive models. Parallel WaveNet [48] distills a trained autoregressive teacher decoder into a flow-based model. WaveGlow [49] is a flow-based generative model based on Glow [50]. WaveGlow is a very high capacity generative flow consisting of multi-coupling and multi-invertible 1x1 convolutions, with each coupling layer consisting of a stack of multi-layers of dilated convolutions. It achieves good quality, but takes much time for training since it needs to train a large number of parameters.

Generative adversarial networks (GANs)-based vocoders generate a high-quality waveform in faster decoding with fewer parameters. GANs [51] have made progress in computer vision such as image generation [52] and image-to-image translation [53]. GAN has two network architectures: generator and discriminator. A generator network maps a latent space into target values of interest, which is a waveform in the vocoder task and produces the outputs that the discriminator cannot distinguish. A discriminator distinguishes the true data from the data generated by the generator. We introduce the GAN objective to train both the generator and discriminator as follows:

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{\mathbf{x}}[\log D(x)] + \mathbb{E}_{\mathbf{x}_{\text{lmfb}}}[\log(1 - D(G(\mathbf{x}_{\text{lmfb}})))] \tag{2.21}$$

where the generator G tries to minimize the objective against an adversarial discriminator D that tries to maximize it. MelGAN [54] is a GAN-based vocoder and a non-autoregressive model. The generator of MelGAN is a fully convolutional feed-forward network with an lmfb feature as input and raw waveform as output. Because the lmfb feature is calculated per a certain interval waveform and at a lower time resolution, it uses a stack of transposed convolutional layers to upsample the input sequence. Unlike traditional GANs, the MelGAN generator does not use a global noise as input. The discriminator adopts a multi-scale architecture with three discriminators. There are several variants of the GAN-based model. Parallel WaveGAN [55] uses a WaveNet-based generator and multi-resolution STFT auxiliary loss for stable training of the generator. In the multi-resolution STFT loss, we use spectral convergence ($\mathcal{L}_{sc}$) and log STFT magnitude loss ($\mathcal{L}_{mag}$) as follows:

$$\mathcal{L}_{sc}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{|\text{STFT}(\mathbf{x})| - |\text{STFT}(\hat{\mathbf{x}})|}{|\text{STFT}(\hat{\mathbf{x}})|} \tag{2.22}$$

$$\mathcal{L}_{mag}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \log|\text{STFT}(\mathbf{x})| - \log|\text{STFT}(\hat{\mathbf{x}})| \tag{2.23}$$

where $|\mathrm{STFT}()|$ and N denote the STFT magnitudes and number of elements in the magnitude, respectively. The multi-resolution STFT loss is the sum of the STFT losses with different analysis parameters (i.e., FFT size, window size, and frame shift).

The GAN-based architecture and the multi-resolution STFT loss are effective for generating a high-quality waveform, and several extensions based on them have been proposed such as VocGAN [56] and HiFi-GAN [57].

Chapter 3

Synthesizing Waveform to Augment Training Data for ASR

3.1 Introduction

The major problems of end-to-end ASR systems are the need for a large amount of training data, data sparseness, and uneven distributions. It is not easy to adapt to a new domains because we need to prepare the paired speech and transcription data of the target domain.

Many attempts [18, 20, 21, 58–62] using text-only data have been made. One of the approaches investigates the integration of a language model which is trained using text-only data. Kannan *et al.* proposed shallow fusion to integrate an external language model and the ASR model in inference [18]. Sriram *et al.* developed a method to train the attention-based ASR model together with a language model [58]. These approaches enhance the language model function of the end-to-end model using the external language model. Other approaches compose the end-to-end model that allows text-only data input to enhance the language model [59, 60]. Masumura *et al.* [59] prepared a shared decoder to train phoneme-to-grapheme (text-only data) and ASR tasks (paired data) and pre-trained the decoder on a phoneme-to-grapheme task. Tang *et al.* [60] jointly trained speech-to-subword and phone-to-subword tasks.

Other approaches utilize back-translation style learning. Karita *et al.* [61, 62] used autoencoders to leverage speech-only data and text-only data. These autoencoders learn features from speech-only and text-only datasets by switching the encoders and decoders used in the ASR and TTS models. These methods can be effective for enhancing the ASR models in the same domain.

Other works combine the ASR and TTS models [20, 21] referred to as speech chain. Given the unlabeled speech features, the ASR model transcribes the unlabeled input speech, while the TTS model reconstructs the original speech waveform based on the output text from ASR. Given only the text input, TTS generates speech waveform, while ASR also reconstructs the original text transcription giving the synthesized speech.

In our previous work [22, 23], we proposed utilizing speech synthesis to generate acoustic features for training end-to-end ASR models from new-domain texts. Recently, seq2seq neural speech synthesis models have also been developed [9, 63–65]. In contrast to conventional TTS, a seq2seq model realizes TTS with very simple architecture, and its training is much easier. Moreover, it has been shown to achieve naturalness comparable to human speech [9, 65]. TTS efficiently makes it possible to generate paired data covering the new target domain. However, speech synthesizers are usually trained with a single speaker and do not have speaker diversity. This may be a serious problem for ASR, which needs to cover a variety of speakers. Therefore, we extended the Tacotron 2-based speech synthesis framework to generate multi-speaker speech using speaker embedding in seq2seq speech synthesis [23]. Training a speech synthesizer with a large number of speakers is expected to generate useful speech data for ASR model training and eventually solve the data sparseness problem.

However, the performance gain by the data augmentation with TTS is still limited compared with real speech data since the quality of speech data generated by TTS is not realistic. We observe that the time resolution of the generated lmfb features is insufficient, and the generated lmfb features result in blurry, particularly on the time dimension. Actually, the time resolution of the artificial spectrum is not sufficient because the TTS model estimates the spectrum of several contiguous frames at one decoding step for stable training and fast inference. In this work, we design the waveform domain data augmentation used for both ASR and TTS. In the TTS, we use a vocoder to improve the generated spectrum. We aim to interpolate speech at time dimension to convert a generated lmfb feature into a waveform. We can again convert the waveform into the lmfb features and then use them as the lmfb-input ASR input. In this work, we also design waveform-input ASR to fully utilize the generated waveform. To realize the waveform-input ASR, we introduce a CNN-based feature extraction network and train it together with a recognition network. We also apply a data masking method in the waveform-based end-to-end ASR because a data masking method for the lmfb-based ASR systems such as SpecAugment [17] significantly affects the ASR performance.

3.2 TTS for ASR Training

We investigate leveraging seq2seq speech synthesis to augment the training data for speech recognition (Fig. 3.1). This data augmentation method basically has three steps. First, we train a TTS model using paired data of speech and transcription (this paired data does not need to be a target domain). We then generate the speech from the target domain texts. We collect text from a target domain where we want to perform speech recognition and generate speech using the text. This scheme not only enhances the language model capability but also learns acoustic patterns of the text, including domain-specific words. Finally, we train an ASR model using the generated data mixed with the real data.

Although the generated speech improves the ASR performance, the improvement of ASR depends on the performance of TTS. For example, speech data generated by a single speaker TTS has less diversity, but the ASR model needs to recognize various speech. In our seminal work [23], we introduced multi-speaker TTS to acquire speaker diversity by adding speaker embedding to the Tacotron 2 architecture. Similar efforts have been made by Rosenberg [24], Nick [66], and Huang [67] based on speaker-ID embedding for data augmentation. Rosenberg *et al.* [24] introduced multi-speaker TTS for data augmentation using hierarchical VAE and the WaveNet-based vocoder. Nick *et al.* [66] used GST-based

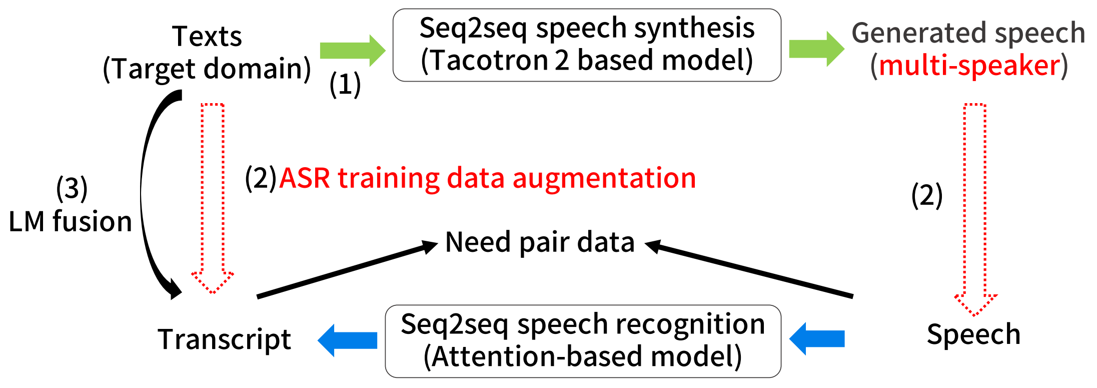


Figure 3.1: Overall architecture of data augmentation for ASR. (1) seq2seq speech synthesis model generates speech. (2) seq2seq speech recognition model is trained using generated speech and text. (3) when seq2seq speech recognition model decodes, text-only data is also used with LM fusion.

speaker embedding and the Griffin-Lim vocoder to synthesize speech waveforms. Huang *et al*. [67] used a multi-speaker TTS for speaker adaptation of ASR and the WaveNet-based vocoder.

Although the multi-speaker TTS enhances ASR model training, the quality of the lmfb features generated by the Tacotron 2 based model is not sufficient compared with the real lmfb features. Fig. 3.2 shows the comparison of real and generated speeches (lmfb features) for the same texts. It is observed that the TTS model generates lmfb features reasonably in the low-frequency regions, but the generated lmfb features in the high-frequency regions are vague or almost blurry compared with the real speech. Moreover, we can see the generated speech sometimes includes unnatural silence in the right example (around 250-th frames). Generally, the magnitudes of low-frequency bins are larger than those of the high-frequency bins and the silence parts. When the Tacotron 2 based model focuses on minimizing the L1 loss between predicted and ground-truth lmfb features, the prediction performance in the high-energy parts is sufficient, but the information of low-energy parts and silence tends to be diminished. Moreover, by using the reduction rate to make the training of the TTS model stable and its inference faster, the predicted lmfb features may be over-smoothed.

Considering that we must generate a huge amount of speech data, it is not

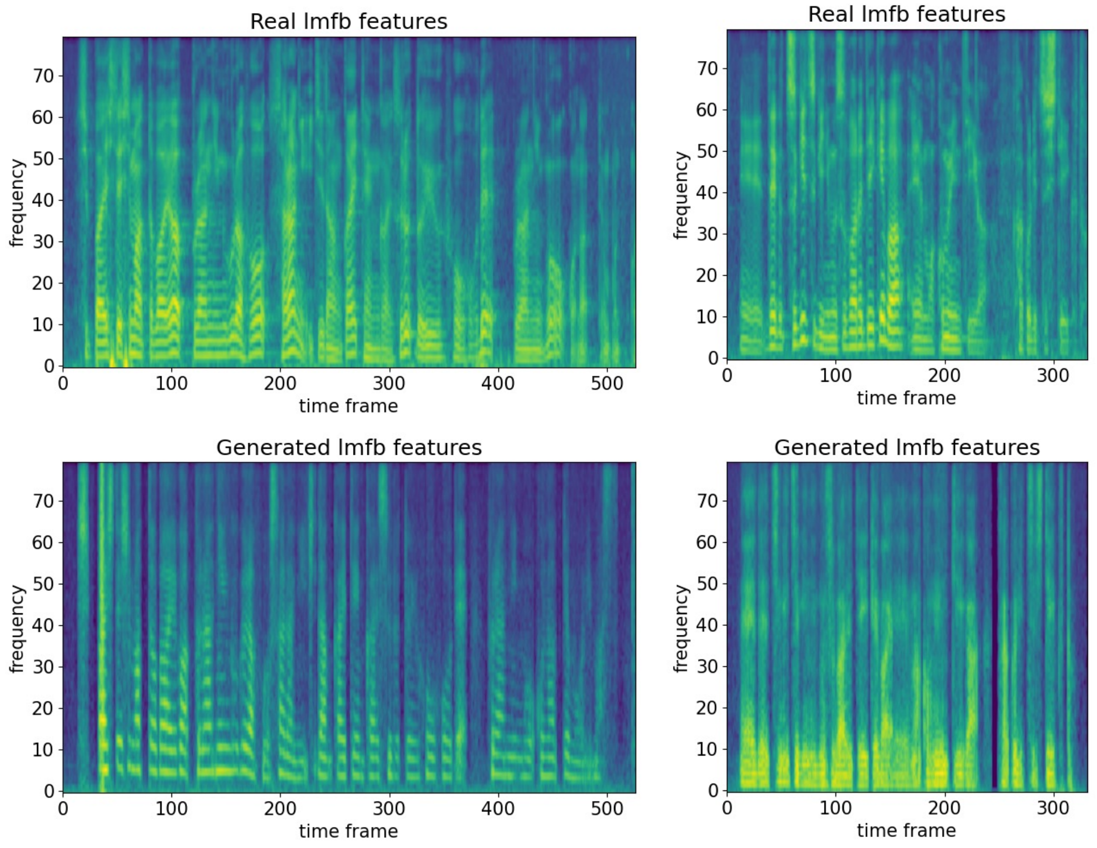


Figure 3.2: Examples of real (top) and generated speech (bottom) from two texts.

easy to fix this problem in the Tacotron 2 side. To generate high-quality speech data, we turn to the WaveNet vocoder. Fig. 5.1 shows lmfb features generated by TTS models. In the top (a), the TTS model generates lmfb feature with a 50 ms window and 12.5 ms shift, which TTS models typically use. In the middle (b), we compute the lmfb feature from the waveform generated by applying WaveNet to (a). In the bottom (c), the TTS model generates the lmfb feature with parameters that ASR models typically use. We observe that the lmfb feature, in particular the harmonic structure in low-energy parts, becomes clear after applying the WaveNet vocoder, and thus expected to be effective for ASR model training. Thus, we investigate the waveform-based data augmentation with TTS. Specifically, we compare the following three configurations.

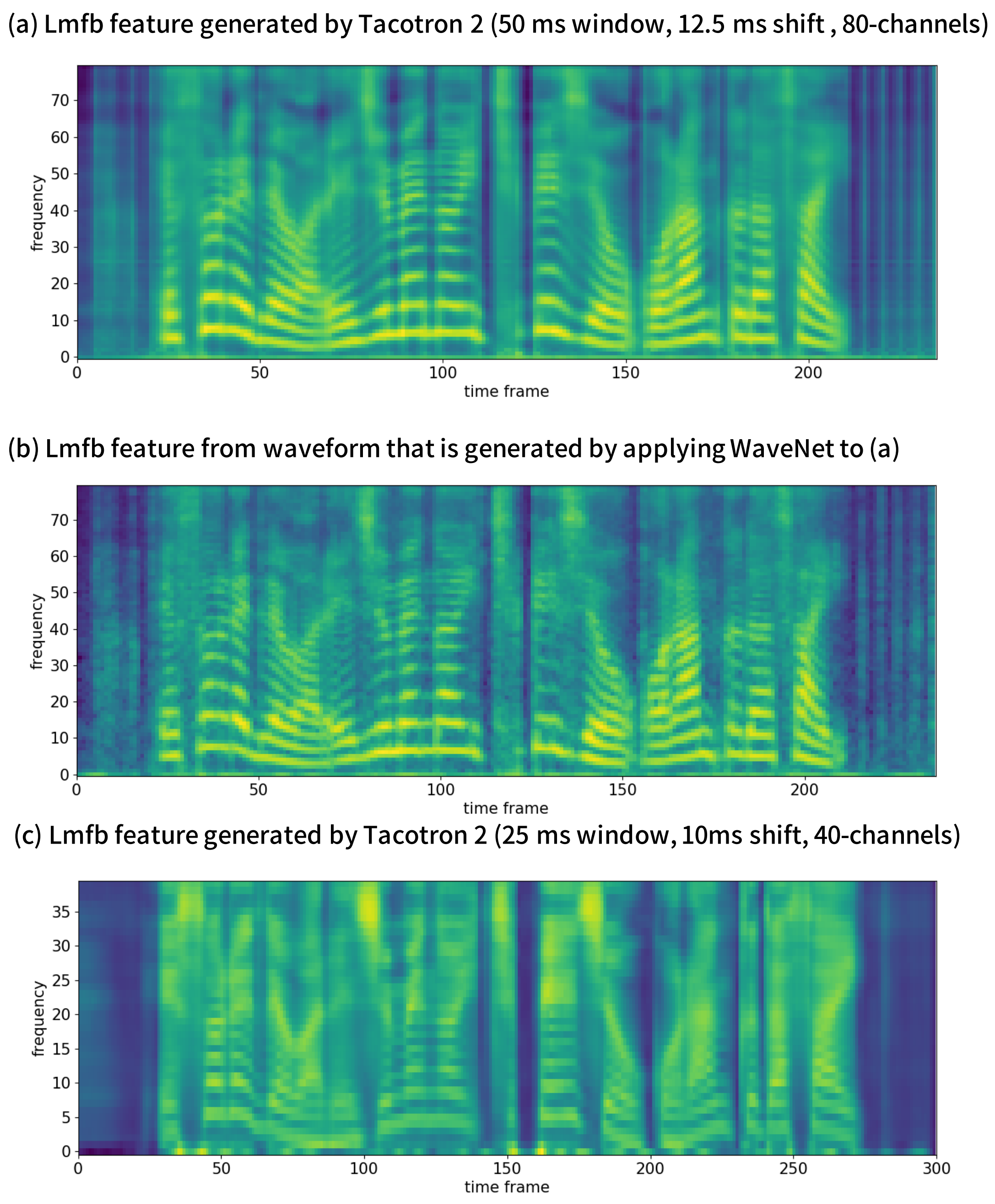


Figure 3.3: Log Mel-scale filterbank (lmfb) features generated by TTS models.

