Doctoral Dissertation

Self-adaptive and Incremental Machine Speech Chain

Sashi Novitasari
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Satoshi Nakamura
Augmented Human Communication Lab.
(Division of Information Science)

Submitted on September 16, 2022

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Sashi Novitasari

Thesis Committee:

Supervisor	Satoshi Nakamura
	(Professor, Division of Information Science)
	Taro Watanabe
	(Professor, Division of Information Science)
	Helen Meng
	(Professor, Chinese University of Hong Kong)
	Sakriani Sakti
	(Associate Professor, JAIST)

Self-adaptive and Incremental Machine Speech Chain*

Sashi Novitasari

Abstract

In human spoken communication, speech production and perception are inseparable. It is reflected in the human speech chain mechanism, showing that humans speak while listening. This mechanism allows them to monitor and improve their speech performance in various situations. It is also important for language acquisition.

Inspired by the human speech chain mechanism, a machine speech chain framework based on deep learning was recently proposed for a semi-supervised development of a text-to-speech (TTS) system and an automatic speech recognition (ASR) system. However, the basic framework was aimed only for non-incremental TTS and ASR training, in which the systems require a long delay when encountering a long input sequence. Moreover, the TTS and ASR still perform separately during inference. They could not do self-adaptation or change the speech by considering environmental situations. By contrast, humans can listen to what they speak in real-time and enhance the intelligibility of their speech, which is called the Lombard effect. If there is a delay in the hearing, they won't be able to continue speaking and adapting to the environment appropriately.

In this thesis, we propose self-adaptive and incremental machine speech chain frameworks for training and inference by mimicking the human speech chain closely. To achieve this, first, we reduce the latency of the basic machine speech chain by replacing the components with an incremental TTS (ITTS) and an incremental ASR (ISR). During speech chain training, we let these systems improve together through a short-term loop. Second, we design a self-adaptation

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, September 16, 2022.

framework focusing on speech synthesis in noisy environments through a speech chain mechanism. It synthesizes the speech not only by taking text input but also the auditory feedback representing the current system performance and the environmental situation. This mechanism allows the TTS to speak in a Lombard effect automatically according to real situations. Finally, we perform experiments of self-adaptive incremental speech synthesis with a low-latency adaptation in noisy environments. Low-latency adaptation is critical for machine to perform optimally in dynamic situations. All this contribution shows that the feedback mechanism is not only essential for the human speech chain but also for machines to dynamically adapt and improve themselves in various situations.

Keywords:

self-adaptation, incremental, machine speech chain, speech synthesis

Acknowledgements

I would like to express my gratitude to Professor Satoshi Nakamura for welcoming me to his lab and providing me the best research environment I have ever been to. He introduced me to the world of scientific research through a summer internship back in 2017 and recommended me for MEXT-IPGP scholarship program. I would not imagine myself in this opportunity without his support. His great insight and leadership inspire me on how I strive to become a good researcher.

I would like to thank Associate Professor Sakriani Sakti for tirelessly supervising and teaching me various things. From her, I learned the knowledge of spoken language technology, research methods, academic writing and presentation, and other things that made my skills today what they are. Her working style and enthusiasm always inspire me, and I am grateful for her continuous support. I thank the thesis committee, Professor Taro Watanabe and Professor Helen Meng for their valuable comments and suggestions for this thesis.

I also want to express my gratitude to each member of the Augmented Human Communication Lab, my first family in Japan. I would like to thank all the faculty for their support, discussion, and constructive feedback during my research progress report. My special thanks to Ms. Manami Matsuda, who has always given me research and daily support since my internship time in NAIST, Japan. Her advice helps me be able to go through my life in Japan.

Finally, I want to thank my family and friends in Indonesia for giving me emotional support. Their encouragement always supports me during hard times and makes me never give up on achieving my dreams.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Human Speech Communication

Communication plays a crucial role in advancing human civilization. It allows humans to share their thoughts and experiences to better understand each other. Among many forms, speech might be the most effective way to communicate our thoughts in most circumstances [1]. In speech communication, messages can be transmitted without delay, enabling quick information exchange between people. It also allows immediate feedback to both speaker and listener and flexibility in deciding their actions during the conversation.

Speech communication involves the activities of speaking (speech production) and listening (speech perception). Although these seem to be simple, there are more complex processes underlying those activities to achieve a successful conversation. To produce speech, one does not simply move the lips and tongue to utter the words; it also involves the adjustment of linguistic and acoustical planning in real-time to speak clearly. Speech perception is also not limited to the act of hearing the sounds, but also to differentiating between the words and noises and comprehending their meaning. These tasks are deeply related through a speech chain mechanism.

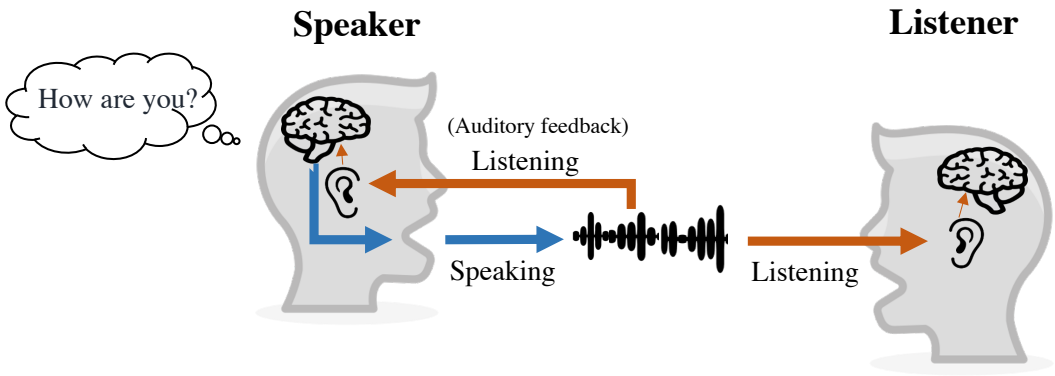


Figure 1.1: Human speech chain [1].

1.1.1 Introduction to Human Speech Chain Mechanism

Human speech communication involves a speech chain process, a chain of events that links the speaker and the listener's mind [1] in Fig. 1.1. To speak, the speaker first decides what to say and how to say it, and then generates the speech by moving the vocal muscles coordinated by the brain. The transmitted speech will be received by the listeners through their hearing system, which then continues to the speech perception process to understand the message. During these events, the speaker also acts as a listener by listening to their voice. By perceiving their speech as mouth-to-ear auditory feedback, the speaker compares the quality of speech with their initial speech plan. Based on this, they can coordinate their vocal efforts to make the next speech match the acoustic goal. The human speech chain shows that speech perception and production tasks are closely related. Here, all processes are done simultaneously. In short, speech communication is established by the activities of speaking while listening.

1.1.2 Functions of Speech Chain

The functions of the speech chain include language acquisition, speech monitoring through incremental feedback, and speech self-adaptation based on the environment, including noisy places such as the Lombard effect.

1.1.2.1 Language Acquisition

Language acquisition is defined as the process of gaining the ability to comprehend and use the words in a language to communicate with others. Here, speaking efforts made during the speech chain process may affect language acquisition and development. The acquisition of the first or native language happens in the childhood period through babbling in the early stages [4, 5, 6]. Infants' babble consists of articulate sounds but has not yet formed into recognizable words. To produce a "sound", infants first differentiate between sounds in the environment by listening to them and then select the"sound" to be uttered. The vocal adjustment might be performed so the produced sound is similar to what they hear, and from here they learn the articulatory control and the mapping between the phonetic and phoneme properties. This shows that the development of speech production and speech perception systems are deeply correlated. By continuously learning the phonetic map and exploring the language properties around them, children will eventually produce their first words and more complex words in the corresponding language structure.

1.1.2.2 Incremental Feedback

When speaking, the auditory feedback that is obtained incrementally allows the speaker to monitor and improve their speech in real-time. Inadequate monitoring of auditory feedback may result in speaking difficulties [7, 8]. Several studies have investigated the importance of auditory feedback in speech perception as well as in speech production [9, 7, 8]. It is done by constructing a delayed auditory feedback (DAF) device that extends the time between speech and auditory perception. A study by Badian et al. [7] found that using DAF with a 175-millisecond delay has been shown to induce mental stress, which was measured as changes in biochemical and cardiovascular variables. Another study also found that the effect of a few hundred milliseconds of delay can disturb people, and this effect disappears immediately by stopping speaking [8]. Thus, if there is a delay in hearing, humans are unable to continue their speech.

1.1.2.3 Self-adaptation and Lombard Effect

Humans maintain their speech quality in various conditions by simultaneously listening to their speech. When an error is detected in their speech, they will adapt or tune their speech plan according to the auditory feedback. Auditory feedback is not only used to maintain the stability between the sound output and the acoustic goal but also helps to make situation-dependent adjustments to the prosody attributes [9]. The adjustment is not only manifested in the next utterance but also could be presented as a correction of the previous utterance through the re-speaking attempt [10].

Speech adaptation notably occurs in noisy conditions. In a noisy environment, humans continuously adjust their vocal effort to improve their speech intelligibility, a phenomenon known as the Lombard effect [11, 12]. This adjustment does not only affect the loudness of the speech but also other aspects such as the speech pitch and speaking rate [13, 14, 15]. As the response to ambient noise, the intensity and pitch tend to increase, while the speaking rate tends to become slower. Several works reported the response latency in the human Lombard effect about 90-287 ms [16, 17, 18].

1.2 Speech Technology for Human-Machine Communication

In recent decades, many speech technologies have been invented by mimicking the mechanisms in humans to support human-machine and human-human communication and through machine mediation. Corresponding to human speech production and perception, several approaches in automatic speech synthesis and speech recognition systems were studied and developed to enable the machine to speak and listen.

1.2.1 Speech Synthesis Systems

Automatic text-to-speech (TTS) is a technology that automatically synthesizes a speech signal given a text, as shown in Fig. 1.2. It acts as a "speaking" system for machines. The research on TTS has progressed to synthesizing speech with

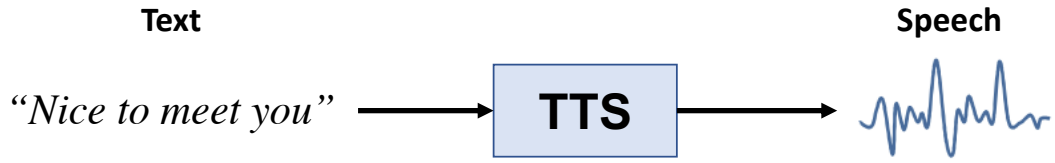


Figure 1.2: Speech synthesis by TTS.

high naturalness and intelligibility. Naturalness shows how close the TTS speech is to human speech, while intelligibility describes how easily the TTS speech can be understood. In earlier times, automatic speech synthesis was done by concatenating segments of recorded speech in the form of diphones by considering the F0, stress, duration, and formant distance between the segments [19, 20]. From concatenative speech synthesis, TTS technology is then gradually shifted into a more flexible approach through statistical modeling to synthesize speech in a spectrogram similar to reference speech. In the statistical approach with a hidden Markov model (HMM), spectrum, pitch, and state duration are modeled simultaneously in a unified framework of HMM [21, 22]. Speech generation is done through speech parameter or feature generation using HMM and Mel-cepstrum-based vocoding techniques.

Recently, end-to-end text-to-speech modeling with neural networks has gained high attention in the research and industry community. Previously, the Tacotron [23, 24] framework was proposed for speech Mel-spectrogram synthesis from character sequences autoregressively using the recurrent neural network-based encoder-decoder network with an attention mechanism. Based on Tacotron, Transformer network-based TTS was also proposed by replacing the recurrent network with the transformer block [25, 26]. Fastspeech [27, 3] in a Transformer framework was also proposed for non-autoregressive speech synthesis by modeling the speech duration inside the network.

1.2.2 Speech Recognition Systems

Automatic speech recognition (ASR) is a technology that transcribes speech into text, which is shown in Fig. 1.3. This system works in a reversed way in TTS and acts as a machine “listening” component. In the classic approach, the HMM-

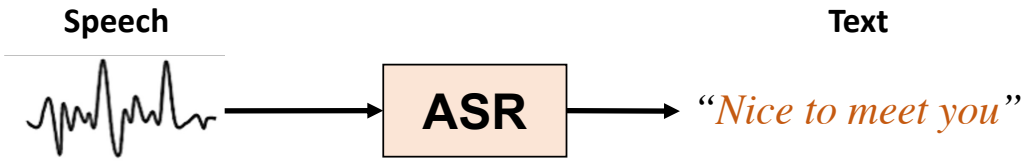


Figure 1.3: Speech recognition by ASR.

based ASR was used for the large vocabulary continuous speech recognition system [28, 29, 30]. HMM-based ASR consists of three separate components: the acoustic model, the lexicon, and the language model. The acoustic model predicts the phoneme sequence from the input speech by modeling the speech features distribution using a Gaussian mixture model (GMM) and finding the optimal phoneme sequence using HMM. The second component, a lexicon consisting of the word-to-phonemes dictionary, proposes the word candidates. From the word candidates, the final word sequence is decided by the language model by maximizing the word sequence probability score.

Similar to the progress in TTS, recent ASR also works in an end-to-end manner using a neural network structure. Connectionist temporal classification (CTC) based ASR was proposed for speech modeling with the recurrent units to directly map the speech into text [31, 32]. A listen, attend, and spell (LAS) framework was also proposed for speech recognition with the encoder-decoder network with the attention mechanism [33]. Recently, Transformer-based ASR also gained high attention in the research fields [34, 35].

1.3 Limitations and Challenges

Problem:
Common TTS and ASR systems perform separately, unable to dynamically adapt to the changing inference environments

TTS and ASR systems commonly perform their tasks separately. It simplifies the construction and utilization of these systems. However, they do not have a feedback mechanism like humans. TTS can only speak and ASR can only listen.

Therefore, during inference, those systems are not aware of the current situation and their performance, and they can not adapt themselves.

In contrast, humans can do self-adaptation dynamically in various situations with speech chain mechanisms. The auditory feedback connection in the speech chain shows that speech production and speech perception systems are deeply related. If we could apply a similar mechanism to the human speech chain to machines, we might be able to make TTS and ASR improve and adapt themselves to various conditions by doing self-monitoring. But despite this potential, unfortunately, these have received less attention in the TTS and ASR research communities.

Existing approaches:
Machine speech chain for joint TTS and ASR construction through auditory feedback loop

A machine speech chain framework [2, 36] was previously proposed to link the TTS and ASR with an auditory feedback loop during training. The motivation was to enable TTS and ASR semi-supervised construction with only a small number of paired speech-text data and unpaired data, which are less complicated to collect than paired data. From the unpaired data, a data reconstruction loss will be computed and regarded as feedback. From this, TTS and ASR could support each other and improve together. In inference, TTS and ASR perform separately without the auditory feedback connection. Although this framework successfully improved TTS and ASR in a semi-supervised training setting, those systems are still unable to do self-adaptation during inference due to the lack of auditory feedback, unlike humans.

Another limitation in the previous machine speech chain is that the framework was originally designed for the utterance-level TTS and ASR systems. Utterance-level systems require a completed utterance input to produce an output. Because of this, the speech chain mechanism to listen while speaking can be done only after receiving the entire input sequences. ASR starts recognition after receiving a complete speech utterance from TTS, and TTS begins its synthesis after receiving a complete sentence from ASR. As a result, there is a significant delay when encountering long utterances. In contrast with machines, humans can listen to

what they speak in real-time, whereas a delay in auditory feedback may cause speaking issues in humans.

Offline fine-tuning on specific environment conditions

Offline fine-tuning is commonly applied to enable the TTS and ASR to perform well in the new environments [37, 38, 39, 40]. Fine-tuning trains a pre-trained model using the dataset representing the new environments. This is done before the system is utilized in inference. By using this method, we can improve system performance when presented in similar environments to the training condition. However, the current fine-tuned TTS and ASR systems might still not be able to perform dynamically because they only take speech or text as the input without knowing the current situation. For example, a TTS fine-tuned to Lombard speech will always produce Lombard speech, with a similar sound to the training data, in any situation. The TTS might perform well if the inference environment is similar to the presumed condition in training. However, environmental noise can change dynamically. Moreover, in quiet conditions, it might not be preferred that TTS produce loud speech.

1.4 Thesis Objective and Contribution

1.4.1 Thesis objective: Machine speech chain framework for machine self-adaptive inference based on environmental conditions

We borrow the concept of the human speech chain to improve the machine's performance in inference dynamically based on feedback observation. The previous machine speech chain focused on joint semi-supervised TTS and ASR training. In this thesis, we propose an advanced version of a machine speech chain that applies the feedback mechanism during training and inference. The auditory feedback mechanism aims to enable self-adaptation in the machine conditioned on the machine's performance and environmental factors. As one of the use-cases, in this thesis, we specifically focus on a self-adaptive and incremental machine speech

chain framework for speech synthesis in noisy situations with the similar idea to the Lombard effect. In inference, TTS synthesizes the speech by considering auditory feedback from ASR and environmental conditions.

To achieve our goal, two challenges are considered:

1. **Low-latency processing**
   Low-latency self-adaptation is necessary to make the machine perform and adapt optimally. In real situations, environmental conditions often change. Systems with a long output and adaptation latency might be unable to catch up with those changes and perform poorly. Therefore, machines should be able to make immediate adaptations by generating and processing short-term output and feedback incrementally.

2. **Auditory feedback during inference**
   Auditory feedback is critical to perceiving the environmental situation and how the system performs now. The existing machine speech chain uses reconstruction loss as feedback. In this thesis, we extend the auditory feedback closer to humans, which also contains environmental information and use it explicitly in inference.

1.4.2 Thesis contribution

We propose self-adapting and incremental machine speech chain for both training and inference. Fig. 1.4 shows the tasks covered in this thesis. The proposed frameworks are the extensions of the basic machine speech chain proposed by Tjandra et al. [2, 36] for semi-supervised TTS and ASR training. Here, we construct three frameworks to achieve our goal step-by-step.

First, we reduce the latency of the basic machine speech framework by replacing the components with an incremental TTS (ITTS) and an incremental ASR (ISR). We formally call the proposed framework the incremental machine speech chain. This framework aims to improve the learning quality of ITTS and ISR through the short-term feedback loop and also demonstrates short-term feedback generation during inference.

Second, we design a machine speech chain inference mechanism for TTS self-adaptation in noisy conditions. The feedback loop between TTS and ASR is

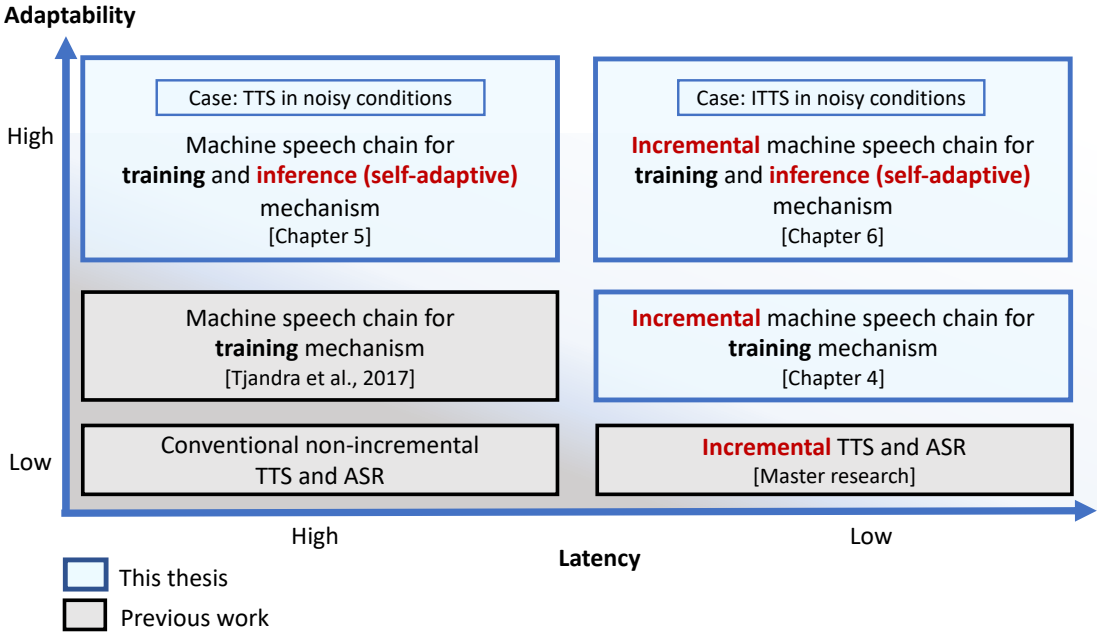


Figure 1.4: Thesis contribution.

maintained during training and inference. We also add a new feedback to the auditory feedback: the speech-to-noise ratio (SNR), which represents the intensity ratio between speech and environmental noises. At this point, the TTS synthesizes the speech at utterance-level with utterance-level feedback.

Lastly, we show a self-adaptive ITTS with machine speech chain inference that performs incremental speech synthesis with the immediate adaptation to environmental noises. The adaptation is done by progressively generating the output and feedback and utilizing the feedback to synthesize and improve the next output.

1.5 Thesis Outline

The structure of the remaining chapters of this thesis is as follows. In Chapter 2, we describe the end-to-end speech modeling technology for speech synthesis and speech recognition. We elaborate on the neural network frameworks that we use in the proposed systems. In Chapter 3, we describe the basic machine speech chain that becomes the basis of the proposed systems.

In Chapter 4, we introduce our proposed incremental machine speech chain framework for ITTS and ISR training. We show our attempts at reducing system latency through the supervised and unsupervised short-term feedback loop.

In Chapter 5, we show the machine speech chain mechanism for TTS inference in noisy situations. We describe our attempt at synthesizing Lombard speech through an auditory feedback mechanism under static noise conditions.

In Chapter 6, we describe our attempt to reduce the adaptation latency in speech synthesis with machine speech chain inference by using an ITTS. Here we perform speech synthesis experiments by considering static and dynamic noises.

Finally, we conclude our thesis in Chapter 7. In addition, we also discuss further possible future research on the topic related to this thesis.

Chapter 2

End-to-end Neural Speech Modeling Framework

This chapter covers the basic knowledge about the end-to-end neural network frameworks that are implemented in the proposed system.

2.1 Sequence-to-sequence Framework

Sequence-to-sequence (seq2seq) framework is a neural network structure that converts an input sequence into another output sequence. This framework is commonly used in end-to-end systems that model the conditional probability directly:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{T} p(y_t|\boldsymbol{x}, y_{<t}) \tag{2.1}$$

using single model given an input sequence $\boldsymbol{x} = [x_1, x_2, ..., x_S]$ with a length S and the target output sequence $\boldsymbol{y} = [y_1, y_2, ..., y_T]$ with length T. Seq2seq frameworks commonly consist of two main components:

- **Encoder**: Encodes the input sequence $\boldsymbol{x}$ into an encoded representation $\boldsymbol{h}^e = [h_1^e, h_1^e, , ..., h_S^e]$.
- **Decoder**: Produce output sequence $\boldsymbol{y}$, where each output at timestep t (y_t) is conditioned on the output from the previous timestep $y1, ..., y_{<t}$ and the encoded representation $\boldsymbol{h}^e$.

The encoder-decoder model, however, has difficulty with long sequences because the decoder has limitations in finding relevant information from $\boldsymbol{h}^e$. To overcome this issue, an attention mechanism [41, 42] was proposed to bridge the information sharing between encoder and decoder. It enables the decoder to soft-search for the relevant information in the encoder sequence. Specifically, when attention mechanism is applied, decoder produces a decoder sequence $\boldsymbol{h}^d = [h_1^d, h_1^d, , ..., h_t^d]$ at the timestep t conditioned on $y1, ..., y_{<t}$. Then attention module calculates the alignment scores between $\boldsymbol{h}^e$ and $\boldsymbol{h}^d$ marking the relevance of each part in the encoder sequence to the current decoder sequence, which scores are then used to help the current output y_t prediction.

In this thesis, two seq2seq frameworks are utilized: seq2seq recurrent neural network (RNN) and Transformer. These frameworks differ mainly in the network type and the attention mechanism. In the next section, the detailed mechanisms applied in the encoder, decoder, and attention modules for each framework are elaborated.

2.1.1 Sequence-to-sequence RNN

Seq2seq RNN [41] in Fig. 2.1 consists of an encoder and a decoder, each consists of a stack of RNN layers, with an attention module between them. All components are optimized jointly during the training process. Given input sequence $\boldsymbol{x}$, first encoder generate the encoder sequence $\boldsymbol{h}^e$ by taking $\boldsymbol{x}$, expressed as

$$\boldsymbol{h}_e = \text{Encoder}(\boldsymbol{x}). \tag{2.2}$$

To predict y_t, decoder generates the decoder state h_t^d conditioned on the previous output y_{t-1}, previous decoder state h_{t-1}^d, and the encoder sequence $\boldsymbol{h}^e$:

$$h_t^d = \text{Decoder}(y_t - 1, h_{t-1}^d, \boldsymbol{h}^e). \tag{2.3}$$

Based on the decoder state, attention module will compute the alignment score between h_s^e for $s = [1, 2, ..., S]$ in $\boldsymbol{h}^e$ and the current decoder state h_t^d, written as

$$\begin{aligned} a_t(s) &= \text{Align}(h_s^e, h_t^d), \\ &= \frac{exp(Score(h_s^e, h_t^d))}{\sum_{s=1}^{S} exp(Score(h_s^e, h_t^d))}, \end{aligned} \tag{2.4}$$

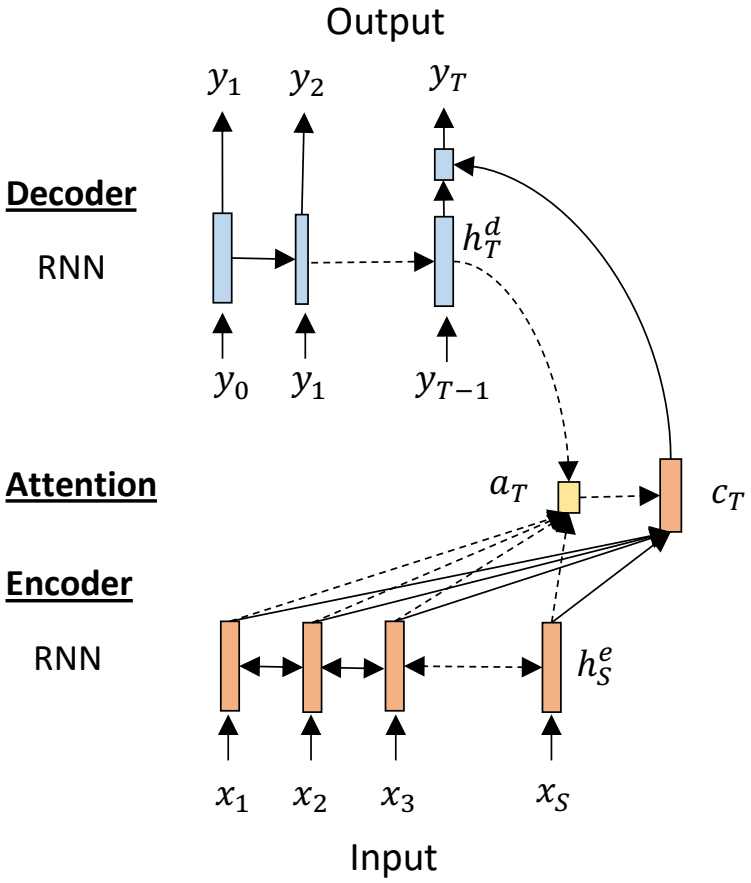


Figure 2.1: Seq2seq RNN with encoder, decoder, and attention module.

$$c_t = \sum_{s=1}^{S} a_t(s) * h_s^e, \tag{2.5}$$

where $a_t(s)$ is the alignment score between h_s^e and h_t^d. c_t is the context information utilized to predict y_t by taking a sum of $a_t(s)$ for all encoder timestep. In Eq.2.4, *Score* is commonly computed using one of the following functions [42]:

$$Score(h_s^e, h_t^d) = \begin{cases} \langle h_s^e, h_t^d \rangle, & \text{dot product} \\ h_s^{e\intercal} W_s h_t^d, & \text{bilinear} \\ V_s^\intercal \tanh(W_s[h_s^e, h_t^d]) & \text{MLP,} \end{cases} \tag{2.6}$$

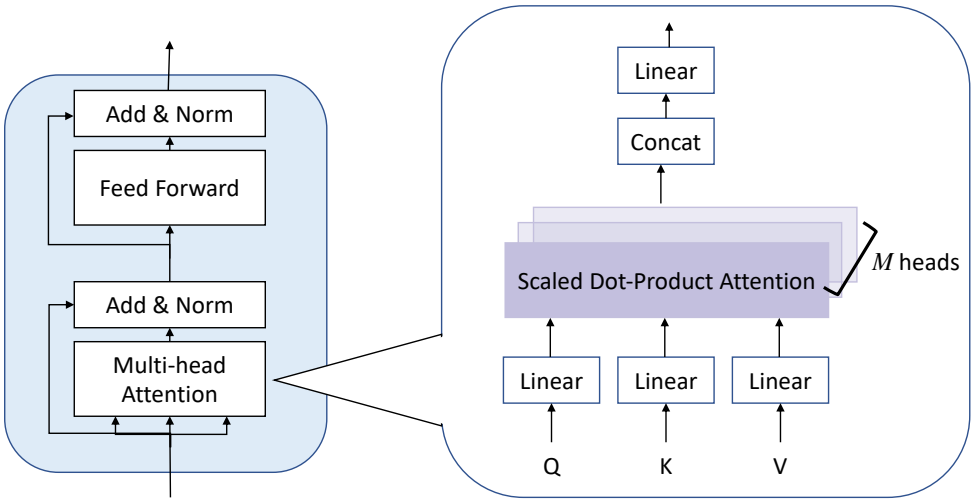


Figure 2.2: (a) Basic transformer block with (b) multi-head self-attention module implemented inside.

where $Score$ is a $(\mathbb{R}^{U_{enc}} \times \mathbb{R}^{U_{dec}}) \rightarrow \mathbb{R}$ function, where U_{enc} is the number of encoder hidden units and U_{dec} is the number of decoder hidden units. After the current context c_t is obtained, the current output y_t can be predicted by concatenating c_t and h_t^d and pass it to the output network layer on top of decoder (W_o),

$$y_t = W_o([c_t, h_t^d]). \tag{2.7}$$

2.1.2 Transformer

Transformer [43] consists of an encoder and a decoder in stack of transformer blocks. A basic transformer block, shown in Fig. 2.2 (a), has a multi-head self-attention layer followed by a feed forward layer with add and normalization layers on top.

Multi-head self-attention in Fig. 2.2 (b) has a different attention mechanism to that of the seq2seq RNN. First, self-attention is an attention mechanism that computes the alignment between different positions in a sequence, or the inner-alignment. Self-attention computation begins by projecting the transformer block

input, with the dimension of d_{model}, into three matrices: query (Q), key (K), and value (V) with the dimensions of d_q, d_k and d_v respectively. These matrices are then processed through a scaled dot-product attention. The operation consists of dot product between Q and K, which is then divided by $\sqrt{d_k}$, passed through a softmax function and multiplication to V. It is formally written as

$$\text{Self-attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{2.8}$$

The scaled-dot product attention aims to map Q and a set of K-V pairs to an output.

Instead of using single attention computation, multi-head self-attention performs the attention computation M times or heads with the different linear projection weights into d_q, d_k and d_v. For the final attention output, all M attention heads are concatenated together and projected for the final values:

$$\text{head}_m = \text{Self-Attention}(QW_m^Q, KW_m^K, VW_m^V), \tag{2.9}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_M)W^O, \tag{2.10}$$

where W_m^Q,W_m^K, and W_m^V are the linear projection network weights for Q, K and V respectively for the m-th head, and W^O is the final projection layer. The multi-head attention approach allows Transformer to encode several relationships within the input sequence.