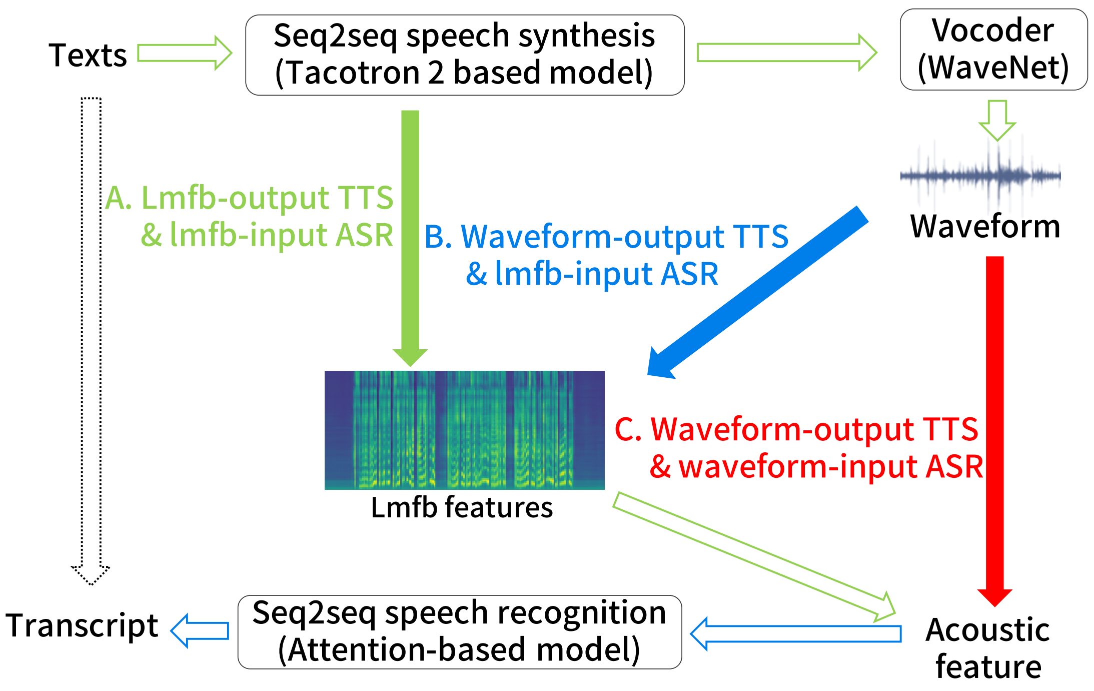

Texts
Seq2seq speech synthesis (Tacotron 2 based model)
Vocoder (WaveNet)
Waveform
A. Lmfb-output TTS & lmfb-input ASR
B. Waveform-output TTS & lmfb-input ASR
C. Waveform-output TTS & waveform-input ASR
Lmfb features
Transcript
Seq2seq speech recognition (Attention-based model)
Acoustic feature

Figure 3.4: Three configurations of data augmentation with TTS.

3.2.1 Lmfb-output TTS and Lmfb-input ASR

In this method, lmfb features are directly generated by TTS (Fig. 3.4 A). This method is computationally efficient since we do not generate waveforms. However, we change the setting of TTS to match the typical ASR systems. While typical TTS systems predict lmfb features based on 80 channels using a 50 ms window with a shift of 12.5 ms, we adopt a 25 ms window with a shift of 10 ms to compute 40-channel lmfb, which is typically used in the ASR model. We train the Tacotron 2 model with this ASR-matched setting. We also train the Tacotron 2 model with TTS-matched setting and the ASR model with TTS-matched setting in an experiment.

3.2.2 Waveform-output TTS and Lmfb-input ASR

As general ASR systems input lmfb features, we convert waveforms generated by TTS to lmfb (Fig. 3.4 B). As shown in Fig. 5.1, this lmfb has better quality than the lmfb generated by TTS. We compute 40-channel lmfb features using a 25 ms window with a shift of 10 ms for the ASR input. It is the widely-used setting in ASR.

3.2.3 Waveform-output TTS and Waveform-input ASR

In this study, we design a seq2seq ASR system whose input is a raw waveform (Fig. 3.4 C). This ASR system is an end-to-end ASR from a waveform to a word sequence. We describe the detail of the feature extraction part and a masking method in Section 5. The Tacotron 2-based model predicts 80-channel lmfb features using a 50 ms window with a shift of 12.5 ms, and then the WaveNet model generates waveforms from these lmfb features.

3.3 Waveform-input ASR

There are many previous works on learning acoustic models from raw waveforms [68–73]. Jaitly *et al.* [68] modeled a waveform using a restricted Boltzmann machine. Palaz *et al.* [69, 71] used CNN for extracting acoustic features from speech signals. Tüske *et al.* [70] analyzed which acoustic features, including waveforms, were effective for training acoustic models. Sainath *et al.* [72] proposed feature extraction that consisted of two convolution layers for reducing temporal variations and reducing frequency variations. Ravanelli *et al.* [73,74] used SincNet, which is a learnable filter based on the band-pass filter. Recently, many works proposed feature extraction in an end-to-end manner. Tjandra *et al.* [20] proposed a CNN-based feature extraction architecture for an attention-based model and pretrained the model by minimizing the mean squared distance between lmfb features and the output of the model. Zeghidour *et al.* [75] presented an end-to-end speech recognition system based on convolution layers without RNNs.

3.3.1 Feature Extraction

We adopt a CNN-based model proposed by Zeghidour *et al.* [75]. First, we apply a 2x1 filter initialized by a pre-emphasis filter ([-0.97, 1]). We then apply a

CNN with several channels (40 in this work) and a 1-time sample stride. After taking absolute values, the Hanning window is applied to unify time frames. After processing with $\log(1 + \mathrm{abs}())$, instance normalization [76] is applied to normalize across each channel. Finally, we use frame stacking [77] in which we stack and skip some frames to make a new super-frame. Fig. 3.5 shows this feature extraction part and the masking method in Section 3.3.2.

After frame stacking, we apply the attention-based encoder-decoder model described in Section 2.2.2 to predict the symbol sequence. Based on the cross-entropy between the predicted sequence and the ground-truth sequence, we can update not only the parameters of the attention-based model but also those of the feature extraction part. We expect that the model can extract more effective features for speech recognition than lmfb features. In this model, we do not conduct any pretraining or particular initialization except for the pre-emphasis filter. The CNN filters are initialized with random values drawn by He initialization [78].

3.3.2 Data Augmentation by Masking

In the attention-based model using lmfb features, we can apply data augmentation methods such as SpecAugment [17]. In the waveform-input model, we apply data augmentation after instance normalization in a similar manner. However, in a preliminary experiment, we observe that the masking was not effective when updating all parameters including feature extraction during the training. Therefore, we first train the entire waveform-input ASR model without masking. We then fix the parameters of the feature extraction part and fine-tune the attention-based model part by using masking. The masks are randomly chosen since the order of filters is meaningless, unlike lmfb features.

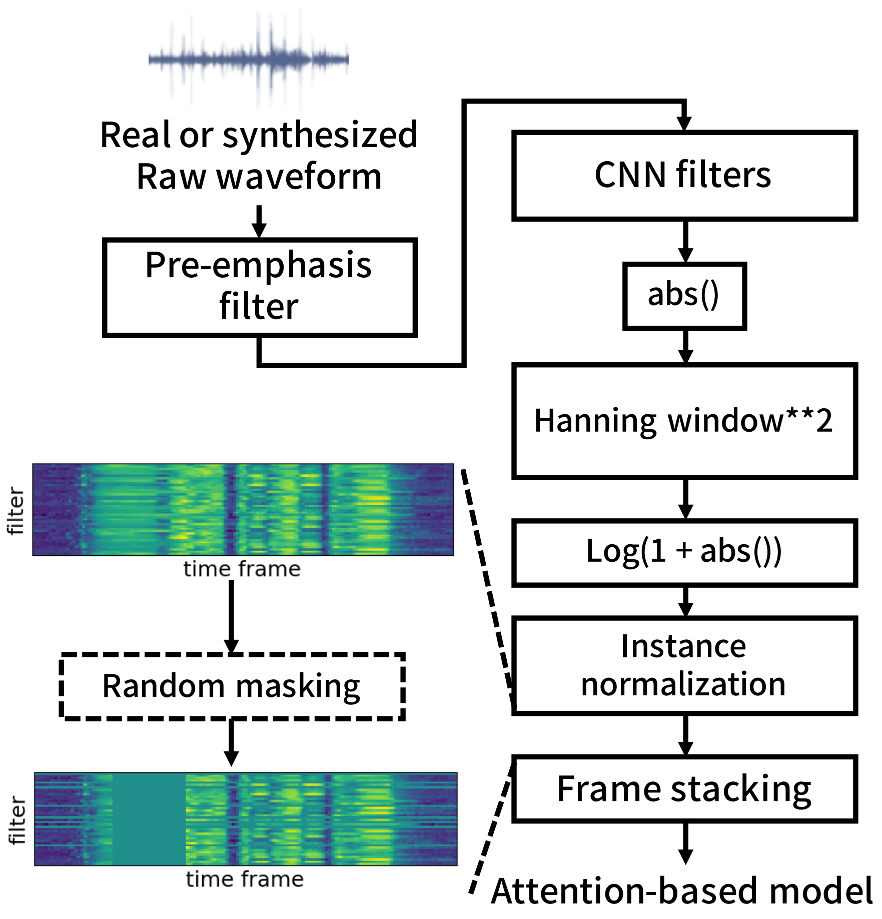


Figure 3.5: Feature extraction part of waveform-input ASR.

3.4 Experimental Evaluations

3.4.1 Datasets and Tasks

All experiments are conducted with Japanese TTS and ASR systems. We use the JSUT corpus [79] to train a single speaker Tacotron 2 model. JSUT has a recording of 7607 utterances of prompt texts read aloud by a female speaker with a total duration of ten hours. We converted the sampling rate to 16 kHz for all datasets. We used 33 phone classes as input. They include special tokens for pause, word boundary, and end of the sentence. For extracting word segmentation and phone sequence of texts, we used MeCab[1], a CRF-based Japanese morphological analyzer.

We primarily used the the Corpus of Spontaneous Japanese (CSJ) [11] to train the ASR model and multi-speaker Tacotron 2 model. CSJ has two distinct sub-corpora, Academic Presentation Speeches (APS) and Simulated Public Speeches (SPS). APS consists of academic presentation speeches in nine different academic societies (engineering, humanities, and social and behavioral sciences). It has 986 speakers (male: 809, female: 177) with 162259 utterances and 247.9 hours. SPS consists of simulated presentation speeches about everyday topics. It has 1704 speakers (male: 799, female: 905) with 238108 utterances and 281 hours. We added all distinct words that occur more than twice in the training data and special tokens ($\langle$sos$\rangle$, $\langle$eos$\rangle$, and $\langle$UNK$\rangle$) to the vocabulary. The vocabulary sizes were 24286 for SPS and 34305 for combination of APS and SPS. CSJ provides the official testsets of APS and SPS. The APS testset has 1.83 hours of speech and 26028 words, and the SPS testset has 1.31 hours of speech and 17134 words. APS contains many technical terms like "F0" and "linear predictive coding."

We used the same dataset for training both Tacotron 2 model and WaveNet vocoder[2]. When generating multi-speaker speeches, we randomly selected a single speaker for each text.

3.4.2 System Configuration

End-to-End Speech Recognition Model

We used 40-channel lmfb features as acoustic features of the ASR models with a shift of 10ms and a width of 25ms. Non-overlapping frame stacking [77] was applied to these features, in which we stacked and skipped three frames to make a super-frame. We implemented an acoustic encoder that consisted of five-layer bidirectional LSTMs with 320 cells. The dropout rate for training each BiLSTM layer was set to 0.2. The attention-based decoder consisted of a one-layer LSTMs with 320 cells, 320-dimensional hidden states with tanh nodes, and a softmax

[1]http://taku910.github.io/mecab
[2]We evaluated the quality of TTS model in [80]

output layer for word entries. We used Adam optimizer with the standard settings described in [81]. We also used a weight decay of 1e-5 and gradient clipping with a threshold of 2.0. The BiLSTM encoders and convolution networks in the decoders were initialized with random values drawn from He initialization [78] and the other networks were initialized in a uniform distribution with the range (-0.1, 0.1). Since providing long input sequences often slows convergence at the beginning of the training, the input data were sorted by the length of frames before creating minibatches. In decoding with the attention-based ASR model, we used a simple beam search with a beam width of 4.

We also trained neural language models with 3 layers of unidirectional LSTMs with 256 memory cells for the language model integration based on shallow fusion. Each word was mapped to a 512-dimensional continuous vector before being fed to LSTMs.

End-to-End Text-to-Speech Model

The phone encoder consists of a 512-dimensional phone embedding, a 256-dimensional speaker embedding, three convolution layers with 512 filters, and one-layer BiLSTM with 256 cells. The location-sensitive attention mechanism [82] summarizes the encoder outputs. The attention weight at each decoding step is calculated by using the 128-dimensional projected vectors of the decoder LSTM state, the encoder output sequence, and the location features. The location features are calculated by convolving 32 one-dimensional convolution filters with a length of 31 to the cumulative vector of the attention weights in all past decoding steps. The pre-net consists of two fully-connected linear layers with 256 ReLU units. We sum the pre-net output, the speaker embedding, and the encoded representation with the attention vector. The decoder network consists of 2-layer unidirectional LSTMs with 1024 memory cells. The decoder LSTM outputs, together with the attention context vector, were passed through a linear projection layer to predict five frames of the target lmfb features.

Fig. 3.6 shows examples of generated speech of the multi-speaker model. We generated these pieces of speech from the same text and different speaker IDs.

Table 3.1: Comparison of lmfb-input ASR and waveform-input ASR [WER (%)].

Training data	SPS		APS+SPS	
Testset	APS	SPS	APS	SPS
lmfb-input ASR w/o masking	23.30	9.69	10.30	9.06
lmfb-input ASR w/ masking	21.97	8.88	9.56	8.75
waveform-input ASR w/o masking	22.56	9.08	10.03	7.97
waveform-input ASR w/ masking	**20.92**	**8.53**	**9.40**	**7.92**

It is confirmed that the multi-speaker model could produce various speakers' speech as these lmfb features and waveform were different in terms of the length of speech and the spectrum patterns because the Tacotron 2 based model not only generates the lmfb features but also estimates the lengths of them.

For synthesizing the waveform, we used WaveNet [83] vocoder[3] with conditioning on 80-dimensional lmfb features of the TTS setting[4]. The upsampling layer assumes the lmfb features with a frame shift of 12.5 ms. The dilation was 1, 2, 4, ..., 128, 1, ..., 128 repeatedly, and the total number of layers is 16. Each dilated CNN has 128 channels with a kernel size of 2. We used the PyTorch [84] toolkit to train all networks with Nvidia TITAN RTX.

The TTS system often fails to generate some pieces of speech correctly. For example, some synthesized speech samples were silent, and their lengths were too short. On the other hand, the lengths of some speech samples were too long. We discarded speech samples if they are are not within some thresholds.

3.4.3 Results of Waveform-input ASR vs. Lmfb-input ASR

We compared the performance of two ASR systems in Table 3.1: one is the standard ASR, whose input is lmfb features. The other is the waveform-input ASR, whose input is raw waveforms. In this table, we evaluated the models trained using the SPS (281 hours) and APS+SPS (528.9 hours) datasets on the APS and SPS testsets. We also applied SpecAugment to the lmfb-input ASR, and the masking method presented in Section 5.2 to the waveform-input ASR. The

[3]https://github.com/NVIDIA/nv-wavenet

[4]In a preliminary experiment, we observed that the WaveNet with the typical ASR setting (40-channel, and 10 ms shift and 25 ms window) does not generate a waveform properly.

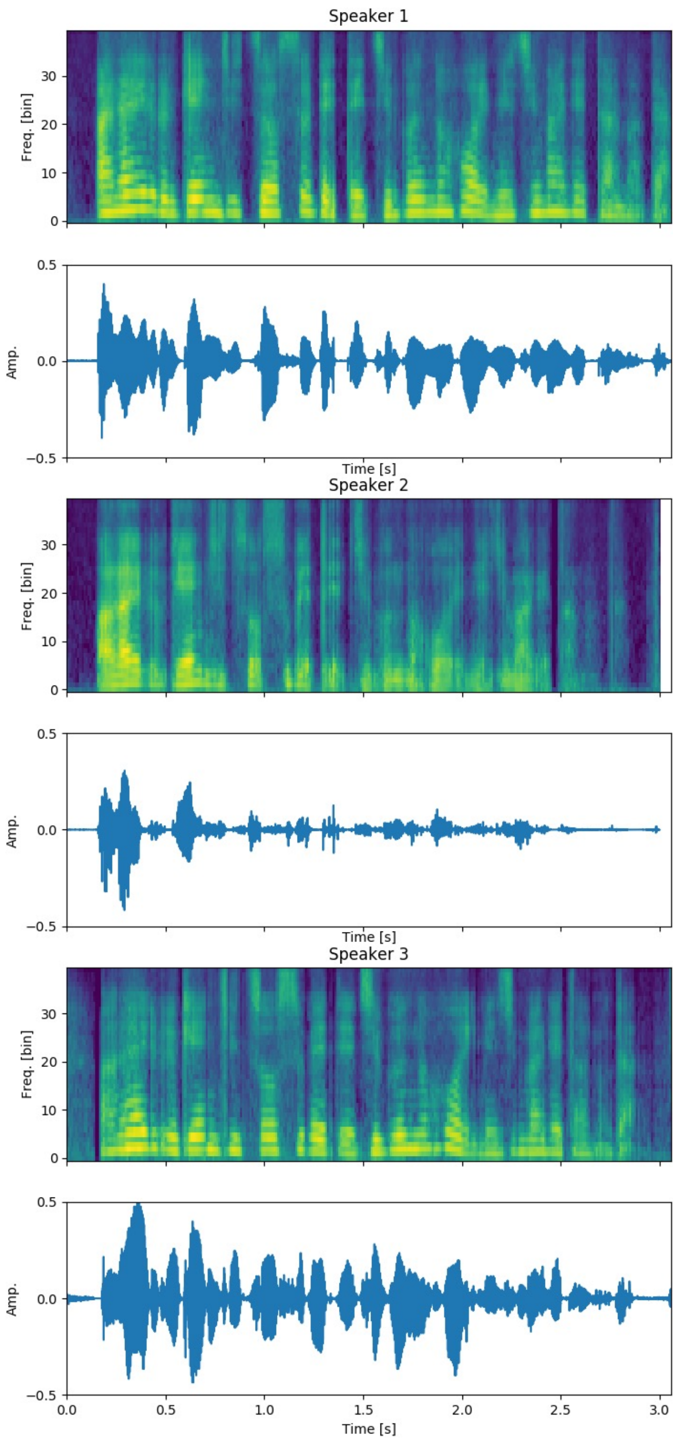


Figure 3.6: Examples of generated speech from the multi-speaker model. These examples were generated from the same text and different speaker IDs.

waveform-input ASR performed comparable to or better than the lmfb-input ASR in all cases. There are significant differences in the WERs for the APS testset using SPS and the SPS testset using APS+SPS. For the APS testset using SPS training set, the waveform-input ASR with masking achieved WER of 20.92%, which is much better than the lmfb-input ASR with SpecAugment. We use the waveform-input ASR with masking and the lmfb-input ASR with SpecAugment as the baseline. We also use the models using APS+SPS dataset as the oracle model.

3.4.4 Results of Simulated Domain Adaptation to APS

In order to simulate a domain adaptation scenario, we chose SPS as a source domain and APS as a target domain since domain adaptation to academic topics from general topics is often required. Using the source domain data, we trained the baseline ASR model and the multi-speaker speech synthesizer. For the target domain, we assumed only text transcription data for adaptation. We generated speech data from the texts of the target domain to retrain the ASR model.

The results are shown in Table 3.2. By using the text-only data of APS, we synthesized about 209 hours of speech data. The baseline WERs were over 20%, as shown in Table 3.2. When we applied an LM shallow fusion using the APS text, the improvement was limited since the APS topics is different from the SPS topics. If we can prepare real speeches for the APS dataset, both the lmfb-input and waveform-input ASR models achieved WER of 9.23%. When we used the single-speaker Tacotron 2 model for synthesizing the additional training data, a large improvement from the baseline is obtained. Among them, “waveform-output TTS and lmfb-input ASR" is better than “lmfb-output TTS and lmfb-input ASR". The best WER in the single speaker setting was 12.90% with the “waveform-output TTS and waveform-input ASR". After the domain adaptation, the LM fusion leads to a large improvement.

The multi-speaker models obtained over 0.6% WER reduction from the single speaker models in all settings. In this experiment, we compared two settings of “lmfb-output TTS and lmfb-input ASR": ASR-matched setting (40-channels

Table 3.2: ASR performance [WER (%)] for the APS testset (eval1). In this table, we used the paired-data of the SPS training set and only transcriptions of the APS training set.

Training data	eval 1	+LM
SPS (real speech, lmfb-input ASR)	21.97	21.72
+ APS (real speech) [oracle]	9.56	9.23
Single speaker TTS:		
lmfb-output TTS and lmfb-input ASR, ASR-matched settings	15.54	14.75
waveform-output TTS and lmfb-input ASR	14.56	13.80
Multi speaker TTS:		
lmfb-output TTS and lmfb-input ASR, ASR-matched settings	13.55	13.08
lmfb-output TTS and lmfb-input ASR, TTS-matched settings	13.32	12.88
waveform-output TTS and lmfb-input ASR	13.05	12.58
SPS (real speech, waveform-input ASR)	20.92	20.80
+ APS (real speech) [oracle]	9.40	9.23
Single speaker TTS:		
waveform-output TTS and waveform-input ASR	**13.68**	**12.90**
Multi speaker TTS:		
waveform-output TTS and waveform-input ASR	**12.77**	**12.27**

Table 3.3: Recognition rate of original OOV in the eval1 (APS testset).

# unknown words for SPS vocabulary	1143
Coverage rate of original OOV	905 / 1143 (80.05%)
Recognition rate of original OOV	782 / 1143 (68.42%)

using a 25 ms window with a shift of 10 ms) and TTS-matched setting (80-channels using a 50 ms window with a shift of 12.5 ms). It is shown that the former is outperformed by the latter, but they are behind the performance of the waveform-output using the WaveNet vocoder, which improved the quality of lmfb features. In this case, too, the best WER of the augmented model was 12.27% with the “waveform-output TTS and waveform-input ASR" with the multi-speaker model. This model improved the WER by 41.01% relatively from the model without domain adaptation. These results show that the waveform-output TTS using WaveNet makes the ASR model better than the lmfb-output TTS. The waveform-input ASR realized further improvement.

Table 3.3 shows how the adapted model recognizes unknown words that do

Table 3.4: ASR performance [WER (%)] for JNAS testset. In this table, we use multi-speaker TTS trained with CSJ. JNAS dataset has 85.5 hours of real speeches.

Training data	WER(%)
CSJ (real speech, lmfb-input ASR) [baseline]	15.79
+ JNAS (real speech, lmfb-input ASR) [oracle]	5.24
+ JNAS (waveform-output TTS and lmfb-input ASR)	10.18
+ newspaper article (waveform-output TTS and lmfb-input ASR)	9.07
CSJ (real speech, waveform-input ASR) [baseline]	13.79
+ JNAS (real speech, waveform-input ASR) [oracle]	5.52
+ JNAS (waveform-output TTS and waveform-input ASR)	10.12
+ newspaper article **(waveform-output TTS and waveform-input ASR)**	**7.95**

not appear in the target domain. When we use only SPS vocabulary, 1143 words of the APS testset cannot be recognized. Among them, 905 words (80.05%) are included in the augmented (SPS + APS) vocabulary. The "data augmentation method (waveform-output TTS and waveform-input ASR)" in Table 3.2 correctly recognized 782 out of 1143 words (68.42%). It enhanced the ASR model's language model capability and recognized a majority of new words.

3.4.5 Results of Adaptation to Newspaper Domain Leveraging A Large Amount of Newspaper Texts

Next, we conducted adaptation to a newspaper domain. For this experiment, we used the JNAS dataset [10]. JNAS is a read speech corpus of Japanese newspaper articles. It has 85.5 hours for training data and 20 minutes for testset. For synthesizing training data for the JNAS testset, we also used an external language resource. We collected 500k sentences randomly from newspaper articles of Mainichi Shinbun, one of the major newspapers in Japan.

Table 3.4 shows the evaluation of adaptation to the newspaper domain using the JNAS testset. In this evaluation, we used the real speech of both CSJ-APS and CSJ-SPS (528.8 hours) to train the ASR and multi-speaker TTS models. We generated 58 hours speeches from the JNAS transcript. The original JNAS data provides 85.5 hours of real speeches. The generated speech is much shorter (58.8

hours) than the real one because we do not use the silence as an input label of TTS, and the silence, including short pause, is not inserted in the synthesized speech. We also generated 768.9-hour data from 500k newspaper articles. The lmfb-input and waveform-input ASR using only the CSJ dataset obtained WERs of 15.79% and 13.79%. They are very high because the CSJ domain is different from the JNAS domain. When we mixed the CSJ and original JNAS dataset, the WERs were 5.24% and 5.52%. The models that used the CSJ and synthetic JNAS dataset improved approximately 4∼5% in WERs from the baseline CSJ model. When a large amount of speech was generated using newspaper articles, a large improvement is achieved. The best WER was 7.95% with the "waveform-output TTS and waveform-input ASR". This result showed that the waveform-input ASR enhanced with the data augmentation is most effective when preparing a large amount of synthesized data.

3.4.6 Analysis on Learned Filter

In the feature extraction part, the role of CNN filters is to extract characteristics from a waveform to minimize the cross-entropy between a predicted label sequence and a ground-truth label sequence. In the conventional ASR systems, we use Mel filter to approximate the human auditory system's response. When it comes to decreasing the dimension of input features, the role of CNN filters is close to the Mel filter. While the Mel filter is used for the frequency-domain spectrogram, the feature extraction is used for the time-domain waveform. Fig. 3.7 shows the examples of learned filters. We extract them from CNN filters in Fig. 3.7. We see that each CNN filter extracts different characteristics. In particular, they extract different frequencies. Fig. 3.8 shows a comparison with the Mel filter (top) and the learned filter of the proposed model (bottom) on the frequency domain. We obtain the frequency-domain learned filter to transform the time-domain weights of CNN filters by an STFT. Note that the order of CNN filters is meaningless, and all filters are not in order of frequency. In Fig. 3.8, we roughly sorted them by frequency. The horizontal axis indicates the frequency, and the vertical axis indicates the channel dimension.

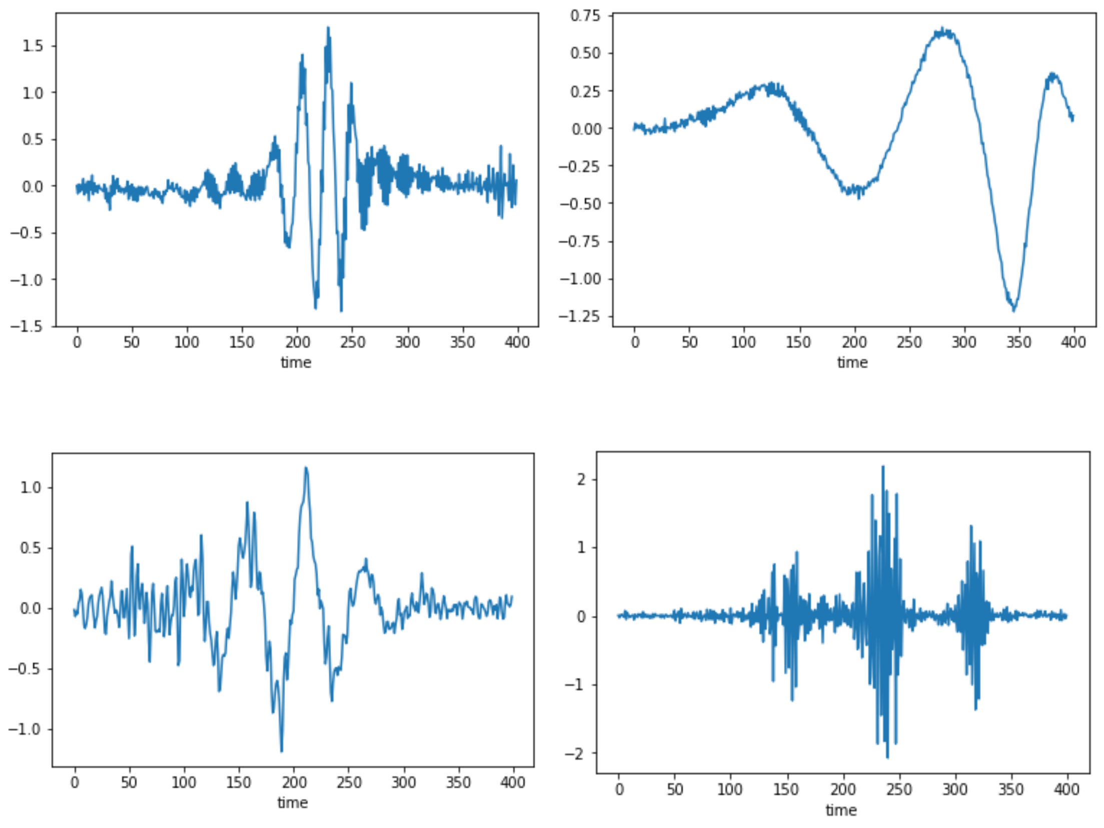


Figure 3.7: Examples of learned filter on time domain. These filters are extracted from weights of CNN filters in Fig. 3.5. The kernel size of each CNN filter is 400 samples for 16kHz waveforms (400/16000 = 25ms).

As in the Mel filter bank, we see that the learned filter is concentrated in the low-frequency range and does not focus on the high-frequency range. This is because the energy of human speech on high frequency is high, and the low-frequency part is important to recognize human speech. It can be seen that many of them look at a wider range of frequencies per filter, and the number of filters that focus on low frequency is larger than the Mel filter.

3.5 Summary

In this chapter, we have presented the waveform-based data augmentation method for end-to-end ASR systems. To realize the waveform-output TTS, we

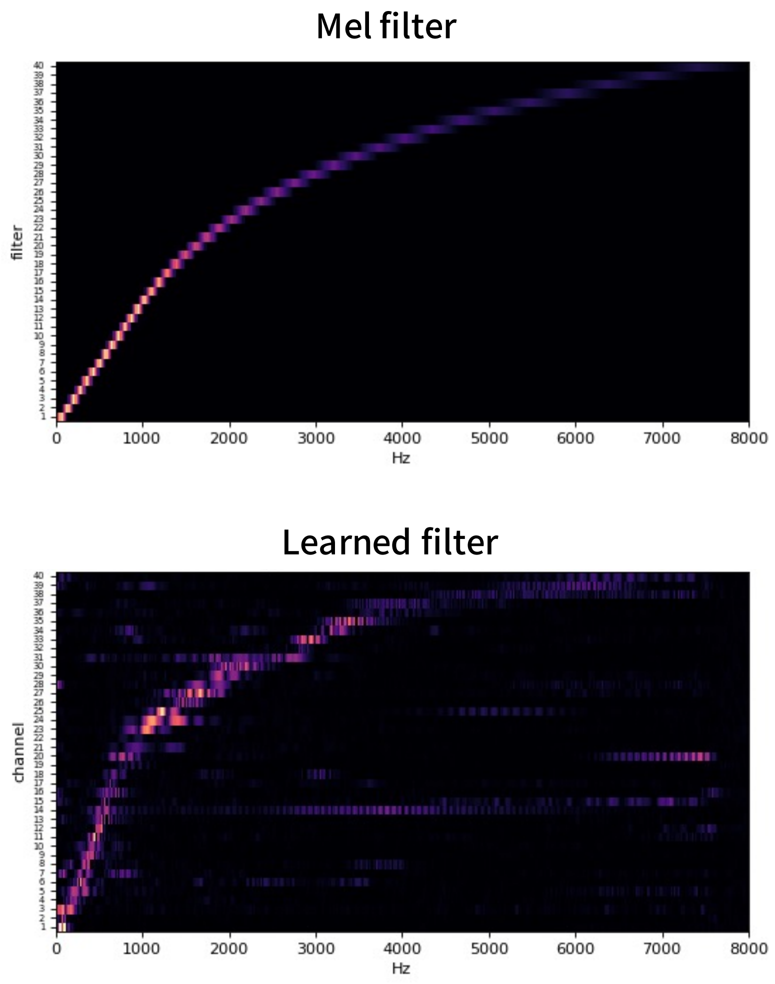


Figure 3.8: Comparison of Mel filter and learned filter. We roughly sorted learned filter by frequency.

use the WaveNet vocoder. The WaveNet vocoder makes better lmfb features, improving the ASR performance. To fully utilize waveform-output TTS, we have also designed the waveform-input ASR and a fine-tuning method by masking. We have shown that the masking method for the wave-input ASR achieved comparable or better performance than the standard lmfb-input ASR with SpecAugment. We have also demonstrated that the waveform-output TTS and waveform-input ASR achieved better performance than the waveform-output TTS and lmfb-input ASR in two domain adaptation scenarios. Future work includes improvement of multi-speaker TTS model for spontaneous speech for generating better and more data set.

Chapter 4

Generating ASR Features via a Discrete Representation

4.1 Introduction

In Chapter 3, we proposed the waveform-input ASR for effective training. Although adding a frontend is effective for improving the ASR performance, the amount of training data is increased since the number of parameters is larger. Above all, it is not easy to prepare the paired data of speech and transcription. Most recently, approaches using speech-only data have been investigated. In particular, unsupervised pre-training or self-supervised learning is attracting increasing interest. Self-supervised learning has emerged as a paradigm to learn general data representations from unlabeled data. It has been particularly successful for natural language processing [85, 86] and computer vision [87, 88]. In the speech processing field, many attempts of the self-supervised manner have been made [12–14, 89–91] in order to learn the representation. Oord *et al.* introduced contrastive predictive coding (CPC) [12]. The CPC combines predicting future observations (predictive coding) with a probabilistic contrastive loss. It predicts the future samples in a latent space and maximizes the loss when feeding negative examples. Schneider *et al.* explored unsupervised pre-training for speech recognition refereed to as wav2vec [89]. The wav2vec is pretrained with a simple multi-layer convolutional neural network optimized via a noisy contrastive binary classification task. Unlike the CPC, the wav2vec is designed for frame-wise phoneme classification and applies the learned representations to improve strong supervised ASR systems. Chung *et al.* proposed autoregressive predictive coding (APC) that uses autoregressive models to encode temporal information of a past acoustic sequence [90]. Beaevski *et al.* introduced vector quantization discrete representation into the wav2vec architecture and BERT architecture to make sophisticated context representation from the discrete representation [13]. Moreover, they masked the speech input in the latent space and solved a contrastive task defined over a quantization of the latent representations, which are jointly learned referred to as wav2vec 2.0 [14]. Liu *et al.* used multi-layer transformer encoders to achieve bidirectional encoding, and this framework allows the model to consider past and future contexts at the same time [91]. These works achieved comparable or better performance than the conventional lmfb-input ASR models.

In this chapter, we investigate the use of self-supervised features for data augmentation using TTS. The use of synthesized lmfb data brings only a limited improvement compared with using real data since the synthesized features are different from real speech. Moreover, the TTS system often generates unrealistic speech.

In this work, we propose a novel data augmentation scheme using a discrete representation. In this scheme, TTS generates a discrete representation instead of log Mel-scale filterbank (lmfb) features and predicts discrete ID sequences from texts. We also use features converted from the discrete ID when we train an ASR model. We adopt vq-wav2vec [13], which is an unsupervised training method to produce a discrete representation. The use of a discrete representation has two benefits over the standard use of TTS by experimental evaluations: (1) a discrete representation is much easier to predict than the lmfb features of continuous values; (2) it reduces speaker dependency, which the TTS has trouble separating from the segmental information.