By stacking the transformer blocks in encoder and decoder, Transformer in Fig. 2.3 is commonly utilized to perform auto-regressive prediction for seq2seq problem. Similar to the seq2seq RNN, auto-regressive Transformer first encodes the input sequence $\boldsymbol{x}$ into a latent encoded representation $\boldsymbol{h}^e$ and then produces the output step-by-step by decoding $\boldsymbol{h}^e$ conditioned on the previous output $y_{<t}$. Not only the self-attention, here decoder also has a cross attention component to the encoder sequence. The cross attention is computed in a similar mechanism as the self-attention but by using the key and value matrices projected from the encoder sequence and query matrix projected from the decoder self-attention output below the cross attention component:

$$\text{Cross-attention}(Q^{dec}, K^{enc}, V^{enc}) = \text{Softmax}(\frac{Q^{dec}(K^{enc})^T}{\sqrt{d_k}})V^{enc}. \tag{2.11}$$

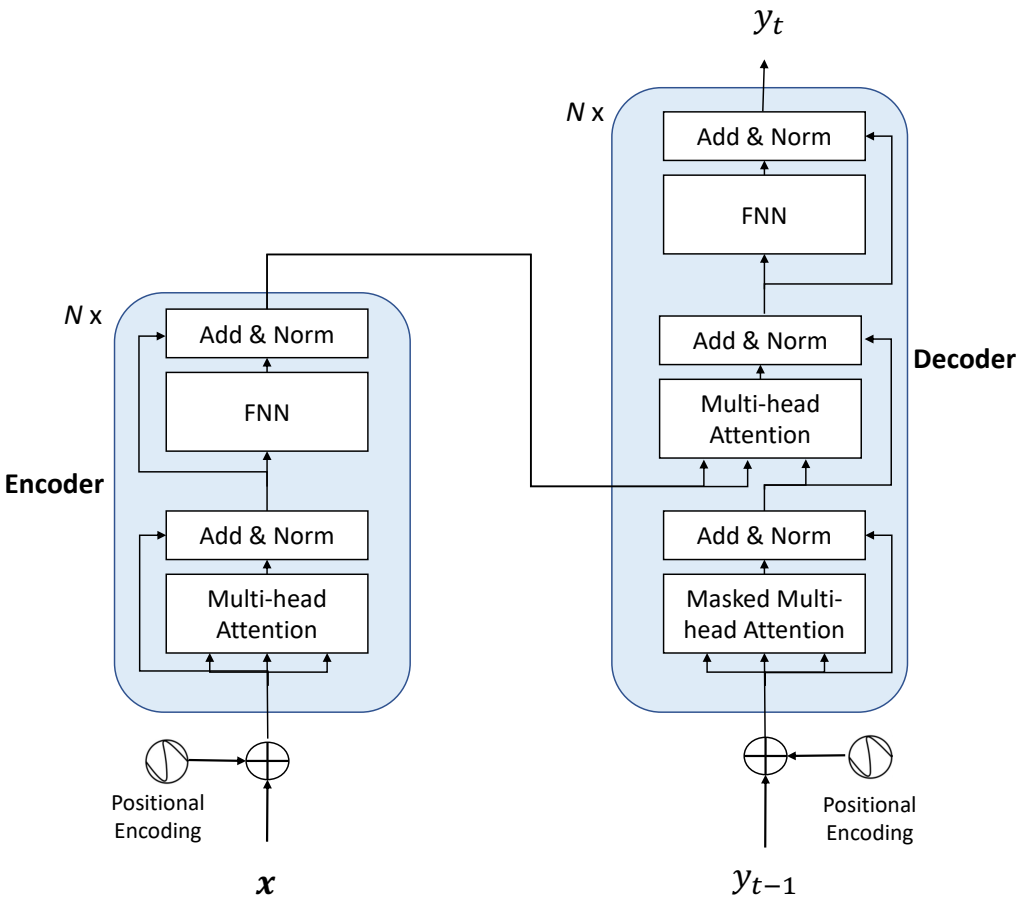


Figure 2.3: Transformer with encoder, decoder, and multi-head attention module for seq2seq prediction tasks.

Here, Q^{dec} is the query projected from decoder self-attention output and K^{enc} and V^{enc} are the key and value projected from encoder sequence.

Since the transformer block does not contains recurrent network that retains the input sequence's positional information, Transformer applies an additional component in the structure to preserve the positional information, called positional encoding (PE) component. PE is placed below the encoder and decoder. It preserve positional information of the input sequence by injecting the positional information into the input sequence. Commonly, PE applies a sine and cosine

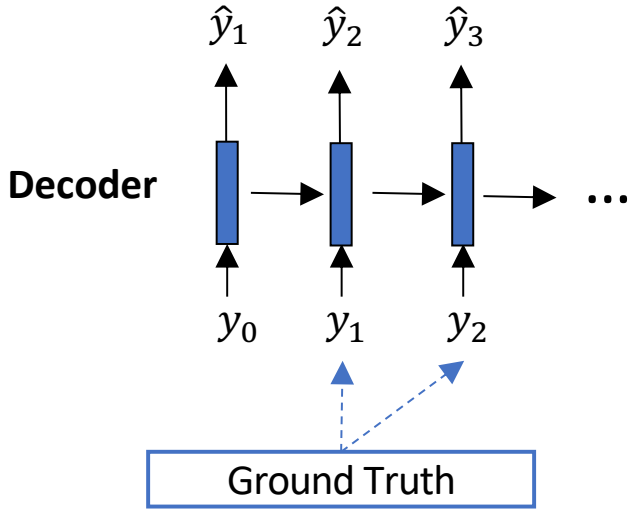


Figure 2.4: Decoding with teacher-forcing strategy for training.

function:

$$\begin{aligned} \mathrm{PE}_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ \mathrm{PE}_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.12}$$

where pos is the position index in the input sequence and i is the dimension index of the input. The output of positional encoding is combined to the transformer block input through summation. From which, the PE-fused input is passed to the main transformer block.

2.2 Training and Inference

Seq2seq neural network model is commonly trained using a teacher-forcing strategy [44] applied to the decoder. Here, we denote the target output as $\boldsymbol{y} = [y_1, ..., y_T]$ and the output predicted by the model as $\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_T]$. Teacher-forcing strategy in Fig. 2.4 is done to predict $\hat{y}_t$ by feeding the decoder the correct output of the previous timestep y_{t-1}.This strategy allows the model to converge fast and keeps the model stability during training. A training loss, which is task-dependent, is computed based on the output probability distribution computed by the model and the correct output.

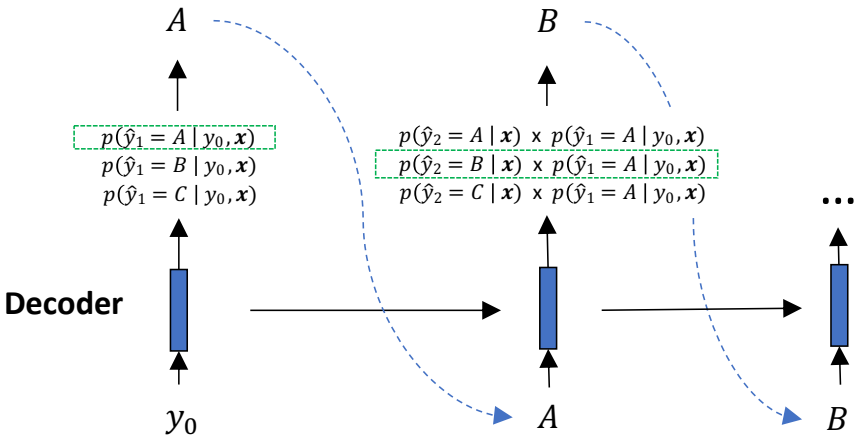


Figure 2.5: Decoding with greedy searching for inference.

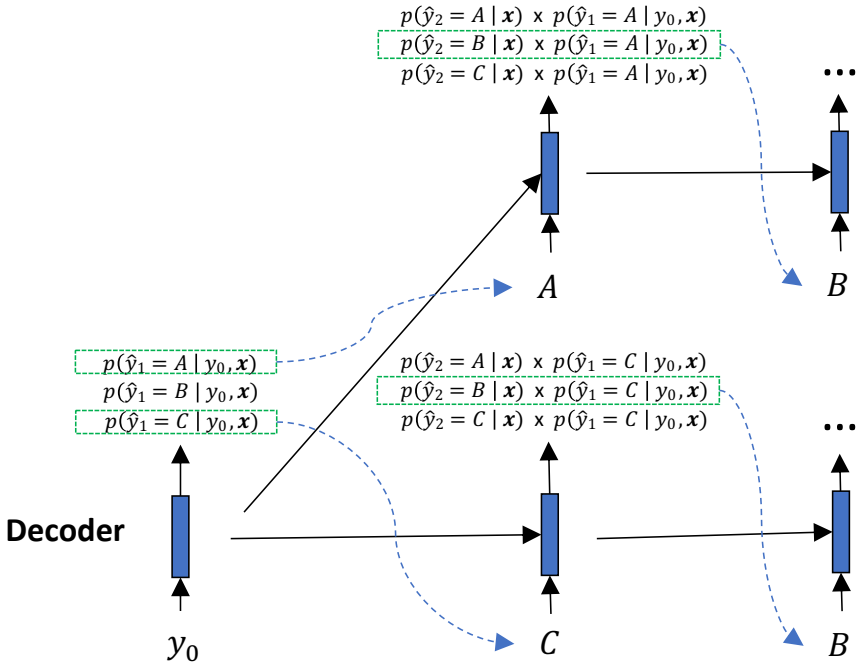


Figure 2.6: Decoding with beam searching with k=2 for inference.

In inference, decoding is done based on the output predicted by the model in the prior timestep. There are two searching algorithms that are commonly implemented in the seq2seq model to find the optimum output sequence during decoding:

- **Greedy search** (Fig. 2.5) determines $\hat{y}_t$ by choosing the output with the highest probability in the predicted probability distribution, written as

$$\hat{y}_t = \arg \max_{1 \leq c \leq C} p(\hat{y}_t | \boldsymbol{x}, \hat{y}_{<t})[c], \tag{2.13}$$

  where C is the output vocabulary or classes. This method excels in speed and output stability than other method because $\hat{y}_t$ is determined at the corresponding timestep. However, the predicted sequence might not be best one.

- **Beam search** (Fig. 2.6) determines the output sequence $\hat{\boldsymbol{y}}$ by keep track of k-best output sequences for each decoding timestep t [45]. In the subsequent decoding step t+1, the algorithm generates all possible output sequences based on the k-best sequence from t, and then select another k-best sequences based on the sequence probability score,

$$\text{Token Sequence Score}(\hat{y}_1, ..., \hat{y}_t) = \sum_{i=1}^{t} \log p(\hat{y}_i | \boldsymbol{x}, \hat{y}_{<i}). \tag{2.14}$$

  The newly selected k sequences are then passed to the next decoding timestep by also repeating the same mechanism. At the end of decoding process, the final output sequence is chosen based on the sequence with the highest score. Beam decoder may predict the final output sequence with a better quality than the greedy decoder. This is because it keep some possible output sequences with the high score at the same time.

2.3 Neural TTS

Neural TTS performs two tasks: text-to-speech features conversion using seq2seq framework and speech features-to-waveform or audio using a vocoder. In this work, we focus on the core text-to-speech features part. In the experiments, our

vocoder is a CBHG (1-D Convolution Bank + Highway + bidirectional gated recurrent unit) module with the Griffin-Lim algorithm, similar to the Tacotron TTS framework [23].

TTS models the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ of the output speech features $\boldsymbol{y}$ given the input sentence text $\boldsymbol{x}$. For the rest of the chapters, we refer $\boldsymbol{x} = [x_1, ..., x_S]$ as the sequence of text tokens and $\boldsymbol{y} = [y_1, ..., y_T]$ as the sequence of speech features

2.3.1 Input and Output

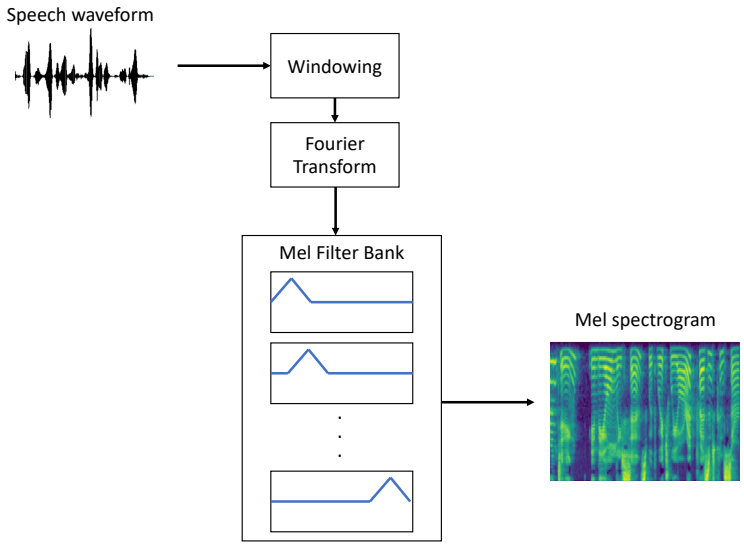


Figure 2.7: Mel-spectrogram extraction.

In this thesis, the TTS input is a text tokenized into character-level tokens. The character sequences of two words are separated by a whitespace token which here denoted as <*spc*>. The sentence is also encapsulated with a beginning-of-sentence (BOS) symbolized as <*s*> and end-of-sentence (EOS) tokens symbolized as </*s*>. An example of text with character-level tokens from a normal word-level text is the following:

Word: *hello nice to meet you*
Character: *<s> h e l l o <spc> n i c e <spc> t o <spc> m e e t <spc> y o u </s>*

The TTS output utilized in this work is the Mel-spectrogram feature of a framed speech sequence. It represents the speech power spectrum in a Mel-scale rendering the frequencies in a certain range logarithmically, which mimics the human perception. In training, Mel-spectrogram feature extration, shown in Fig. 2.7, is done by applying the short-time Fourier transform to the segmented target speech signal via windowing and then multiplying it to a Mel filter bank.

2.3.2 Structure

2.3.2.1 Sequence-to-sequence RNN-based TTS

One of the commonly used seq2seq RNN-based TTS is Tacotron. In our implementation in Fig. 2.8 (a), TTS encoder consists of a character embedding module followed by a feedforward neural network (FNN) layer and CBHG module. The decoder part consists of FNN pre-net layer, unidirectional long short-term memory (LSTM) layer, and linear output layers on top for the Mel-spectrogram and the speech end flag prediction. The speech end flag is represented as a binary value marking the end of Mel-spectrogram sequence. When the flag is positive, the decoding for Mel-spectrogram can be stopped, so it can be passed to the vocoder to compute the final speech waveform $\boldsymbol{y}^R = [y_1^R, ..., y_T^R]$. Mel-spectrogram and speech end flag sequences are predicted in parallel.

Based on the multi-speaker TTS utilized in the previous machine speech chain work [46], a structure modification is done on the decoder to do multi-speaker speech synthesis, shown in Fig. 2.8 (b). Here, a speaker embedding z_{SPK} that represents the speaker identity is summed to the decoder pre-net output and also the Mel-spectrogram linear output layer.

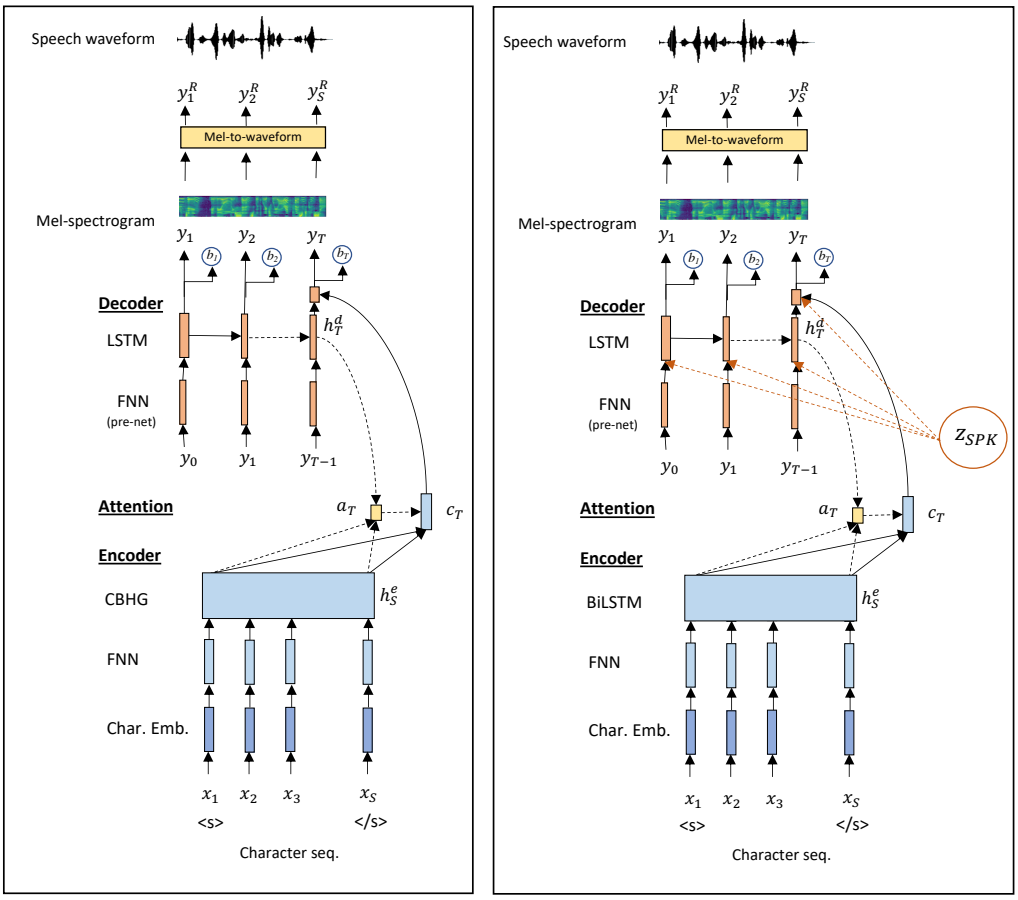
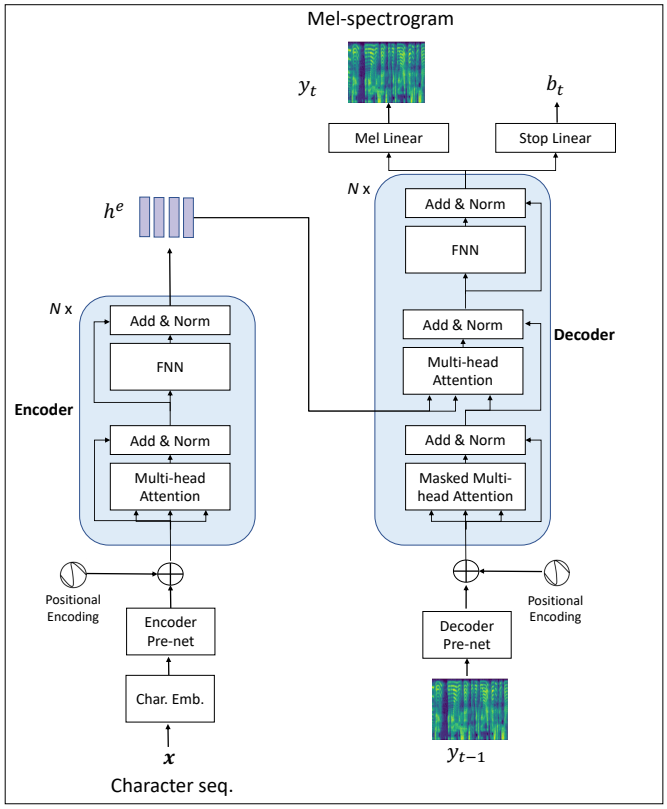


Figure 2.8: Seq2seq RNN-based TTS for (a) single-speaker and (b) multi-speaker speech synthesis.

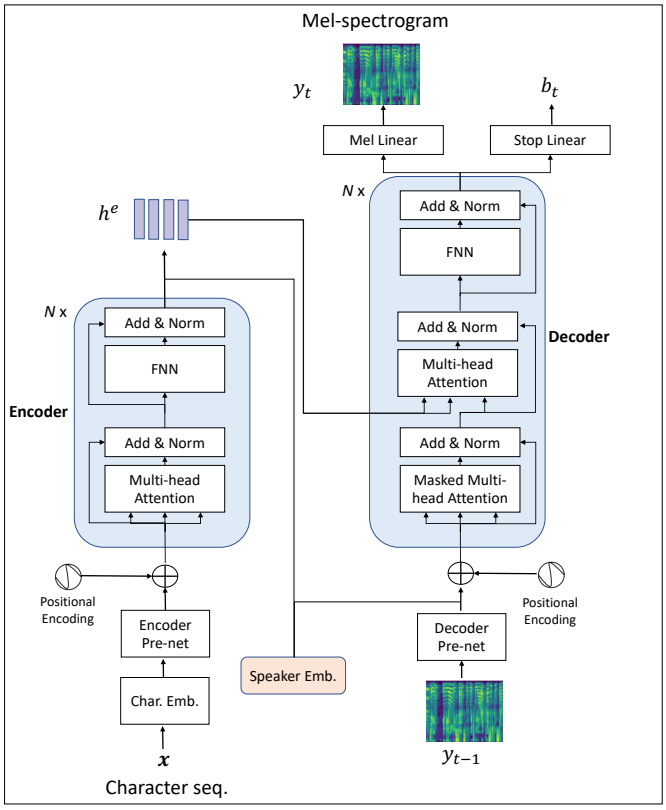
2.3.2.2 Transformer-based TTS

Our Transformer-based TTS is a encoder-decoder Transformer with an autoregressive decoder [26, 25], shown in Fig. 2.9 (a) for single-speaker and Fig. 2.9 (b) for multi-speaker TTS. The structure is based on the Tacotron by replacing the encoder and decoder into transformer blocks. This framework also output two sequences: Mel-spectrogram and speech end flag.

Transformer-based TTS encoder consists of an encoder pre-net followed by positional encoding and stacks of transformer blocks. The encoder pre-net consists



(a)



(b)

Figure 2.9: Transformer-based TTS for (a) single-speaker and (b) multi-speaker speech synthesis.

of character embedding and the convolutional layers with a batch normalization layer and ReLU activation.

On top of encoder is a decoder. Decoder contains a decoder pre-net below the positional encoding and then followed by stack of transformer blocks. The transformer block in decoder also contains multi-head cross attention that calculate the attention between the encoder and decoder sequence. On the last transformer block, two parallel linear layers are placed for Mel-spectrogram and speech end flag predictions respectively. For multi-speaker speech synthesis, a speaker embedding also injected to the encoder state and the decoder input.

2.3.3 Training

Seq2seq RNN-based and Transformer TTS are commonly trained using the teacher-forcing strategy. In both of the frameworks, model optimization is done by back-propagating the loss calculated using L2 function that includes L2 distances for the speech feature and the speech end flag, expressed as

$$\begin{aligned} &Loss_{TTS}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \\ &\frac{1}{T}\sum_{t=1}^{T}((y_t - \hat{y}_t)^2 - (b_t \log(\hat{b}_t) + (1 - b_t)\log(1 - \hat{b}_t))), \end{aligned} \tag{2.15}$$

where $\boldsymbol{Y} = (\boldsymbol{y}, \boldsymbol{b})$ and $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{y}}, \hat{\boldsymbol{b}})$. Here, $\boldsymbol{y}$ is the target speech feature and $\hat{\boldsymbol{y}}$ is the predicted speech feature. $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ are the reference and the predicted probability of the speech end flag.

2.4 Neural ASR

Neural ASR converts a sequence of speech features into a sequence of text using a neural network. It models the conditional probability $p(\boldsymbol{x}|\boldsymbol{y})$ of the output sentence text $\boldsymbol{x}$ given the input speech features $\boldsymbol{y}$.

2.4.1 Input and Output

The input and output of ASR are the reverse of TTS. In this thesis, the ASR input speech features is also the Mel-spectrogram of speech and the output is a

text with the character-level tokenization. Similar to the TTS text, ASR output text is also encapsulated with a BOS and an EOS token. The character sequences of different words are also separated by a whitespace token. From the final character sequence, the word-level output text can be obtained by concatenating the characters and separating the words based on the predicted whitespace tokens.

2.4.2 Structure

2.4.2.1 Sequence-to-sequence RNN-based ASR

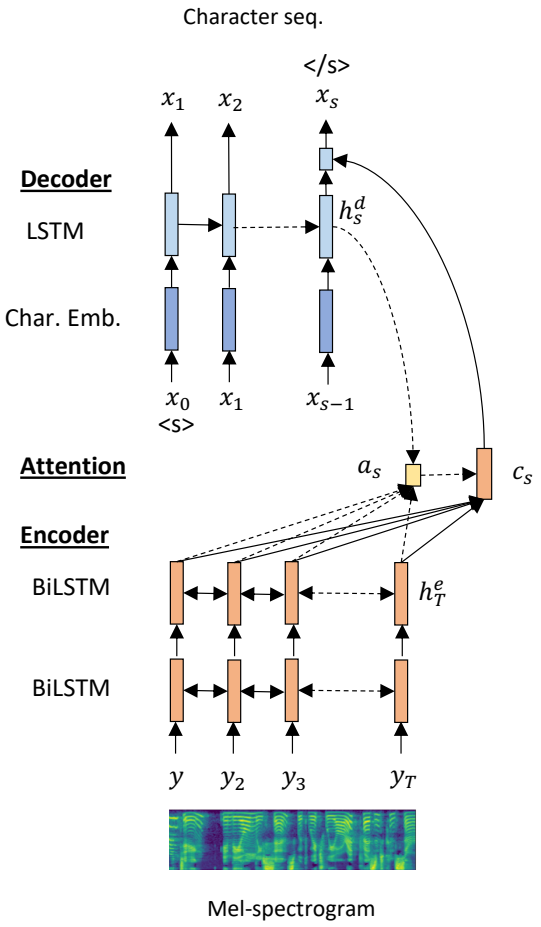


Figure 2.10: Seq2seq RNN-based ASR.

In the seq2seq RNN-based ASR, as shown in Fig. 2.10, the encoder encodes

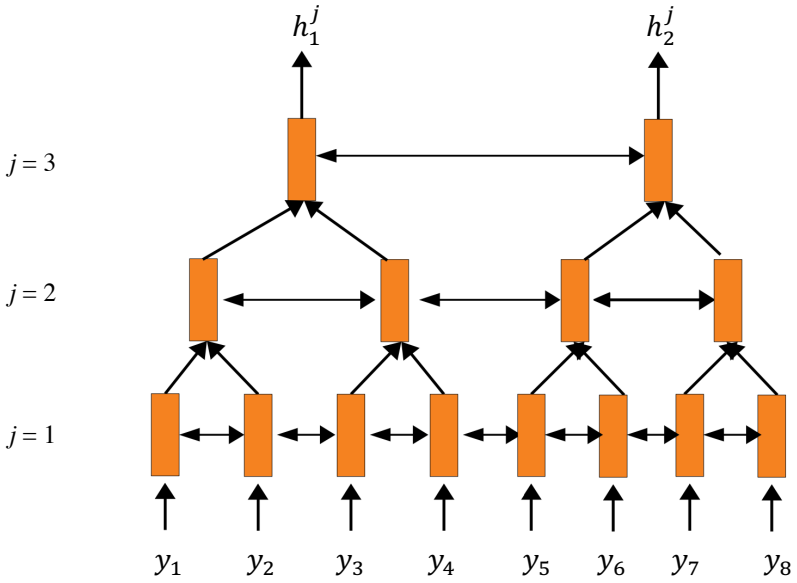


Figure 2.11: Example of encoder structure with hierarchical sub-sampling.

the whole speech utterance using bidirectional LSTM (BiLSTM) layers and then the decoder decodes for the output autoregressively using LSTM layers.

In encoder, sequence hierarchical sub-sampling [33, 47, 48] is commonly applied. Since the length of a framed speech features sequence can be very long, it can cause the encoder to converge very slow, thus, unable to achieve the optimum performance. This is because the relevant information extraction from a long sequence is difficult. By using speech sequence sub-sampling mechanism, the sequence's time resolution will be reduced by a factor as it proceeds to the higher layer in the encoder. In general, this is done by concatenating some consecutive outputs in the previous encoder layer to calculate the output of the current layer. An example of encoder that applies hierarchical sub-sampling with a factor of two for each layer can be seen in Fig. 2.11. If hierarchical sub-sampling is applied, for example a sub-sampling by a factor of two for each layer, the i-th output computation in j-th BiLSTM layer may follow Eq. 2.16.

$$h_i^j = BiLSTM(h_{i-1}^j, [h_{2i-1}^{j-1}, h_{2i}^{j-1}]) \tag{2.16}$$

Decoder predicts the transcription of speech as a sequence of text tokens. This component can be considered as a 'speller'. Here, decoding starts with a BOS

token as the initial input x_0 and stops when the decoder predict an EOS token.

2.4.2.2 Transformer-based ASR

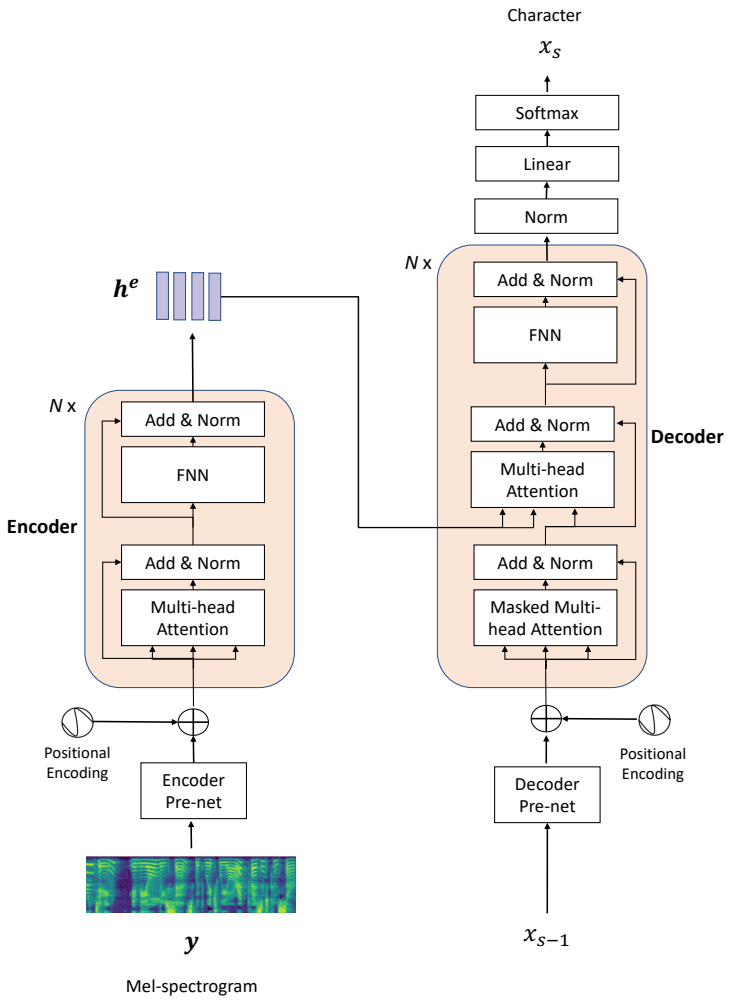


Figure 2.12: Transformer-based ASR.

Similar to the TTS, Transformer-based ASR [34] also performs speech recognition using encoder and decoder with the transformer blocks as shown in Fig. 2.12. Here, encoder is a stack of encoder pre-net with positional encoding and transformer blocks. The encoder pre-net consist of convolutional layers and a linear layer. Encoder pre-net also performs sequence sub-sampling through the convolutional layers, in which each convolutional layer reduces the sequence time

resolution by a factor of two. The decoder comprised of decoder pre-net, positional encoding, transformer blocks, and softmax layer. Decoder pre-net is a module that contains the character embedding layer that processes the decoder character input. Decoding steps is done autoregressively by feeding the previous output to the decoder, in which x_s is predicted conditioned on x_{s-1}.

2.4.3 Training

Seq2seq ASR is also generally the teacher-forcing strategy. The model optimization is done based on cross-entropy loss function between the target text $\boldsymbol{x}$ and the predicted text $\hat{\boldsymbol{x}}$

$$Loss_{ASR}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = Loss_{ASR}(\boldsymbol{x}, \boldsymbol{p}_{\boldsymbol{x}}) = -\frac{1}{S}\sum_{s=1}^{S}\sum_{c=1}^{C} \mathbb{1}(x_s = c) * \log p(\hat{x}_s|\boldsymbol{y}, x_{<s})[c]. \tag{2.17}$$

C is the number of class or the size output vocabulary

Chapter 3

Basic Machine Speech Chain

3.1 Overview

The human speech chain shows that the relationship between the speech production and perception systems is critical. The development and performance of one greatly affects another. Despite that, research on TTS and ASR has progressed more or less independently. Although the general TTS and ASR can achieve high accuracy by performing independently, this is limited to the models trained on large data. On the other hand, humans can learn new languages or words and improve their performance in real-time thanks to the speech chain mechanism.