4.2 VQ-wav2vec

Vq-wav2vec learns a discrete representation of speech frames through a self-supervised future time-step prediction task. The model is based on three convolutional neural networks, in which the encoder produces a representation $\boldsymbol{z_i}$ for each time step i with a rate of 100 Hz, a quantization module converts $\boldsymbol{z_i}$ to a discrete representation $\hat{\boldsymbol{z}}_i$, and the aggregator combines the multiple encoder time steps into a new representation $\boldsymbol{c_i}$. The quantization module replaces the original representation $\boldsymbol{z_i}$ by $\hat{\boldsymbol{z}}_i = \text{concat}(\boldsymbol{e}_{i,1}, ..., \boldsymbol{e}_{i,G})$ from a shared fixed-size codebook $\boldsymbol{E} \in \mathbb{R}^{V \times d/G}$, which contains V representations of size d/G, where d is the dimension of $\hat{\boldsymbol{z}}_i$, and G is the number of groups. We represent each row by an integer index and then represent the feature vector $\hat{\boldsymbol{z}}_i$ by the indices $\boldsymbol{w}_i \in [V]^G$, where each element $w_{i,g}$ corresponds to a fixed codebook vector. For instance, when $G = 2$, two-dimensional code is generated e.g. (15, 24) and (36, 87). We concatenate the elements, e.g. "15-24" and "36-87" when feeding to BERT.

In the context prediction, wav2vec loss [89] is defined as follows:

$$\mathcal{L}_k^{wav2vec} = -\sum_{i=1}^{T-k} (\log \sigma(\hat{\boldsymbol{z}}_{i+k}^{\mathsf{T}}\, h(\boldsymbol{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\hat{\boldsymbol{z}}} \sim p_n} [\log \sigma(-\tilde{\hat{\boldsymbol{z}}}^{\mathsf{T}}\, h(\boldsymbol{c_i})]) \tag{4.1}$$

where T is the sequence length, σ is a sigmoid function, h is an affine transformation, and thus $\sigma(\hat{\boldsymbol{z}}_{i+k}^{\mathsf{T}}\, h(\boldsymbol{c}_i))$ is the probability of a sample $\boldsymbol{z}_{i+k}$, that is k-steps in the future, being correctly predicted. $\tilde{\hat{\boldsymbol{z}}}$ are negative samples uniformly drawn from the same uttrance. In addition to wav2vec loss, Gumbel-Softmax or K-means loss is added between $\hat{\boldsymbol{z}}_i$ and $\boldsymbol{z}_i$. In this work, we used K-means clustering and added the loss of vector quantization [92] to $\mathcal{L}_k^{wav2vec}$. As a result, the vq-wav2vec model maximizes $\mathcal{L}^{vq-wav2vec}$ as follows:

$$\mathcal{L}^{vq-wav2vec} = \sum_{k=1}^{K} \mathcal{L}_k^{wav2vec} + (\|\text{sg}(\boldsymbol{z}) - \hat{\boldsymbol{z}}\|^2 + \gamma \|\boldsymbol{z} - \text{sg}(\hat{\boldsymbol{z}})\|^2) \tag{4.2}$$

where $\text{sg}(x) \equiv x$, $\frac{\text{d}}{\text{d}x}\text{sg}(x) \equiv 0$, and γ is a hyperparameter.

4.2.1 BERT for VQ Codes

BERT is also used for training a more sophisticated representation of the context features in vq-wav2vec architecture. BERT [86] is originally one of the self-supervised learning approaches in natural language processing. The main principal characteristics of BERT are introducing bidirectional encoding based on Transformer and a masked language model (MLM) objective. While a standard language model objective (softmax cross entropy) makes the representation of left-to-right context, the MLM objective enables the representation of both the left and the right context, which allows us to pretrain a deep bidirectional Transformer. In the MLM task, we randomly replace the label with a special [MASK] label. The BERT model predicts the labels of [MASK] instead of predicting the next timestep label. In addition, it is pre-trained using a next sentence prediction task to capture the relationship between sentences. In the fine-tuning step, we add a linear layer to predict the target label of interest and update the whole parameters.

In the vq-wav2vec task, we use the BERT for discrete representations and apply the MLM task. We randomly make the [MASK] labels for the concatenated vq-codes. We do not use next sentence prediction because the BERT model for vq-wav2vec does not need to train the relationship between two speeches. In this work, we also do not fine-tune the BERT model and use the BERT model as the feature extraction model. We use the last layer's hidden states as the input features of the ASR model.

4.3 Data Augmentation via Discrete IDs

4.3.1 Conventional Data Augmentation by TTS

The conventional data augmentation scheme using TTS [22–24, 30, 66] has four steps.

1. Training TTS which predicts lmfb features from a phone sequence.
2. Generating lmfb features using unpaired texts.

3. Converting lmfb features into waveform by a vocoder.
4. ASR training using the generated waveform data mixed with real data.

In this scheme, the vocoder is used for bridging the gap between the lmfb features of the ASR model and the TTS model.

Although the generated speech data give some improvement of the ASR performance, the gain is limited since the TTS does not completely reproduce the real speech. To alleviate the problem, Mimura *et al.* [22] froze the ASR acoustic encoder when training with the synthesized data. Wang *et al.* [27,28] investigated the consistency regularization when incorporating TTS with ASR. Zheng *et al.* [30] introduced the loss for regularization of the decoder when finetuning to improve the ASR model for out-of-vocabulary words. Fazel *et al.* [32] investigated the multi-stage training strategy by combining weighted multi-style training, data augmentation, encoder freezing, and parameter regularization. Several studies used speaker information to generate multi-speaker speech such as speaker ID [23], VAE latent variables [24], pre-trained speaker verification model [32], and global style token (GST) [66]. Chen *et al.* [26] jointly trained the pre-trained TTS and ASR using a GAN-based model to increase the acoustic diversity in the synthesized data.

4.3.2 Data Augmentation via Discrete ID Sequences

A major problem with the conventional data augmentation using TTS is the mismatch between the synthetic and real speech data. Conventionally, lmfb features are used as an intermediate representation for both TTS and ASR. Since an lmfb feature is a continuous value and the loss used for training does not address phonetic constraints, the TTS model often generates unrealistic data that do not exist in real speech. Moreover, neural TTS tends to generate many errors such as too short or repeated speech, and multi-speaker TTS model causes more errors than the single speaker model. It is much more difficult to train a multi-speaker TTS model since the amount of training data available is usually quite limited per speaker and multi-speaker features have more variety. These

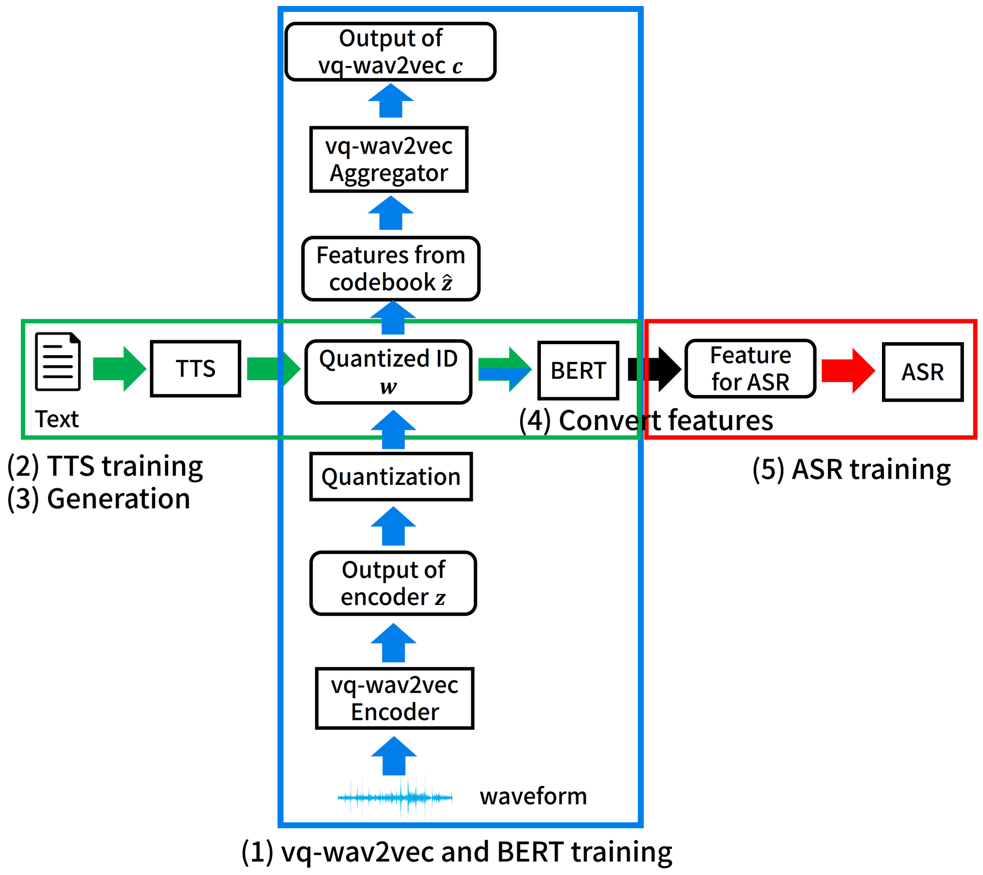


Figure 4.1: Overall architecture of the proposed data augmentation for ASR. (1) Train vq-wav2vec and BERT using the real waveform. (2) Perform TTS training using discrete IDs and texts (phones). (3) Generate discrete IDs from texts (phones). (4) Generate features from discrete IDs via BERT (5) Perform ASR training using the final hidden states of BERT and texts (subwords).

problems pose a bottleneck for effective data augmentation for ASR model training.

To address the above two problems, we introduce a discrete representation to be used for both ASR and TTS. Theoretically, the unreal features which do not exist in real features are not generated because the TTS model selects the discrete IDs from a finite set. Moreover, using discrete IDs for a TTS target makes the TTS task easier since selecting IDs from the fixed classes is considered to be easier than predicting a continuous values. Fig. 4.1 shows an overview of the proposed method. In this work, we use the vq-wav2vec module for the

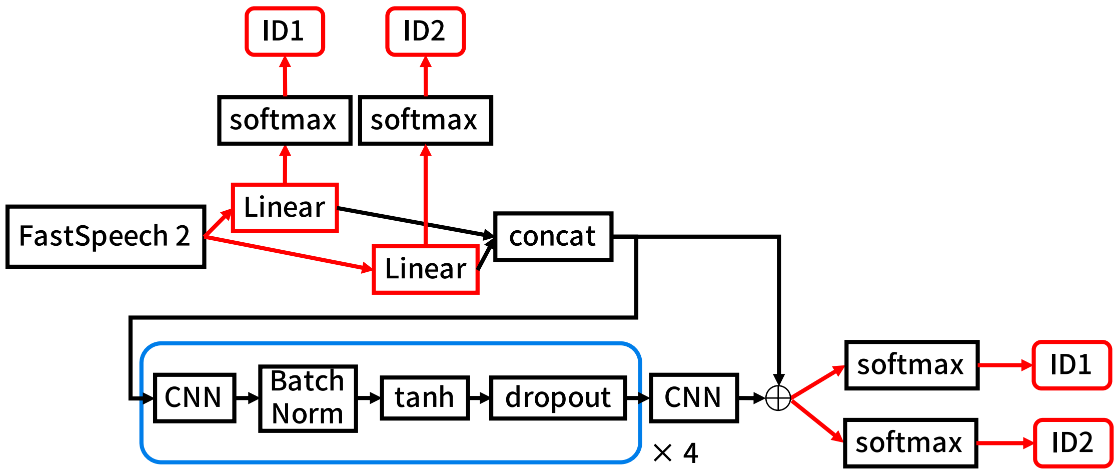


Figure 4.2: Postnet architecture of the proposed TTS system. The vq-wav2vec model generates two IDs from a 10ms segment of speech, which the TTS system predicts.

intermediate representation to convert waveform into the discrete IDs because the vq-wav2vec achieved promising performance of ASR [13]. The output layer of the TTS is a softmax layer corresponding to the discrete IDs. The proposed architecture has five steps.

1. **Train vq-wav2vec and BERT**.
2. Train TTS to predict **discrete IDs** from texts using paired training data.
3. Generate **discrete ID sequences** from text-only training data for ASR.
4. Convert the ID sequences to ASR features through BERT.
5. Perform ASR training using the generated data mixed with real data.

In step 1, we use the original vq-wav2vec with which the two discrete IDs ($G = 2$) are generated every 10ms.

Then, we use the FastSpeech 2-based model to synthesize discrete ID sequences. For this purpose, we replace the lmfb prediction layer of FastSpeech 2 with two output layers to predict these discrete IDs. We add a Postnet in order to divide the hidden state from FastSpeech 2 into two (G) streams of representations for two (G) IDs and to smooth this sequence of representations. Fig. 4.2 shows the Postnet architecture of the proposed method. The Postnet concatenates the hidden states of the linear layers corresponding to ID1 and ID2 and then applies five convolution layers. We finally sum the outputs of the linear layers and CNN and separate them into two outputs corresponding to ID1 and ID2. For the training, we used two softmax cross-entropy losses for the FastSpeech 2 output and Postnet output. When generating IDs, we use the Postnet output.

For the FastSpeech 2 model training, we also need to align between the transcriptions and audio in advance. For this purpose, we train a CTC-based ASR model on the same data and conduct forced alignment. The original FastSpeech 2 model uses additional prosodic information such as F0 and energy. In particular, we use F0 and energy to build a baseline FastSpeech 2 model that generates lmfb features. However, we do not use this information in our method that predicts discrete ID sequences. In inference, we generate an ID sequence from a phone sequence. Unlike the standard lmfb-output FastSpeech 2 model, we predict IDs by selecting index which have maximum value from outputs. The generated IDs are concatenated and fed into BERT to generate the ASR features.

4.4 Experimental Evaluations

4.4.1 Datasets and Tasks

All experiments were conducted with English TTS and ASR models using the LibriSpeech [93] corpus. We converted the sampling rate of the waveforms to 16 kHz for all datasets. To train the TTS model, we used *train-clean-100* of the LibriTTS corpus [94], which is derived from LibriSpeech and designed for TTS tasks. In LibriTTS, *train-clean-100*, which is subset of LibriSpeech *train-clean-100*, has 53 hours' worth of paired data from 247 speakers (male: 123, female: 124).

We used 85 phones and the speaker ID as inputs to the TTS. We converted each word sequence to a phone sequence with an open-source grapheme-to-phone tool[1]. To obtain the ground-truth alignment for FastSpeech 2 training, we also used a CTC-based ASR model trained with LibriTTS *train-clean-100*. We also trained a standard FastSpeech 2 model as a baseline, which generated

[1]https://github.com/Kyubyong/g2p

80-dimensional lmfb features based on a 50-ms window with a shift of 12.5ms. To obtain F0, we used WORLD [45]. We also used MelGAN [54] conditioning on the lmfb features trained with LibriTTS *train-clean-100* to generate a waveform and then converted it into the ASR-matched lmfb features again.

After data augmentation with the TTS, we trained and evaluated the ASR models with LibriSpeech and TED-LIUM release 2 (TED-LIUM 2) [95]. We used real speech and transcription data of LibriSpeech *train-clean-100*.

We augmented the data by using the text data of *train-clean-360* for LibriSpeech testset and TED-LIUM 2 training set for TED-LIUM 2 testset. We trained ASR models on three training data on each corpus.

- **Baseline model**: *train-clean-100* (real).
- **Augmented model**: *train-clean-100* (real) + synthesized data of *train-clean-360* or TED-LIUM 2.
- **Oracle model**: *train-clean-100* (real) + **real** data of *train-clean-360* or TED-LIUM 2.

We also prepared standard lmfb-input ASR systems for comparison. We used 80-dimensional lmfb features based on a 25-ms window with a shift of 10 ms. In the vq-input ASR systems, we used the 1024-dimensional final hidden states of BERT as the input. We used the pre-trained vq-wav2vec with K-means clustering and RoBERTa$_{\text{BASE}}$ models[2], which were trained with waveforms of LibriSpeech 960h. In the experiments on the vq-input ASR, we did not finetune the vq-wav2vec and BERT models. In all tasks, we used 1000-class subwords based on byte-pair encoding [96]. We used the transcription and the official text data for training the language model of each dataset on the basis of the same subwords.

4.4.2 Network Configurations

We built an lmfb-output TTS model and a discrete ID-output TTS model based on FastSpeech 2. The FastSpeech 2-based models had six transformer layers in

[2]https://github.com/pytorch/fairseq/tree/master/examples/wav2vec

the encoder and decoder with 384 model dimensions, 1536 feed-forward network dimensions, and 4 attention heads. The Postnet has five CNN layers with kernel size 5. The learning rate was warmed up over the first 1000 updates and then linearly decayed. In the discrete ID-output TTS, we used an L1 loss for the alignment prediction and two softmax cross-entropy losses for the FastSpeech 2 output and Postnet output, which corresponded to IDs. In the lmfb-output TTS, we added the Postnet without blocks for generating discrete IDs for fair comparison. We used five L1 losses in order to predict the alignment, F0, energy, FastSpeech 2-output lmfb features, and Postnet-output lmfb features.

We also built an lmfb-input ASR model and vq-input ASR, whose input consisted of BERT's hidden states. The ASR models were attention-based encoder-decoder models. The encoder had 5-layer BiLSTMs with 320-dimensional hidden states. The decoder was composed of a 1-layer unidirectional LSTM with an attention mechanism. SpecAugment [17] is applied, with two frequency masks with $F = 27$ and two time masks with $T = 100$ in the lmfb-input ASR model and two frequency masks with $F = 260$ and two time masks with $T = 100$ in the vq-input ASR model. We sorted all of the training data in ascending order of speech length and trained the ASR model epoch by epoch. In the augmentation, we also sorted the mixed data and did not try to balance the real and synthetic speech in a batch.

In inference, we set the beam search width to 4 and applied shallow fusion [97] with an LM weight of 0.2. The language model was composed of four unidirectional LSTM layers with 512 model dimensions.

The vq-wav2vec is composed of eight CNN encoder layers with 512 channels and 12 CNN aggregator layers with 512 channels [13]. K-means clustering with two groups of 320 classes ($V = 320, G = 2, d = 512$) was used for vector quantization. For BERT-based ASR feature generation, we used RoBERTa$_{\mathrm{BASE}}$ models with 12 layers, 768 model dimensions, 3072 feed-forward network dimensions, and 12 attention heads to generate ASR features [13].

Table 4.1: ASR results (WER) on LibriSpeech.

		dev-clean	dev-other	test-clean	test-other
Baseline					
Real 100h	lmfb	9.40	30.22	9.76	32.08
	vq	9.59	25.34	10.14	26.20
Augmented					
Real 100h	lmfb	7.12	29.41	7.87	30.16
+ TTS 360h	**vq**	**6.50**	**21.00**	**6.96**	**22.33**
Oracle					
Real 100h	lmfb	4.70	18.40	5.06	18.65
+ Real 360h	vq	5.54	18.16	5.76	18.92

4.4.3 Results

Results of LibriSpeech

Table 4.1 lists the word error rates (WERs) on the LibriSpeech dev and test sets. With the baseline model using LibriSpeech 100h, the WERs of lmfb-input and vq-input ASR were over 9% in the clean settings and 25% in the other settings. When augmented with 360 hours of TTS data, the conventional lmfb-based model achieved 2.28-, 0.81-, 1.89-, and 1.92-point improvements on each test set from the baseline lmfb-input ASR model. On the other hand, the proposed vq-based augmentation achieved 3.09-, 4.34-, 3.18-, and 3.87-point improvements from the baseline. We presume that the oracle model, which we trained with the real data of *train-clean-100* and *train-clean-360*, was the upper bound of the augmented models. The reduction in WER from that of the baseline augmentation achieved by the proposed augmentation method relative to the error between the baseline and oracle turned out to be 76.3%, 60.4%, 73.6%, 53.2%, for dev-clean, dev-other, test-clean, and test-other, respectively. These ratios are much better than those achieved by lmfb-input data augmentation, which were 48.5%, 6.9%, 40.2%, and 14.3%, respectively. These results show that the proposed method mitigated the mismatch between synthetic and real features. In particular, it was more effective on the *dev-other* and *test-other* sets. This suggests that vq-wav2vec is more robust in adverse conditions for ASR.

Table 4.2: ASR results (WER) on TED-LIUM 2

		dev	test
Baseline			
Real 100h	lmfb	34.58	33.75
	vq	31.29	31.99
Augmented			
Real 100h	lmfb	28.11	30.05
+ TTS TED-LIUM 2	**vq**	**21.49**	**22.49**
Oracle			
Real 100h	lmfb	10.56	9.88
+ Real Tedilum 2	vq	13.24	13.33

Results of domain adaptation to TED-LIUM 2

In the experiment described in the previous section, we used LibriSpeech data to train all of the vq-wav2vec, TTS, and ASR models. In the experiment described in this section, we applied the proposed model to a completely different task (TED-LIUM 2). Table 4.2 shows the WERs on the dev and test set of TED-LIUM 2. With the baseline model using only LibriSpeech 100h, the WERs of the lmfb-input and vq-input ASR models were 30% apparently due to the speaking style difference. The conventional data augmentation yielded 6.5-point and 3.7-point improvements on the respective test sets. The proposed data augmentation yielded a 9.8-point improvement on the dev set and 9.5 points on the test set. The proposed model filled 54.3% of the WER gap between the baseline and the oracle. It was also effective in a completely unknown domain. These results show that the discrete representation is robust against domain mismatches.

ASR Training Using Only Synthetic Features

Table 4.3 shows WERs when we trained the ASR models using only synthetic data. Even when the TTS model was trained with the LibriSpeech dataset, the WERs of the lmfb-input ASR model were over 50% for the LibriSpeech clean data sets. This is because there is a serious mismatch between the generated and real lmfb features. The proposed model had an WER of 15.18% in the dev-clean set and 16.85% on the test-clean set. These results show that the discrete ID-based

Table 4.3: ASR results (WER) using only synthetic features. In the LibriSpeech task, we evaluated the ASR models on dev-clean and test-clean.

		dev	test
TTS LibriSpeech 460h	lmfb	53.97	52.16
	vq	**15.18**	**16.85**
TTS TED-LIUM 2 211h	lmfb	90.41	88.34
	vq	**40.04**	**43.54**

TTS model generates features similar to those of real data. The lmfb-based ASR model did not work at all on TED-LIUM 2.

4.5 Discussions

Fig. 4.3 shows a t-SNE visualization calculated for the same phones using 5-speaker lmfb features and vq-wav2vec features. We can clearly distinguish speakers in the lmfb features, but have trouble identifying speakers in the vq-wav2vec features. This result suggests that vq-wav2vec reduces speaker-dependent information. This property is good for stable training of the multi-speaker TTS.

Fig. 4.4 compares t-SNE visualizations of the real and synthetic features. The real features were extracted from one speaker. We can see that the features are separated by phones in both features. In the proposed method, the same phone's synthetic and real features are much more similar compared to the lmfb features. Accordingly, the proposed approach improved the ASR performance because of its characteristics of (1) reducing speaker diversity and (2) using codebook indices rather than generating speech.

4.6 Summary

We have proposed a novel data augmentation method for ASR that leverages TTS via a discrete representation. The conventional method has a serious mismatch between the generated and real speech, which results in a limited improvement in ASR. To mitigate this mismatch, we introduce vq-wav2vec-based IDs as an

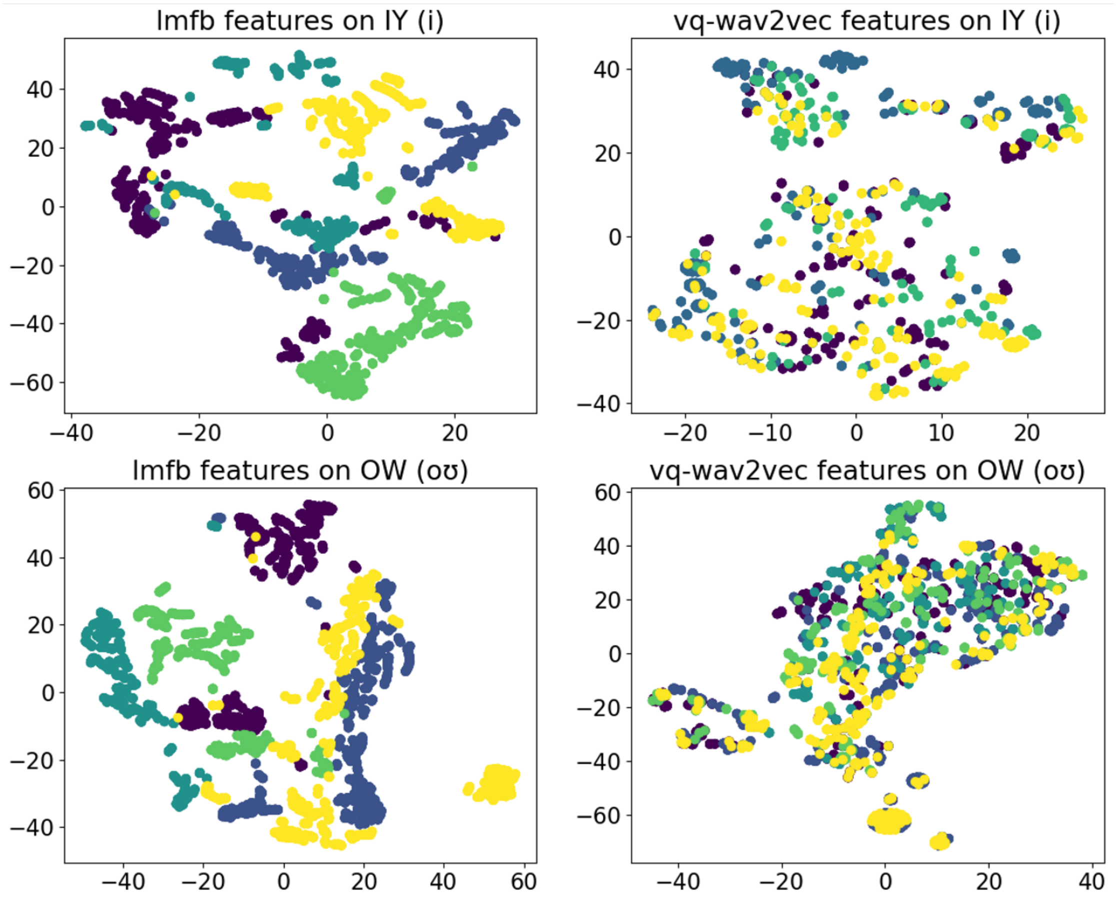


Figure 4.3: Comparison of t-SNE visualizations of lmfb and vq-wav2vec features. Each color represents a different speaker.

intermediate representation instead of lmfb features. In the experimental evaluations, the proposed model gave a more effective data augmentation. Moreover, it reduced some speaker-dependent information and generated features that were close to the real data.

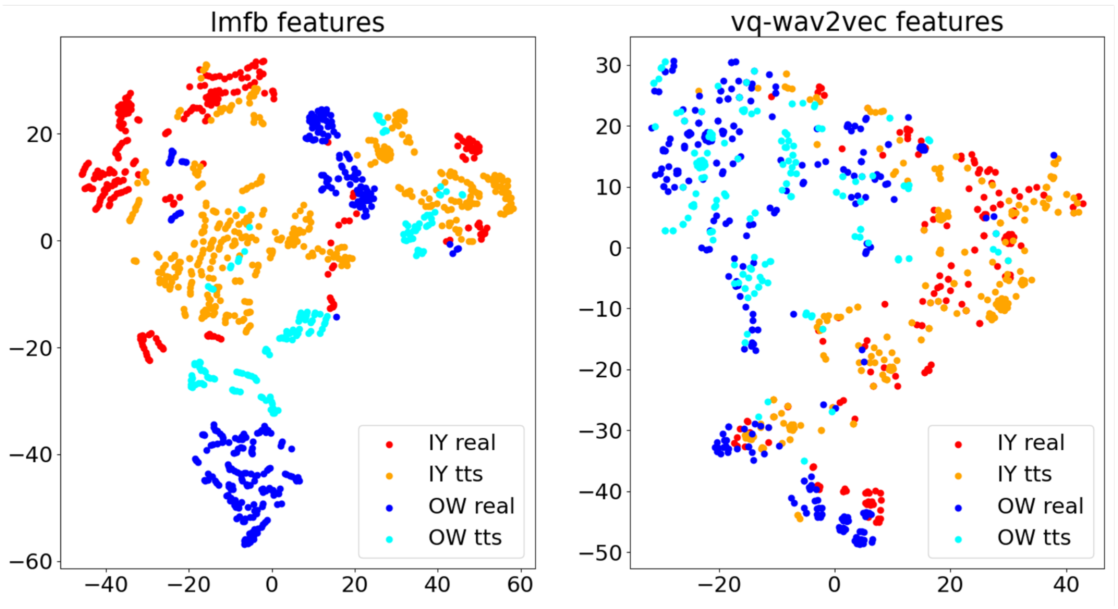


Figure 4.4: Comparison of t-SNE visualizations using lmfb and vq-wav2vec features on two phones (IY and OW). Bluish colors represent OW sounds and reddish colors represent IY sounds, darker being real and lighter synthetic.

Chapter 5

Mel-to-Mel Network to Refine Generated Speech

5.1 Introduction

In Chapter 3, we investigated waveform-based data augmentation using the vocoder. Other works also used a vocoder to convert the lmfb feature into a waveform [24,25,29–32], which is again converted into an lmfb feature used for the ASR input. An alternative way is to directly use the synthesized lmfb features without any post-processing [22,23,26–28], but it results in worse performance.

The neural network-based vocoder is generally used since it delivers better performance than the conventional vocoders such as the Griffin-Lim algorithm [29]. The benefit of using the vocoder is that we can design ASR and TTS models independently since we can use an optimal setting for lmfb features for respective systems. Moreover, the vocoder improves the quality of data and performance of augmentation compared with the direct use of lmfb features. In this way, the vocoder is regarded as a post-processing network for enhancing the lmfb feature.

However, synthesizing waveforms takes a huge amount of time because the waveform has much longer sequence lengths than the lmfb feature. Moreover, we need to generate a huge amount of the ASR training data, and the ASR system needs not waveforms but lmfb features. In this Chapter, we propose a phone-informed post-processing network to refine the synthesized lmfb features. The proposed network focuses on filling the gap between real and synthesized lmfb features and is used instead of the vocoder. Refinement on the lmfb feature takes less inference time than the vocoder synthesizing waveforms. For improved enhancement, we use text information, specifically phone information of the speech, which is readily available in the TTS task.

5.2 Phone-Informed Post-Processing Network for Speech Refinement

5.2.1 Baseline Architecture of Data Generation

For data augmentation for ASR, we compose a multi-speaker text-to-mel network, which is generally used [23–25, 32]. There are some options for the multi-speaker embeddings such as speaker IDs [23], VAE latent variables [24], pre-trained speaker verification model [32], and a global style token (GST) [25]. In this work, we use a speaker ID embedding.

5.2.2 Phone-Informed Mel-to-Mel Network

In the standard TTS task, the role of the vocoder is to generate a waveform that people can hear and evaluate. On the other hand, in the data augmentation task, the vocoder aims to fill the discrepancy between the lmfb feature settings of ASR and TTS without changing the input of each model. Moreover, the vocoder can alleviate the quality gap between the real and synthesized lmfb features. We observe that synthesized lmfb feature become clear after applying the vocoder. However, the vocoder model takes a long time for inference. Moreover, the text-to-mel and vocoder models must be applied step by step. We also need to convert the waveform to a lmfb feature again.

In this work, we propose a phone-informed post-processing network instead of the vocoder, whose data generation time is much smaller than the vocoder. Specifically, we compose a mel-to-mel network to directly refine the synthesized lmfb feature and fill the gap from the real speech. Refining the speech on the lmfb features domain takes less time than that on the waveform domain. For general speech enhancement, masking is widely applied to noisy (not Mel) spectrograms [98], but it cannot use text information because it is not usually available. However, it is well known that enhancement will be improved given phone information of the speech [99, 100], which is available in TTS and data augmentation tasks. Thus, we use phone embedding information.

Fig. 5.1 shows an architecture of the proposed phone-informed post-processing network. We train the FastSpeech 2-based model at the first stage. After training it, we do not update its parameters. Next, we compose a Transformer-based network that consumes the generated lmfb feature and the output of the variance adaptor which corresponds to phone embedding information. The generated lmfb feature and the output of the variance adaptor are taken from the FastSpeech 2-based model. The residual block is adopted in the proposed method as in the postnet [9] in FastSpeech 2. We use an L1 loss between the predicted and the ground-truth lmfb feature for the objective of training the proposed network. Although the FastSpeech 2-based model is trained on the same criteria, it must learn a complex mapping from a text to the lmfb feature together with the duration, pitch, and energy. On the other hand, the proposed post-processing network is expected to minimize the L1 loss more efficiently since it is given an approximate spectrogram. Moreover, we also feed phone information, which is readily available, unlike general speech enhancement. However, the length of the text is much shorter than that of the lmfb feature. In this work, we use the output of the variance adaptor in the FastSpeech 2 model, which predicts the duration of each phone and extends the outputs of the encoder to the duration. The output length of the variance adaptor is the same as the predicted lmfb feature length.

When we use the vocoder network, we can change the setting of the synthesized lmfb feature to that used in the ASR via a waveform. On the other hand, the proposed method needs to use the same setting such as the FFT size, the frame length and shift[1]. However, recent ASR networks such as Transformer use some CNN sub-sampling layers, and thus the difference of the settings in TTS and ASR can be filled.

[1]We must match only the number of frequency bins. In this work, we used 80-dimensional frequency bins in both ASR and TTS tasks.

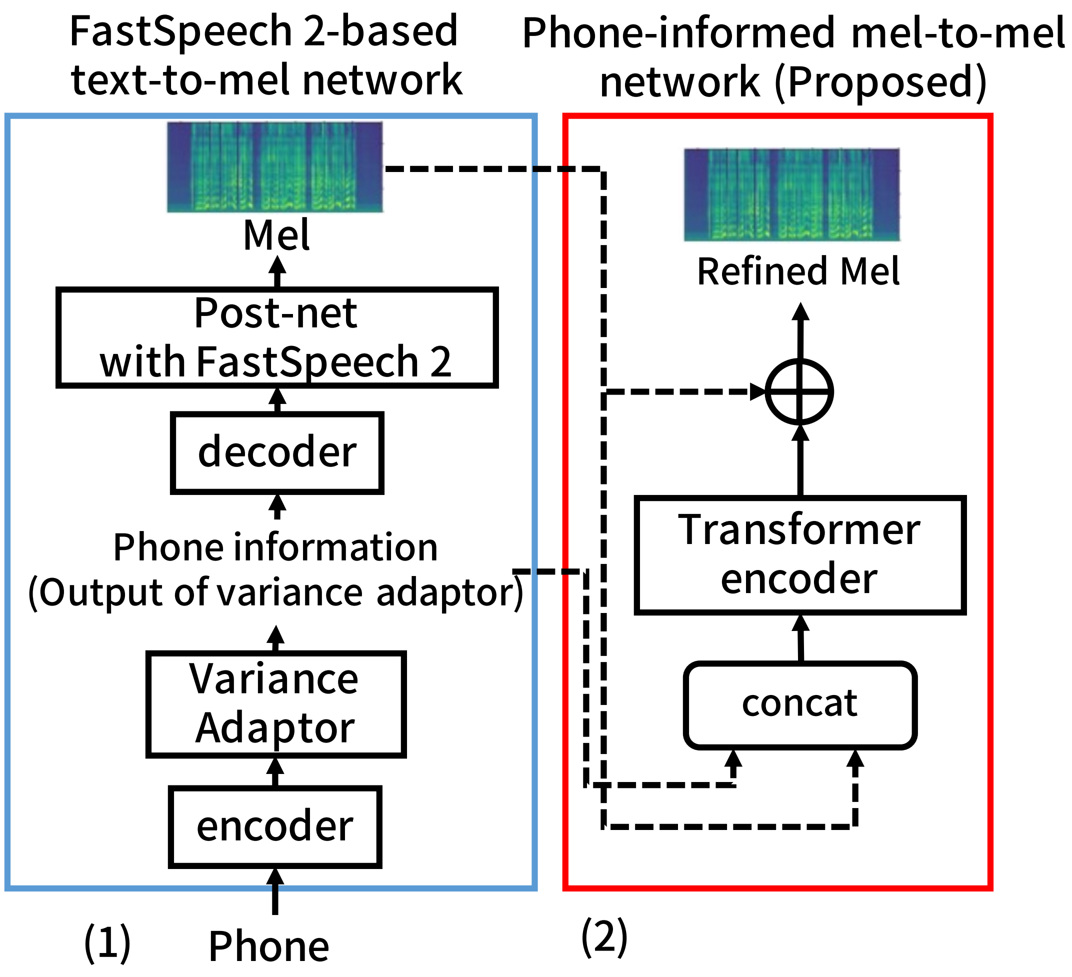


Figure 5.1: The architecture of the proposed phone-informed post-processing network. (1) FastSpeech 2-based model. (2) Proposed post-processing network using the synthesized lmfb feature and phone information (the output of the variance adaptor).