Motivated by the human speech chain, a machine speech chain framework [2, 36] in Fig. 3.1 was proposed for semi-supervised seq2seq TTS and ASR construction. This framework connects the TTS and ASR during the training process through a closed feedback loop. The feedback loop is disconnected during inference so the system can be used independently.

Feedback connection allows us to train the models using both the labeled (paired) and unlabeled (unpaired) speech and text data in a semi-supervised way. To be precise, the semi-supervised training is composed of the supervised and unsupervised training process as the following:

- Paired speech-text for ASR and TTS training (supervised)
- Speech data only for ASR-to-TTS (unsupervised)
- Text data only for ASR-to-TTS (unsupervised)

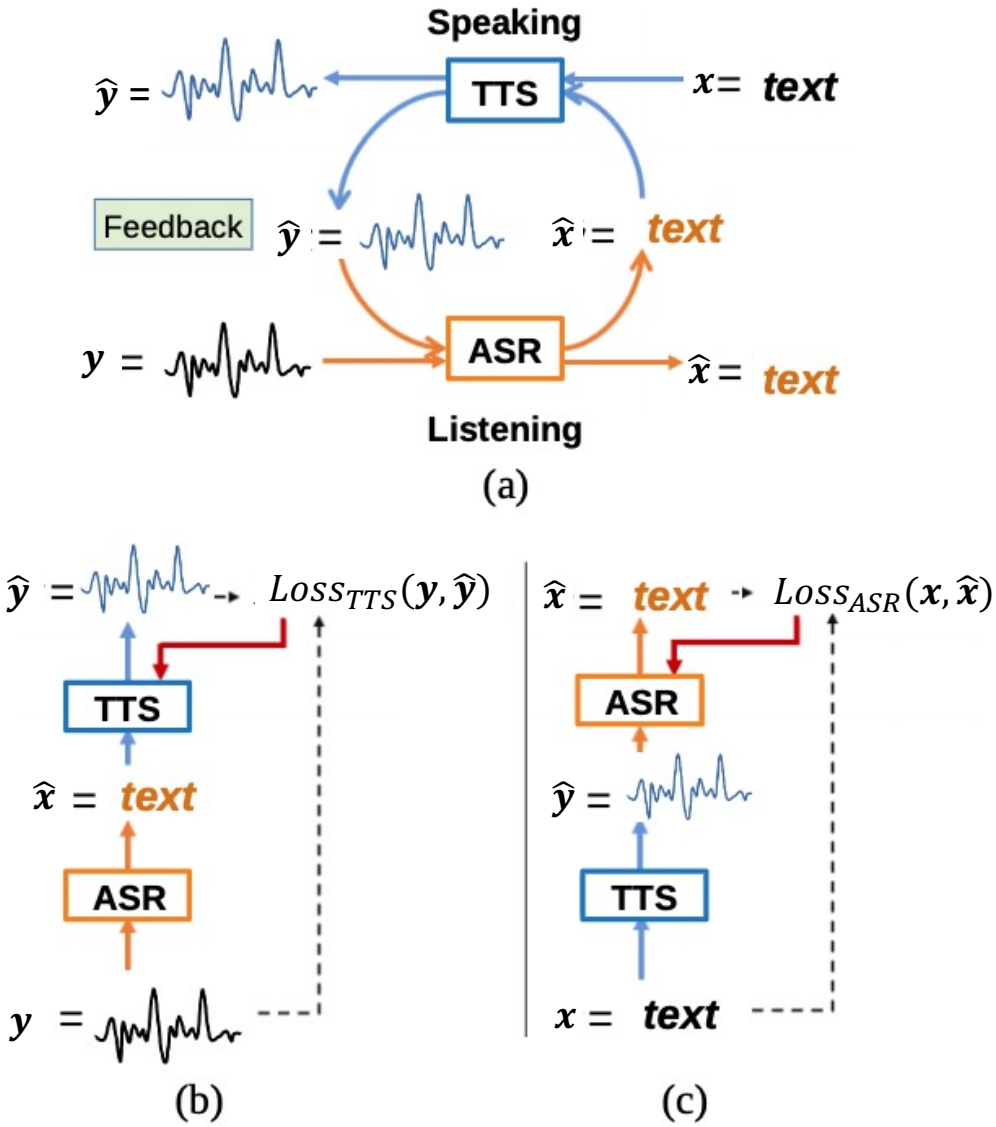


Figure 3.1: Overview of machine speech chain [2] (a). The feedback loop is unrolled into two processes: ASR-to-TTS (b) and TTS-to-ASR (c).

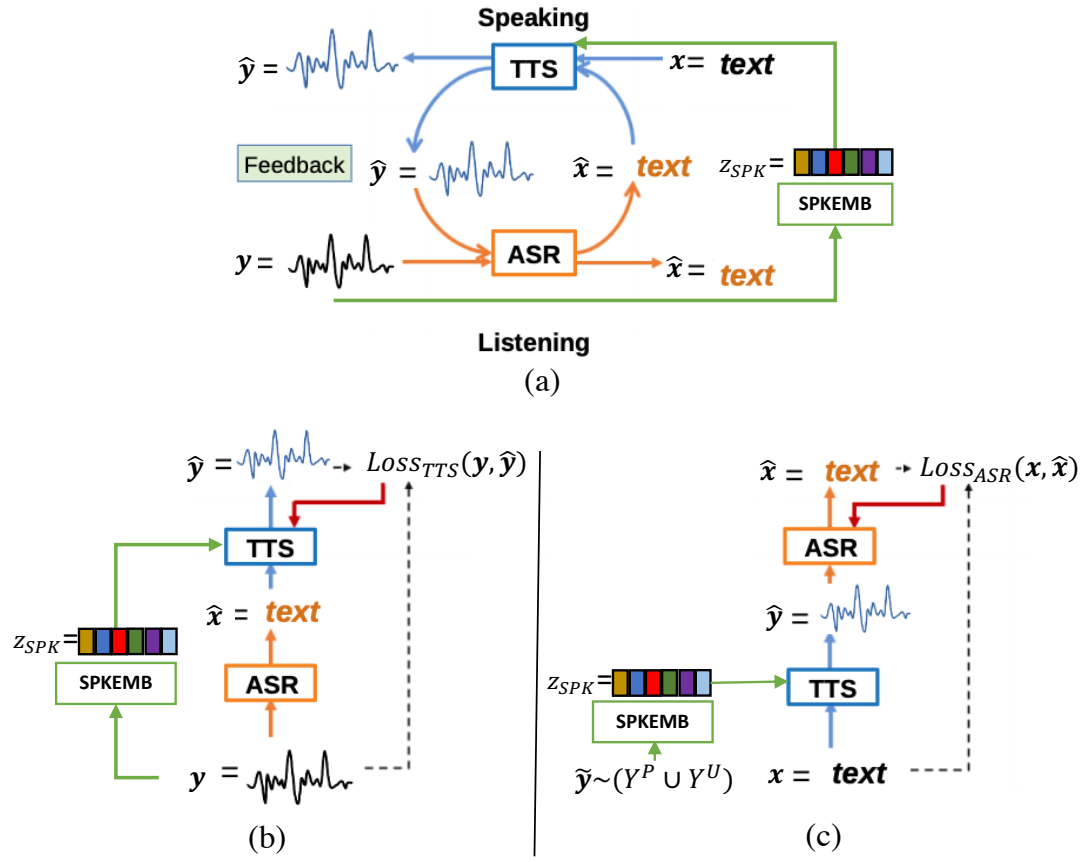


Figure 3.2: Overview of machine speech chain with speaker recognition(a). The feedback loop is unrolled into two processes: ASR-to-TTS with the speaker vector generated based on ASR speech input (b) and TTS-to-ASR with the speaker vector by sampling the available speech data (c).

At the earlier stage of framework development, machine speech chain was initially utilized for single-speaker TTS and ASR training [2] (Fig. 3.1). Then, this framework progressed for multi-speaker TTS and ASR training by extending the framework with a speaker recognition system [46] (Fig. 3.2).

Machine speech chain could be an important milestone in spoken language processing technology because a large amount of labeled data, which is necessary for neural network model training, is expensive to prepare. The success in learning from unlabeled data through a feedback loop shows that the integration of human speech perception and production behaviors improves the human-machine

interaction systems.

3.2 Architecture

The basic machine speech chain consists of seq2seq TTS and ASR. The details of the TTS, ASR, and speaker recognition modules inside the framework are followings.

3.2.1 TTS

TTS architecture follows the seq2seq TTS framework for end-to-end text-to-speech features generation. The text and speech representations utilized in TTS are the same as the representations used in ASR. Inside the closed feedback loop, TTS might synthesize the speech based on the text from the training material and also the ASR output text. Therefore, data representation uniformity between TTS and ASR is required to link these components. In multi-speaker condition, TTS also receives a speaker embedding vector as an auxiliary input in addition to the text to generate the speech with the speaker-specific characteristic. The multi-speaker TTS structure and the speaker embedding injection method may follow the structures showed in Chapter 2.3. In the original machine speech chain work, multi-speaker TTS structure was based on seq2seq RNN-based Tacotron extended with a Deep-Speaker module as the speaker recognition module for speaker embedding generation. Here, speech vocoder is not included in the feedback loop and it is trained separately.

3.2.2 ASR

ASR performs end-to-end conversion of speech features sequence into text sequence. The ASR input features are the same as TTS output, and the representation of ASR output text is also the same as TTS input. Inside the speech chain, ASR might also perform the recognition based on the features extracted from natural speech and also the direct TTS output. Unlike the speech synthesis tasks, speech transcription does not depend on speaker. Therefore, ASR structure for single-speaker and multi-speaker tasks is the same. In the original work,

ASR structure followed the seq2seq RNN-based framework.

3.2.3 Speaker Recognition

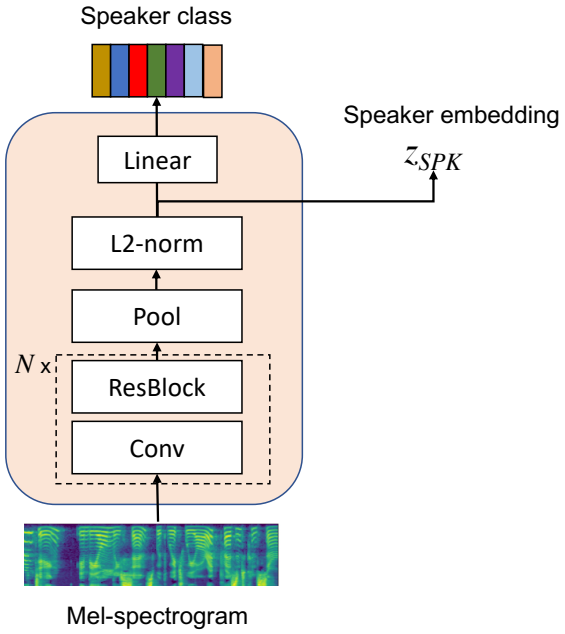


Figure 3.3: Speaker recognition and speaker embedding generation using Deep-speaker.

The speaker recognition module is implemented in TTS for machine speech chain training in multi-speaker data condition. It determines the speaker identity from the speech utterance. In the machine speech chain framework, this module is based on Deep-Speaker [49] framework that performs the speaker recognition using a deep learning architecture given the speech features, as shown in Fig. 3.3. Deep-speaker consists of stack of convolutional layers followed by a pooling operation and output the speaker posterior. The speaker embedding z_{SPK} for TTS is obtained from Deep-Speaker intermediate output vector. In the machine speech chain work, Deep-Speaker model configuration followed the same setting as the Deep-Speaker work.

3.3 Training Mechanism

The basic machine speech chain framework is trained in a semi-supervised approach consisting of supervised training and unsupervised training, which are done based on the paired or unpaired data scenario. The details of the algorithm can also be found in Alg. 1. From the supervised and unsupervised training phases, we obtain the cumulative loss and use them to optimize the TTS and ASR parameters as shown in Alg. 2.

3.3.1 Supervised Training

Supervised training serves as a knowledge initialization phase for TTS and ASR, which is done before the unsupervised training. These systems are trained independently by using the paired speech-text data that is defined as $\mathcal{D}_P = (\mathcal{Y}^P, \mathcal{X}^P)$, where $\mathcal{Y}^P$ is speech dataset with the corresponding text pair in $\mathcal{X}^P$ text dataset. The amount of the paired data is smaller than the unpaired data that is utilized in the unsupervised training phase. For example, in the original work [2] that conducted the experiment on BTEC data [50], the ratio of labeled data and unlabeled data was 1:4. TTS and ASR optimization is done using the same method as the standard supervised training (see Chapter 2). Here, given the input, each model generates the output through teacher-forcing mechanism and the loss is calculated based on the ground truth label and the model output.

In the multi-speaker tasks, TTS generates the speech by also using a speaker embedding input. The speaker embedding is extracted from the speech data that also is used as the TTS output reference.

3.3.2 Unsupervised Training

After the supervised training, ASR and TTS are trained jointly by using the unlabeled training data. Unlabeled data could be either of speech-only data ($\mathcal{Y}^U$) without the transcription or text-only data ($\mathcal{X}^U$) without the corresponding speech utterance. During unsupervised training, ASR and TTS support each other by doing feedback passing though a loop that connects them. The loop between ASR and TTS consists of two unrolled processes: ASR-to-TTS and

Algorithm 1 Speech Chain Algorithm (part 1)

Input:Paired speech and text dataset $\mathcal{D}^P$, text-only dataset $\mathcal{X}^U$, speech-only dataset $\mathcal{Y}^U$, supervised loss coefficient α, unsupervised loss coefficient β

REPEAT ...

A. Supervised training with speech-text data pairs

Sample paired speech and text $(\boldsymbol{y}^P, \boldsymbol{x}^P) = ([y_1^P, .., y_{T_P}^P], [x_1^P, .., x_{S_P}^P])$ from $\mathcal{D}^P$ with speech length T_P and text length S_P.

Generate a text probability vector by ASR using teacher forcing:

$p_{x_s} = p(x_s|\boldsymbol{y}^P, x_{<s}^P; \theta_{ASR}), \forall s \in [1..S_P]$

Generate best predicted speech by TTS using teacher forcing:

$\hat{y}_t^P = p(z|\boldsymbol{x}^P, y_{<t}^P; \theta_{TTS}); \forall t \in [1..T_P]$

Calculate the loss for ASR and TTS ▷ Eq. 2.15 & 2.17

$$Loss_{ASR}^P = Loss_{ASR}(\boldsymbol{x}^P, \boldsymbol{p}_x; \theta_{ASR}) \tag{3.1}$$

$$Loss_{TTS}^P = Loss_{TTS}(\boldsymbol{y}^P, \hat{\boldsymbol{y}}^P; \theta_{TTS}) \tag{3.2}$$

B. Unsupervised training with unpaired speech and text

 # Unpaired speech data (ASR-to-TTS):

Sample speech $\boldsymbol{y}^U = [y_1^U, .., y_{T_U}^U]$ from $\mathcal{Y}^U$

Generate text by ASR: $\hat{\boldsymbol{y}}^U \sim p_{ASR}(\cdot|\boldsymbol{y}^U; \theta_{ASR})$

Generate speech by TTS from ASR's predicted text using teacher forcing: $\hat{y}_t^U = p_{TTS}(z|\boldsymbol{y}_{<t}^U, \hat{\boldsymbol{x}}^U; \theta_{TTS}), \quad \forall t \in [1..T]$

Calculate the loss between original speech $\boldsymbol{y}^U$ and generated speech $\hat{\boldsymbol{y}}^U$

$$Loss_{TTS}^U = Loss_{TTS}(\boldsymbol{y}^U, \hat{\boldsymbol{y}}^U; \theta_{TTS}) \tag{3.3}$$

 # Unpaired text data (TTS-to-ASR):

Sample text $\boldsymbol{x}^U = [x_1^U, .., x_{S_U}^U]$ from $\mathcal{X}^U$

Generate speech by TTS: $\hat{\boldsymbol{y}}^U \sim p_{TTS}(\cdot|\boldsymbol{x}^U; \theta_{TTS})$

Generate text probability vector by ASR from TTS's predicted speech using teacher forcing: $p_{x_s} = p(x_s|\hat{\boldsymbol{y}}^U, x_{<s}^U; \theta_{ASR}), \quad \forall s \in [1..S_U]$

Calculate the loss between original text $\boldsymbol{x}^U$ and reconstruction probability $\boldsymbol{p}_x$

$$Loss_{ASR}^U = Loss_{ASR}(\boldsymbol{x}^U, \boldsymbol{p}_x; \theta_{ASR}) \tag{3.4}$$

Algorithm 2 Speech Chain Algorithm (part 2)

19: ***# Loss combination:***

20: Combine all weighted loss into a single loss variable

$$Loss_{ALL} = \alpha * (Loss^{P}_{TTS} + Loss^{P}_{ASR}) + \beta * (Loss^{U}_{TTS} + Loss^{U}_{ASR}) \quad (3.5)$$

Calculate TTS and ASR parameters gradient with

21: the derivative of $Loss_{ALL}$ w.r.t $\theta_{ASR}, \theta_{TTS}$

$$G_{ASR} = \nabla_{\theta_{ASR}} Loss \quad (3.6)$$

$$G_{TTS} = \nabla_{\theta_{TTS}} Loss \quad (3.7)$$

Update TTS and ASR parameters with gradient descent

22: optimization (SGD, Adam, etc)

$$\theta_{ASR} \leftarrow Optim(\theta_{ASR}, G_{ASR}) \quad (3.8)$$

$$\theta_{TTS} \leftarrow Optim(\theta_{TTS}, G_{TTS}) \quad (3.9)$$

23: **UNTIL** convergence of parameter $\theta_{TTS}, \theta_{ASR}$

TTS-to-ASR, where each process can be considered as auto-encoding process.

3.3.2.1 ASR-to-TTS

ASR-to-TTS process in Fig. 3.1 (b) is done by using speech data only. Here, given a speech utterance $\boldsymbol{y}^U$ from $\mathcal{Y}^U$, ASR generates the transcription $\hat{\boldsymbol{x}}^U$ through beam or greedy decoding strategy. After that, TTS synthesize a speech $\hat{\boldsymbol{y}}^U$ from $\hat{\boldsymbol{x}}^U$. A speech reconstruction loss between $\boldsymbol{y}^U$ and $\hat{\boldsymbol{y}}^U$ will be computed using TTS loss function in Eq. 2.15. In multi-speaker tasks (Fig. 3.2 (b)), speaker embedding for TTS is generated based on the ASR speech input $\boldsymbol{y}^U$ by using the speaker recognition module.

3.3.2.2 TTS-to-ASR

This process is done by using text data $\mathcal{X}^U$ only. As shown in Fig. 3.1 (c), first, TTS synthesize a speech $\hat{\boldsymbol{y}}^U$ through greedy decoding strategy given the text sentence $\boldsymbol{x}^U$ from training material $\mathcal{X}^U$. From the synthesized speech, ASR is performed to produce the transcription $\hat{\boldsymbol{x}}^U$. Here the text reconstruction loss is calculated by comparing the original text $\boldsymbol{x}^U$ and ASR output text $\hat{\boldsymbol{x}}^U$ through ASR loss function in Eq. 2.17.

In multi-speaker tasks (Fig. 3.2 (c)), speaker embedding is generated by, first, sampling a speech from the available speech data set $\tilde{\boldsymbol{x}} \sim (\mathcal{Y}^P \bigcup \mathcal{Y}^U)$ and generate a random speaker embedding vector based on the sampled speech $\tilde{\boldsymbol{x}}$.

Chapter 4

Proposed Incremental Machine Speech Chain Training Mechanism

4.1 Overview

Previously, a machine speech chain framework based on the human speech chain mechanism was proposed. The basic framework enables semi-supervised training of seq2seq TTS and ASR through a closed feedback connection, in which those components support each other when learning from the unpaired speech-only or text-only data. Thus, this loop allows the machine to learn not only to listen or speak but also to listen while speaking.

Although the machine speech chain was able to improve TTS and ASR performances, those components require long latencies to produce the output, similar to conventional systems. Due to the global attention mechanism inside them, they have to wait for a complete input sequence to generate the output sequence. ASR starts recognition after receiving a complete speech utterance from TTS, and TTS begins its synthesis after receiving a complete sentence from ASR. As a result, there is a significant delay when encountering long utterances.

Prediction latency could be reduced by replacing the ASR and TTS with the incremental systems: ISR and ITTS. In their mechanisms, ISR and ITTS use the segment-by-segment prediction approach. Therefore, their output can

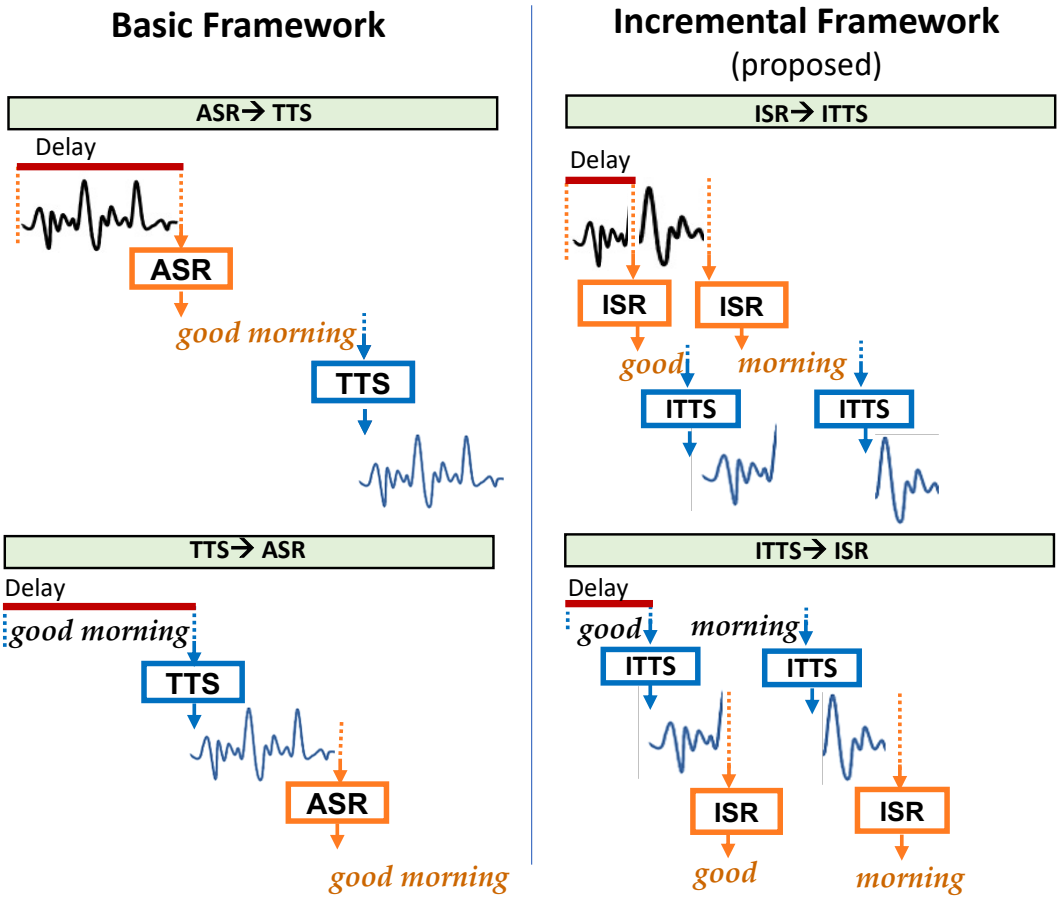


Figure 4.1: An overview of the comparison between the basic machine speech chain and the proposed incremental machine speech chain.

be given without waiting for the complete input sequence. However, low-latency processing could harm the system's performance because the output has to be produced based on the short and limited input.

In this chapter, we propose an incremental machine speech chain mechanism to improve the learning quality of end-to-end ITTS and ISR through a short-term closed loop, which is shown in Fig. 4.1. The proposed mechanism also aims to enable real-time feedback generation during inference. Although the proposed framework only uses the feedback explicitly during training, by enabling the real-time feedback generation, we can move a step closer to achieving a TTS or ASR that can adapt simultaneously to the environment unsupervisedly, similar

to humans.

4.2 Related Works

Previously, several frameworks were proposed for a low-latency ASR and also TTS. Low-latency ASR can be done using a conventional approach with the HMM and hybrid systems [51, 52, 53, 54]. However, the HMM-based ASR cannot perform end-to-end recognition, whereas the current state-of-the-art approach is deep learning. Recently, end-to-end incremental or streaming ASR with a low recognition latency has gained attention in the speech community. Jaitly et al. [55] might be the first group that has proposed a neural transducer framework that can recognize speech segment-by-segment with a fixed window. Another study has investigated attention-transfer ISR (AT-ISR) [56, 57], which learns from attention-based non-incremental ASR for end-to-end speech recognition with a low latency. Other works have also proposed ISR in neural transducer [58, 59] and Transformer [58, 60] neural network structures.

Developing ITTS is also very challenging; the standard framework commonly requires language-dependent contextual linguistics of a full sentence to produce a natural-sounding speech waveform. Existing ITTS studies have mainly been conducted on a model based on HMM [61, 62, 63, 64]. The first study that attempted to synthesize speech in real-time using neural ITTS was proposed by Yanagita et al. [65]. Recently, another ITTS was proposed based on a prefix-to-prefix framework [66].

The previously published works were only concerned with ITTS and ISR tasks individually. By contrast, this thesis investigates the joint incremental learning between ITTS and ISR that attempts to mimic the human speech chain. The idea here is to train well-performing low-latency systems by evaluating the short-term output of a system using another system and jointly improving both of them. The proposed mechanism is based on the basic machine speech chain shown in Chapter 3 and the incremental steps during supervised training are learned through attention transfer [56]. In the experiment, we performed the inference process with separate ITTS and ISR, and also inference with connected ITTS and ISR to do feedback generation.

4.3 Incremental Machine Speech Chain

4.3.1 Architecture

The proposed incremental machine speech chain consists of ITTS and ISR, which are connected through a feedback loop during training. The structure of ITTS and ISR are based on the seq2seq RNN framework with the same setting as the standard TTS and ASR utilized in the basic machine speech chain. Here, the main difference is that, in the incremental system, the task is done by performing segment-by-segment processing. To produce a complete input, ITTS or ISR will first take the first segment of input and then produce a segment of output. The same process is then repeated on the next input segment to generate the next output segment, and so on.

4.3.1.1 Incremental ASR (ISR)

We use a block-wise seq2seq ISR that recognizes speech by transcribing it segment-by-segment with a fixed window length [67, 56]. ISR predicts sentence text $\hat{\boldsymbol{x}}$ with length of S from a full speech utterance $\boldsymbol{y}$ with length of T in N recognition steps, as shown in Fig. 4.2. This is done by first dividing the speech $\boldsymbol{y}$ into N segments, denoted as $\boldsymbol{y} = [\boldsymbol{y}^1, ..., \boldsymbol{y}^N]$, where the length of $\boldsymbol{y}^n$ is W frames and $W < T$. For this factor, ISR's latency is equal to W frames.

The recognition procedure for each recognition step $n = [1, ..., N]$, where $N = \frac{T}{W}$, is below:

1. Encode $\boldsymbol{y}^n$.
2. Decode and predict $\hat{\boldsymbol{x}}^n$, a segment of K_n text tokens from $\hat{\boldsymbol{x}}$, where $0 \leq K_n < S$, until an *end-of-block* token, denoted </m>, is predicted by attending encoder states from $\boldsymbol{y}^n$.
3. Shift the input window W frames and keep the model states.

The prediction steps are repeated until an EOS token is predicted. We use AT-ISR in our incremental machine speech chain to limit ISR construction complexity

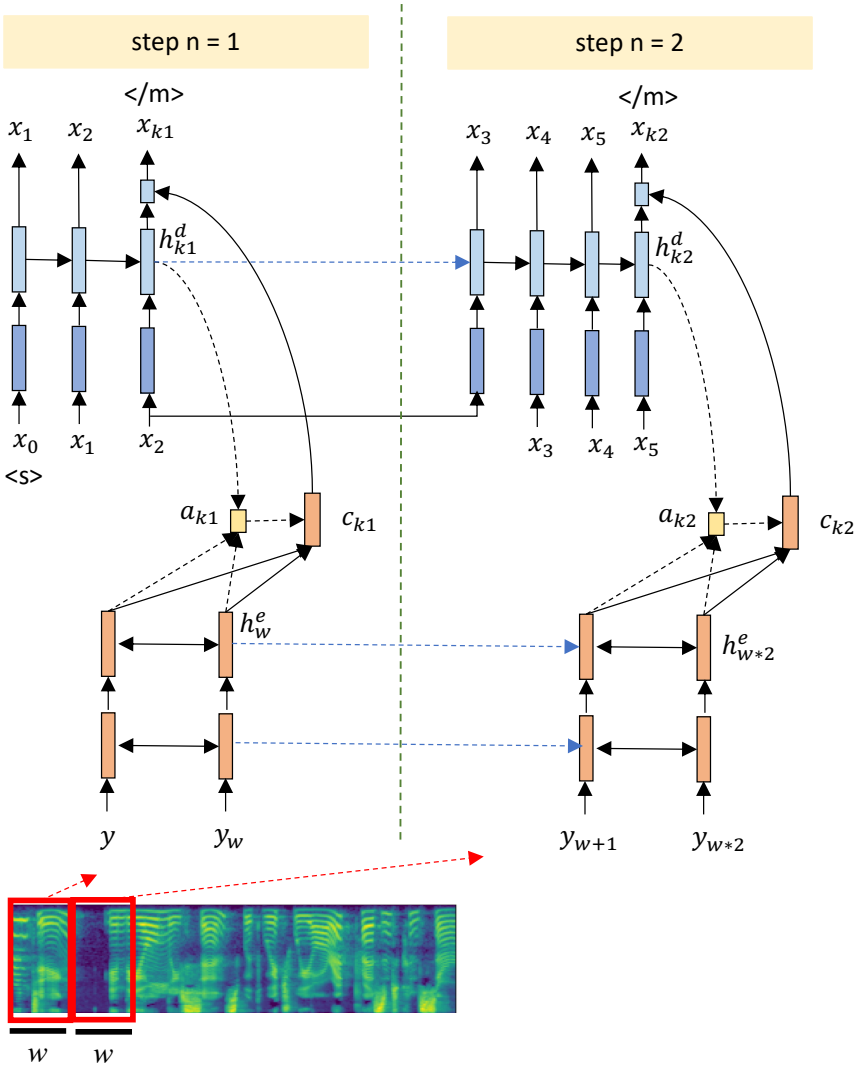


Figure 4.2: Incremental speech recognition.

while maintaining the recognition performance. Attention transfer teaches AT-ISR, a student model, to mimic the alignment from a teacher model or non-incremental ASR that provides $\boldsymbol{y}^n$-$\boldsymbol{x}^n$ pairs based on the attention alignment. AT-ISR learns $\boldsymbol{x}^n$ and an *end-of-block* token as the output target of $\boldsymbol{y}^n$. All $\boldsymbol{y}^n$ lengths are uniform (W) in the attention-based alignment, but the length of each text segment $\boldsymbol{x}^n$ can vary.