5.3 Experiment Evaluations

5.3.1 Datasets and Tasks

We conducted two domain adaptation experiments, one in English and the other in Japanese. In training the TTS and ASR models in English, we used LibriTTS [94] and LibriSpeech corpus [93]. We downsampled waveforms of LibriTTS to match the sampling rate to 16kHz in all datasets. In LibriTTS and LibriSpeech, we used the train-clean-100 subset. The train-clean-100 of LibriSpeech contains 100 hours of speech data. The train-clean-100 of LibriTTS contains 53.8 hours of speech data including 247 speakers (Female: 123, Male: 124)

For the TTS model, a word sequence was converted into 85-class phones by an open-source grapheme-to-phone tool[2]. To obtain the alignment for training the variance adaptor, we also trained a CTC-based ASR model with the train-clean-100 and conducted forced alignment. A pitch (F0) was predicted by WORLD [45].

For ASR tasks, a word sequence was converted into 10k-class byte-pair-encoding (BPE) units. In this experiment, we suppose that speech generation is used for domain adaptation from read speech (LibriSpeech) to spontaneous speech. For the target domain, we used TED-LIUM release-2 corpus [95] of 91,967 utterances (211 hours). We used only transcription for generating speech. The generated speech was mixed with the real speech of LibriSpeech when training the ASR model. For language model integration, we used official TED-LIUM 2 text data.

In the task in Japanese, we used the CSJ [11], which has two different domain subsets named SPS (Simulated Public Speaking) and APS (Academic Presentation Speech). While SPS is speech on everyday topics, APS is live recordings of academic presentations. SPS has 324.1 hours of speech including 1704 speakers. We trained the TTS and ASR models using the real speech of SPS[3]. For the TTS model, a word sequence was converted into 33-class phones. In the ASR task, we used 10k-class BPE units. We tried to adapt the ASR model

[2]https://github.com/Kyubyong/g2p

[3]We used about 160 hours in training the TTS model since 324.1-hour speech is too large for training the TTS model.

to the APS subset using the transcription of 151,627 utterances (299.5 hours) as the target domain. For evaluation, we used eval1, which is APS domain speech.

5.3.2 FastSpeech 2-Based TTS and Proposed Network

We used a FastSpeech 2-based model as the text-to-mel model. The encoder consisted of a 6-layer Transformer block with 384 model dimensions, 1,536 feed-forward network dimensions, and four attention heads. The variance adaptor consisted of three variance predictors which have two CNN layers with a ReLU activation and layer normalization to predict the duration, pitch, and energy. The 6-layer Transformer with 4-head and 384-dimensional hidden states which consumes the output of the variance adaptor predicted 80-dimensional lmfb features with a shift of 12.5 ms. We added the post-net which had five CNN layers with a kernel size 5. For the multi-speaker TTS model, speaker IDs were fed to the encoder and decoder. In speech generation, we randomly selected one speaker ID per one sentence. We used a linear warmup for the 4k steps. The TTS models were trained with a gradient norm clipping of 1.0, and each batch contains totally 10k frames.

The proposed post-processing network consisted of 6-layer Transformer blocks with 384 model dimensions, 1,536 feed-forward network dimensions, and four attention heads. It consumes the predicted lmfb feature and the output of variance adaptor. It is trained on the same setting as the FastSpeech 2-based model encoder.

For comparison of our proposed method, we used the VocGAN vocoder [56] which converts lmfb features into a waveform. We implemented it based on an open-source code[4] and trained it using LibriTTS. We changed the up-sampling rates of the generator to 5, 5, 2, 2, and 2 to generate a 16kHz sampling waveform.

5.3.3 Transformer-Based ASR System

The ASR model consisted of two CNN subsampling layers (each subsampling factor is 2), 12-layer Conformer-based encoder [40] with 4-head and 256-dimensional

[4]https://github.com/rishikksh20/VocGAN

Table 5.1: Results of TED-LIUM 2 dev and test set (WER [%]) and data generation time of the TTS step.

Method	dev	test	time
Baseline model: Real (train-clean-100)	30.19	27.60	–
Adapted Model: Real (train-clean-100) + TTS (TED-LIUM 2)			
w/o vocoder and post-processing	17.12	17.79	×1
w/ vocoder	16.71	16.76	×2.75
Proposed method	**16.71**	**16.62**	×**1.26**
Proposed method + vocoder	16.87	**16.37**	×3.01
Oracle Model: Real (TED-LIUM 2)	9.28	8.56	–

hidden states, and 1-layer unidirectional LSTM decoder with an attention mechanism which had 256-dimensional hidden states. We used the 80-dimensional lmfb feature with a frame shift of 10ms as the input features for the real speech. In training, we applied a label smoothing [38] with a factor of 0.1, SpecAugment [17], and multi-task learning with the CTC loss. We used a linear warmup for the 25k steps. In the adaptation, we did not try to balance the real and synthetic speech in a batch. In decoding, we set the beam search width to 10. For shallow fusion, we composed a language model with a 4-layer unidirectional LSTM with 512-dimensional hidden states, and the LM weight was set to 0.2.

5.3.4 Results

Table 5.1 shows the word error rates (WERs) for TED-LIUM 2 dev and test set and the data generation time relative to that of the FastSpeech 2 model. The data generation time of the model with the vocoder includes conversion of the generated waveform to the lmfb features. When we did not use any generated speech, WERs were not good because there was a serious domain mismatch between LibriSpeech and TED-LIUM 2. By using the generated speech by the TTS model, we observe 43.3% and 35.5% relative improvement on the dev and test set of TED-LIUM 2 without any post-processing. Applying the vocoder yielded further improvement (44.6% and 39.3% relative improvement). Our proposed post-processing network achieved sightly better performance than the

Table 5.2: Effect of phone information in the proposed method.

	dev	test
w/ phone information (w/ F0 and energy)	**16.54**	**16.62**
w/ phone information (w/o F0 and energy)	16.52	16.88
w/o phone information	16.89	17.16

vocoder (45.2% and 39.8% relative improvement) in a much smaller amount of data generation time. We confirmed that our proposed method enhanced the effect of data generation with a simple framework. When we use both proposed mel-to-mel and vocoder, that is we first refine the lmfb features and convert it to a waveform, the ASR performance is also improved in the testset. However, the improvement is slight, and it is not synergistic since the effects of refinement are overlapped and both models have similar errors.

In Table 5.2, we evaluated the effect of the use of the output of the variance adaptor, which has phone information together with F0 and energy. The model without phone embedding information uses only lmfb features generated by the FastSpeech 2 model. In this case, improvement of the ASR performance is limited and worse than the case using the vocoder. On the other hand, when we remove the additional acoustic prediction (F0 and energy), the result is not changed so much. These results show that the use of the phone embedding information is critical for improving the speech refinement and ASR performance.

Table 5.3 presents an investigation which frequency bins we should refine. In this experiment, we refined the lmfb features of the specified bins. Enhancing all bins ("1-80") achieved the best performance. Partial refinement improved the performance, but the improvement was limited. Fig. 5.2 shows the L1 loss of the FastSpeech 2 and proposed model. It indicates that the loss at high frequency bins are larger than that at low frequency bins because the low frequency bins have a constant high energy, which is more easily learned. The generated lmfb feature at high frequency bins still has a large gap with the real lmfb feature and filling the gap improves the ASR performance. We also confirmed the loss of the proposed model is lower than that of the FastSpeech 2 model in all frequency bins. This suggests the proposed model improves lmfb feature effectively.

Table 5.3: Comparison of frequency bins (80-dim, 0-8kHz) selectively refined in the proposed method. All models used the phone information. Low bin corresponds to low frequency.

Method	dev	test
1-20	17.20	16.65
21-80	17.21	16.85
1-80	**16.54**	**16.62**

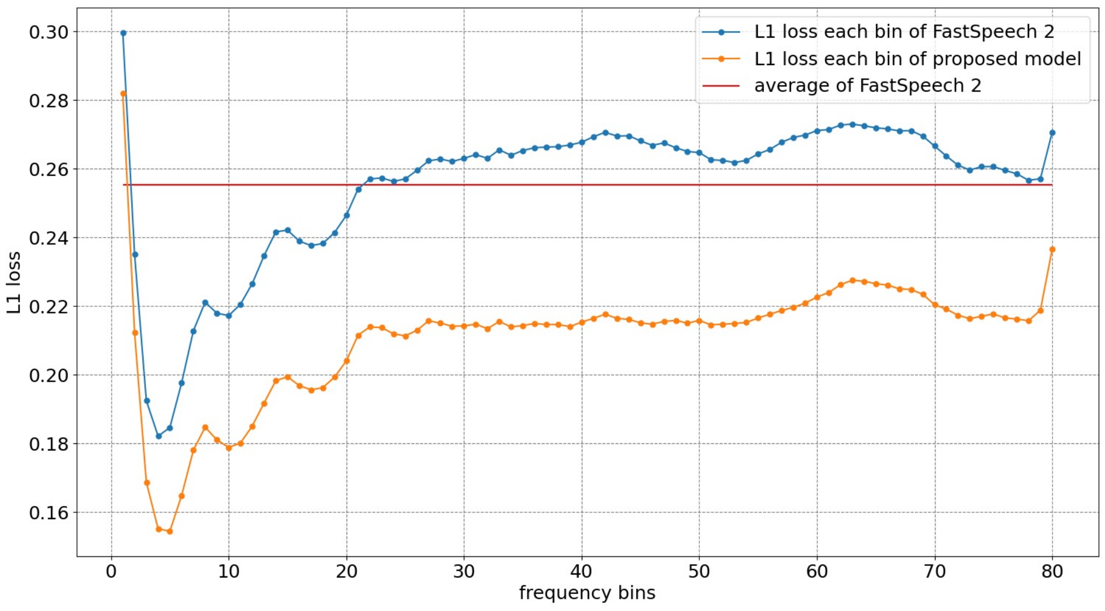


Figure 5.2: Values of L1 losses of the FastSpeech 2 (blue), proposed model (yellow), and the average of frequency bins in the FastSpeech 2 (red). The loss was calculated for each frequency bin from random 1,000 dev-clean samples.

In the proposed method, we used a residual block and did not predict lmfb feature directly. We used a replacement block as an alternative. The network with a replacement block directly predicts lmfb feature to be replaced. Table 5.4 shows that the residual block model achieves higher performance than the replacement block.

Table 5.5 shows the results of domain adaptation in the Japanese data sets. The model without any processing achieves 40.4% relative improvement from the baseline model. When we compare the augmented and oracle models, the absolute WERs difference is lower than 2 points. This is because the speaking style

Table 5.4: Comparison of the residual and replacement blocks in the proposed method. These networks used the phone information and refined all bins.

Method	dev	test
replacement block	16.92	16.80
residual block	**16.54**	**16.62**

Table 5.5: Results of CSJ test set (WER [%]) and data generation time.

Method	eval1	time
Baseline model: Real (SPS)	17.09	–
Adapted Model: Real (SPS) + TTS (APS)		
w/o vocoder and post-processing	10.19	$\times 1$
w/ vocoder	10.09	$\times 2.03$
Proposed method	**9.74**	$\times$**1.26**
Oracle Model: Real (SPS+APS)	8.37	–

of SPS is spontaneous and similar to that of APS. The proposed model realizes a large improvement in much less data generation time. In the CSJ experiments, the data generation time of the vocoder is shorter than in TED-LIUM 2 cases since the duration of TED-LIUM 2 speech is longer than that of CSJ (the average duration of TED-LIUM 2 synthesized speech is 7.3s and that of CSJ is 5.3s). We confirm the proposed network refines the synthesized speech effectively in different kinds of data sets.

5.4 Discussions

Fig. 5.3 and 5.4 show examples of generated lmfb features. They are based on lmfb features which the FastSpeech 2 model generated (top). In the middle, we used the vocoder network to convert the lmfb feature into the waveform and made the lmfb feature again. In the bottom, we used the proposed mel-to-mel network and refined the lmfb feature. We see that an abrupt transition (around $100 \sim 150$ timesteps) of F0 is improved by both the vocoder and proposed network in Fig. 5.3. The FastSpeech 2 model sometimes generates an unnatural transition since it predicts the lmfb features in parallel. While the vocoder network sometimes decreases the energy of lmfb features at high-frequency bins (around 100 $\sim$ 150 timesteps), the proposed model keeps the structure of the lmfb feature. In Fig. 5.4, we see that the output of FastSpeech 2 around 100 timesteps is collapsed. This collapse is not improved when using the vocoder. It is enhanced by our proposed model. Our proposed method refines the lmfb features better since the refinement task given a generated lmfb feature is easier than the waveform prediction task. It results in ASR performance improvement.

5.5 Summary

In this chapter, we have proposed the phone-informed post-processing network for data augmentation for the ASR model using the TTS model without the vocoder network. Unlike the vocoder network, we directly refine the generated lmfb feature derived from the text-to-mel network (FastSpeech 2-based model). The proposed network uses not only the predicted lmfb feature but also the output of the variance predictor which corresponds to the phone information. In the experimental evaluations, the proposed method resulted in a large improvement from the baseline and better performance than the vocoder in a much smaller amount of data generation time. We also showed that the use of the phone information is critical for improving the performance.

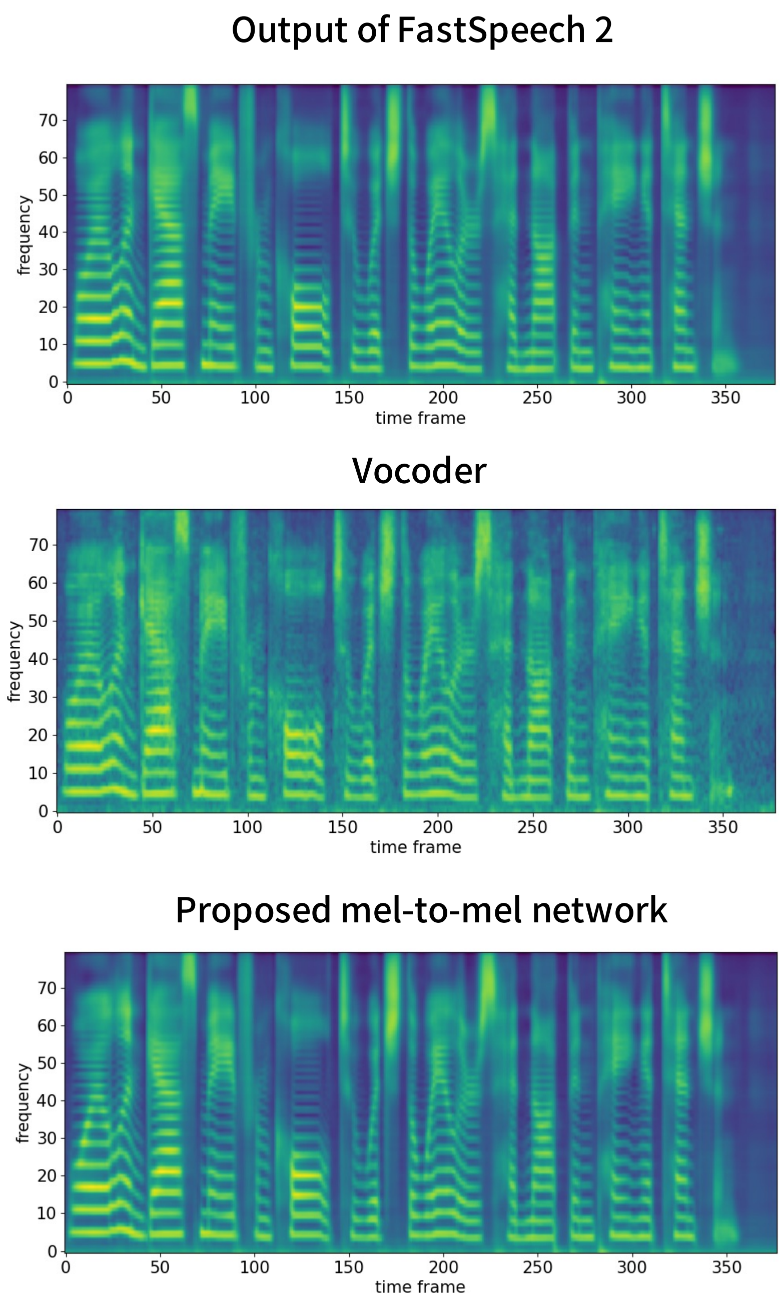


Figure 5.3: Example of generated lmfb features. The output of the FastSpeech 2 (top) is generated given the text "oil and gas made from coal to which the whalers had not been paying attention." In the vocoder (middle), we converted the top lmfb features into the waveform and again made the lmfb feature. In the proposed mel-to-mel network, we refined the lmfb feature given the top lmfb feature (bottom).

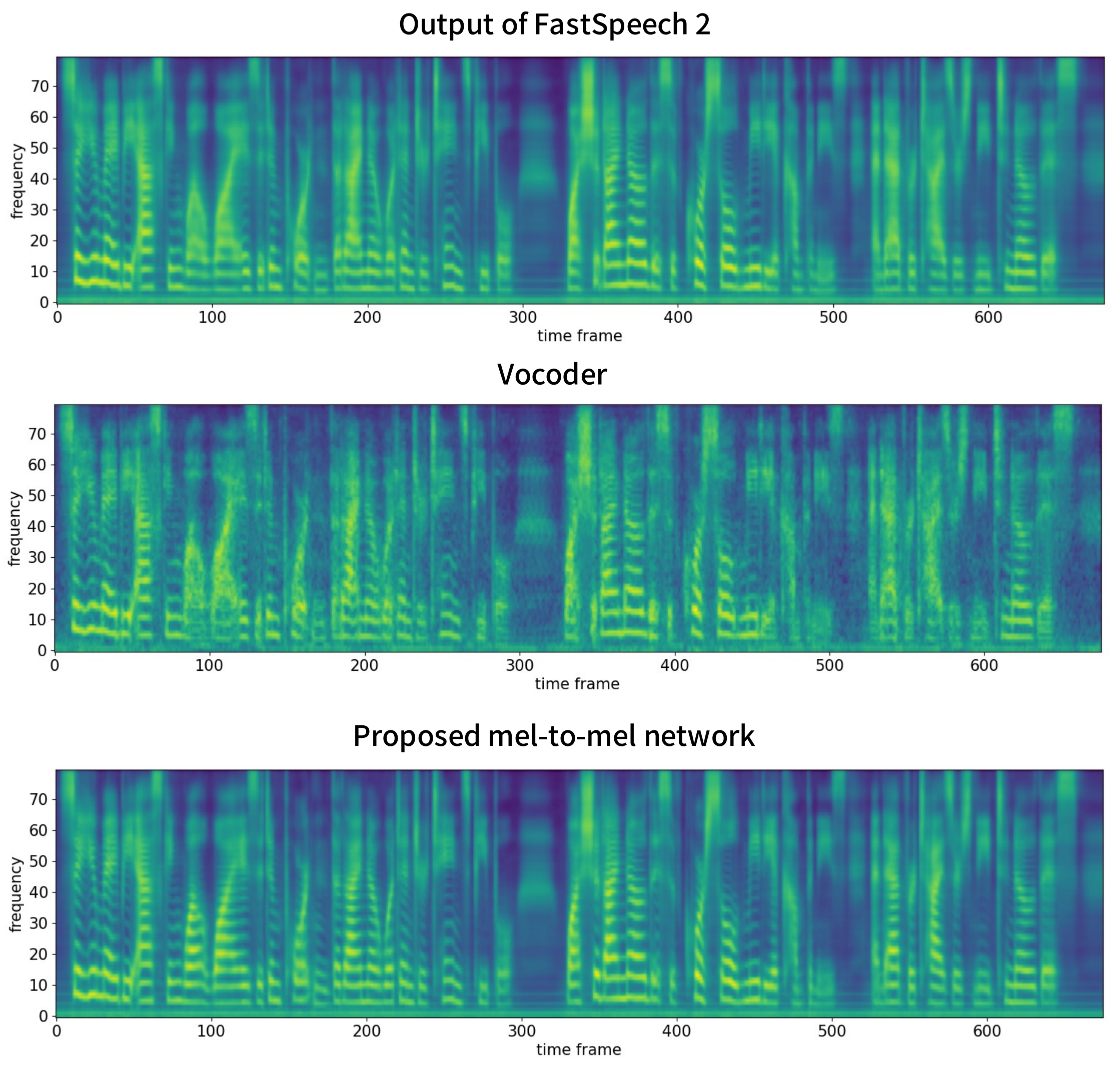


Figure 5.4: Example of the output of the FastSpeech 2 (top), the vocoder (middle), and the proposed mel-to-mel network (bottom). The input text is "so you may be asking well why is it important that I know what entertains people why should I know this of course old media companies and advertisers need to know this."

Chapter 6

Conclusions

This chapter reviews the contributions of this thesis, the comparison of three methods, and the future directions.

6.1 Contributions

The end-to-end ASR models achieve high performance when a large amount of training data is available. However, the training of end-to-end ASR requires paired data of speech and transcription, and it costs a lot of manual works and time to prepare them. We focus on the data augmentation methods that use the TTS system, and have investigated three approaches.

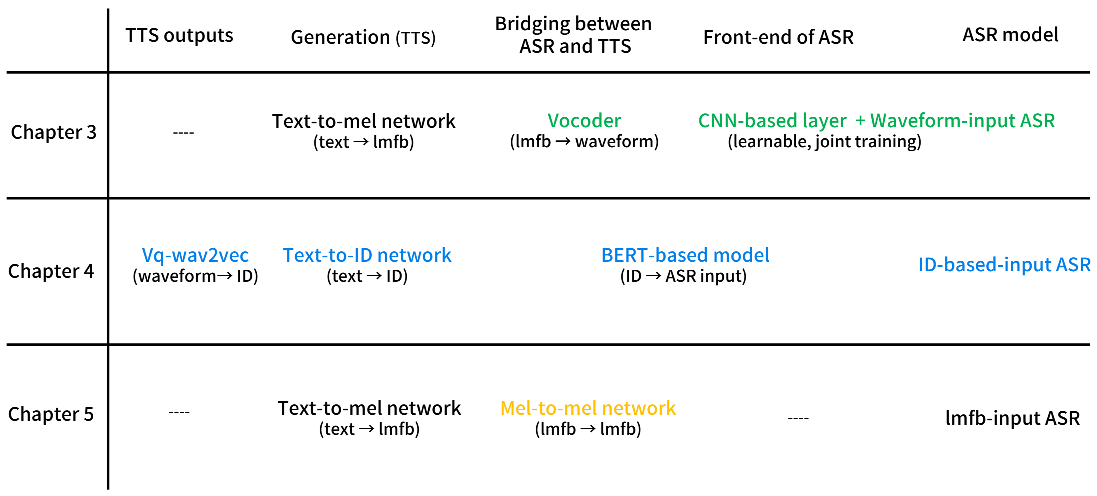
In Chapter 3, we presented waveform-based data augmentation on both ASR and TTS. The proposed approach was based on a waveform-output TTS model and a waveform-input ASR. In a waveform-output TTS, we used not only a text-to-mel network but also a vocoder network. In the waveform-input ASR, we introduce CNN filters and a masking method similar to SpecAugment. We compared the waveform-input ASR with two kinds of lmfb-input models: (1) lmfb features are directly generated by TTS, and (2) lmfb features are converted from the waveform generated by TTS. Experimental evaluations show that the combination of waveform-output TTS and the waveform-input end-to-end ASR model outperforms the lmfb-input ASR models in two domain adaptation settings.

In Chapter 4, we presented a data augmentation method via a discrete speech representation. Thanks to the development of an unsupervised approach, particularly a self-supervised approach, we can use another acoustic feature instead of lmfb features. In this work, we used the discrete ID produced by vq-wav2vec as an intermediate representation instead of lmfb features. The TTS model predicts a discrete ID sequence that is easier to train than a continuous value of lmfb features. Using a discrete representation based on vq-wav2vec not only makes TTS training easier but also mitigates the mismatch with real data. The ASR model also consumes an ID-based feature. Experimental evaluations show that the proposed method outperforms the data augmentation method using the conventional TTS. We found that it reduces speaker dependency, and the generated features are distributed more closely to the real ones.

In Chapter 5, we presented a post-processing network that refines lmfb features, where the model directly predicts lmfb features given the lmfb features generated by the text-to-mel network. In Chapter 3, we used a vocoder network to convert the generated feature into a waveform. It improved the ASR performance, but requires a huge amount of runtime since a waveform has much longer than the lmfb feature. Moreover, converting into a waveform is not necessary for data augmentation. Therefore we proposed a mel-to-mel network to refine the lmfb features directly. The direct refinement takes less runtime than the vocoder. Experimental evaluations demonstrated that the proposed network achieves better word error rates than the vocoder network in an English domain adaptation task (read speech to spontaneous speech) in a much smaller amount of data generation time. It was also shown that phone information is critical for the improvement. We also confirmed the effect of the proposed model in a Japanese domain adaptation task.

6.2 Comparison of Approaches

In this thesis, we proposed three data augmentation approaches for ASR using TTS. We compare the architecture of each approach and their advantages and disadvantages in this section.



	TTS outputs	Generation (TTS)	Bridging between ASR and TTS	Front-end of ASR	ASR model
Chapter 3	----	Text-to-mel network (text → lmfb)	Vocoder (lmfb → waveform)	CNN-based layer (learnable, joint training)	+ Waveform-input ASR
Chapter 4	Vq-wav2vec (waveform→ ID)	Text-to-ID network (text → ID)	BERT-based model (ID → ASR input)		ID-based-input ASR
Chapter 5	----	Text-to-mel network (text → lmfb)	Mel-to-mel network (lmfb → lmfb)	----	lmfb-input ASR

Figure 6.1: The role of each network and comparison of each chapter. "—-" means we do not use any learnable network.

6.2.1 Architecture

Fig. 6.1 shows the comparison of the network architecture in each approach. In Chapter 3, we must prepare three models: a waveform-input ASR model, a text-to-mel network, and a vocoder network. The waveform-input ASR model includes learnable front-end CNN layers for a waveform and an encoder decoder architecture that predicts a target linguistic label sequence given the output of the front-end. We update the parameters of the two modules using the same criterion (likelihood between predicted and ground-truth labels). We do not change the architecture of a text-to-mel network and a vocoder network.

In Chapter 4, we composed four models: a vq-wav2vec network, a BERT-based model, a text-to-ID network, and an ID-based-input ASR model. The vq-wav2vec network consumes a waveform to convert a discrete representation. It is regarded as a front-end. Unlike Chapter 3, we separately train the vq-wav2vec model and the ASR model. We also use the BERT-based model, which acquires contextualized information for the ASR given an ID sequence. In the data augmentation, the role of the BERT-based model is close to a vocoder. In addition to vq-wav2vec, we change the output of the generation network. While the original text-to-mel network predicts an lmfb feature sequence (continuous values), the text-to-ID network predicts an ID sequence. We change the MAE criterion into likelihood between a ground-truth ID sequence and a predicted ID sequence. The input of the ASR model is the hidden state of the BERT-based model.

In chapter 5, we composed three modules; an ASR model, a text-to-mel network, and a mel-to-mel network. The mel-to-mel network refines the generated lmfb features. We do not change the architecture of the ASR model and the text-to-mel network and do not use a vocoder network.

6.2.2 Advantages and Disadvantages

Chapter 3 work has three advantages.

- The vocoder network improves the TTS quality, and we fully utilize the generated waveform to add the learnable front-end.
- We do not need to consider the differences of lmfb features between ASR and TTS since raw waveforms can be converted into lmfb features with any setting.
- We optimize the front-end by the likelihood for the linguistic label, which is the goal of the ASR task, while the lmfb features are not designed for minimizing the ASR loss.

However, there are two disadvantages compared with other works.

- The waveform-input ASR has a large amount of parameters and the ASR model needs a lot of time to train and recognize.
- In this method, we need to compose a vocoder network and convert an lmfb feature into a waveform. Generating a waveform takes a long time since a waveform has a longer sequence than the lmfb features.

Chapter 4 work has three advantages.

- While waveform-input ASR models need the paired data to train the front-end, the self-supervised model can utilize waveform-only data to train the self-supervised model (vq-wav2vec).

- The TTS task is much easier since the target of TTS is changed to a discrete representation from a continuous value.
- The generated speech exists in the real speech because the text-to-ID model predicts IDs which is derived from the real speech.

There are three disadvantages.

- We need to compose four models separately
- Vq-wav2vec and BERT-based model have a large amount of parameters, and it takes a long time for training and generation.
- Self-supervised model needs a lot of speech data to achieve high performance of the ASR models.

Chapter 5 work has two advantages.

- The generation is faster than a vocoder since we do not need to predict a waveform that has a long-time sequence.
- We do not have to change the basic architecture of both the ASR and TTS models.

Chapter 5 work has two disadvantages.

- The mel-to-mel model cannot fill the differences between ASR and TTS lmfb settings.
- It is possible that the generated lmfb features do not exist in the real lmfb features.

6.3 Investigation on Mismatch of TTS

When we compare Table 3.2 in Chapter 3 and Table 5.5 in Chapter 5, the performance gap between the augmented model and oracle model still exists because of the TTS quality. We used Tacotron 2 in Chapter 3 and FastSpeech 2 in Chapter 5 for a text-to-mel model. Training the text-to-mel model using spontaneous speech is especially difficult for the autoregressive model (Tacotron 2) since it generates speech using previous timesteps and needs to train a wide variety of speech. Moreover, we used WaveNet in Chapter 3 and VocGAN in Chapter 5 as a vocoder. We observe that generated waveform using WaveNet contains a lot of unnatural noises.

The performance gap between the augmented model and oracle model is also different for English and Japanese (e.g., Table 5.1 and Table 5.5). While we used readout speech (LibriTTS) in English to train the TTS model, we used spontaneous speech (CSJ-SPS) for Japanese. The speaking style of CSJ-SPS is close to that of the target domain (CSJ-APS). These results suggest that the mismatch between the real and generated speech depends on the TTS quality and speaking style. In this section, we investigate the effects of TTS for ASR performance using LibriTTS and TED-LIUM 2. The architecture of TTS model and ASR model is the same in Chapter 5. While LibriTTS provides read speeches, the target domain (TED-LIUM 2) is spontaneous lecture speech. The generated speech is basically read-style speech and there are differences in the speaking style. Moreover, converting an lmfb feature into a waveform has some errors when we use a vocoder. In Table 6.1, we compare the effect of the TTS model settings, training data (speaking style) and vocoder. In "LibriSpeech + TTS (trained by LibriTTS)", we train the FastSpeech 2 model using both speech and transcription of LibriTTS. In "LibriSpeech + TTS (trained by LibriTTS) w/ vocoder", we also train a vocoder using LibriTTS in addition to the FastSpeech 2 model and convert an lmfb feature into a waveform.. In "LibriSpeech + TTS (trained by TED-LIUM 2)", we train the FastSpeech 2 model using both speech and transcription of TED-LIUM 2. In "LibriSpeech + TTS (trained by TED-LIUM 2) w/ vocoder", we also train a vocoder using TED-LIUM 2 in addition to the FastSpeech 2 model and convert an lmfb feature into a waveform. In "LibriSpeech + TED-LIUM 2 (Real speech) w/ vocoder", we use real speeches but just lmfb features and convert them into a waveform with a vocoder. We then make lmfb features from a converted waveform. It aims to confirm the effect of speaking style, and the quality of mel-to-mel networks, and the quality of vocoder networks. We see that the TTS

Table 6.1: Comparison of dataset for TTS training. We evaluate the WER [%] on TED-LIUM 2 test set. In "LibriSpeech + TTS (trained by TED-LIUM 2)", we train the FastSpeech 2 model using both speech and transcription of TED-LIUM 2. In "LibriSpeech + TTS (trained by TED-LIUM 2) w/ vocoder", we use a vocoder in addition to the FastSpeech 2 model. In "LibriSpeech + TED-LIUM 2 (Real speech) w/ vocoder", we use real speeches but just lmfb features that are converted into a waveform.

	dev	test
LibriSpeech (train-clean-100)	31.95	28.92
LibriSpeech + TTS (trained by LibriTTS)	17.12	17.79
LibriSpeech + TTS (trained by LibriTTS) w/ vocoder	16.71	16.76
LibriSpeech + TTS (trained by **TED-LIUM 2**)	15.06	14.02
LibriSpeech + TTS (trained by **TED-LIUM 2**) w/ vocoder	14.23	13.81
LibriSpeech + TED-LIUM 2 (real speech) w/ vocoder	10.90	10.77
LibriSpeech + TED-LIUM 2 [oracle]	9.28	8.56

model using TED-LIUM 2 dataset yields 2.95 points improvement compared with that using the LibriTTS dataset. This improvement can be attributed to the fact that the speaking style is matched to the target domain. However, the improvement is limited and there is 5.25-point difference from the oracle model in the test set. This result shows that the discrepancy between real and generated speech affects the ASR performance much. When we converted the real lmfb features into waveforms using a vocoder and trained the ASR system based on the converted waveforms, the performance is slightly degraded from the oracle model. This result indicates that the vocoder network also causes the discrepancy between the real and generated features, but the difference is small (2.21 points) compared to the oracle model.

When we compare Table 4.2 in Chapter 4 and Table 5.1 in Chapter 5, we also observe that the effects of the data augmentation are different. We used different encoders of ASR model and Conformer encoder achieves higher performance than the LSTM encoder. We consider one reason of the difference is SpecAugment. In Chapter 5, we applied a larger amount of masks than in Chapter 4. The masking can alleviate the mismatch of the generated speech.

6.4 Future Work

This section describes several open problems regarding the methods developed in this thesis and future research directions.

In all the proposed methods, we comprise the ASR and TTS systems separately, and the TTS system does not consider the ASR performance. One direction is to integrate the TTS architecture with the ASR model and optimize them jointly. However, there is the possibility that the joint training loss causes over-fitting of the ASR model or the TTS model. Instead of the shared criterion, we will introduce GAN-based architecture to improve the TTS quality and utilize the synthesized speech effectively.

Basically, we assume that the high quality of generated speech is good for speech recognition. However, noise injection such as SpecAugment and changing tempo such as speed perturbation methods positively impact speech recognition. These speeches are unreal, but the ASR performance is improved. Moreover, a recent scalable end-to-end speech synthesis model has been investigated to control the synthesized voice. For example, a FastSpeech 2 model can change the pitch of generated speech when we change the pitch prediction. These changed speeches are less natural, but we can generate various speech. It is thus necessary to investigate the use of generated speech for improving the ASR models. Moreover, it is worth investigating the relationship between unreal speech and ASR performance.

In this thesis, we mainly investigate approaches to improve the generated speech quality. One problem of all the works is that the generated speech does not have diversity compared with the real speech. A multi-speaker TTS model alleviates the problem, but we need to investigate a TTS system that can generate variant speech. We will leverage waveform-only data to generate more variety of speech to train the TTS model. Another way may be to use a voice conversion model to acquire more diverse speech from the generated speech.