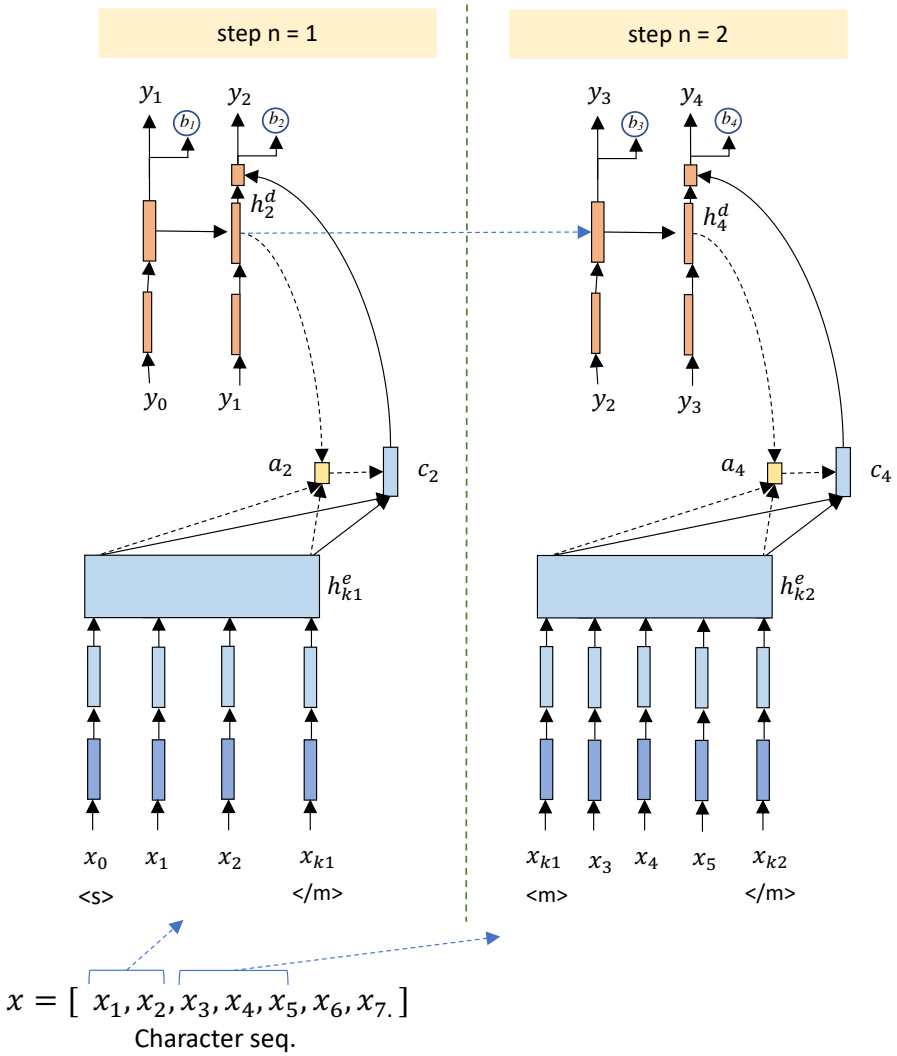


Figure 4.3: Incremental speech synthesis.

4.3.1.2 Incremental TTS (ITTS)

Seq2seq ITTS (Fig. 4.3) performs speech generation without waiting for a complete sentence text input [65, 66]. We construct ITTS using attention transfer from non-incremental ASR, in which ITTS is trained using pairs of speech and text segments. We apply the same alignment for ITTS and ISR in the incremental machine speech to reduce incompatibility between ITTS and ISR incremental units during the joint training. Here, non-incremental ASR provides attention-based alignment between fixed-size speech segments and variable-length text segments. ITTS, with the attention transfer from non-incremental ASR, learns how

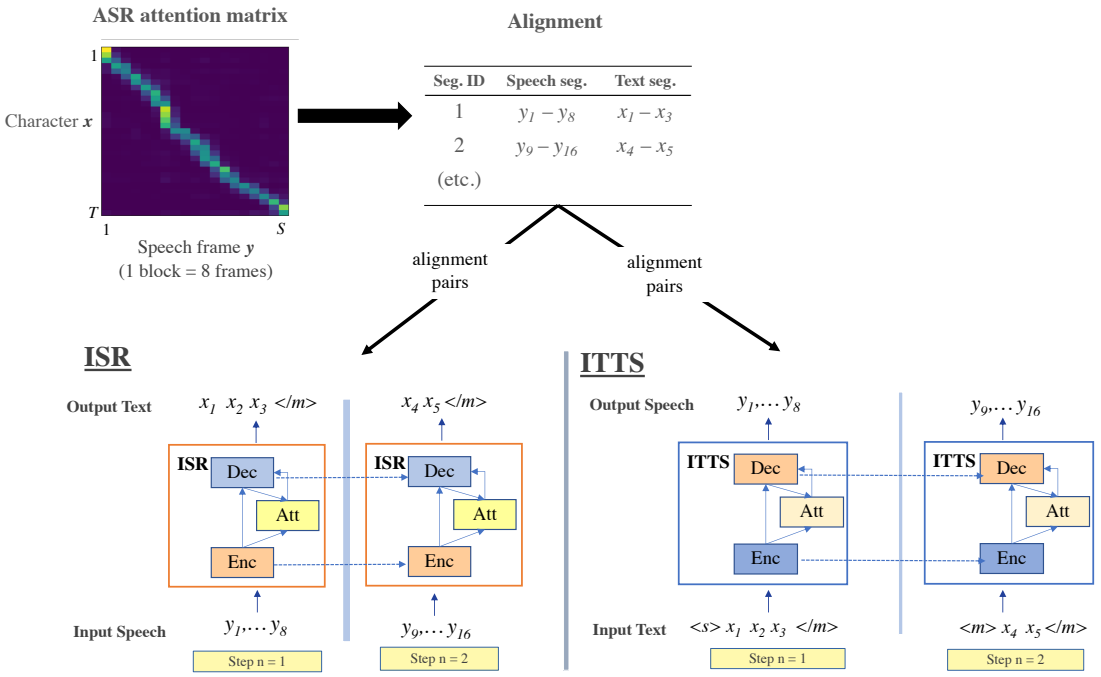


Figure 4.4: Supervised ITTS and ISR training via attention transfer.

to process a variable-length token sequence to produce at least W speech frames, given the length of a speech segment in the attention-based alignment during training is W. The ITTS output length here is at least W frames because, during training, we combine subsequent speech segments that did not align with any token (K_n=0) with the neighboring segment that aligns with an output token.

Speech generation with our ITTS follows the following procedure in each step $n = [1, ..., N]$:

1. Encode $\boldsymbol{x}^n$, a segment of K_n tokens from token sequence $\boldsymbol{x}$, where $1 \leq K_n < S$.

2. Decode and predict $\hat{\boldsymbol{y}}^n$, a segment of W_n speech frames from speech utterance $\hat{\boldsymbol{y}}$, where $W \leq W_n < T$, by attending $\boldsymbol{x}^n$ until a stop flag is predicted.

3. Shift the input window K_n tokens and keep the model states.

4.3.2 Training Mechanism

4.3.2.1 ITTS and ISR Independent Training

ISR and ITTS are trained independently by using paired speech-text data. We apply the attention transfer training mechanism to train ISR and ITTS with the non-incremental ASR as the teacher (Fig. 4.4). Both of the incremental systems are trained using the same training data as the teacher model.

4.3.2.2 ITTS and ISR Joint Training Through Short-term Closed Feedback Loop

ISR and ITTS support each other to jointly improve themselves by establishing a short-term closed-loop. Here, each time the first component finishes an incremental step, it passes the output to the second component. The second component then processes the passed data in an incremental step. The closed-loop between ISR and ITTS is unrolled into the following processes:

- **ISR-to-ITTS**.

  In each recognition step n, ISR processes a speech segment $\boldsymbol{y}^n$ and generates a text segment $\hat{\boldsymbol{x}}^n$, which is then encoded by ITTS to generate a reconstructed speech segment $\hat{\boldsymbol{y}}^n$ (Fig. 4.5). Model parameters are updated using the average ITTS loss from each incremental step, as formulated in Eq. 4.1.

$$Loss_{ITTS} = \frac{1}{N}\sum_{n=1}^{N} Loss_{TTS}(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n) \tag{4.1}$$

- **ITTS-to-ISR**.

  For each step n, ITTS firstly synthesizes a speech segment $\hat{\boldsymbol{y}}^n$ by taking a text segment input $\boldsymbol{x}^n$. The ITTS output $\hat{\boldsymbol{y}}^n$ is then transcribed by ISR to reproduce the ITTS input text $\hat{\boldsymbol{x}}^n$ (Fig. 4.6). Training loss is calculated between $\boldsymbol{x}^n$ and $\hat{\boldsymbol{x}}^n$ pair, and then averaged

$$Loss_{ISR} = \frac{1}{N}\sum_{n=1}^{N} Loss_{ASR}(\boldsymbol{x}^n, \hat{\boldsymbol{x}}^n). \tag{4.2}$$

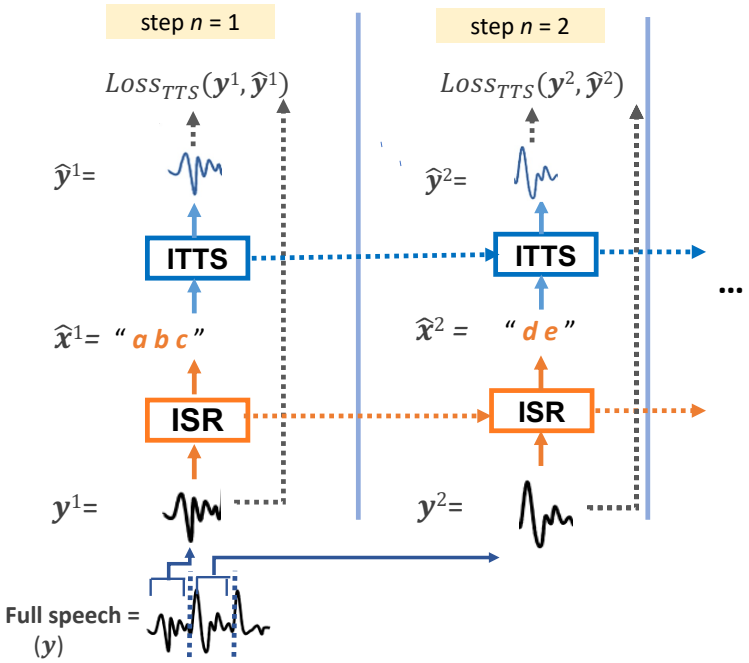


Figure 4.5: ISR-to-ITTS.

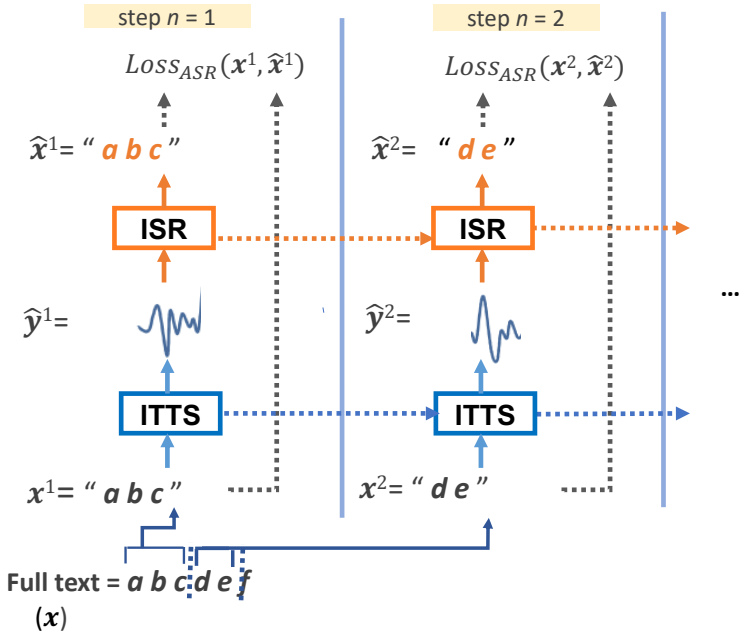
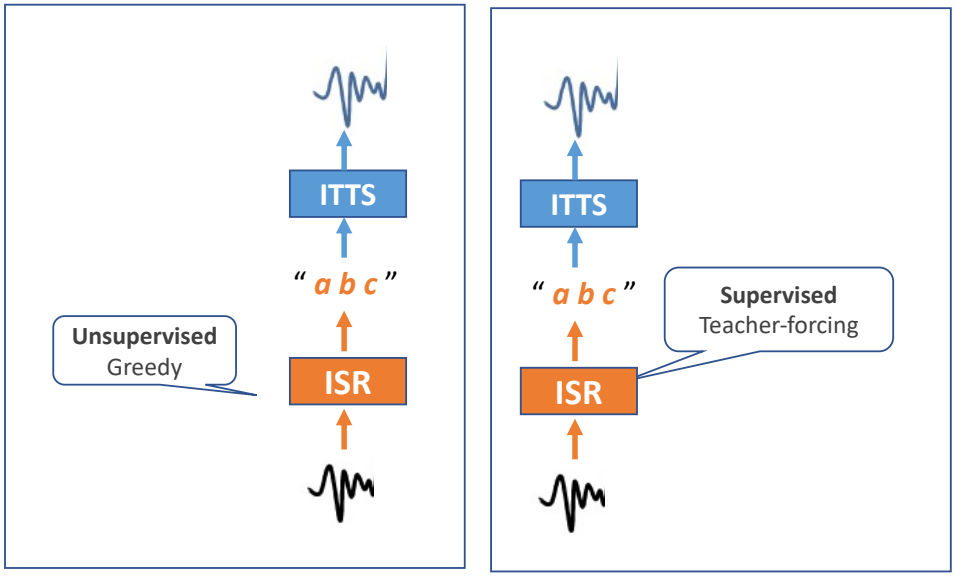


Figure 4.6: ITTS-to-ISR.



(a) (b)

Figure 4.7: An example of an unrolled feedback loop in (a) the unsupervised chain and (b) the supervised chain.

4.3.2.3 Learning Approach in Supervised and Unsupervised Chain

The basic machine speech chain framework was originally proposed for semi-supervised learning by combining the supervised and unsupervised training steps. In the unsupervised training phase with a closed-feedback loop, the intermediate output in an unrolled feedback loop process is generated through a greedy decoding mechanism.

In this thesis, as our focus is not on semi-supervised learning, we explore two approaches for intermediate output generation during joint training via closed-loop: 1) unsupervised chain via greedy approach and 2) supervised chain via teacher-forcing approach. In the unsupervised chain approach shown in Fig. 4.7 (a), the intermediate output is generated by the first component through greedy or beam-decoding decoding based on unpaired data. On the other hand, the supervised chain in Fig. 4.7 (b) generates the intermediate output using teacher-forcing decoding. The supervised chain training is done using paired speech-text data.

4.3.3 Inference Mechanism

ITTS and ISR inference are done separately without the feedback loop. ITTS synthesizes speech segment-by-segment by using the procedure in Chapter 4.3.1.2. Here the decoding is done using greedy searching. In our experiment, the ITTS input text during inference was segmented based on a text segment dictionary constructed from the training data. On the other hand, ISR also recognizes speech segment-by-segment with the procedure in Chapter 4.3.1.1. The size of the speech segments is fixed, with the same configuration as in the training phase. Decoding is done using greedy searching to reduce the computational latency and to keep output consistency.

4.4 Experimental Setup

4.4.1 Dataset

We used the Wall Street Journal (WSJ) [68] dataset for ITTS and ISR construction with the following settings: *SI-84*, *SI-200*, and *SI-284* as the training sets, *dev93* as the development set, and *eval92* as the test set. The *SI-84* set consisted of 16 hours of speech by 83 speakers, and the *SI-200* set consisted of 66 hours of speech by 200 speakers that did not overlap with *SI84* set. The *SI-284* set was a combination of *SI-84* and *SI-200* sets. The *SI-84* and *SI-284* were utilized to train the ITTS and ISR during independent training, while the *SI-200* set was utilized for systems joint training with a closed-loop. All speech utterances had a sampling rate of 16 kHz. For the ISR input and ITTS output, speech utterances were represented as 80-dimension Mel-spectrograms, where each feature frame had a length of 50 ms and was shifted by 12.5 ms from the previous frame. The text was represented as a sequence of character units.

4.4.2 Model Configuration

4.4.2.1 ITTS

Our TTS followed the TTS structure in previous machine speech chain work [2], which was a modification of TTS Tacotron [23]. The model hyperparameters

were generally the same as those in the original Tacotron. The modification was made by replacing the rectified linear unit (ReLU) function with the leaky ReLU (LReLU) function. The CBHG module used $K = 8$ filter banks. The decoder consisted of two LSTM (256 units) layers. Our TTS generated 4 consecutive frames for each decoding step, thus reducing the number of total decoding steps.

We allow ITTS to take contextual inputs, which consisted of look-back and look-ahead blocks [56, 65, 66, 67], which are the blocks before and after the main segment, respectively, to enrich the information in the main input segment. The ITTS input segment in an incremental step consisted of the main character block with two look-back and four look-ahead character blocks. The ITTS main input size range was between one and four blocks, with an average of two blocks. One text character block consisted of five characters, the average word length in the training data.

4.4.2.2 ISR

The seq2seq model structures of non-incremental ASR and ISR were identical. The encoder consisted of a FNN layer (512 units) that was followed by three BiLSTM layers (256 units each). Each BiLSTM layer applied hierarchical sub-sampling [48, 47]. As a result, an encoder state in the encoder's final layer represented eight speech frames. Here we defined eight speech frames (0.14 sec) as a speech block. The ASR decoder consisted of a character embedding layer (256-dims), an LSTM layer (512 units) with an attention mechanism, and a softmax layer. We applied an MLP-scoring function that used a previously proposed multi-scale alignment and contextual history [69] in the attention component. The text generation during inference was done by greedy-decoding to prevent an additional delay.

Similar to ITTS, ISR input also included the contextual input blocks. The ISR input segment for an incremental step consisted of four main speech blocks with two look-back and four look-ahead blocks, which we decided based on optimum latency configuration in previous AT-ISR work [56].

4.4.2.3 Speaker Recognition

Our TTS implemented a speaker recognition component similar to that in previous machine speech chain work [46]. Here we used a Deep-Speaker model with the same hyperparameters as in the previous machine speech chain research. The speaker recognition component was trained using *SI-84* set.

4.4.3 Evaluation Metrics

In this experiment, ITTS was evaluated using L2 loss of the Mel-spectrogram and the ISR was evaluated using character error rate (CER):

- **Mel-spectrogram L2 loss**
  The L2 loss on Mel-spectrogram was used to compare the similarity of natural $\boldsymbol{y}$ and synthesized $\hat{\boldsymbol{y}}$ speech utterances with the same transcription and length. The loss calculation follows

$$Loss_{L2}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{T}\sum_{t=1}^{T}((y_t - \hat{y}_t)^2. \tag{4.3}$$

- **Character error rate**
  CER is the minimum number of edits in the hypothesis that are required to make the hypothesis exactly match the reference. CER equals the character-level edit distance that follows Eq. 4.4.

$$Edit\ Distance = \frac{Sub + Del + Ins}{N_{ref}} \times 100\%. \tag{4.4}$$

  In the CER calculation, Sub, Del, and Ins represent the number of character substitutions, deletions, and insertions required to correct the hypothesis, respectively, and N_{ref} represents the number of characters in the reference text. The ISR CER was measured by comparing the model's complete output sequence against the full reference transcription.

Table 4.1: Performances of ASR (CER (%)) and TTS (L2-norm^2 between ground truth and predicted Mel spectrogram).

Data	**ASR (CER(%))**				**TTS (L2-norm^2)**			
	Non-incr (latency: 7.88 sec)		**Incr** (latency: 0.84 sec)		**Non-incr** (latency: 103 char.)		**Incr** (latency: 30 char.)	
	nat-sp	**syn-sp**	**nat-sp**	**syn-sp**	**nat-txt**	**rec-txt**	**nat-txt**	**rec-txt**
ASR and TTS with independent training								
indep-trn (*SI-84*)	17.33	27.03	17.81	44.54	0.99	1.02	1.04	3.62
indep-trn (*SI-284*)	7.16	9.60	7.97	19.99	0.75	0.77	0.84	1.31
ASR and TTS with machine speech chain								
indep-trn (*SI-84*) + chain-unsup (*SI-200*)	11.21	11.52	14.23	32.43	0.80	0.82	0.86	1.35
indep-trn (*SI-84*) + chain-sup (*SI-200*)	7.27	6.30	9.43	12.78	0.77	0.80	0.79	1.26

4.5 Experiment Results

Our experiment results can be seen in Table 4.1. In this table, *Non-incr* denotes the standard non-incremental system, *Incr* denotes the incremental systems, *nat-sp* denotes natural speech input, *nat-txt* correct text input, *syn-sp* denotes TTS speech as the ASR input, and *rec-txt* denotes ASR text as TTS input. Here *indep-trn* is independent training, *chain-unsup* is joint training with an unsupervised chain, and *chain-sup* is joint training with a supervised chain.

In this experiment, the baselines are ISR and ITTS that were trained independently using *SI-84* set and the toplines are the systems that were trained independently with *SI-284* set. The machine speech chain mechanism for the non-incremental systems followed the basic mechanism (see Chapter 3), while the incremental systems followed the incremental mechanism (see Chapter 4.3). System evaluation was done based on natural and synthetic inputs. The synthetic input was generated by the target system's counterpart system, which was trained under the same training condition by processing a natural input. For the incremental system, the synthetic input of an incremental step was the output from the processing of a short segment of natural input. We can consider the output of synthetic input processing as the feedback for the system that produced the synthetic input.

4.5.1 ITTS Performance

The proposed incremental machine speech chain improved the ITTS performance on the correct text input. The ITTS model was trained by firstly pre-training it independently using *SI-84* dataset, where the loss in that phase was 1.04. By using the proposed mechanism, the loss could be decreased to 0.86 using the unsupervised chain and to 0.79 using the supervised chain. The ITTS with the supervised chain also outperformed the ITTS trained supervised without the speech chain using the full *SI-284* set. In this case, ITTS performed similarly to non-incremental TTS too, with the exception that the character-level latency in ITTS was shorter. An improvement was also observed when the ITTS input was the ISR output text. This might show that ITTS was able to understand the ISR output better than the model without the feedback-based training. From this,

we could expect an improvement as well in the ISR.

4.5.2 ISR Performance

The proposed incremental machine speech chain training improved ISR performance as well. CER of the initial ISR (baseline) after it was trained independently with *SI-84* data was 17.81%. Given a natural input, the CER could be reduced into 14.23% and 9.43% using unsupervised and supervised chains, respectively, in the incremental speech chain training. But the topline ISR still performed the best when the input was natural speech. The proposed ISR learned from synthetic speech input during the joint training, whereas there is a gap between natural and synthetic speech. Therefore, the proposed method resulted in an improvement in synthetic speech recognition, outperforming the baseline and also the topline ISR.

4.6 Summary

The incremental machine speech chain training framework successfully reduces the systems latency in the machine speech chain and improves ITTS and ISR performances. The improvement of ITTS and ISR occurred in both natural and synthetic input processing. It shows that the short-term feedback loop between the incremental systems is able to leverage their training quality. We also demonstrated real-time feedback generation. Although the feedback is not yet utilized in the inference to improve the system performance, the quality of feedback processing and understanding could be critical in the feedback-based inference, which is our next task. This could be an important step towards achieving a system that can listen while speaking in real-time. The resulting systems can be used in incremental processing tasks, for example, ISR and ITTS for a simultaneous speech translation system.

Chapter 5

Proposed Self-adaptive Machine Speech Chain in Noisy Environment

5.1 Overview

Machine self-adaptation to the real environment requires the machine to be aware of the situation and their performance. TTS systems have been developed to mimic human speech production. But unlike the human mechanism, which considers speech production and perception connections important, TTS development focuses only on speech production without considering to perceive and evaluate the synthesized speech and the environment. Under clean conditions, neural TTS successfully synthesizes highly natural speech given only the text [23, 26, 25, 70]. However, in noisy conditions, TTS speech intelligibility degrades because most systems have not been designed to handle noisy environments. Furthermore, since TTS only learns to speak without listening to and understanding the situation, they cannot adapt to the situation. A widely used solution for achieving TTS with high intelligibility in noisy places is to adapt the system offline using Lombard speech from a particular noisy condition [38, 39].

In contrast to machines, humans can improve their speech when it is necessary during the speech chain process. Particularly in a noisy environment, speakers tend to change their speaking effort to increase their speech audibility while

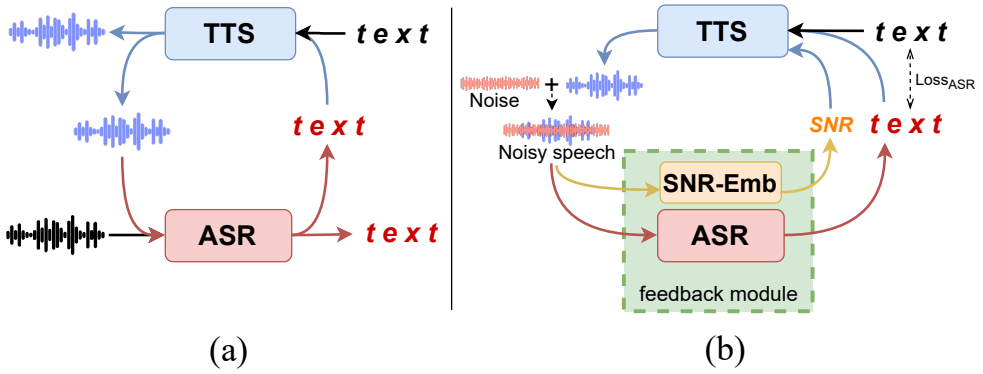


Figure 5.1: (a) Basic machine speech chain semi-supervised training; (b) proposed machine speech chain for training and dynamically adaptive inference.

simultaneously listening to their speech and the noise [12]. This change, which is known as the Lombard effect [11], includes not only a change in speech intensity but also changes in speech pitch and speed [13].

Previously, the basic machine speech chain in Fig. 5.1 (a) for semi-supervised TTS and ASR training was proposed, which was inspired by the human speech chain mechanism. However, the basic machine speech chain was only utilized as a semi-supervised training method for TTS and ASR. During inference, TTS and ASR are still performed separately as in the conventional manner, so they could still not adapt based on the actual condition dynamically, unlike the human speech chain.

In this chapter, we propose an advanced version of a machine speech chain that utilizes a feedback mechanism not only during training but also during inference. Simulating the Lombard effect, we implemented a machine speech chain for a self-adaptive end-to-end neural TTS in noisy environments (Fig. 5.1 (b)) that enables TTS to speak in Lombard effect in noisy conditions given the auditory feedback. The auditory feedback for our TTS includes the ASR loss as a speech intelligibility measurement and the SNR prediction as a power measurement. Based on feedback, the proposed TTS will generate speech while adapting the speech prosody, focusing on pitch, intensity, and speed to improve the overall speech quality.

Machine speech chain inference mechanism can be applied to improve the

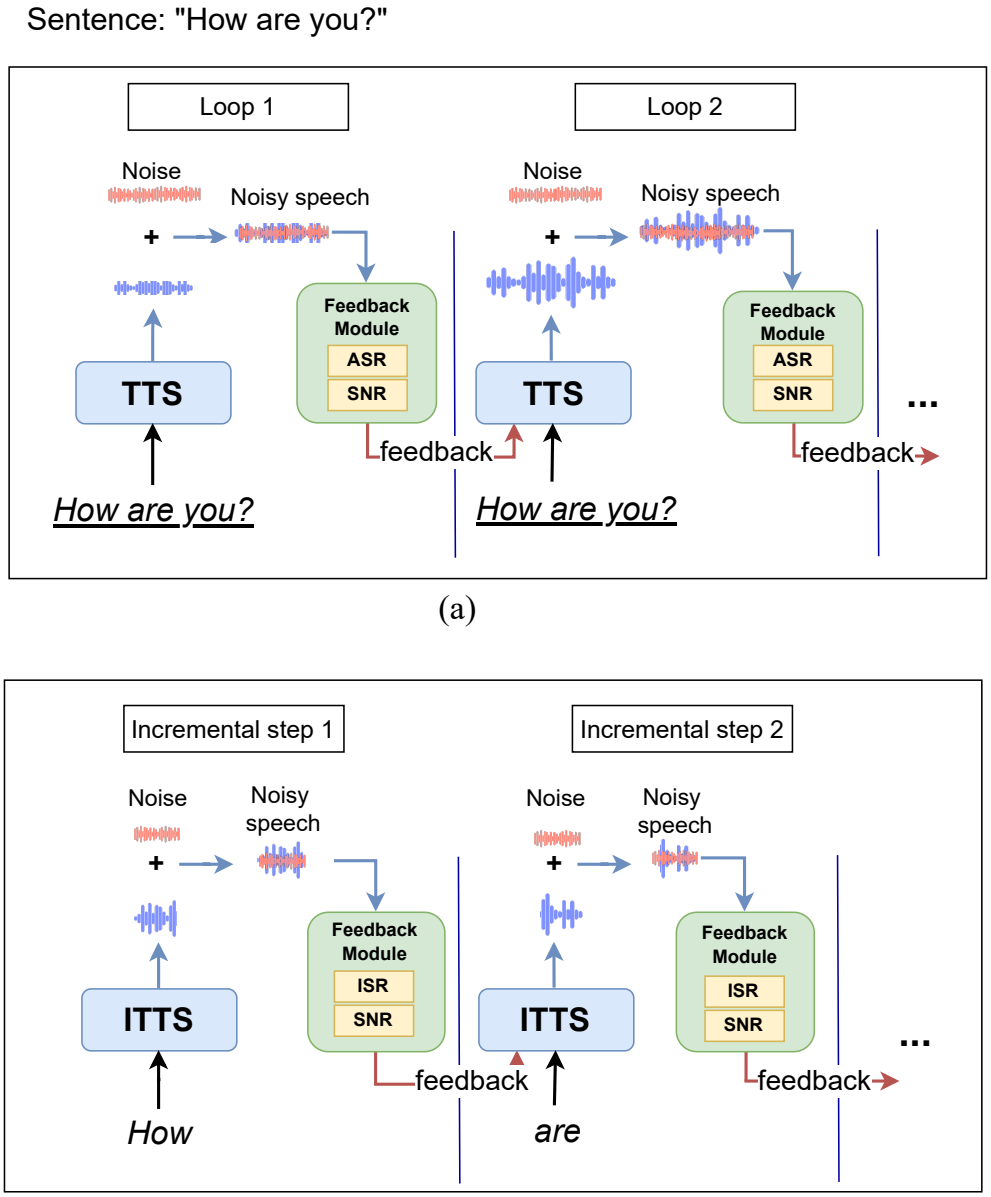


Figure 5.2: Unrolled machine speech chain inference mechanism for (a) non-incremental TTS with the utterance-level feedback and (b) ITTS with the short-term feedback.

speech of the same text or to make the speech of the next text better than the current one. In this thesis, we consider two approaches for TTS with an auditory feedback mechanism. As shown in Fig. 5.2 (a), the first approach is non-incremental TTS with utterance-level feedback in the re-speaking mechanism. The second approach is the ITTS with short-term feedback for progressive adaptation, as shown in Fig. 5.2 (b). This chapter focuses on the first method with the non-incremental TTS and the utterance-level feedback. ITTS is discussed in Chapter 6.

The main difference compared to the basic framework in Chapter 3 and also the proposed framework in Chapter 4 is that, in this chapter, the auditory feedback is explicitly utilized as TTS input for self-adaptive speech synthesis. The proposed self-adaptive TTS process in this chapter is analogous to the TTS-to-ASR process in the basic framework, in which it starts with inputting the text into TTS and then calculating the text reconstruction loss based on ASR. The basic framework uses the reconstruction loss to update the neural network weights in ASR only. In contrast, the proposed method described in this chapter aims to improve the TTS intelligibility in noisy situations dynamically using auditory feedback that includes the ASR loss. We give the ASR loss to TTS as an auxiliary input, instead of for ASR optimization, to make TTS aware of how they perform and estimate how they should change the output speech style. In this thesis, the proposed self-adaptive TTS does not use the reconstruction loss to optimize the neural network parameters, so we can focus on the feedback utilization mechanism to change the TTS speech style. The proposed TTS' neural network weight is optimized using TTS loss only. The neural network weight is maintained during inference.

5.2 Related Works

Lombard speech synthesis is designed to produce intelligible speech in the presence of noise. This system has gained attention within the speech community, which was reflected in the Hurricane Challenge [71, 72] for speech synthesis and the evaluation of speech enhancement systems under noisy conditions.

5.2.1 Lombard TTS with Speech Post Modification

The related works on speech enhancement applied signal processing on speech to improve the intelligibility in noisy conditions. The earlier works performed speech modification through a statistical method with fixed parameters based on known noises. The enhancement operations include modifications of duration [73], pitch, energy contour, formant sharpness, and intensity [74]. Several works also proposed other spectral modification approaches, such as spectral tilt, spectrum contrast enhancement, and harmonic component preservation at in the low-frequency region to emphasize the speech features that are important for speech perception [75]. Spectral shaping and a dynamic range compression method were also studied [76]. Next, AdaptDRC [77] was proposed for speech enhancement controlled by the short-term speech intelligibility index. It enhanced the speech content at high frequencies by also boosting the low-energy speech content through time- and frequency-dependent dynamic range compression and frequency-shaping. Another work also proposed a noise-dependent AdaptDRC with the reverberation-dependent onset enhancement and overlapping masking reduction [78]. Although the above approaches could be applied to both natural and synthesized speech, a noise signal separated from the speech was required. Their experiments were generally carried out by assuming perfect noise was available. Speech and noise separation in real situations might be challenging, especially in dynamic noise conditions. In our proposed approach, we use a TTS to directly synthesize the Lombard speech given the text and feedback based on synthesized speech with the noise.