In this thesis, we do not investigate the balance of synthesized speech and real speech or a method to select the text for effective training. It is better to train the ASR model effectively using a small amount of generated data. It is worth investigating the methods to consider the ratio of synthesized speech and real speech. We will also investigate the effect of the data augmentation when we have a small amount of in-domain data.

Bibliography

[1] T. Sakai and S. Doshita, "The Phonetic Typewriter: Its Fundamentals and Mechanism," in *IFIP Congress 62,*, pp. 445–450, 1962.

[2] K.-F. Lee, *Automatic speech recognition: the development of the SPHINX system*, vol. 62. Springer Science & Business Media, 1988.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] A. Graves and N. Jaitly, "Towards End-To-End speech recognition with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, pp. 1764–1772, 2014.

[5] J. Olive, "Rule synthesis of speech from dyadic units," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, pp. 568–570, 1977.

[6] É. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," in *Speech Communication*, vol. 9, pp. 453–467, 1990.

[7] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *European Conference on Speech Communication and Technology (Eurospeech)*, pp. 2347–2350, 1999.

[8] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," *IEEE International Conference on Acoustics, Speech, and Signal Processing*

(ICASSP), vol. 3, pp. 1315–1318, 2000.

[9] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, *et al.*, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *INTERSPEECH*, pp. 4779–4783, 2017.

[10] K. Itou, M. Yamamoto, K. Takeda, T. Takezawa, T. Matsuoka, T. Kobayashi, K. Shikano, and S. Itahashi, "Jnas: Japanese speech corpus for large vocabulary continuous speech recognition research," *Journal of the Acoustical Society of Japan (E)*, vol. 20, no. 3, pp. 199–206, 1999.

[11] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous Speech Corpus of Japanese," in *LREC*, pp. 947–9520, 2000.

[12] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[13] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *International Conference on Learning Representations (ICLR)*, 2020.

[14] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 12449–12460, Curran Associates, Inc., 2020.

[15] G. Zavaliagkos and T. Colthurst, "Utilizing untranscribed training data to improve performance," in *DARPA Broadcast News Transcription and Understanding Workshop, Landsdowne*, pp. 301–305, 1998.

[16] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, B. Li, Y. Wu, and Q. V. Le, "Improved Noisy Student Training for Automatic Speech Recognition," in *Proc. Interspeech 2020*, pp. 2817–2821, 2020.

[17] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *INTERSPEECH*, pp. 2613–2617, 2019.

[18] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar,

"An analysis of incorporating an external language model into a sequence-to-sequence model," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5824–5828, IEEE, 2018.

[19] J. Shin, Y. Lee, and K. Jung, "Effective sentence scoring method using bert for speech recognition," in *Proceedings of The Eleventh Asian Conference on Machine Learning* (W. S. Lee and T. Suzuki, eds.), vol. 101 of *Proceedings of Machine Learning Research*, pp. 1081–1093, 2019.

[20] A. Tjandra, S. Sakti, and S. Nakamura, "Attention-based wav2text with feature transfer learning," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 309–315, 2017.

[21] A. Tjandra, S. Sakti, and S. Nakamura, "Listening while speaking: Speech chain by deep learning," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 301–308, 2017.

[22] M. Mimura, S. Ueno, H. Inaguma, S. Sakai, and T. Kawahara, "Leveraging sequence-to-sequence speech synthesis for enhancing acoustic-to-word speech recognition," in *Workshop on Spoken Language Technology (SLT)*, pp. 477–484, 2018.

[23] S. Ueno, M. Mimura, S. Sakai, and T. Kawahara, "Multi-speaker sequence-to-sequence speech synthesis for data augmentation in acoustic-to-word speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6161–6165, 2019.

[24] A. Rosenberg, Y. Zhang, B. Ramabhadran, Y. Jia, P. Moreno, Y. Wu, and Z. Wu, "Speech recognition with augmented synthesized speech," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 996–1002, 2019.

[25] N. Rossenbach, A. Zeyer, R. Schluter, and H. Ney, "Generating Synthetic Audio Data for Attention-based Speech Recognition Systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7064–7068, 2020.

[26] Z. Chen, A. Rosenberg, Y. Zhang, G. Wang, B. Ramabhadran, and P. J. Moreno, "Improving Speech Recognition Using GAN-Based Speech Synthe-

sis and Contrastive Unspoken Text Selection," in *INTERSPEECH*, pp. 556–560, 2020.

[27] G. Wang, A. Rosenberg, Z. Chen, Y. Zhang, B. Ramabhadran, Y. Wu, and P. Moreno, "Improving speech recognition using consistent predictions on synthesized speech," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7029–7033, 2020.

[28] G. Wang, A. Rosenberg, Z. Chen, Y. Zhang, B. Ramabhadran, and P. J. Moreno, "SCADA: Stochastic, Consistent and Adversarial Data Augmentation to Improve ASR," in *INTERSPEECH*, pp. 2832–2836, 2020.

[29] A. Laptev, R. Korostik, A. Svischev, A. Andrusenko, I. Medennikov, and S. Rybin, "You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation," *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2020.

[30] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5659–5663, 2021.

[31] G. Kurata, G. Saon, B. Kingsbury, D. Haws, and Z. Tüske, "Improving Customization of Neural Transducers by Mitigating Acoustic Mismatch of Synthesized Audio," in *INTERSPEECH*, pp. 2027–2031, 2021.

[32] A. Fazel, W. Yang, Y. Liu, R. Barra-Chicote, Y. Meng, R. Maas, and J. Droppo, "SynthASR: Unlocking Synthetic Data for Speech Recognition," in *INTERSPEECH*, pp. 896–900, 2021.

[33] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *INTERSPEECH*, 2012.

[34] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[35] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), (Berlin, Germany), pp. 1715–1725, Association for Computational Linguistics, Aug. 2016.

[36] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *International Conference on Machine Learning (ICML)*, pp. 369–376, 2006.

[37] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4835–4839, 2017.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

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

[40] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *INTERSPEECH*, pp. 5036–5040, 2020.

[41] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "FastSpeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations (ICRL)*, 2020.

[42] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[43] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. Weiss, and Y. Wu, "Parallel tacotron: Non-autoregressive and controllable tts," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5694–5698,

2021.

[44] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. Skerry-Ryan, and Y. Wu, "Parallel Tacotron 2: A non-autoregressive neural tts model with differentiable duration modeling," in *INTERSPEECH*, pp. 141–145, 2021.

[45] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. E99.D, no. 7, pp. 1877–1884, 2016.

[46] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016.

[47] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *CoRR*, vol. abs/1802.08435, 2018.

[48] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 3918–3926, PMLR, 10–15 Jul 2018.

[49] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 3617–3621, 2019.

[50] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[51] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in*

Neural Information Processing Systems (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[52] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

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

[54] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "MelGAN: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[55] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6194–6198, 2020.

[56] J. Yang, J. Lee, Y. Kim, H. Cho, and I. Kim, "VocGAN: A high-fidelity real-time vocoder with a hierarchically-nested adversarial network," in *INTERSPEECH*, pp. 200–204, 2020.

[57] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[58] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," in *INTERSPEECH*, pp. 387–391, 2018.

[59] R. Masumura, N. Makishima, M. Ihori, A. Takashima, T. Tanaka, and

S. Orihashi, "Phoneme-to-Grapheme Conversion Based Large-Scale Pre-Training for End-to-End Automatic Speech Recognition," in *INTERSPEECH*, pp. 2822–2826, 2020.

[60] Y. Tang, J. Pino, C. Wang, X. Ma, and D. Genzel, "A general multi-task learning framework to leverage text data for speech to text tasks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6194–6198, 2021.

[61] S. Karita, S. Watanabe, T. Iwata, A. Ogawa, and M. Delcroix, "Semi-supervised end-to-end speech recognition," in *INTERSPEECH*, pp. 2–6, 2018.

[62] S. Karita, S. Watanabe, T. Iwata, M. Delcroix, A. Ogawa, and T. Nakatani, "Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6166–6170, 2019.

[63] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2962–2970, 2017.

[64] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," in *International Conference on Learning Representations (ICRL)*, 2018.

[65] N. Li, S. Liu, Y. Liu, S. Zhao, M. Liu, and M. Zhou, "Close to human quality TTS with transformer," *arXiv preprint, 1809.08895*, 2018.

[66] N. Rossenbach, A. Zeyer, R. Schluter, and H. Ney, "Generating Synthetic Audio Data for Attention-based Speech Recognition Systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7064–7068, 2020.

[67] Y. Huang, L. He, W. Wei, W. Gale, J. Li, and Y. Gong, "Using Personalized Speech Synthesis and Neural Language Generator for Rapid Speaker Adaptation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7394–7398, 2020.

[68] N. Jaitly and G. Hinton, "Learning a Better Representation of Speech Soundwaves using Restricted Boltzmann Machines," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5884–5887, 2011.

[69] D. Palaz, R. Collobert, and M. Magimai-Doss, "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks," in *INTERSPEECH*, pp. 1766–1770, 2013.

[70] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Acoustic modeling with deep neural networks using raw time signal for LVCSR," in *INTERSOEECH*, pp. 890–894, 2014.

[71] D. Palaz, M. Magimai-Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4295–4299, 2015.

[72] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *INTERSPEECH*, pp. 1–5, 2015.

[73] M. Ravanelli and Y. Bengio, "Speech and speaker recognition from raw waveform with SincNet," *arXiv preprint arXiv:1812.05920*, 2018.

[74] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028, IEEE, 2018.

[75] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully Convolutional Speech Recognition," in *arXiv preprint arXiv:1812.06864*, 2018.

[76] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[77] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition," in *INTERSPEECH*, pp. 1468–1472, 2015.

[78] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1026–1034, 2015.

[79] R. Sonobe, S. Takamichi, and H. Saruwatari, "JSUT corpus: free large-scale Japanese speech corpus for end-to-end speech synthesis," *arXiv preprint, 1711.00354*, 2017.

[80] S. Ueno, M. Mimura, and T. Kawahara, "End-to-end speech synthesis for multiple speakers using the corpus of spontaneous japanese," in *Acoustical Society of Japan (ASJ)*, pp. 1085–1086, 2018.

[81] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint, 1412.6980*, pp. 1–15, 2014.

[82] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 577–585, 2015.

[83] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016.

[84] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Advances in Neural Information Processing Systems (NIPS) Workshop on Deep Learning*, 2017.

[85] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, June 2018.

[86] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[87] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[88] O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord, "Data-efficient image recognition with contrastive predictive coding," *CoRR*, vol. abs/1905.09272, 2019.

[89] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[90] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An Unsupervised Autoregressive Model for Speech Representation Learning," in *Proc. INTERSPEECH*, pp. 146–150, 2019.

[91] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[92] A. van den Oord, O. Vinyals, and k. kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[93] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

[94] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A corpus derived from librispeech for text-to-speech," *arXiv preprint arXiv:1904.02882*, 2019.

[95] A. Rousseau, P. Deléglise, and Y. Estève, “Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks,” in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, pp. 3935–3939, 2014.

[96] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, 2016.

[97] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, “On using monolingual corpora in neural machine translation,” *arXiv preprint arXiv:1503.03535*, 2015.

[98] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, “Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 708–712, 2015.

[99] K. Kinoshita, M. Delcroix, A. Ogawa, and T. Nakatani, “Text-informed speech enhancement with deep neural networks,” in *INTERSPEECH*, pp. 1760–1764, 2015.

[100] K. Schulze-Forster, C. S. J. Doire, G. Richard, and R. Badeau, “Joint phoneme alignment and text-informed speech separation on highly corrupted speech,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7274–7278, 2020.

List of Publications

Refereed International Journal Papers

1) Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara: Synthesizing Waveform Sequence-to-Sequence to Augment Training Data for Sequence-to-Sequence Speech Recognition, *Acoustical Science and Technology*, Vol.42, No.6, pp.333–343, 2021.

Refereed International Conference Papers

2) Sei Ueno and Tatsuya Kawahara: Phone-Informed Refinement of Synthesized Mel Spectrogram for Data Augmentation in Speech Recognition, *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), accepted, 2022.
3) Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara: Data Augmentation for ASR Using TTS via a Discrete Representation, *IEEE Automatic Speech Recognition and Understanding Workshop* (ASRU), pp.68–75, 2021.
4) Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara: Multi-Speaker Sequence-to-Sequence Speech Synthesis for Data Augmentation in Acoustic-to-Word Speech Recognition, *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), pp.6161–6165, 2019.
5) Sei Ueno, Takafumi Moriya, Masato Mimura, Shinsuke Sakai, Yoshikazu Yamaguchi, Yushi Aono, and Tatsuya Kawahara:Encoder Transfer for Attention-Based Acoustic-to-Word Speech Recognition, *INTERSPEECH*, pp.2424–2428 2018.

6) Sei Ueno, Hirofumi Inaguma, Masato Mimura, and Tatsuya Kawahara: Acoustic-to-Word Attention-Based Model Complemented with Character-Level CTC-Based Model. *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP),pp.5804-5808, 2018.

Refereed International Conference Papers (co-authored works)

7) Kohei Matsuura, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara:Speech Corpus of Ainu Folklore and End-to-End Speech Recognition for Ainu Language. *International Conference on Language Resources and Evaluation* (LREC), pp.2622-2628, 2020.
8) Viet-Trung Dang, Tianyu Zhao, Sei Ueno, Hirofumi Inaguma, and Tatsuya Kawahara: End-to-End Speech-to-Dialog-Act Recognition. *INTERSPEECH*, pp.3910–3914, 2020.
9) Hayato Futami, Hirofumi Inaguma, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara: Distilling the Knowledge of BERT for Sequence-to-Sequence ASR. *INTERSPEECH*, pp.3635–3639, 2020.
10) Han Feng, Sei Ueno, and Tatsuya Kawahara: End-to-End Speech Emotion Recognition Combined with Acoustic-to-Word ASR Model. *INTERSPEECH*, pp.501—505, 2020.
11) Masato Mimura, Sei Ueno, Hirofumi Inaguma, Shinsuke Sakai, and Tatsuya Kawahara: Leveraging Sequence-to-Sequence Speech Synthesis for Enhancing Acoustic-to-Word Speech Recognition. *IEEE Spoken Language Technology Workshop* (SLT), pp.477-484, 2018.
12) Takafumi Moriya, Sei Ueno, Yusuke Shinohara, Marc Delcroix, Yoshikazu Yamaguchi, and Yushi Aono: Multi-Task Learning with Augmentation Strategy for Acoustic-to-Word Attention-Based Encoder-Decoder Speech Recognition. *INTERSPEECH*, pp.2399-2403, 2018.

Domestic Conferences

13) 上乃 聖, 河原達也: 音声認識のデータ拡張のための音素情報を用いた合成音声の強調, 日本音響学会研究発表会講演論文集, 1-3-4, 春季 2022.
14) 上乃 聖, 河原達也: 音声認識のデータ拡張のための合成音声の周波数スペクトログラム強調, 情報処理学会研究報告, SLP-139-28, 2021.
15) 上乃 聖, 三村正人, 河原達也: 音声合成による wav2vec 2.0 を用いた音声認識のデータ拡張, 日本音響学会研究発表会講演論文集, 1-3-5, 秋季 2021.
16) 上乃 聖, 三村正人, 河原達也: 複数話者を対象とした非自己回帰型ニューラル音声合成, 日本音響学会研究発表会講演論文集, 3-2-25, 春季 2021.
17) 上乃 聖, 三村正人, 河原達也: vq-wav2vec による離散 ID を扱う音声認識のデータ拡張, 日本音響学会研究発表会講演論文集, 1-2-16, 春季 2021.
18) 上乃 聖, 三村正人, 坂井信輔, 河原達也: Wave2word: 音声波形を入力とする単語単位 End-to-End 音声認識, 日本音響学会研究発表会講演論文集, 1-3-6, 秋季 2019.
19) 上乃 聖, 三村正人, 坂井信輔, 河原達也: 多数話者コーパスを用いた End-to-End 音声合成による 単語単位 End-to-End 音声認識のデータ拡張, 日本音響学会研究発表会講演論文集, 2-9-2, 春季 2019.
20) 上乃 聖, 三村正人, 坂井信輔, 河原達也: 音声波形を入力とする単語単位 End-to-End 音声認識, 情報処理学会研究報告, SLP-129-2, 2019.
21) 上乃 聖, 三村正人, 坂井信輔, 河原達也:End-to-End 音声合成を用いた単語単位 End-to-End 音声認識のデータ拡張, 情報処理学会研究報告, SLP-125-2, 2018, 学生論文賞受賞.
22) 上乃 聖, 三村正人, 河原達也: End-to-End 音声合成を用いた単語単位 End-to-End 音声認識の学習データ拡張, 日本音響学会研究発表会講演論文集, 1-2-6, 秋季 2018.
23) 上乃 聖, 三村正人, 河原達也: 『日本語話し言葉コーパス』を用いた多数話者 End-to-End 音声合成, 日本音響学会研究発表会講演論文集, 1-4-2, 秋季 2018.
24) 上乃 聖, 森谷崇史, 三村正人, 坂井信輔, 篠原雄介, 山口義和, 青野裕司, 河原達也: 転移学習による注意機構付き単語単位音声認識の適応, 電子情報通信学 会技術研究報告, SP2018-23, 2018, 学生ポスター賞受賞.

25) 上乃 聖, 森谷崇史, 三村正人, 坂井信輔, 篠原雄介, 山口義和, 青野裕司, 河原達也: 単語単位エンコーダデコーダ音声認識モデルの転移学習を用いた適応, 日本音響学会研究発表会講演論文集, 1-2-5, 秋季 2018.

26) 上乃 聖, 稲熊寛文, 三村正人, 河原達也: CTC による文字単位のモデルを併用した attention による単語単位の End-to-End 音声認識, 情報処理学会研究報告, SLP-120-16, MUS-118-16, 2018.

27) 上乃 聖, 稲熊寛文, 三村正人, 河原達也: 文字単位のモデルを併用した単語単位の End-to-End 音声認識, 日本音響学会研究発表会講演論文集, 3-8-5, 春季 2018, 学生優秀賞受賞.