5.2.2 Lombard TTS with Offline Fine-tuning

In the conventional approach, Lombard speech synthesis was commonly done using the parametric model with HMM. GlottHMM [79, 80] applies a glottal inverse filtering technique in the vocoder of HMM TTS to improve speech intelligibility in the presence of noise. Speech was synthesized by filtering the glottal excitation with a vocal tract filter, where the excitation signal was generated from the real glottal flow extracted from natural speech. A speaking style adaptation approach has also been studied, in which the HMM TTS system is adapted with a small amount of Lombard speech after training with normal speech [37, 38]. The per-

formance of the statistical approach, however, has been limited by poor acoustic modeling [39].

Recently, the neural network approach has also gained attention for synthesizing Lombard speech in an end-to-end manner. A recent study proposed Lombard TTS by tuning a Tacotron model on the Lombard speech data [39]. The model was first pre-trained on the normal speech data. In another study, a multi-style Tacotron TTS was proposed with a framework that could synthesize speech in normal, whispered, and Lombard speech styles [81]. In their experiment, TTS training was done by including speech spoken in these three styles by a single speaker in the training material. The TTS generates the styled speech by treating the three speaking styles as three different speakers, so the output speech style is decided based on the speaker embedding vector.

5.3 TTS with Auditory Feedback in Machine Speech Chain Framework

When the environment becomes noisy, the proposed TTS tries to synthesize the speech with higher intensity, higher pitch, and slower speed than the speech before the adaptation. In this chapter, the basic TTS is the autoregressive Transformer-based multi-speaker TTS in a MultiSpeech framework [25]. To achieve dynamic adaptation, we extended the basic structure with auditory feedback modules (ASR-loss embedding and SNR embedding) and a variance adaptor. An overview of this architecture is given in Fig. 5.3 (a). The proposed TTS generates the speech Mel-spectrogram $\boldsymbol{y} = [y_1, y_2, ..., y_T]$ with a length of T given the character sequence $\boldsymbol{x} = [x_1, x_2, ..., x_S]$ with length S. TTS adapts the speech prosody attributes by also taking the auditory feedback in SNR embedding (z_{SNR}) and ASR-loss embedding (z_{ASR}) as input. In inference, adaptation is done in several feedback iterations until ASR loss converges. The conversion of the Mel-spectrogram into a waveform is done using a CBHG module and the Griffin-Lim algorithm, similar to the Tacotron framework. We use a speaker recognition module implementing the Deep-Speaker framework to generate the speaker embedding vector z_{SPK} for our multi-speaker TTS, following the implementation of TTS in the basic machine speech chain framework.

5.3.1 Achitecture

In this thesis, we construct three TTS models with different feedback configurations to investigate the effect of auditory feedback in the machine speech chain framework. All systems are trained using normal speech and Lombard speech. The overview of the proposed architecture is shown in Fig. 5.3. The details are below.

5.3.1.1 TTS with SNR Feedback

TTS synthesizes speech based on text input and SNR feedback as embedding. The SNR feedback represents the SNR or speech and noise intensity ratio, which is a measure of how well the TTS speech can be heard in a noisy environment. Commonly SNR can be calculated by measuring the intensity of the speech and noise separately. However, separating speech and noise in a real-world situation could be challenging, for example, because noises might dynamically change. In our approach, we use machine learning as a neural network to obtain the SNR directly from a noisy speech where the speech and noise are mixed. Given an SNR embedding feedback, TTS attempts to re-synthesize speech with a higher SNR ($\geq$ 20 dB), indicating that the speech is louder than the noise.

We implement the SNR embedding module using convolution network layers with an average pooling operation in Fig. 5.3 (c). It generates an utterance-level embedding z_{SNR} from noisy TTS speech features $\boldsymbol{y}^{noisy}$:

$$z_{SNR} = \textit{SNR Embedding}(\boldsymbol{y}^{noisy}). \tag{5.1}$$

Before training the TTS, we pre-train the SNR embedding module (Conv + ReLU and ResBlock) as an SNR recognition module so that the TTS can converge faster. We can initialize the SNR recognition model as a classification or a regression model. The SNR recognition model recognizes the average SNR in an utterance. In SNR classification, we first define several SNR classes. It generates SNR embedding vectors by learning to classify the SNR given noisy speech utterances. Model optimization is done by minimizing the cross-entropy loss:

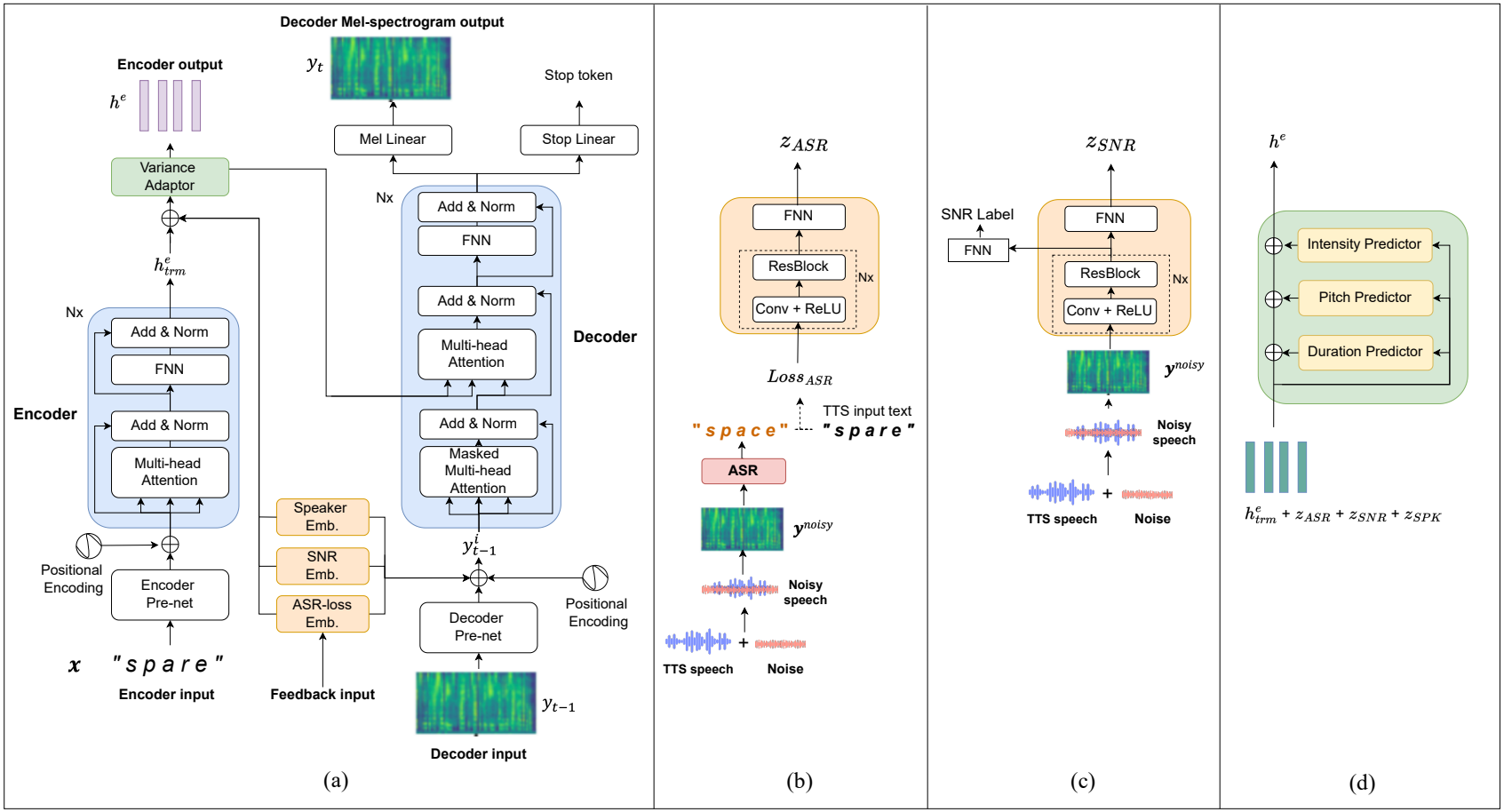


Figure 5.3: Architecture: (a) proposed TTS with an autoregressive transformer-based encoder-decoder structure, extended with (b) ASR-loss embedding, (c) SNR embedding, and (d) variance adaptor [3] modules.

$$Loss_{SNR-CLS}(l, p_l) = -\sum_{c_l=1}^{C_l} \mathbb{1}(l = c_l) * \log p_l[c_l], \tag{5.2}$$

where l is the reference SNR label, p_l is the predicted SNR probability, and C_l is the number of SNR classes.

On the other hand, the SNR regression model is trained to estimate the average SNR of an utterance as a real value. It is optimized using L2 loss:

$$Loss_{SNR-REG}(l, \hat{l}) = (l - \hat{l})^2, \tag{5.3}$$

where $\hat{l}$ is the predicted SNR at the utterance level.

Inside the TTS encoder side, SNR embedding vector z_{SNR} is integrated with the TTS encoder transformer output h^e_{trm} and speaker embedding z_{SPK} to obtain the final TTS encoder output h^e, written as

$$h^e = h^e_{trm} + z_{SPK} + z_{SNR}. \tag{5.4}$$

On the decoder side, embedding vectors z_{SPK} and z_{SNR} are also combined with the decoder pre-net output and the positional encoding PE to obtain the decoder intermediate input y^i_{t-1}:

$$y^i_{t-1} = prenet(y_{t-1}) + z_{SPK} + z_{SNR} + PE. \tag{5.5}$$

Following this, the decoder multi-head attention query, key, and value are the encoder output and decoder input that have been embedded with the auditory feedback. TTS model optimization is done based on the standard seq2seq TTS loss function in Eq. 2.15.

5.3.1.2 TTS with ASR Loss and SNR Feedback

TTS generates speech based on text input and auditory feedback in SNR and ASR-loss embedding. The ASR-loss embedding, shown in Fig. 5.3 (b), represents the speech intelligibility measurement of how well the noisy TTS speech can be recognized by an ASR. ASR-loss embedding vector z_{ASR} is generated by transcribing a noisy TTS speech using an ASR, which written as

$$\boldsymbol{p}_x = p(\boldsymbol{x}|\boldsymbol{y}^{noisy}), \tag{5.6}$$

where $\boldsymbol{p}_x$ is the ASR posterior, and then calculating the loss between the ASR hypothesis and the TTS input text. The ASR loss embedding module, which is a stack of convolutional layers with average pooling, produces z_{ASR} as an utterance-level embedding by taking $Loss_{ASR}(\boldsymbol{x}, \boldsymbol{p}_x)$ that is a sequence of character-level loss in a sentence:

$$z_{ASR} = ASR\ Loss\ Embedding(Loss_{ASR}(\boldsymbol{x}, \boldsymbol{p}_x)), \tag{5.7}$$

Here suppose a sentence text $\boldsymbol{x}$ consists of S characters, the s-th character (x_s) loss is calculated by

$$Loss_{ASR}(x_s, p_{x_s}) = -\sum_{c=1}^{C} \mathbb{1}(x_s = c) * \log p_{x_s}[c], \tag{5.8}$$

where $Loss_{ASR}(x_s, p_{x_s})$ is the character-level loss and C is the size of ASR output vocabulary. ASR text decoding is done by teacher-forcing mechanism based on the TTS text input.

Inside the main part of TTS, ASR-loss embedding is combined with the TTS encoder output and the decoder input along with the speaker and the SNR embedding vectors:

$$h^e = h^e_{trm} + z_{SPK} + z_{SNR} + z_{ASR}, \tag{5.9}$$

$$y^i_{t-1} = prenet(y_{t-1}) + z_{SPK} + z_{SNR} + z_{ASR} + PE. \tag{5.10}$$

In TTS training and inference, we use a pre-trained ASR to transcribe TTS speech. The ASR-loss embedding module is trained directly during TTS training without a pre-training step. TTS optimization is done by minimizing the TTS loss in Eq. 2.15.

5.3.1.3 TTS with ASR Loss and SNR Feedback and Variance Adaptor

In addition to the SNR and ASR-loss embedding feedback, we implement a variance adaptor module in the proposed TTS with a similar approach to FastSpeech2 [3]. The variance adaptor is intended to guide the prosody adaptation by predicting the prosody attributes from the encoded text input and the auditory feedback. The variance adaptor, shown in Fig. 5.3 (d), consists of three components: a pitch predictor, an intensity predictor, and a duration predictor. This

module is applied in the TTS encoder and provides the following output:

$$h^e = \mathit{Var\ Adaptor}(h^e_{trm} + z_{SPK} + z_{SNR} + z_{ASR}). \tag{5.11}$$

The decoder input follows Eq. 5.10. In our duration predictor, instead of predicting the token duration as an integer to regulate the encoder output length like in the original FastSpeech2 framework, it estimates the duration as a real value similar to the other predictors. The encoder output length in our model follows the standard autoregressive transformer TTS.

The proposed TTS with variance adaptor is trained with the standard TTS loss function combined with the variance predictor losses. The variance predictor loss is calculated by the MSE loss function:

$$Loss_{pred}(\boldsymbol{v}, \hat{\boldsymbol{v}}) = \frac{1}{S}\sum_{s=1}^{S}(v_s - \hat{v}_s)^2, \tag{5.12}$$

where v_s is the normalized reference value for the predictors inside the variance adaptor and $\hat{v}_s$ is the predictor output at timestep s. The reference intensity, pitch, and duration are estimated from the TTS reference output speech. The TTS training loss function becomes

$$\begin{aligned} &Loss_{TTS}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \\ &\frac{1}{T}\sum_{t=1}^{T}((y_t - \hat{y}_t)^2 - (b_t \log(\hat{b}_t) + (1 - b_t)\log(1 - \hat{b}_t)))+ \\ &Loss_{pred}(\boldsymbol{v}^P, \hat{\boldsymbol{v}}^P) + Loss_{pred}(\boldsymbol{v}^G, \hat{\boldsymbol{v}}^G) + Loss_{pred}(\boldsymbol{v}^D, \hat{\boldsymbol{v}}^D), \end{aligned} \tag{5.13}$$

where $\boldsymbol{Y} = [\boldsymbol{y}, \boldsymbol{b}, \boldsymbol{v}^P, \boldsymbol{v}^G, \boldsymbol{v}^D]$ and $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{v}}^P, \hat{\boldsymbol{v}}^G, \hat{\boldsymbol{v}}^D]$. Here, $\boldsymbol{v}^P$, $\boldsymbol{v}^G$, and $\boldsymbol{v}^D$ are the reference pitch, the intensity, and the duration, and $\hat{\boldsymbol{v}}^P$, $\hat{\boldsymbol{v}}^G$, and $\hat{\boldsymbol{v}}^D$ are the pitch, the intensity, and the duration predicted by the predictor.

5.3.2 Training Method

The proposed TTS training method is illustrated in Fig. 5.4. To enable dynamic adaptation, we train the proposed TTS using inputs, consisting of text and auditory feedback embedding vectors, and an output target, which is reference speech representing the speech after adaptation. Auditory feedback represents

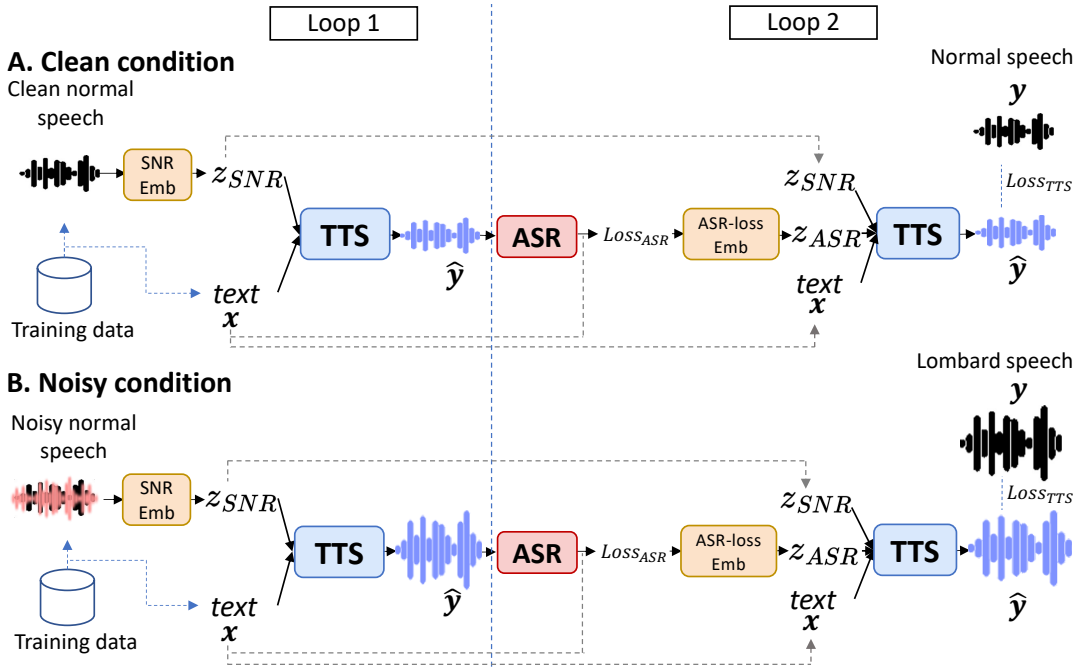


Figure 5.4: Proposed TTS training in two feedback loops based on clean and noisy conditions.

the speech condition before it is adapted into the target speech. In training, the SNR embedding is pre-computed from clean or noisy normal speech in the training data, while ASR-loss embedding is computed from TTS speech generated during training. Therefore, the speech data required for training are the clean normal speech, the normal speech with additive noise (noisy normal speech), and the clean Lombard speech.

For speech synthesis and adaptation with the re-speaking mechanism, we trained the proposed TTS in one or two feedback loops based on the type of architecture:

1. **TTS training with SNR feedback**: For the proposed TTS without the ASR-loss embedding module, we apply one-loop training using pre-computed SNR embedding and text based on the training data.

2. **TTS training with SNR-ASR feedback**: For the proposed TTS with the ASR-loss embedding module, we generate speech in two feedback loops.

The SNR embedding vector is calculated in the first loop based on the training data, and we use the same vector in the second loop. The ASR-loss embedding is calculated in the second loop based on the TTS speech generated in the first loop, thus, the TTS can learn the ASR-loss pattern based on the synthesized speech for a more realistic ASR feedback processing.

We consider two target conditions: clean and noisy. In the clean condition, the proposed TTS produces normal speech without the Lombard effect. In the noisy condition, the proposed TTS produces Lombard speech. We apply batch training to train the proposed TTS in which a batch consists of a mix of speech samples for clean and noisy conditions. The details of the training mechanism are described below with two feedback loops based on the type of target condition:

1. **TTS training in clean condition**: Speech generation is learned using text and clean normal speech data. Normal speech is speech which is uttered in a clean condition without the Lombard effect. It has a lower intensity, a lower pitch, and a faster speaking rate than Lombard speech. In the training, the SNR embedding and the output speech reference are based on clean normal speech. Therefore, before starting the training, we first compute the SNR embedding from clean normal speech. In the first feedback loop, TTS generates normal speech by taking the text, pre-computed SNR embedding, and ASR-loss embedding in a zero vector as the input. In the second feedback loop, we repeat the same process but use the ASR-loss embedding computed from the TTS speech features predicted in the first loop.

2. **TTS training in noisy condition**: Speech generation is learned using text, noisy normal speech, and clean Lombard speech with high SNR and low ASR loss in the corresponding noisy condition. Clean Lombard speech is a speech under the Lombard effect but without noise in the audio. In our experiment, the clean Lombard speech is a synthetic Lombard speech generated by modifying the prosody of normal speech (intensity, pitch, duration) into Lombard speech using SoX audio manipulation toolkit [82, 83]. Noises were not included in the resulting audio. The detail is discussed in Chapter 5.4.1.1. In the training loop, we use SNR embedding generated

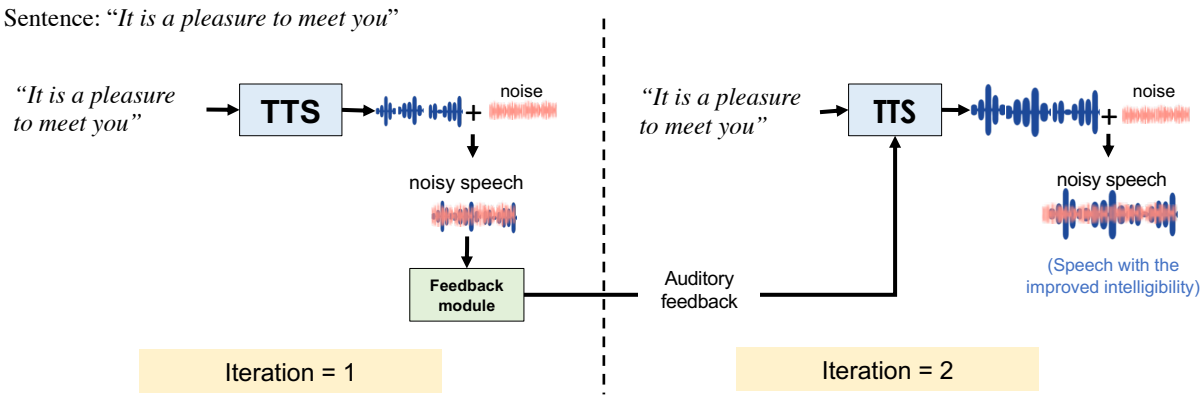


Figure 5.5: Proposed TTS inference mechanism

from noisy normal speech in the training data and the text as the input and the Lombard speech as the target. The ASR-loss embedding generation and utilization method are the same as those in the clean condition case. In the first loop, ASR-loss embedding is a zero vector. In the second loop, the ASR-loss embedding is generated from TTS speech features generated in the previous loop.

5.3.3 Inference Mechanism

The proposed TTS in the machine speech chain mechanism synthesizes the speech using feedback loop, which unrolled loop is shown in Fig. 5.5. In the first iteration in the loop, TTS synthesizes the speech only from the text input, the same as in standard TTS. The synthesized speech is then combined with the noises, from which the noisy speech is passed to the auditory feedback modules to generate the auditory feedback vectors. In the second loop iteration, TTS once again synthesizes the speech from the same text by also taking the auditory feedback from the first loop as an input. Here, TTS will try to improve the speech intelligibility considering the feedback that it has received. Further speech improvement can be made by repeating the same feedback mechanism for the third loop and so on.

5.4 Experimental Setup

5.4.1 Dataset

Our systems are constructed using normal speech, normal speech with additive noise, and Lombard speech datasets. Since the availability of Lombard speech data is limited, we constructed a synthetic Lombard speech dataset by observing the natural Lombard speech and modifying the normal speech into Lombard speech.

Our experiment was based on WSJ corpus. TTS training was done based on three static noise conditions containing noises from 1) clean, 2) SNR 0 dB, and 3) SNR -10 dB conditions, where SNR is relative to normal speech of 44.44 dB in WSJ. The proposed TTS with a variance adaptor was also trained using the character-level prosody attribute labels. These labels were generated by first extracting a character-level speech timing using Montreal forced-alignment toolkit [84]. The details of the data we constructed from WSJ and then utilized for model training are given below.

5.4.1.1 WSJ Corpus

Natural Normal Speech The WSJ dataset consists of multi-speaker English speeches recorded by reading news text in a clean condition, sampled at 16 kHz. We utilized the *SI-284*, *dev93*, and *eval92* sets as the training, development, and evaluation sets. The *SI-284* set consists of 81 hours of speech. The average speech intensity in WSJ utterances was 44.44 dB. A speech utterance length is 7.88 sec and 17 words on average.

Normal Speech with Additive Noise We combined the WSJ normal speech with noisy sounds to train our system. The noises were white[1] and restaurant babble[2] noises with SNR levels of 0 dB and -10 dB relative to the WSJ speech. This dataset was mainly utilized to train the SNR recognition model and ASR.

[1]Generated using white-noise-generator toolkit (https://github.com/jannispinter/white-noise-generator)

[2]From the noise sounds in AURORA-2 corpus [85]

5.4.1.2 Lombard Speech Dataset Construction

Natural Lombard Speech To learn how human vocalization changes in noisy conditions, we recorded natural Lombard speech with a single male speaker who read the WSJ *dev93* and *eval92* set transcriptions in noisy conditions. The noises in the recording were the same noises generated for our normal speech dataset with additive noise. The noise level is considered constant within an utterance. Given only the noise signals, the speaker read aloud the WSJ text as if it were aimed at someone in a noisy condition. Then, we estimated prosody attributes from the normal and the Lombard speech in phoneme-level detail, and the averages of these values can be seen in Table 5.1. For comparison, we also conducted the recording in both clean and dynamic noise environments from the same speaker.

Table 5.1: Statistic of the natural Lombard speech spoken by a single male speaker

Noise source condition	**Noise Intensity (dB)**	**Speech**		
		Intensity (dB)	**Pitch (Hz)**	**Speaking rate (words/sec)**
Clean	-	56.92	124.63	2.05
SNR0	44.44	59.73	132.56	1.99
SNR-10	54.44	63.68	143.23	1.93

Synthetic Lombard Speech Next, based on the prosody attribute changes observed in the recorded Lombard speech, we constructed synthetic Lombard speech of a full set of WSJ data. Synthetic Lombard speech was made by modifying the original WSJ speech pitch, duration, and intensity[3]. First, since the WSJ speech consists of multi-speaker data, to maintain the speaker characteristics, we modified the speech pitch and duration based on the attribute shift between the clean and noisy conditions as shown in Table 5.1. The modification was done by, first, aligning the Lombard speech and the normal speech in our recording data, and then estimating the attribute shifts at phoneme-levels. Next, we estimate the pitch and duration of normal WSJ speech at phoneme-level. Then, based on

[3]The speech pitch, duration, and intensity were modified using the SoundExchange (SoX) toolkit (http://sox.sourceforge.net/).

the database of prosody shifts in the recording data, the target noise, and the prosody level of the WSJ normal speech, we estimate the target pitch and duration and modify the corresponding normal speech using SoX commands. After that, we modified the speech intensity into a target SNR of 20 dB relative to the noise level. The maximum intensity of the resulting Lombard speech was 75 dB to avoid clipping. This dataset was utilized as the target Lombard speech in the TTS training.

5.5 Experiment

5.5.1 Model Configuration

5.5.1.1 TTS

Our TTS model consists of a transformer-based encoder and autoregressive decoder. The TTS input was the character sequence, and the output was the 80 dimensions of the Mel-spectrogram. The encoder character embedding layer consists of 256 units, followed by an encoder pre-net that consists of three convolution layers. In the decoder part, the decoder pre-net consists of three linear layers. For both the encoder and decoder, the transformer module consists of six transformer blocks with a dimension of 512, eight attention heads, and a feed-forward inner dimension size of 2048.

5.5.1.2 ASR

Our ASR takes a sequence of speech Mel-spectrogram input to predict its character level transcription. The model configuration follows a similar configuration to the big model proposed in Speech-Transformer [34]. It consisted of twelve encoder layers and six decoder layers. The transformer dimension was 512 with the feed-forward inner dimension of 2048. The attention module consisted of multi-head self-attention with four heads.

We prepared two non-incremental ASR systems which were trained on the WSJ dataset to evaluate the TTS in the ASR objective measure. The first system was a clean-condition ASR, which was trained using speech utterances in clean

conditions. The second system was a multi-condition ASR that was trained using speech in both clean and noisy conditions. For the proposed TTS, we used the multi-condition ASR to generate the ASR feedback.

5.5.1.3 SNR Recognition

The SNR recognition model consisted of four stacks of convolution and residual blocks and a linear layer, which was trained with a learning rate of 1e-4. As mentioned earlier, we experimented on two SNR recognition tasks: classification and regression. The difference between those models lies in the output layer size. The SNR classification model output layer dimension was three based on the number of SNR classes: SNR 0 dB, SNR -10 dB, and clean (no noise). SNR classification model was specifically designed for the use case in static noise conditions. It was trained using normal speech with static noises. Next, the SNR regression model output layer dimension was set to one, and the SNR level was output as a real number scaled in the range of -1 to 1. This model was designed for more precise SNR prediction in the static and dynamic noise conditions (see the details in Chapter 6), and it was trained using the normal speech added with static and dynamic noises. SNR recognition models were trained to recognize the average SNR of a noisy speech utterance.

5.5.1.4 Speaker Recognition

The speaker recognition module was based on Deep-Speaker framework with the same configuration as the one implemented in the basic machine speech chain work [46]. The speaker recognition component was trained using *SI-284* set consisting of 282 speakers.

5.5.2 Evaluation Metrics

We focus on evaluating TTS speech intelligibility. The speech intelligibility was measured through two metrics: ASR CER and the short-term objective intelligibility (STOI) measure:

- **Character error rate**
  CER is estimated by transcribing a noisy TTS speech using ASR. The error

calculation is done using Eq. 4.4 with the TTS input text as the reference text.

- **Short-term objective intelligibility**
  STOI [86] estimates the temporal envelope correlation between the speech signal disturbed by noise (noisy) and the reference speech signal before being disturbed (clean). A higher correlation indicates a higher speech signal intelligibility. STOI score is represented as a scalar $-1 \leq d \leq 1$ denoting the envelope loss $\mathcal{L}$ of the entire speech signal, formally written as

$$d = \frac{1}{J(M-N+1)} \sum_{j=1}^{J} \sum_{m=N}^{M} \mathcal{L}(\underline{a}_{j,m}, \hat{\underline{a}}_{j,m}), \tag{5.14}$$

$$\mathcal{L}(\underline{a}_{j,m}, \hat{\underline{a}}_{j,m}) = \frac{\left(\underline{a}_{j,m} - \mu_{\underline{a}_{j,m}}\right)^T \left(\hat{\underline{a}}_{j,m} - \mu_{\hat{\underline{a}}_{j,m}}\right)}{\left\|\underline{a}_{j,m} - \mu_{\underline{a}_{j,m}}\right\| \ \left\|\hat{\underline{a}}_{j,m} - \mu_{\hat{\underline{a}}_{j,m}}\right\|}, \tag{5.15}$$

  where $\|\cdot\|$ denotes the Euclidean L2-norm and $\mu_{\underline{a}_{j,m}}$ and $\mu_{\underline{a}_{j,m}}$ denote the sample means of $\underline{a}_{j,m}$ and $\hat{\underline{a}}_{j,m}$, respectively. Here $\underline{a}_{j,m}$ is the short-time temporal envelope vector of the reference speech from the $(m-N+1)$-th frame to m-th frame:

$$\underline{a}_{j,m} = [a_j(m-N+1),\ a_j(m-N+2), \ldots, a_j(m)]^T \tag{5.16}$$

$$a_j(m) = \sqrt{\sum_{k=k_1(j)}^{k_2(j)} a(k,m)^2}, \tag{5.17}$$

  where a is the reference speech magnitude spectrum, and $k_1(j)$ and $k_2(j)$ denote the first and last STFT bin index, respectively, of the jth one-third octave band. In Eq. 5.14, $\hat{\underline{a}}_{j,m}$ are those of the noise-disturbed speech. In the experiment, the reference speech is the TTS speech before combined with noise and the noise-disturbed speech is the TTS speech combined with noise signal.

Table 5.2: Average TTS speech intelligibility (CER %) at different SNR levels in babble- and white-noise conditions evaluated using clean- and multi-condition training ASR. SNR levels denote the SNR condition before adaptation was performed.

System	Clean-condition WSJ ASR			Multi-condition WSJ ASR		
	Clean	SNR 0	SNR -10	Clean	SNR 0	SNR -10
Baseline TTS						
Std. TTS (Clean)	18.92	118.72	106.25	18.32	70.54	77.07
+ Rule-based modification	18.92	102.96	104.69	18.32	43.25	55.79
+ Fine-tuning (SNR 0 + SNR -10)	13.58	68.53	94.75	14.82	21.99	37.41
Std. TTS (Clean+SNR 0+SNR -10)	11.04	114.36	102.83	12.89	56.57	70.41
Proposed TTS						
TTS in speech chain framework	11.04	114.36	102.83	12.89	56.57	70.41
+ SNR (cls)	10.21	83.15	101.41	11.58	22.82	42.00
+ SNR (cls) + ASR	10.76	52.51	87.72	12.55	16.11	25.62
+ SNR (cls) + ASR + var. adaptor	10.47	55.70	92.75	11.99	14.70	24.96
+ SNR (reg) + ASR + var. adaptor	12.63	66.84	86.41	13.52	18.57	31.19
Topline (human natural speech)						
Normal speech	5.77	92.56	98.98	7.43	22.17	58.81
+ Rule-based modification	5.77	58.40	67.78	7.43	13.24	15.15
Lombard speech	5.77	25.38	59.25	7.43	11.46	20.46

5.6 Experiment Results

5.6.1 Speech Intelligibility in Character Error Rate

In this study, we focused on improving TTS speech intelligibility in noisy conditions. The TTS speech intelligibility measured in ASR CER is shown in Table 5.2. In this experiment, CER was calculated by firstly transcribing TTS speech using a clean-condition or a multi-condition ASR model that was trained on the WSJ dataset. The ASR feedback for all proposed TTS systems was generated using the multi-condition ASR. All TTS systems generated speech using speaker embedding extracted from the normal natural speech in our recording data. The clean condition testing was done using TTS speech without noise, and the noisy condition testing was done using noise signals of the corresponding SNR condition. SNR levels here are the initial SNR conditions before adaptation. The

proposed TTS with '*SNR (cls)*' generated SNR feedback based on SNR classification, while the system with '*SNR (reg)*' was based on SNR regression. We allow the proposed TTS to refine the speech in five feedback iterations at most, and we present the speech that achieved the lowest ASR loss.

We compared our proposed systems with several baselines: 1) the standard TTS denoted as '*Std. TTS*' trained using normal speech from a clean condition, in which the speech in noisy testing was merged with the noise without any modification; 2) the rule-based modification into the Lombard speech, in which the original output of the standard TTS was modified by the same method as that used in the synthetic Lombard WSJ speech construction; 3) the standard TTS that was fine-tuned to Lombard speech [39]; and 4) the standard TTS trained on normal and Lombard speech. Note that these systems did not have feedback components and were also trained based on static noise conditions. The topline speech is the natural clean and Lombard speech produced by a human. We also included synthetic modifications from the natural human speech.

From the baseline results, we found that the speech CER of the standard TTS from clean condition training could be reduced by post-processing the speech prosody into Lombard speech-like. We also obtained further improvement by fine-tuning the standard model using Lombard speech data. Next, the proposed models with SNR and ASR feedback were able to outperform the fine-tuned baseline models and more closely approached the CER of topline human speech. In this experiment, the best TTS performance was achieved by a TTS with SNR classification-based feedback ('*SNR (cls)*'), ASR-loss feedback, and a variance adaptor.

In overall, TTS with SNR classification feedback and the TTS with SNR regression feedback show different results. The performance difference between them might be caused by the SNR recognition performance and the complexity of SNR embedding. First, SNR classification accuracy on noisy natural normal speech was 100%. This model achieved high accuracy because the number of classes was small (three classes) and the training data was sufficiently large. On the other hand, the SNR regression prediction error was 0.90 dB. The SNR embedding generated by the regression model might be more complex than the embedding based on the classification model. This is because the embedding from

SNR regression represents a real value, while SNR classification represents a discrete class based on three classes only. This might affect the TTS convergence, in which TTS with SNR classification-based feedback was easier to converge and resulted in higher speech intelligibility than with the SNR regression-based feedback in the static noise conditions.

5.6.1.1 The Impact of Auditory Feedback

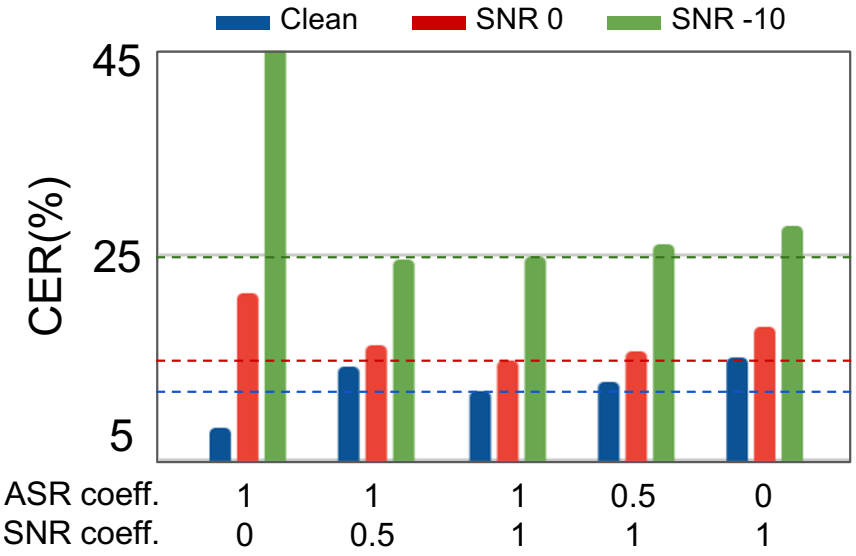


Figure 5.6: Effect of auditory feedback on the TTS speech intelligibility based on the embedding coefficients.

Using the best proposed system, we analyzed how the SNR and ASR-loss embedding affected the TTS performance. To clarify this, we experimented on the various embedding coefficients to scale the SNR and ASR-loss embedding when they were combined into TTS encoder output h^e and the decoder's first transformer layer input y^i_{t-1}. From the results shown in Fig. 5.6, SNR feedback alone is shown to be sufficient to improve the speech intelligibility in noise, but it did not result in the best performance in our setting. Using only ASR feedback could result in the best TTS speech when the condition is clean. This shows that ASR feedback also contributed to speech enhancement. But when the environment becomes noisy, SNR feedback becomes critical to the system. The

Table 5.3: ASR performance on natural speech and the corresponding TTS speech intelligibility in SNR -10 dB condition. (*: CER assessed using multi-condition WSJ ASR. ASR feedback was based on the ASR model in the same row.)

ASR training data	**ASR CER (%) on natural speech**		***TTS intelligibility (CER %)**
	Noisy Lombard speech	**Noisy normal speech**	
TTS + SNR (cls)			
No ASR	-	-	42.00
TTS + SNR (cls) + ASR + variance adaptor			
Clean-condition WSJ	73.33	103.35	25.01
Multi-condition WSJ	11.85	41.85	24.96
Multi-condition WSJ+Hurricane	11.81	40.89	24.47

optimum performance is when the SNR embedding and ASR-loss feedback embedding coefficients are equal to one, indicating that both feedbacks are crucial to the Lombard effect by TTS.

5.6.1.2 The Impact of Auditory Feedback Module Performance

The effect of ASR and SNR recognition performance on the proposed TTS was investigated. The details are below.

ASR Our experiment results in Table 5.3 show that TTS speech intelligibility tends to be better as the ASR performance becomes higher. In this experiment, we experimented with the proposed TTS using feedback from three ASR models with different training data in clean and noisy conditions based on WSJ and also the Hurricane dataset (see Chapter 6.4.1.2). The highest performance was achieved when the ASR feedback was generated based on the multi-condition ASR trained using the WSJ and Hurricane datasets, which was our best ASR. The tendency in this experiment's result was similar to the previous experiment in Chapter 5.6.1.1 with the embedding coefficient, where TTS intelligibility with the SNR embedding coefficient of one also tends to be higher when using the higher ASR-loss embedding coefficient.

Although different ASR performances led to different TTS performances, the difference was not significant. This might be affected by the ASR loss generation

Table 5.4: SNR recognition accuracy and the corresponding TTS speech intelligibility in SNR -10 dB condition.

SNR model/input	**SNR accuracy (%)**	**TTS intelligibility (CER%)**
Model A/natural speech	100	23.32
Model A/synthesized speech	97.69	24.96
Model B/synthesized speech	96.85	26.30

method, which was based on teacher-forcing decoding. ASR performance has a small effect on TTS intelligibility, but rather, it is largely affected by whether the ASR feedback module is implemented in the proposed TTS or not. In our experiment results, the TTS model with SNR feedback only had the lowest performance among other TTS systems that used SNR and ASR feedback together. This might suggest that ASR-loss embedding as the auxiliary input feature improved the proposed TTS convergence during training and the TTS performance.

SNR Recognition Higher TTS speech intelligibility was influenced by higher SNR recognition accuracy. We investigated the TTS performance difference between the different SNR recognition models and inputs, which is shown in Table 5.4. 'Model A' in this table was an SNR classification model trained with a learning rate of 1e-4, whereas 'Model B' was trained with a learning rate of 1e-3. In this table, SNR accuracy with the natural speech input was evaluated using noisy natural speech at SNR -10 dB, while the accuracy with synthesized speech input was based on the normal TTS speech added with SNR -10 dB noise. The 'Model A', which was also our default SNR classification model, predicted the SNR more accurately than 'Model B' and the TTS produced more intelligible speech. Then, we further investigated the effects of different input for the SNR model on the TTS speech intelligibility: natural and synthesized speech. In the first experiment with natural speech as the SNR model input, we first extracted the SNR embedding from a noisy natural normal speech sample, and then we used the same SNR embedding throughout the TTS feedback loop. Its SNR recognition accuracy was 100%. As shown in Fig. 5.7, it resulted in a consistently degrading ASR loss in the loop and the best final TTS speech intelligibility (Table 5.4).

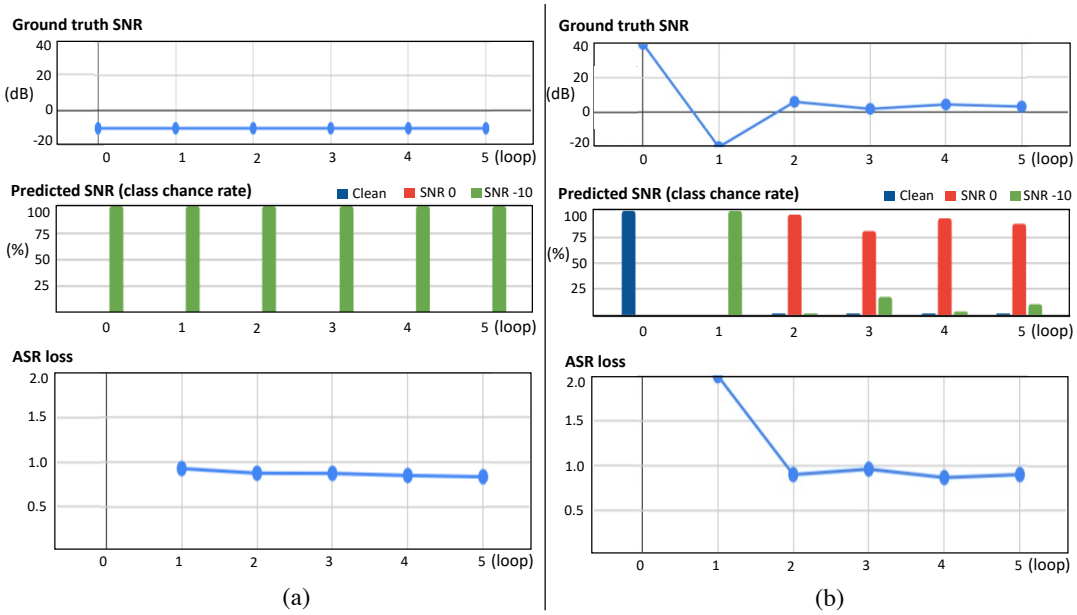


Figure 5.7: The comparison of the SNR prediction result and ASR loss between TTS feedback loops based on (a) constant SNR embedding based on natural speech and (b) updated SNR embedding based on synthesized speech in the loop. The noise intensity was 54.44 dB (SNR -10 dB).

Then, in the second experiment, we fed the proposed TTS with the SNR embedding updated based on synthesized speech. Here, the SNR embedding input in the first loop was based on clean natural normal speech, and the SNR embedding was updated accordingly based on the TTS output in the loop. It resulted in more fluctuated SNR prediction output than in the first experiment, but with an ASR loss that also tended to degrade.

5.6.1.3 The Impact of Feedback Loop

The number of feedback loop iterations also affected our system. Interestingly, the training loop only consisted of two iterations but, in inference, a higher number of loops resulted in better speech intelligibility as shown in Fig. 5.8. Here for each inference iteration, TTS continuously received z_{ASR} and z_{SNR} from the speech that needs to be improved, from which the TTS obtained the current intelligibility information, leading to a better speech performance along with the increased loop number. In comparison to humans, during conversation humans might speak with the Lombard effect in several trials so that the speech could be heard over the

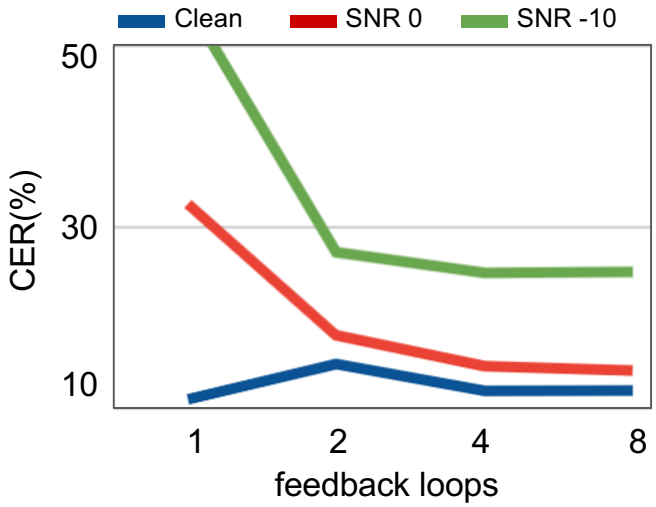


Figure 5.8: Proposed TTS speech intelligibility in different numbers of feedback loop iterations.

noise. Our results reveal that a machine can also dynamically adapt in several loops iteration acted similarly to re-speaking attempt.

5.6.2 Speech Intelligibility in Short-term Objective Intelligibility Measure

Next, we evaluated our TTS speech intelligibility based STOI as shown in Table 5.5. As expected, the standard TTS has the lowest STOI because it could only synthesize normal speech. Similar to CER results, this STOI results show that TTS trained using standard normal data could not perform in noisy situation. Therefore, fine-tuning is necessary to improve the system under such condition. Here the proposed system with SNR-ASR feedback and variance adaptor achieved the higher score than baselines system, showing that the TTS speech signal correlation before and after disturbed by speech is high.

5.6.3 Analysis on Speech Prosody Adaptation

We analyzed the improvement in speech prosody attributes in the proposed Lombard TTS as well as human Lombard speech. Here, we focus on improving speech

Table 5.5: STOI scores (%).

System	SNR 0	SNR-10
Baseline TTS		
Standard TTS (Clean)	49.69	38.07
+ Fine-tuning (SNR0 + SNR-10)	83.56	71.07
Proposed TTS		
TTS + SNR(cls)	81.18	65.66
TTS + SNR(cls) + ASR + var. adaptor	91.09	80.10
TTS + SNR (reg) + ASR + var. adaptor	87.57	76.08
Topline (human natural speech)		
Normal speech	60.31	47.57
+ Rule-based modification	90.48	86.99

intensity, pitch, and duration. First we analyze the TTS speech intensity level, which is shown in 5.6 with a visualized example in Fig. 5.9. Here the proposed TTS intensity changed in the different SNR conditions, where the speech intensity was higher in the noisy condition. The systems with SNR and ASR feedback show a higher improvement than the system with SNR feedback only. Here TTS with '*SNR (reg)*' resulted in more dynamic intensity than the TTS with 'SNR (cls)'. This could be related to the model output precision in representing the SNR. In Table 5.2, TTS with '*SNR (cls)*', ASR feedback and variance adaptor shown higher intelligibility because its Lombard speech was louder than those from TTS with the similar auditory feedback using '*SNR (reg)*'. Next, we analyzed the TTS speech duration in Table 5.7 and speech pitch in Table 5.8, which also illustrated in Fig. 5.10. The proposed TTS adaptation also followed by a shift in the pitch level and also slower speaking rate. The F0 mean squared-error (MSE) evaluation between TTS and human speech in Table 5.9 also shows that the synthesized Lombard speech made the pitch closer to human Lombard speech.

Table 5.6: Average TTS speech intensity level (dB).

System	Clean	SNR 0	SNR -10
Baseline TTS			
Std. TTS (Clean)		43.58	
+ Fine-tuning (SNR 0 + SNR -10)		66.30	
Proposed TTS			
TTS + SNR (cls)	45.59	61.01	61.10
TTS + SNR (cls) + ASR + var. adaptor	43.77	67.28	67.37
TTS + SNR (reg) + ASR + var. adaptor	54.32	62.65	64.20
Topline (human natural speech)			
Normal/Lombard speech	56.92	59.73	63.68

Table 5.7: Average TTS speaking rate (words/sec).

System	Clean	SNR 0	SNR -10
Baseline TTS			
Std. TTS (Clean)		3.37	
+ Fine-tuning (SNR 0 + SNR -10)		3.05	
Proposed TTS			
TTS + SNR (cls)	3.14	3.07	3.07
TTS + SNR (cls) + ASR + var. adaptor	3.35	2.99	3.00
TTS + SNR (reg) + ASR + var. adaptor	2.93	2.80	2.79
Topline (human natural speech)			
Normal/Lombard speech	2.05	1.99	1.93

Table 5.8: Average TTS speech pitch level (Hz).

System	Clean	SNR 0	SNR -10
Baseline TTS			
Std. TTS (Clean)		120.98	
+ Fine-tuning (SNR 0 + SNR -10)		122.62	
Proposed TTS			
TTS + SNR (cls)	123.79	123.31	123.09
TTS + SNR (cls) + ASR + var. adaptor	116.80	123.69	123.86
TTS + SNR (reg) + ASR + var. adaptor	122.81	124.88	125.15
Topline (human natural speech)			
Normal/Lombard speech	124.63	132.56	143.23

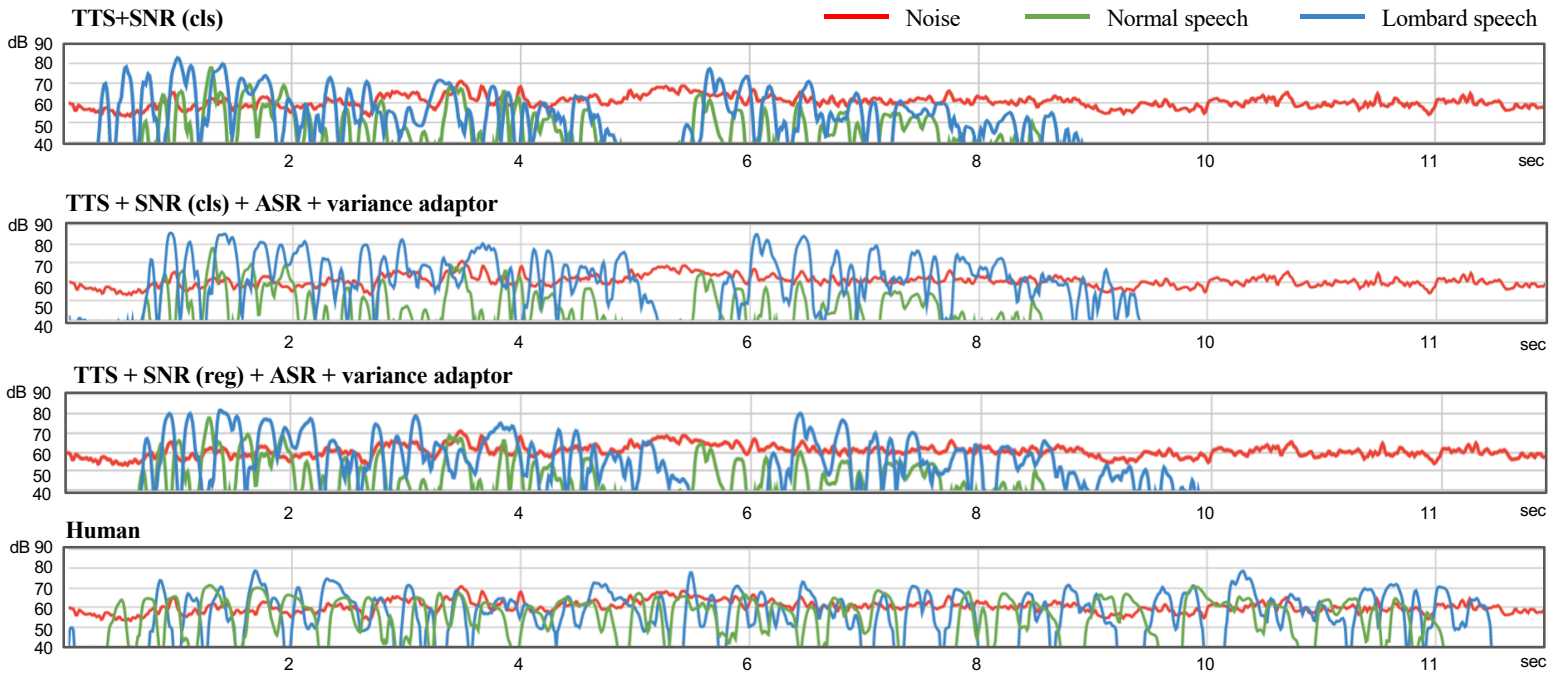


Figure 5.9: Intensities of the normal speech and Lombard speech produced by human and TTS in the babble-noise condition. The speech transcription is the same.

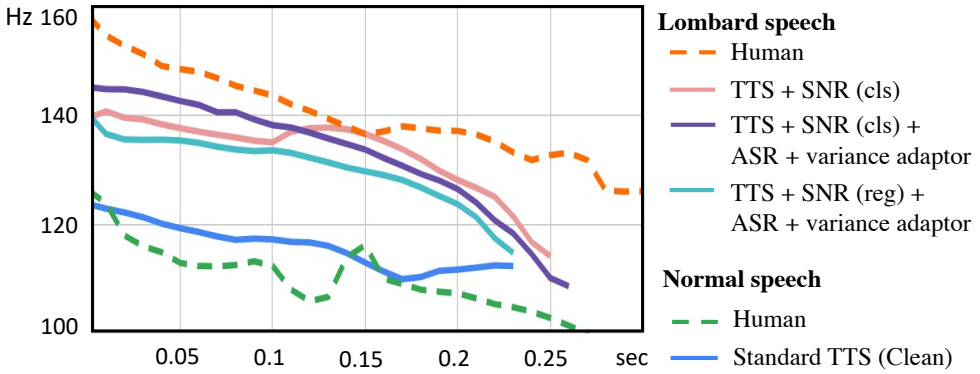


Figure 5.10: Normal and Lombard speech pitch of the word "*ruling*" produced by human and TTS. The Lombard speech was produced in a babble noise with an intensity of 60.35 dB. Natural speech was spoken by a male speaker. TTS speech was generated using a speaker embedding of the same speaker.

Table 5.9: F0 MSE between TTS speech and natural speech.

System	Clean	SNR 0	SNR -10
Baseline TTS			
Std. TTS (Clean)	0.231	0.283	0.380
+ Fine-tuning (SNR 0 + SNR -10)	0.216	0.278	0.368
Proposed TTS			
TTS + SNR(cls)	0.256	0.318	0.406
TTS + SNR(cls) + ASR + var. adaptor	0.239	0.296	0.369
TTS + SNR(reg) + ASR + var. adaptor	0.214	0.266	0.367

Interestingly, our proposed TTS Lombard speech was louder than the human Lombard speech, but human speech had better intelligibility. Here human spoke more slowly with a higher pitch than TTS. This shows that the simultaneous enhancement of these three attributes is necessary. Our proposed TTS speech was shorter than human speech, perhaps due to the speaking rate difference between the speech in WSJ training materials and our natural Lombard speech data. In natural speech, the Lombard effect is not simply a temporal envelope expansion from normal speech, in which it is not limited to intensity, pitch, and duration enhancement. For example, amplitude modulations in Lombard speech are more pronounced than the normal speech [87]. This might also be the reason

for the performance difference between human and TTS speech, since our TTS was trained using Lombard speech with a focus on prosody improvement. Speech naturalness might have also contributed to intelligibility.

5.7 Summary

We constructed a dynamically adaptive machine speech chain inference framework to support TTS in noisy conditions. One key for a machine to adapt to the situation is by understanding the situation and performance, which is represented as an auditory feedback mechanism in the proposed TTS. Our proposed system with auditory feedback and a variance adaptor successfully produced highly intelligible speech that outperformed the standard TTS with a fine-tuning method. These results reveal that dynamic adaptation with auditory feedback is critical not only for human speech production mechanisms but also in speech generation by machines.

The proposed TTS in the machine speech chain framework successfully improved speech intelligibility in noisy conditions. However, a high adaptation delay still occurs because feedback is processed at the utterance level. Thus, if the environment becomes noisier in the middle of an utterance, TTS has to wait for the utterance to finish to begin the adaptation. In the next Chapter 6, we focus on ITTS with a machine speech chain mechanism to start the adaptation with a short latency, i.e., approximately one sec, or three words in our setting.

Chapter 6

Proposed Self-adaptive and Incremental Machine Speech Chain in Noisy Environment

6.1 Overview

Machine self-adaptation to the real environment not only requires the system to understand the situation but also cope with the environmental changes immediately. The machine speech chain inference mechanism in Chapter 5 has enabled TTS with a dynamic adaptation based on the environmental noises. It mimics the human speech chain mechanism with the Lombard effect, in which humans adjust their speaking efforts when they speak in a noisy place. Humans perform the adjustment in real-time by listening to their speech and also the noises simultaneously. A high adaptation delay could cause speaking issues, such as stuttering and the degraded speech intelligibility when noise changes. Several studies reported a response time of about 90-287 ms in the human Lombard effect [16, 17, 18].

One of the challenges for Lombard TTS is that the system has to adapt to static and dynamic noise conditions fast. Most studies on the Lombard TTS [38, 39], including the proposed TTS in Chapter 5, only discuss speech synthesis in static noise conditions. However, noises in reality are dynamic, such as the noise intensity pattern examples shown in Fig. 6.1. Here, noise intensity could change

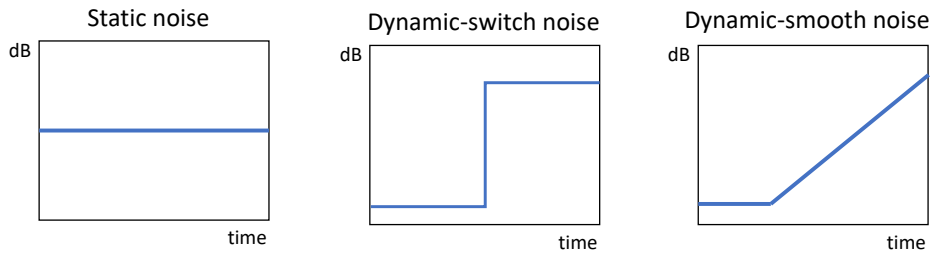


Figure 6.1: Examples of noise intensity pattern in the static and dynamic noise conditions.

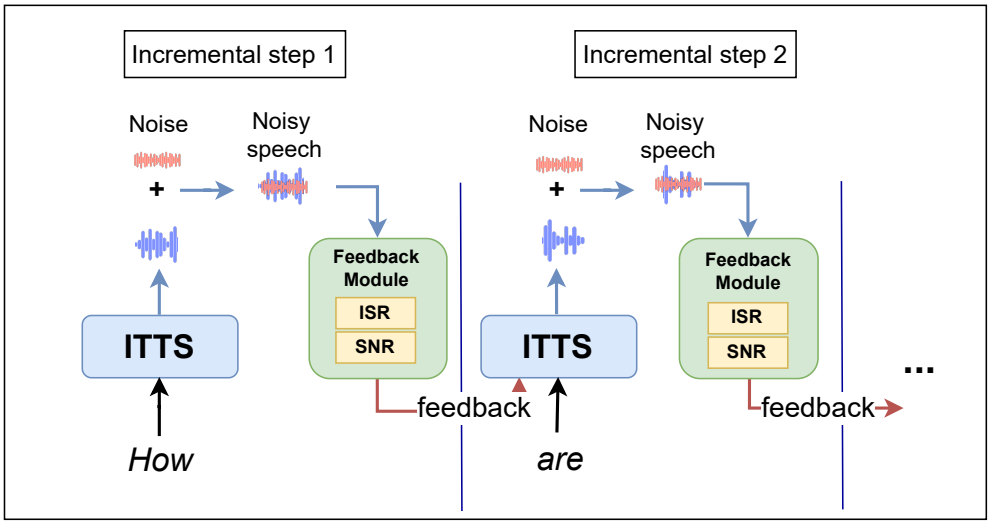


Figure 6.2: Unrolled feedback loops of the proposed the machine speech chain inference mechanism in noisy conditions for ITTS.

suddenly with a switch-like transition (dynamic-switch noise) or change gradually with a smooth transition (dynamic-smooth noise). Although the proposed non-incremental TTS with a machine speech chain inference mechanism could adapt to environmental noises, it requires a long delay to begin the adaptation due to the utterance-level feedback. This implies that, in dynamic noise conditions, TTS needs to wait for the previous utterance to finish before doing the adaptation. In such a situation, the unadapted speech part at the noise transition point might have low intelligibility. Therefore, a dynamic and low-latency or online adaptation is required.

In this chapter, we address the adaptation problem in the dynamic noise condition by focusing on intelligibility and latency. One way to reduce adaptation latency is by reducing the output latency and feedback latency. Therefore, we propose an ITTS in machine speech chain inference, as shown in Fig. 6.2, for incremental speech generation and adaptation. The proposed ITTS works with a short-term feedback mechanism based on short-term output without sacrificing significant performance. It incrementally synthesizes the speech by progressively taking a short text segment and providing feedback based on the current speech segment to the next incremental step. By using short-term feedback, ITTS immediately adapts to environmental changes, thus improving speech intelligibility. In the experiments, we perform the speech synthesis evaluation in both the static and dynamic noise conditions.

6.2 Related Works

To the best of our knowledge, this might be the first deep learning framework for ITTS that mimics the human Lombard speech mechanism in a noisy environment. Similar to the general TTS system, ITTS is commonly constructed using speech data from a clean condition and synthesizes speech by only taking text input. The main difference to the non-incremental TTS is that ITTS performs the speech synthesis incrementally using a partial text sequence, which usually consists of several words, step-by-step to produce the output at low latency. Several ITTS systems in the seq2seq RNN [65, 66] and also Transformer [88] frameworks have been proposed previously. Unfortunately, ITTS in noisy conditions as well as dynamic adaptation have not been investigated yet in previous works.

Previously, in Chapter 4, an incremental machine speech chain for ITTS and ISR training was proposed to support the construction of ITTS and ISR jointly using a short-term auditory feedback loop [89]. However, the feedback loop is disconnected during the inference, so the systems do the task separately. For this reason, the previous framework could still not adapt to environmental conditions. In this thesis, we investigate neural ITTS with short-term input and auditory feedback to dynamically adapt to the environment at a low latency.

6.3 ITTS with Short-term Auditory Feedback in Machine Speech Chain Framework

The proposed ITTS in the machine speech chain framework synthesizes the speech incrementally with the auditory feedback, as shown in Fig. 6.2(b). The proposed ITTS incremental unit is based on fixed words, in which the text segment length in an incremental step is W words. Incrementally synthesizing a well-performed speech based on short text could be a complex task because ITTS has to decide on the output based on a short information sequence. Meanwhile, speech has a continuous representation and heavily depends on context. A general approach to improving the performance is by introducing contextual look-back words and look-ahead words in the input text window [66, 90, 88], which is also applied in the proposed ITTS.

6.3.1 Architecture

The proposed ITTS architecture is based on the autoregressive Transformer TTS. The configuration is similar to the proposed non-incremental TTS in Chapter 5 but with a modification based on how the system treats the intensity-based context. Speech intensity or power context might be required for ITTS because ITTS does not only have to speak louder when noises come but also keep speaking loud while the noise still exists. To keep the Lombard speech prosody in such condition, a feature that represents the speech prosody state based on past speech is necessary. The SNR information, which includes environmental information, only tells the ratio of speech intensity to noise intensity and might be insufficient to keep ITTS speaking loud. Therefore, we propose to do incremental speech synthesis by preserving power context information, which contains the speech intensity from the past output. Using this feature, ITTS will decide whether to keep speaking in Lombard effect or not by also considering the environmental situation. Here, we propose and investigate two ITTS structures accordingly: 1) the power context independent ITTS (PCI-ITTS) that does not consider the power context and 2) the power context dependent ITTS (PCD-ITTS) that considers the power context information.

6.3.1.1 Power Context Independent ITTS (PCI-ITTS)

The PCI-ITTS architecture is the same as the proposed non-incremental TTS with the SNR embedding, the ASR-loss embedding, and the variance adaptor (see Chapter 5.3.1.3). PCI-ITTS reduces the latency of the proposed non-incremental TTS without considering the power context. Here, the SNR and ASR-loss embedding vectors are also generated from the noisy speech segment in the previous step. Those embeddings are utilized as the feedback to generate the speech segment in the current step.

6.3.1.2 Power Context Dependent ITTS (PCD-ITTS)

The PCD-ITTS architecture, shown in Fig. 6.3 (a), is based on the non-incremental TTS structure with SNR embedding, ASR-loss embedding, and variance adaptor (see Chapter 5.3.1.3) with a modification. PCD-ITTS synthesizes speech by also using additional features that preserve the power context. For that reason, in addition to SNR and ASR-loss embeddings, ITTS also takes a power context embedding (Fig. 6.3 (b)) that contains the intensity information of the previous speech output. In our framework, the intensity cues along with the auditory feedback are used to help ITTS control the speech better. It not only helps to control the intensity but also the Lombard speech in general.

Power context embedding (z_{POW}) is generated by using a neural network model with clean speech input. It takes ITTS speech generated in the previous incremental step and then produces an embedding vector representing the input speech's intensity, written as

$$z_{POW} = \textit{Power Context Embedding}(\boldsymbol{y}). \tag{6.1}$$

where $\boldsymbol{y}$ is the clean speech segment generated in the previous incremental step. The power context embedding module consists of the convolution network layers. Before training the ITTS, the power context embedding module could be pretrained for a speech intensity recognition task.

Inside the ITTS, feedback embeddings are utilized to compute the encoder output h^e, written as

$$z = z_{SPK} + z_{SNR} + z_{ASR}, \tag{6.2}$$

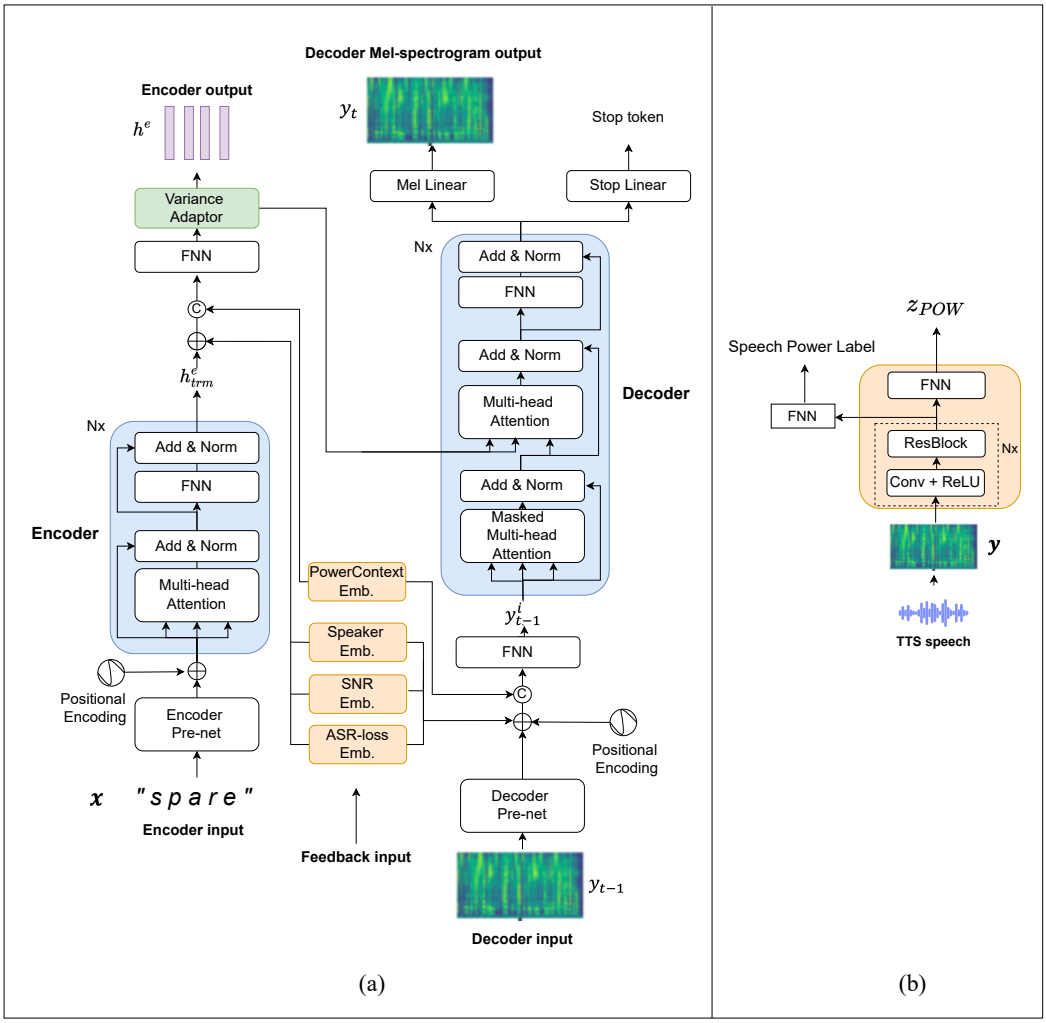


Figure 6.3: (a) The proposed PCD-ITTS structure with (b) power context embedding module.

$$h^e = Var\ Adaptor(FNN([h^e_{trm} + z, z_{POW}])), \tag{6.3}$$

and also the decoder first transformer layer input

$$y^i_{t-1} = FNN([prenet(y_{t-1}) + PE + z, z_{POW}]), \tag{6.4}$$

where z_{POW} is the power context embedding vector. To produce accurate embedding, we pre-train all feedback components for incremental tasks.

6.3.2 Training Mechanism

The proposed ITTS is trained with a similar method as the proposed non-incremental TTS, with the exception that the training is done by using pairs of speech and text segments instead of utterances. ITTS training is done through one feedback loop iteration, in which the SNR and power context embeddings were pre-computed from the training samples and the ASR-loss embedding was based on ITTS speech in the earlier incremental step. ITTS is also trained with batch training, where a batch consists of samples from clean conditions and also noisy conditions.

The speech synthesis in noisy conditions is learned through two cases. The first case is to learn how to change the speech prosody attributes from normal to Lombard speech. This is done by first generating the pre-computed auditory feedback from normal speech with additive noise. The training method is the same as the non-incremental version. The second case is to learn how to maintain the Lombard speech prosody attributes. The training is done using the pre-computed auditory feedback generated from noisy Lombard speech. In this case, the noise condition inside the auditory feedback and the target condition is the same.

6.3.3 Inference Mechanism

6.3.3.1 Basic Inference

Incremental speech synthesis and adaptation are done by generating and utilizing auditory feedback incrementally or progressively, which is illustrated in Fig. 6.2 (b). In the first incremental step, ITTS synthesizes W words speech. In

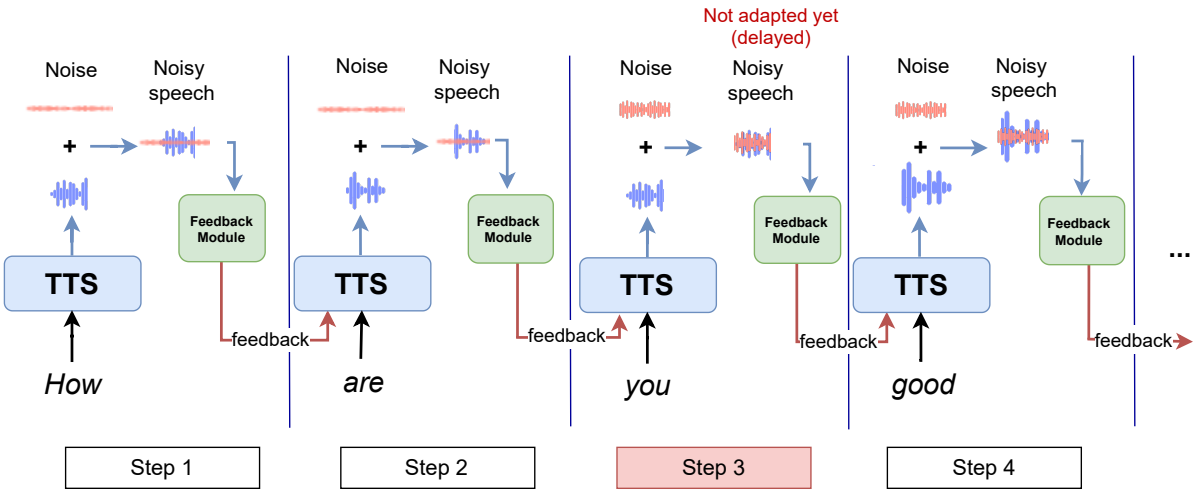


Figure 6.4: An example of delayed adaptation in the proposed ITTS with the basic inference mechanism in a dynamic noise condition.

the second incremental step, we compute the feedback embedding from the first incremental step's speech and use it to synthesize the next W words speech. For the third incremental step and so on, we repeat the same process by taking the previous step's output as the feedback.

The adaptation latency of the proposed ITTS is still high during the basic inference, as shown in Fig. 6.4. ITTS synthesizes a speech segment in an incremental step by looking at the speech generated in the previous step. This implies the adaptation is delayed by one incremental step. In an environment with increasing noise, the unadapted speech segment could have low audibility.

6.3.3.2 Inference with Short-term Intensity Post-adaptation

To address the issue of adaptation latency, we use a simple additional incremental power or intensity modification after the ITTS synthesizes the speech segment. The speech intensity is modified also incrementally on the M ms unit for each ITTS incremental step (Fig. 6.5). The system first plays the M ms speech segment and then estimates the SNR of that segment, which has been fused with noise. The SNR of the most recent M ms speech segment is then used to improve the next M ms speech segment, and so on. Intensity modification is done when the SNR is below a pre-defined threshold.

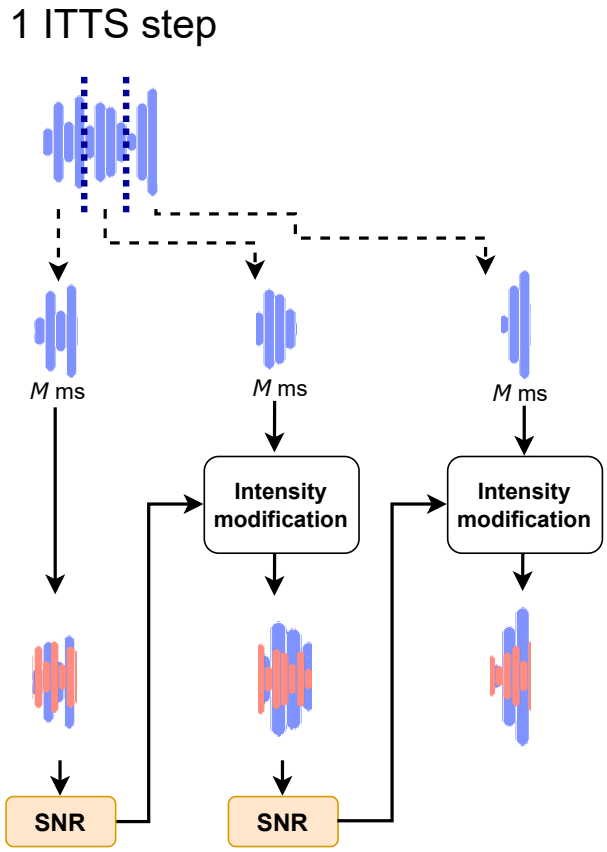


Figure 6.5: ITTS speech intensity post-adaptation in a noisy condition. The process is done incrementally with an M ms unit.

6.4 Experimental Setup

6.4.1 Dataset

6.4.1.1 WSJ Corpus

We trained the proposed ITTS using the same WSJ dataset shown in Chapter 5 for the self-adaptive non-incremental TTS. However, as our ITTS works with a word-level segment, all speech utterances were divided into word segments before starting the training. The speech word segmentation was done by cutting the speech utterance using the list of word timing in the speech. The word timings were extracted by applying a forced-alignment mechanism from the Montreal

forced-alignment toolkit.

6.4.1.2 Hurricane Corpus

In addition to the WSJ dataset, we also conducted experiments using natural Lombard speech data in the Hurricane natural speech corpus [91]. The Hurricane corpus consists of normal and Lombard speech recorded from single male native British-English speaker. The first data, normal speech, was recorded while the speaker reading a transcription in clean environments, which resulted in 2.3 hours of recorded speech consisting of 3034 utterances. The average speech utterance length was 2.67 sec and 7.32 words. The average speech intensity was 65.27 dB. For TTS training, we partitioned the data into 2842 utterances, 72 utterances, and 120 utterances as the training set, development set, and evaluation set, respectively. The second data, Lombard speech, was recorded while the speaker listening to a single static noise via headphones and reading a text. The noise sound was a speech-shaped noise. The Lombard speech audio only contains the speech utterance without the noise recorded. Here, the transcriptions were a subset of the transcription list utilized in the normal speech recording. The Lombard speech data amount was 0.7 hours of speech, which consists of 1020 utterances. The average speech intensity was 70.19 dB. For TTS training, we partitioned the data into 800 utterances as the training set, 100 utterances as the development set, and 120 utterances as the evaluation set, respectively. Here, the speech transcription in the evaluation set for normal and Lombard speech is the same. All speech data was sampled at 16 kHz.

In addition, we also created noisy speech by combining noise sounds with normal speech. Since the noise audio utilized in the original recording was not publicly available, we used the same noise sounds (babble noise and white noise) and SNR (0 dB and -10 dB) as our WSJ data to generate the noisy speech. The SNR here is relative to the normal speech intensity in the Hurricane data. Aside from noise addition, no speech modification was performed on the Hurricane data.

6.4.2 Model Configuration

6.4.2.1 ITTS

The ITTS model configuration, as well as the feedback component architecture, was the same as for the non-incremental TTS. The ITTS incremental unit was three words, with the previous ten words as the look-back input and the next two words as the look-ahead input. In the training material, the average speech segment length in an incremental unit was 1.40 sec. A word sequence was converted into a character sequence before being given to the ITTS.

6.4.2.2 ISR

The ISR architecture is the same as the one used in the previous experiment on non-incremental TTS. In the ITTS, we utilized the ISR trained on short speech segments in which we treat a speech segment as an utterance. The segment length in ASR training was randomized in a range from one to five words.

6.4.2.3 SNR Recognition

The model setting is the same as the one utilized in the previous non-incremental TTS experiment. For the ITTS, we also trained the model on short speech segments. Here, the speech segment length was randomized among lengths of one to five words.

6.4.2.4 Power Recognition

The speech power or intensity recognition model was first trained for the intensity regression task. It consisted of four stacks of convolution and residual blocks and a linear layer. Before ITTS training, this module was trained to do short-speech intensity recognition, where the speech length was randomized among lengths of one to five words. The training label was the speech intensity scaled in the range of -1 to 1.

6.4.2.5 Speaker Recognition

The speaker recognition model to generate speaker embedding has the same architecture and training setting as the model utilized in the non-incremental TTS in Chapter 5. Speaker embedding for ITTS was pre-computed from the target speaker's full-utterance speech.

6.4.3 Intensity Post-adaptation

The power or intensity post-adaptation incremental unit was 200 ms. We modified the speech intensity to reach SNR 20 dB in noisy conditions using the SoX toolkit. SNR estimation was done using the SNR regression model trained on short speech segments, which model is the same model utilized as the SNR embedding module in the proposed ITTS. In this thesis, we do not take computation time into account in the speech performance evaluation.

6.4.4 Evaluation Metrics

We evaluated the proposed systems through objective and subjective evaluation. The objective evaluation is done based on the CER and STOI measures. On the other hand, the subjective evaluation was performed by asking human evaluators to evaluate the TTS speech. The speech aspects evaluated in the subjective evaluation were speech naturalness and speech intelligibility.

- **Subjective speech intelligibility test**
  A speech intelligibility test was carried out by asking the human listener to write a transcription of noisy speech. In this test, we used a semantically unpredictable sentence (SUS) [92] as the TTS input text. SUS is a syntactically correct but semantically unpredictable sentence. This ensures that the listener does not guess the unintelligible speech based on the sentence context.

- **Subjective speech naturalness test**
  A speech-naturalness evaluation was done through a mean opinion score (MOS) test by asking the listener to score the speech naturalness on a scale of 1-5 points. The speech signals were also mixed with noises. The sentences

used in the MOS test were the normal sentences obtained from the WSJ evaluation set.

6.5 Experiment Results

We evaluated our systems under static noise and two types of dynamic noise conditions as shown in Fig. 6.1 : 1) switch noise and 2) smooth noise, as well as in the static-noise condition. In switch noise condition, the noise intensity changed without transition, while in smooth noise conditions the noise intensity changed gradually.

6.5.1 WSJ

In this section, we discuss the experiment results with TTS models that were trained on WSJ data. The evaluated systems were the baseline TTS model shown in Chapter 5, the proposed non-incremental self-adaptive TTS in Chapter 5, and the proposed ITTS in this chapter.

6.5.1.1 Speech Intelligibility in Character Error Rate

The TTS intelligibility measured in ASR CER are shown in Table 6.1. CER was measured by transcribing the synthesized speech using non-incremental multi-condition WSJ ASR. The ASR feedback for the proposed non-incremental TTS was generated with the same method in Chapter 5.6.1 with the multi-condition ASR, while the ASR feedback for ITTS was generated using the same ISR in Chapter 6.4.2.2. In this experiment, the proposed ITTS synthesized the speech with the basic inference mechanism by default. The synthesis with the intensity post-adaptation mechanism is shown in the row with '*+intensity post-adaptation*' below the corresponding model.

Table 6.1: Average TTS speech intelligibility (CER%) on WSJ data in babble- and white-noise conditions evaluated using multi-condition WSJ ASR. SNR embedding in the proposed systems was generated using SNR classification (cls) or regression (reg).

System	**Static Noise**			**Dyn. Switch Noise**		**Dyn. Smooth Noise**	
	Clean	**SNR 0**	**SNR -10**	**Clean, SNR 0, SNR -10**	**SNR 0, Clean, SNR -10**	**Clean, SNR 0, SNR -10**	**SNR 0, Clean, SNR -10**
Baseline Non-incremental TTS							
Standard TTS (Clean)	18.32	70.54	77.07	53.64	49.77	47.39	45.11
+ Fine-tuning	14.82	21.99	37.41	20.30	20.33	19.41	18.74
Proposed Non-incremental TTS + SNR + ASR + var.adaptor							
TTS + SNR (cls) (speak 5x)	11.99	14.70	24.96	61.94	70.98	28.09	17.88
TTS + SNR (reg) (speak 1x)	14.76	32.91	56.42	27.88	26.60	28.48	28.22
+ intensity post-adaptation	14.76	21.43	27.23	20.32	20.57	20.22	19.70
TTS + SNR (reg) (speak 5x)	13.52	18.57	31.19	16.95	18.70	18.00	17.57
+ intensity post-adaptation	13.52	16.16	22.37	14.54	14.62	13.94	12.97
Proposed Incremental TTS (ITTS) + SNR (reg) + ASR + var.adaptor							
PCI-ITTS (speak 1x)	18.96	38.26	60.64	34.90	32.00	34.02	33.69
PCD-ITTS (speak 1x)	14.42	23.32	41.89	26.96	28.13	23.48	22.53
+ intensity post-adaptation	14.42	20.59	31.05	20.64	17.30	20.10	20.99
Topline (human natural speech)							
Normal speech	7.43	22.17	58.81	32.10	32.93	15.04	14.97
+ Rule-based modification	7.43	13.24	15.15	22.41	23.25	12.37	12.60
Lombard speech	7.43	11.46	20.46	22.92	17.77	-	-

First, we evaluated the non-incremental TTS intelligibility. We ran our proposed TTS to speak five times at most, assuming that the noise conditions in all re-speaking attempts were the same. The evaluated model was the same as the model presented in Chapter 5. Then, we evaluated the proposed ITTS with one-time speaking. The details of system performance in each environment are below.

Static Noise Condition The non-incremental system performance reported in this experiment is the same as the one reported in the Table 5.2 in Chapter 5 with the multi-condition ASR. The TTS intelligibility improved with a higher number of re-speaking attempts and intensity post-adaptation.

In incremental speech synthesis, given that intensity post-adaptation was not applied, PCD-ITTS in static noises performed closely to or better than the proposed non-incremental TTS that spoke only once. PCD-ITTS produced more intelligible speech than the PCI-ITTS. This indicates that power context embedding in PCD-ITTS is critical to maintaining speech intelligibility. When we manually inspect the utterances made under the static noise conditions, we hear an intensity fluctuation in the PCI-ITTS speech. For example, PCI-ITTS speaks with a Lombard effect at the initial incremental step, louder than the noise. In the next step, it produces a speech segment with a reduced intensity since the SNR might suggest a less noisy condition. Meanwhile, the PCD-ITTS tracks the previous speech intensity so that the system has better control of the speech. When intensity post-adaptation was applied, PCD-ITTS was able to perform closely to the non-incremental TTS that require a full sentence input and also synthesize the speech through several re-speaking attempts.

Dynamic Noise Condition In dynamic noise, the proposed non-incremental TTS with SNR regression outperformed TTS with SNR classification. As mentioned in Chapter 5.5.1.3, the SNR regression model was trained using speech with static and dynamic noise added for a more accurate and flexible SNR recognition model. The SNR regression error in dynamic noise with natural speech was 2.1 dB. On the other hand, TTS with SNR classification focused on the use case in static conditions with static noise training data. In dynamic noise conditions, the SNR classification performance degraded, notably in dynamic switch noise

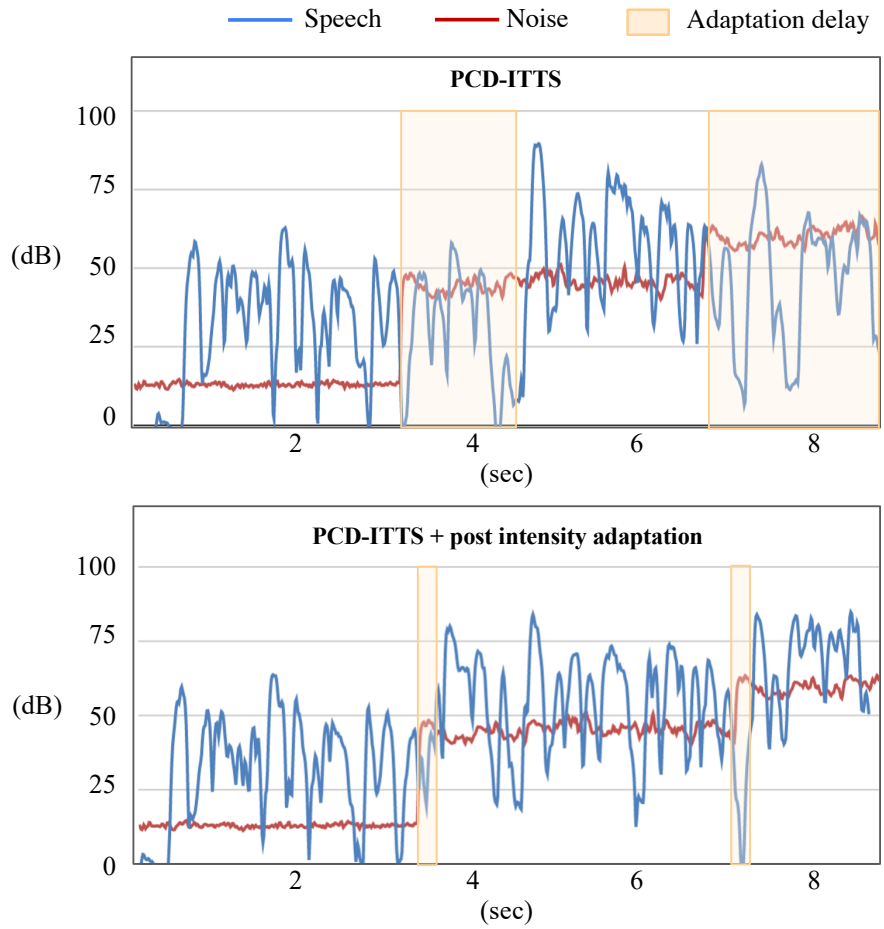


Figure 6.6: PCD-ITTS speech intensity with and without the intensity post-adaptation with a 200-ms incremental unit in the dynamic switch noise condition.

conditions. In that condition, SNR classification accuracy was 0%, in which the SNR class was constantly predicted as 'clean' in our experiment. We analyzed that this is related to the portion of clean speech in the noisy speech. In our experiment, one-third of the dynamic switch noisy speech was clean speech. On the other hand, in dynamic smooth noise, the SNR was classified as 'SNR 0' because the clean speech part was shorter than the dynamic switch noise due to the gradual noise transition. The SNR classification performance, as well as the corresponding proposed TTS, in dynamic switch noise conditions could be improved by including the dynamic switch noise in the SNR classification training data

(see Appendix A). The SNR classification model, however, is less customizable than the regression model because we have to train the model each time the SNR class changes, meanwhile SNR is a real value. In further experiments, we focus on systems with SNR regression feedback only.

In dynamic switch and smooth noise conditions, PCD-ITTS outperformed the PCI-ITTS. However, PCD-ITTS with the basic inference mechanism had an adaptation latency issue, where the adaptation was delayed by one incremental step (Fig. 6.6). The average speech segment length produced by our system in an incremental step was 1.11 sec on average. Here, PCD-ITTS with the basic inference shows a lower intensity than the noise in the undapted part, which occurs when the noise intensity changes, so ITTS could not catch up with those changes. After ITTS obtained the auditory feedback based on the unadapted part, ITTS could improve the intelligibility in the subsequent incremental steps. But by applying an intensity post-adaptation, we were able to reduce the adaptation latency, which was from 1.11 sec on average to 200 msec, and improve the PCD-ITTS intelligibility.

6.5.1.2 Speech Intelligibility in Short-term Objective Intelligibility Measure

Table 6.2 shows the STOI measurement of our system's Lombard speech under noisy conditions. Interestingly, PCD-ITTS had a better STOI than the non-incremental TTS, but for intelligibility in ASR CER, this relationship was reversed. Our analysis suggests that this is related to speech intelligibility as a signal and sentence. PCD-ITTS speech signals before and after disturbance with noises show a high correlation, implying the speech signal is audible in noises, for example, because the speech is very loud. But its comprehensibility as a sentence is not as high as the non-incremental TTS. This is because the proposed non-incremental TTS was allowed to synthesize the speech by using a complete sentence's text with the complete context and sentence-level feedback with the re-speaking. But then PCD-ITTS with intensity post-adaptation achieved a close ASR CER and higher STOI to the non-incremental TTS, although PCD-ITTS did not perform re-speaking. This illustrates how speech adaptation within a short time frame improved the speech intelligibility in incremental speech synthesis.

Table 6.2: STOI scores (%) on WSJ

System	Static Noise		Dynamic Switch Noise	Dynamic Smooth Noise
	SNR0	SNR-10		
Baseline Non-incremental TTS				
Standard TTS(Clean)	49.69	38.07	65.13	71.93
+ Fine-tuning (SNR0 + SNR -10)	83.56	71.07	89.97	92.10
Proposed TTS + SNR (reg) + ASR + variance adaptor				
Non-incremental TTS	87.57	76.08	91.76	92.17
Incremental TTS (PCI-ITTS)	82.62	65.87	79.00	76.44
Incremental TTS (PCD-ITTS)	89.95	77.04	84.31	82.71
+ intensity post-adaptation	93.00	82.85	94.46	94.58
Topline (human natural speech)				
Normal speech	60.31	47.57	76.06	88.88
+ Rule-based modification	90.48	86.99	84.34	93.84

6.5.1.3 Subjective Evaluation

In the next experiment, we evaluated our system through a subjective evaluation of speech intelligibility and naturalness. The SUS intelligibility and MOS tests were done through crowd-sourcing, with 61 participants for the intelligibility test and 138 participants for the MOS test. All participants were located in the United States.

In this work, we mainly focused on improving TTS speech intelligibility. In related work, it has been suggested that speech intelligibility and naturalness do not always imply each other [93], and thus improvement in intelligibility might not necessarily improve naturalness. Overall, our subjective evaluation results also revealed that the proposed systems achieved a significant improvement in speech intelligibility while preserving speech naturalness. The details are below.

Speech intelligibility The SUS intelligibility test results in CER are shown in Table 6.3, and they are also visualized in Fig. 6.7. We conducted a statistical t-test to show the significance of the improvement in the proposed system by

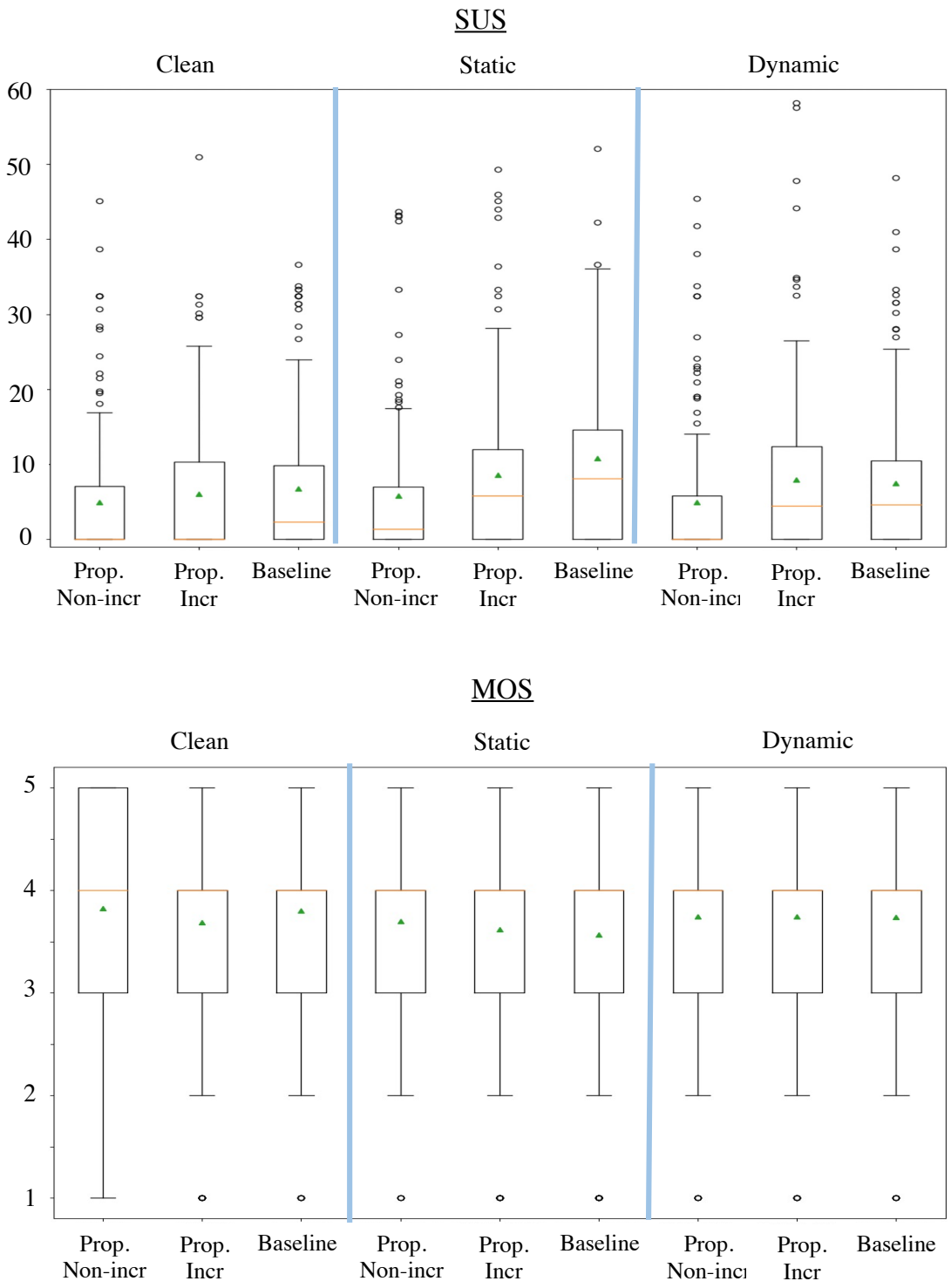


Figure 6.7: SUS intelligibility and MOS naturalness scores.

Table 6.3: SUS intelligibility evaluation results in CER (%) ($\star$: statistically different from the baseline in the same environment)

System	**Noise**	**Objective (ASR)**	**Subjective (Human)**
Baseline Non-incremental TTS			
Fine-tuning (SNR 0 + SNR -10)	Clean	21.76	6.74
	Static	35.31	10.80
	Dynamic	27.69	7.44
Proposed TTS + SNR (reg) + ASR + variance adaptor			
Non-incremental TTS	Clean	14.72	4.94 $\star$
	Static	16.94	5.78 $\star$
	Dynamic	14.29	4.94 $\star$
Incremental TTS (PCD-ITTS + intensity post-adaptation)	Clean	17.73	6.05
	Static	26.26	8.58 $\star$
	Dynamic	24.14	7.92

comparing this system to the fine-tuned baseline system in the same environment. The significance level was 0.05. In Table 6.3, the systems with a statistically different result against the baseline are marked with a star "$\star$". TTS speech was generated based on three noisy conditions: 1) clean, 2) static noise from the SNR -10 dB condition, and 3) dynamic smooth noise consisting of noise transitions from clean, SNR 0 dB, and then SNR -10 dB noises. We also present the ASR CER as the objective measure. Here, the PCD-ITTS was applied with intensity post-adaptation to shorten the adaptation latency. We did not use the intensity post-adaptation in the non-incremental system to see how our basic framework would perform, with the speech improvement solely done within the TTS. Based on the evaluation results, in the clean condition, the proposed non-incremental TTS was more intelligible than the other systems, while the PCD-ITTS and the baseline TTS performed similarly. In the static noise condition, all proposed systems were also more intelligible than the baseline. In the dynamic noise condition, the proposed non-incremental TTS showed the best intelligibility performance.

Table 6.4: MOS evaluation results ($\star$: statistically different from the baseline in the same environment)

System	Noise	MOS
Baseline Non-incremental TTS		
Fine-tuning (SNR 0 + SNR -10)	Clean	3.80
	Static	3.56
	Dynamic	3.74
Proposed TTS + SNR (reg) + ASR + variance adaptor		
Non-incremental TTS	Clean	3.82
	Static	3.70
	Dynamic	3.74
Incremental TTS (PCD-ITTS + intensity post-adaptation)	Clean	3.69
	Static	3.61
	Dynamic	3.75

Speech naturalness The MOS scores are shown in Table 6.4 and in Fig. 6.7. We performed a Mann-Whitney U statistical test whose results show that the presented systems have statistically similar MOS scores, indicating that they preserved naturalness. Here, the proposed non-incremental TTS achieved the highest average score in general. PCD-ITTS naturalness was lower than that of the proposed non-incremental TTS. When we inspected the audio, the naturalness degradation was mostly caused by speech discontinuities in the ITTS speech, which often occur in incremental speech synthesis. But by incorporating feedback into the system, our ITTS achieved higher average scores under noisy conditions than did the non-incremental baseline system, which also spoke once and loudly. This also demonstrates that auditory feedback has a positive impact on incremental speech synthesis.

6.5.2 Hurricane

In this section, we show the TTS performance trained on the Hurricane dataset. Since the data was too small, we trained the TTS model by initializing the model parameters using the WSJ-based model in the same system framework.

6.5.2.1 Speech Intelligibility in Character Error Rate

Table 6.5: Average TTS speech intelligibility (CER%) on Hurricane data in babble- and white-noise conditions based on multi-condition training ASR.

System	Static Noise			Dyn. Switch Noise		Dyn. Smooth Noise	
	Clean	SNR 0	SNR -10	Clean, SNR 0, SNR -10	SNR 0, Clean, SNR -10	Clean, SNR 0, SNR -10	SNR 0, Clean, SNR -10
Baseline Non-incremental TTS							
Standard TTS + Fine-tuning	8.95	24.94	40.76	34.18	30.77	24.32	27.57
Proposed TTS + SNR (reg) + ASR + var.adaptor							
Non-incremental TTS	10.53	19.18	30.63	24.78	26.07	20.70	22.22
Incremental TTS (PCD-ITTS)	13.77	24.82	37.23	50.52	35.52	37.66	26.02
+ intensity post-adaptation	13.77	23.79	33.85	40.83	33.15	32.34	24.15
Topline (human natural speech)							
Normal/Lombard speech	6.85	15.40	29.16	22.17	22.90	15.61	16.49

The overall TTS speech intelligibility scores in CER are shown in Table 6.5. CER was measured by using a non-incremental multi-condition ASR trained on WSJ and Hurricane data.

The best intelligibility was achieved by the proposed non-incremental TTS. In noisy conditions, it showed close intelligibility to natural speech and outperformed the baseline. But in clean conditions, its performance was slightly behind the baseline model. This was because the baseline always produced Lombard speech, while the proposed TTS spoke with a normal style without the Lombard effect in the clean conditions, like humans do. Naturally, Lombard speech is more intelligible than normal speech when noises are removed [87]. However, in the previous WSJ results, the proposed TTS in clean conditions was better than the baseline. It could be affected by the multi-speaker data condition in WSJ, which is more challenging than the single-speaker. The Hurricane data only consisted of single-speaker speech, and it resulted in higher baseline performance. Here, the proposed non-incremental TTS was able to produce more intelligible Lombard speech and also produce normal speech with close intelligibility to the baseline Lombard speech.

PCD-ITTS aided with the intensity post-adaptation was also able to improve the incremental speech synthesis performance on the Hurricane data. In the

dynamic noise experiments, PCD-ITTS without the intensity post-adaptation could not achieve a CER as low as the CER in the WSJ results because the sentence length in the Hurricane data was too short. The average sentence length was 7 words, while the PCD-ITTS incremental step was 3 words. Therefore, the speech might have ended before the adaptation could start. Adding the intensity post-adaptation mechanism sufficiently improved the PCD-ITTS performance.

6.5.2.2 Speech Intelligibility in Short-term Objective Intelligibility Measure

Table 6.6: STOI scores (%) on Hurricane data.

System	Static Noise		Dynamic Switch Noise	Dynamic Smooth Noise
	SNR0	SNR-10		
Baseline Non-incremental TTS				
Standard TTS + Fine-tuning	81.61	64.63	85.92	88.21
Proposed TTS + SNR (reg) + ASR + variance adaptor				
Non-incremental TTS	88.68	75.91	91.70	93.73
Incremental TTS (PCD-ITTS)	87.15	71.68	79.25	84.73
+ intensity post-adaptation	87.27	72.55	80.20	85.86
Topline (human natural speech)				
Normal/Lombard speech	83.36	63.98	85.02	88.68

TTS STOI scores are shown in Table 6.6. In the static noise conditions, all proposed systems resulted in a higher STOI than the baseline and the topline. This shows that the speech signal correlation before and after being disturbed by noise was high. In this condition, PCD-ITTS with intensity post-adaptation achieved a close STOI to the proposed non-incremental TTS. The proposed non-incremental TTS required re-speaking to adapt the speech, while PCD-ITTS only spoke once and adapted the speech in one attempt. In the dynamic noise conditions, the proposed non-incremental TTS achieved the highest STOI scores. The low STOI on PCD-ITTS in dynamic noise conditions could be affected by

the utterance length as stated previously. The noise changed quicker than the noises in the WSJ experiments, and it was more challenging for the incremental speech synthesis. Here, intensity post-adaptation also successfully improved the PCD-ITTS STOI.

6.6 Summary

Low-latency adaptation and environmental understanding are important keys for a machine's self-adaptation in a real environment. In this chapter, we showed a machine speech chain inference mechanism for ITTS for low-latency speech adaptation in noisy conditions. The speech synthesis and adaptation were done progressively by using the past output as feedback. Here, the best ITTS was achieved by a system that used SNR, ASR-loss, and intensity context as the feedback, along with intensity post-adaptation. Although incremental speech synthesis was challenging, the proposed ITTS was able to perform without significant performance loss to non-incremental TTS even though the latency was shorter. The proposed ITTS was able to produce highly intelligible speech by performing dynamic adaptations according to environmental changes at low latency, thus enabling the TTS to more closely resemble the human speech chain mechanism and improve speech quality. Our experimental results reveal that dynamic adaptation with auditory feedback could be an essential tool for optimal speech generation by machines.

Chapter 7

Conclusions and Future Direction

In this chapter, we conclude our thesis and discuss future directions for the proposed framework.

7.1 Problem Reiteration

In human spoken communication, speaking and listening are inseparable tasks. The connection between them, which is reflected in a speech chain mechanism, plays an important role in language acquisition and speech production. The development and performance of one component affects the other. The concept of speech chain shows that, during a conversation, humans can improve and adapt their speech by listening to their speech and also by considering other factors, for example, environmental noises. Despite its importance in the human communication system and its potential, the relationship between machine speech production (TTS) and machine speech perception (ASR) systems gets less attention in the research community. The general TTS and ASR perform the task separately. Although they can perform well, it is only limited to systems trained using a large amount of paired speech-text data and performing under ideal conditions. The current systems cannot grasp environmental changes, causing performance degradation in realistic conditions, such as in noisy places.

Among many TTS and ASR works, the basic machine speech chain was proposed for TTS and ASR joint development using an auditory feedback connection. It was motivated by the human speech chain mechanism. This framework allows

TTS and ASR to be trained using unpaired speech and text data through a closed feedback loop mechanism, in which it successfully improves the components' performance in semi-supervised training conditions. However, the feedback connection is discarded in inference. Therefore, TTS and ASR still perform separately and leave the self-adaptation problem to remain.

7.2 Conclusions

In this section, we review our work from the perspective of theoretical, application, and experimental issues.

7.2.1 Theoretical Issues

We take advantage of the relationship between speech production and perception in humans to develop a spoken language processing system that can do self-adaptation. We generalized the idea of a self-adaptive and incremental machine speech chain by connecting TTS, ASR, and another listening component, which was an SNR recognition system in this thesis. The proposed self-adaptive framework aims to synthesize speech by dynamically adapting to noise in the inference environment. It follows a similar mechanism in the human Lombard effect, in which humans adjust their speaking effort to increase their speech intelligibility in a noisy place based on the auditory feedback.

7.2.2 Application Issues

We showed some proof-of-concept of our proposed self-adaptive and incremental machine speech chain, starting from the previously proposed basic machine speech chain by Tjandra et al. [2, 36] for non-incremental TTS and ASR semi-supervised training. First, we reduced the components' latency in the basic framework by using ITTS and ISR as the machine speech chain components. Second, we showed a machine speech chain mechanism for non-incremental TTS inference to improve TTS intelligibility in a noisy environment. The system performs self-adaptation based on environmental noise by using auditory feedback consisting of SNR and ASR loss information. Lastly, we developed a self-adaptive ITTS with a machine

speech chain inference mechanism, in which the system successfully performed incremental speech synthesis and noise-adaptation at a low latency.

7.2.3 Experimental Issues

We verified the proposed approaches in speech synthesis and recognition tasks in clean and noisy environments. In Chapter 4, we reported ITTS improvement from a Mel L2 loss of 1.04 into 0.86 and 0.79 using the unsupervised chain and supervised chain, respectively, in the incremental machine speech chain training. Here, the ITTS incremental unit was 30 characters, and it was able to perform closely to non-incremental TTS with the 103 characters latency. ISR performance also improved from CER 17.81% into 14.23% using the unsupervised chain and 9.43% with the supervised chain. ISR latency was 0.84 sec and its performance approached those of non-incremental ASR with a 7.88 sec latency. Then, in Chapter 5, our TTS with the machine speech chain inference achieved speech intelligibility of 14.70% and 24.96% CER in SNR 0 dB and SNR -10 dB conditions, outperforming the baseline with the offline fine-tuning. Finally, in Chapter 6, our self-adaptive PCD-ITTS with intensity post-adaptation was able to adapt to the noises with a latency below 1 sec, achieving a speech intelligibility of 6.05%, 8.58%, and 7.92% in CER, evaluated by humans in the clean, static, and dynamic noise conditions. Here, the self-adaptive non-incremental TTS with a latency of 7 sec performed with a CER of 4.94%, 5.78%, and 4.94% in clean, static, and dynamic noise conditions through re-speaking attempts.

7.3 Summary of Contribution

The original contributions of this thesis are listed as follows:

- **A new framework for incremental TTS and incremental ASR construction with a short-term closed feedback loop (in Chapter 4)**
  We showed that ITTS and ISR systems could jointly improve during training by establishing a short-term closed feedback loop between them. These systems have shorter latency than the standard systems. Therefore, the waiting time required when taking a long sequence input is not long. The

proposed framework not only improved over the natural speech and text inputs, but also synthetic inputs generated by the machine. This shows that the feedback loop has widened the input domain from natural only to natural and synthetic, which could be important in feedback-based systems.

- **Synthetic multi-speaker Lombard data construction (Chapter 5)**
  Since the availability of Lombard speech data is limited, we constructed a multi-speaker Lombard speech dataset by modifying normal speech into Lombard speech. This was done by first observing natural speech intensity, pitch, and duration differences between normal and Lombard speech.

- **A new machine speech chain training and inference framework for self-adaptive TTS in noisy condition based on auditory feedback (Chapter 5)**
  We showed that TTS speech dynamic adaptation in noisy conditions could be done through a speech chain mechanism. The proposed TTS synthesizes speech by not only taking the text input but also the auditory feedback representing the environmental noise information. With that, TTS is able to decide the characteristics of speech by itself to improve the intelligibility in noisy situations.

- **ASR-loss and SNR embedding for TTS feedback (Chapter 5)**
  ASR loss and SNR embeddings have proven to be effective in supporting TTS self-adaptation in noisy conditions. The ASR loss embedding denotes the TTS speech intelligibility, while the SNR embedding represents the noise situations. These embeddings are estimated from the noisy TTS speech in an end-to-end manner without the complicated mechanism.

- **Machine speech chain training and inference mechanism with ITTS for low-latency adaptation (Chapter 6)**
  The proposed ITTS in machine speech chain inference mechanism performs speech synthesis and adaptation with a low latency. Since environmental noises are often dynamic, a low-latency adaptation is necessary to catch up with the dynamic changes. Our experiment shows that the proposed ITTS was able to perform adaptation in 1 sec.

- **ITTS post-intensity adaptation method for a faster adaptation (Chapter 6)**
  We enabled a faster adaptation in the self-adaptive ITTS by applying post-intensity adaptation to the synthesized speech segment. In the basic self-adaptive ITTS inference, an adaptation latency of 1 sec affected the speech intelligibility, in which the unadapted segment performed poorly. The post-intensity adaptation method measures the SNR of a 200 ms speech segment and uses the result to convert the next speech intensity to an appropriate level incrementally, and results in intelligibility improvement.

7.4 Future Directions

The reference point for our self-adaptive TTS system in noisy conditions is the human Lombard effect. The human Lombard effect does not increase the vocal effort uniformly throughout the utterance, but it depends on the content and context of speech and the environment in a broader context than just noise. Human Lombard speech is a result of two kinds of speech adjustment mechanisms: private loop and public loop [11]. Private loop is a mechanism to regulate the vocalization and the speech fluidity based on the speaker's own hearing or auditory feedback, which occurs involuntarily. In the Lombard effect, the private loop is manifested as a reflex to speak louder in regards to the noise intensity. On the other hand, the public loop is a vocal regulation mechanism at a higher cognitive level, which is based on the observation on the audience or the context. For example, the Lombard effect has more influence on the words that are important for the audience to understand the speaker. More generally, public loop is also associated with the paralanguage or emotion expressed in the speech in order to engage the audience. Inevitably, public loop affects the private loop. Private and public loops are performed simultaneously based on the real-time conditions. Therefore, humans can quickly adjust their speech according to the situation and their purpose to communicate effectively.

In this thesis, the proposed system still has a gap compared to the reference in respect of mechanism and output performance. First, for the mechanism, the proposed systems currently only use the analogy of the private loop mechanism

for self-adaptive speech synthesis by considering only the SNR and ASR feedback and improving the synthesized speech signal intelligibility in general. The proposed system has not considered other factors yet, for example, the speech content and cognitive-level factors similar to the public loop in the human Lombard effect. Second, for the output quality, we are currently focused on improving speech intelligibility and lowering the speech synthesis and adaptation latency with the baseline of standard TTS. Although the proposed systems outperformed the baseline, they are still behind the quality of human speech. To make the system output quality closer to the reference point, we also have to improve the naturalness and prosodic patterns (e.g., intonation, co-articulation) with regard to the speech content and environments in a broader context. The more similar the natural and synthesized Lombard speech are with respect to the subjective evaluation by humans (intelligibility, naturalness, latency, intonation), the more it indicates that the machine has become closer to the reference point in terms of performance.

In the future, we can improve the proposed system to make it perform more closely to the reference point and also expand the applicability of the concept of machine speech chain in more broader tasks.

7.4.1 Short-term Future Works

In the short-term future works, we may improve the proposed TTS' Lombard speech quality without changing the main part of the proposed framework.

- **Synthetic Lombard speech data construction with spectral modification**

  The current synthetic Lombard speech data was constructed only by modifying the speech prosody attributes (intensity, pitch, and duration). However, the human Lombard effect does not only affect those prosody attributes but also includes spectral modifications, for example, spectral tilt, amplitude modulations, format shift, and energy shift from low frequency bands to higher bands. In the next work, we can improve the synthetic Lombard speech data construction by applying spectral modification operations.

- **Multi-speaker natural Lombard speech data collection**
  Lombard effect is speaker- and gender-dependent [94]. Therefore, we would like to do multi-speaker natural Lombard speech data collection for a better Lombard speech analysis in the synthetic Lombard speech construction. Furthermore, we intend to consider a better speech modification approach than the current prosody modification with SoX. We also intend to carry out TTS training using the multi-speaker natural Lombard speech data.

- **Speech generation with neural vocoder**
  We currently generate the Mel-spectrogram using the proposed TTS with the Griffin-Lim vocoder. Therefore, we could also expect more improvement using an advanced neural vocoder, such as WaveRNN [95] or HiFi-GAN [96], which we plan to investigate in future work.

7.4.2 Long-term Future Works

In the long-term future, we go deeper on the machine speech chain mechanism (Fig. 7.1). We address several things the current proposed framework cannot do, such as:

- **Flexible latency**
  The proposed ITTS synthesizes the speech based on the fixed incremental unit (in word unit) learned during the training. We have to train the system each time we change the incremental unit, for example, into a shorter unit to decrease the delay or into a longer unit to improve the speech quality. In the next work, we can consider self-adaptive ITTS with robust incremental units, such as changing the incremental unit based on the user requirements flexibly or adjusting the incremental unit based on the textual content and the environmental situations.

- **Self-adaptive speech synthesis based on cognitive-level feedback and public loop**
  One of the possible improvements that we can make for the proposed self-adaptive system to perform closer to our reference point is by expanding the "awareness" of the machine into broader contexts of environments. We

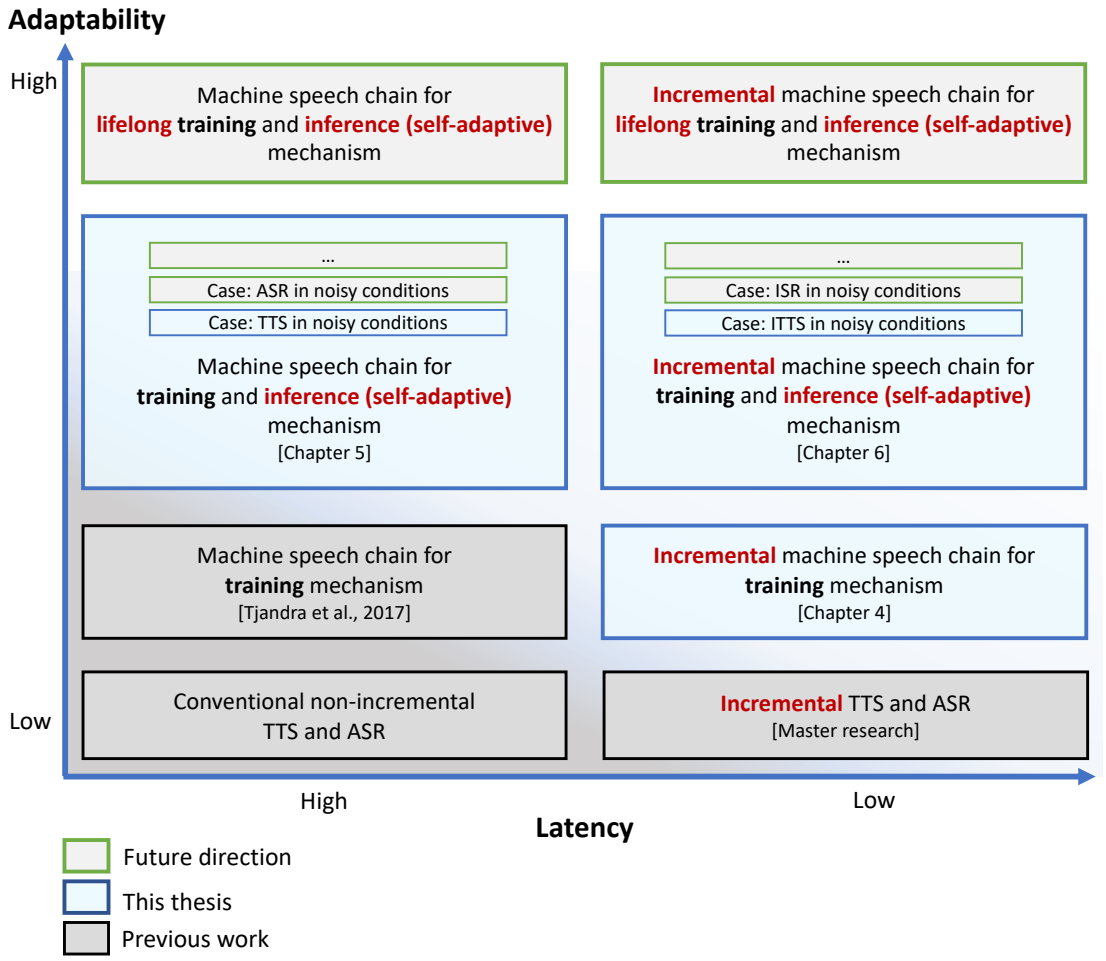


Figure 7.1: Long-term research direction.

may extend the current system with modules that provide cognitive-level information by using the analogy of the public loop in the human speech production mechanism. For example, we can implement new modules that analyze the speech content to improve the speech signal with regard to the important words by considering the situation and the target audience. We may also use a speech emotion feedback module to regulate the speech prosody based on the target emotion, considering the purpose of the speech, the target audience, and situation. We can also implement a module that captures the audience's response, such as an audio response or visual re-

sponse, so the system can regulate the speech vocalization by understanding the target audience and their response better.

- **Self-adaptation with an online learning approach**
  The proposed self-adaptive system is trained only through supervised training. Under unseen conditions, the performance might degrade (see Appendix A and Appendix B). But in reality, noises vary in type and intensity. Moreover, the dynamics of the environment can be extreme and unpredictable, and we could not cover all conditions in the labeled training data. On the other hand, humans can adapt to the situation robustly in various kinds of conditions. This is enabled by humans' life-long learning, in which, for the speaking skills, they learn at the same time as speech adaptation so they can behave similarly or better when speaking in similar conditions in the future. In the next task, we are interested in implementing a machine speech chain framework to improve the system's robustness and adaptability by adding an online learning mechanism to the framework. For example, we can combine the concept of the basic machine speech chain for unsupervised training (Chapter 3 and Chapter 4) and the proposed self-adaptive machine speech chain inference (Chapter 5 and Chapter 6).

- **A proof-of-concept on other kinds of task**
  This thesis focuses on a self-adaptive machine speech chain inference specifically for speech synthesis in noisy situations. We may also apply the concept of the framework to other speech synthesis cases by modifying the feedback modules. For example, we can use it for an expressive speech synthesis by replacing the SNR feedback module in the current framework with an emotion feedback module based on the emotion recognition task. We might also apply the concept of a feedback mechanism to a self-adaptive ASR, for example, speech recognition in noisy conditions. Humans might adapt their focus to listen to the target speech when having a conversation in a place with multiple sounds, such as noise or speech from other people. This is also known as the cocktail party effect. To realize this in the machine, we can connect the ASR with feedback modules such as TTS, SNR recognition, or speaker recognition. Also in another example, we can build a self-adaptive

ASR for code-switching speech by connecting the ASR with a TTS feedback module and a language identification feedback module that compares the language information that is identified based on the input speech and the output text. Another task that could benefit from the concept of auditory feedback mechanism is signal enhancement, such as noise removal using the ASR output difference between the original and enhanced audio.

Appendices

Appendix A

Further analysis on SNR recognition

A.1 SNR Recognition Model Training Data

The SNR recognition model (classification and regression) was trained using normal speech in the WSJ dataset with additive noise. The noise types were babble- and white-noise. The SNR classification model's class consisted of clean, SNR 0 dB, and SNR -10 dB. The SNR regression output value range was -1 to 1 (normalized SNR). This experiment compares TTS intelligibility based on the SNR recognition model training condition.

- **Static noise only**: The SNR recognition model was trained using clean speech and noisy speech with static noise. The SNR of noisy speech was SNR 0 dB or SNR -10 dB. The data size was 60 hours for clean conditions and 120 hours each for SNR 0 dB and SNR -10 dB.
- **Static noise and dynamic noise**: The SNR recognition model was trained using clean speech, noisy speech in static noise (SNR 0 dB and SNR -10 dB), and noisy speech in dynamic-switch noise. The dynamic-switch noise patterns (8 patterns) were:
  - SNR -10 dB → SNR 0 dB → clean
  - SNR -10 dB → clean

- SNR 0 dB → SNR -10 dB
- SNR 0 dB → clean
- SNR 0 dB → SNR -10 dB
- Clean → SNR -10 dB
- Clean → SNR 0 dB
- Clean → SNR 0 dB → SNR -10 dB

The SNR values in the patterns above are relative to the normal speech intensity in WSJ. In the SNR classification model, we keep the SNR class member as in the static noise classifier. The SNR class label was based on the initial SNR in the pattern. For example, a noisy speech with the pattern of "clean → SNR 0 dB → SNR -10 dB" was labeled as a "clean" class. On the other hand, the SNR regression label was based on the average SNR value in an utterance. The data amount for each dynamic-switch noise pattern was 15 hours, making a total of 120 hours for 8 dynamic-switch noise conditions. The data amounts for clean speech and static noisy speech were the same as training data based on static noise only.

A.2 SNR recognition performance

We evaluated the SNR recognition performance, which is shown in Table A.1. The details are below.

A.2.1 Static noise only training

Both of the SNR classification and regression models performed well on input in static noise conditions but could not correctly recognize the SNR in dynamic noise conditions. First, the SNR classification model constantly classified the input with the dynamic noise as "clean". Similarly, SNR regression outputted an SNR value that was close to the SNR in the clean condition.

Table A.1: SNR classification and SNR regression output based on natural speech with additive static and dynamic noises.

SNR recognition model and training condition	**Static**		**Dynamic-switch**	
	SNR 0	**SNR -10**	**Clean, SNR 0, SNR -10 (ground truth: -5.16 dB)**	**SNR0, Clean SNR -10 (ground truth: -3.02 dB)**
SNR (cls) - chance rate				
Static noise	SNR 0: 100%	SNR-10: 100%	Clean: 100% SNR 0: 0% SNR -10: 0%	Clean: 100% SNR 0: 0% SNR -10: 0%
Static + dynamic noise	SNR 0: 100%	SNR -10: 100%	Clean: 70.57% SNR 0: 29.28% SNR -10: 0.15%	Clean: 88.59% SNR 0: 10.81% SNR -10: 0.60%
SNR (reg) - average SNR output				
Static noise	-1.11 dB	-11.21 dB	39.98 dB	37.07 dB
Static + dynamic noise	-0.52 dB	-10.95 dB	-5.51 dB	-6.89 dB

A.2.2 Static and dynamic noises training

The SNR recognition model trained on static and also dynamic noises improved the model's performance on dynamic noisy speech input. In the SNR classification, the chance for dynamic noisy speech to be predicted as non-clean speech was increased. This might lead to the resulting SNR embedding helping the proposed TTS produce a Lombard speech in dynamic noise conditions. The SNR regression performance on dynamic noisy speech also improved significantly, in which the predicted SNR was close to the actual SNR value.

A.3 TTS performance

Table A.2 compares TTS intelligibility using the SNR recognition models shown in Table A.1. In this experiment, the TTS model was trained based on clean and static noise conditions only.

As expected, TTS that was connected to an SNR recognition model trained in static condition only could not perform well in the dynamic noise conditions because the speech condition was constantly predicted as a clean condition. Then, the SNR recognition models trained on static and dynamic noises successfully improved the TTS intelligibility in dynamic noise conditions. This experiment result shows that expanding the domain of the SNR recognition model could make

the TTS perform well in dynamic noise conditions, although those conditions were not learned explicitly during the TTS training.

Table A.2: Comparison of TTS intelligibility in CER (%) between the proposed TTS + SNR feedback + ASR feedback + variance adaptor with the different SNR recognition models.

SNR recognition model and training condition	**Static**		**Dynamic-switch**	
	SNR 0	**SNR -10**	**Clean, SNR 0, SNR -10**	**SNR 0, Clean, SNR -10**
SNR (cls)				
Static noise	14.70	24.96	61.94	70.98
Static + dynamic noise	11.88	24.74	26.36	21.88
SNR (reg)				
Static noise	16.05	29.65	66.03	64.34
Static + dynamic noise	18.57	31.19	16.95	18.70

Appendix B

TTS speech intelligibility in unseen SNR

We evaluated the proposed TTS in several SNR conditions. The TTS systems were trained based on clean, SNR 0 dB, and SNR -10 dB. The proposed TTS (non-incremental and PCD-ITTS) maintained speech intelligibility in noisy conditions with the same or higher SNR than those included in the training data, as shown in Figure B.1 for the WSJ-based models and Figure B.2 for the Hurricane-based model.

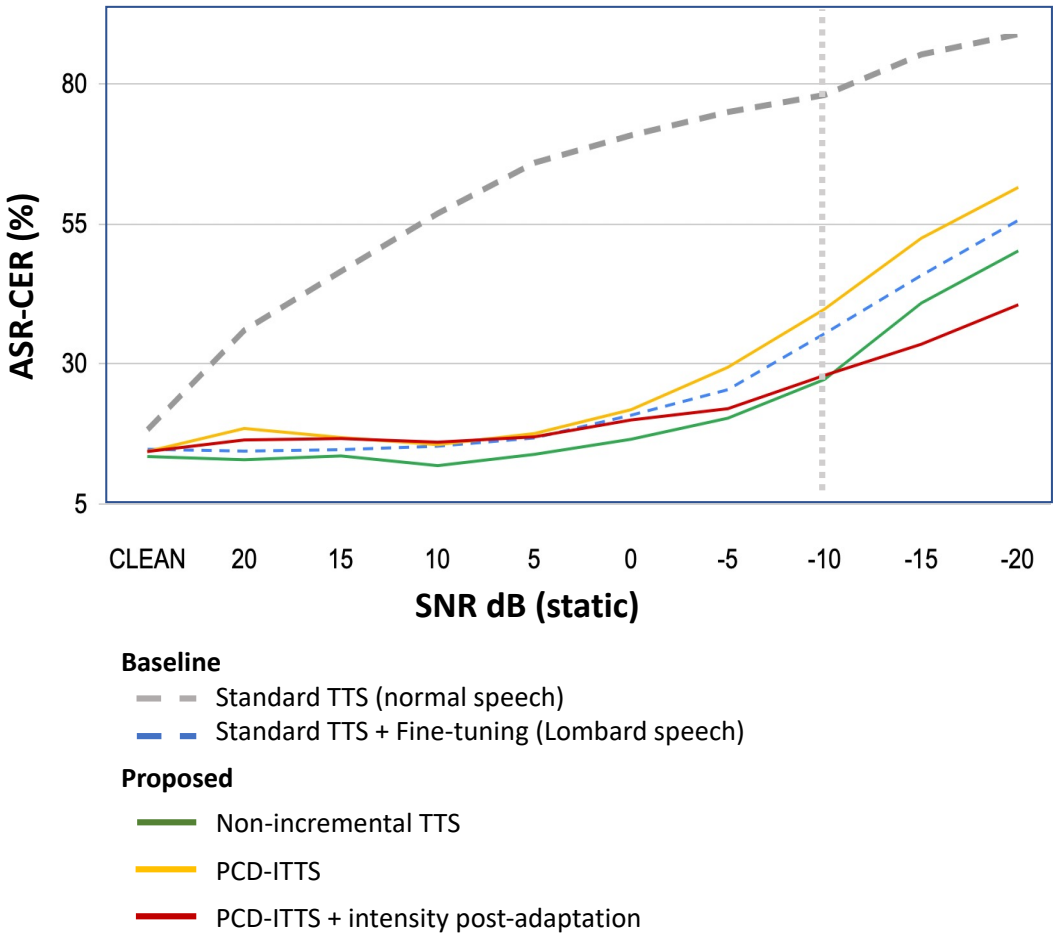


Figure B.1: WSJ TTS intelligibility at different SNR level.

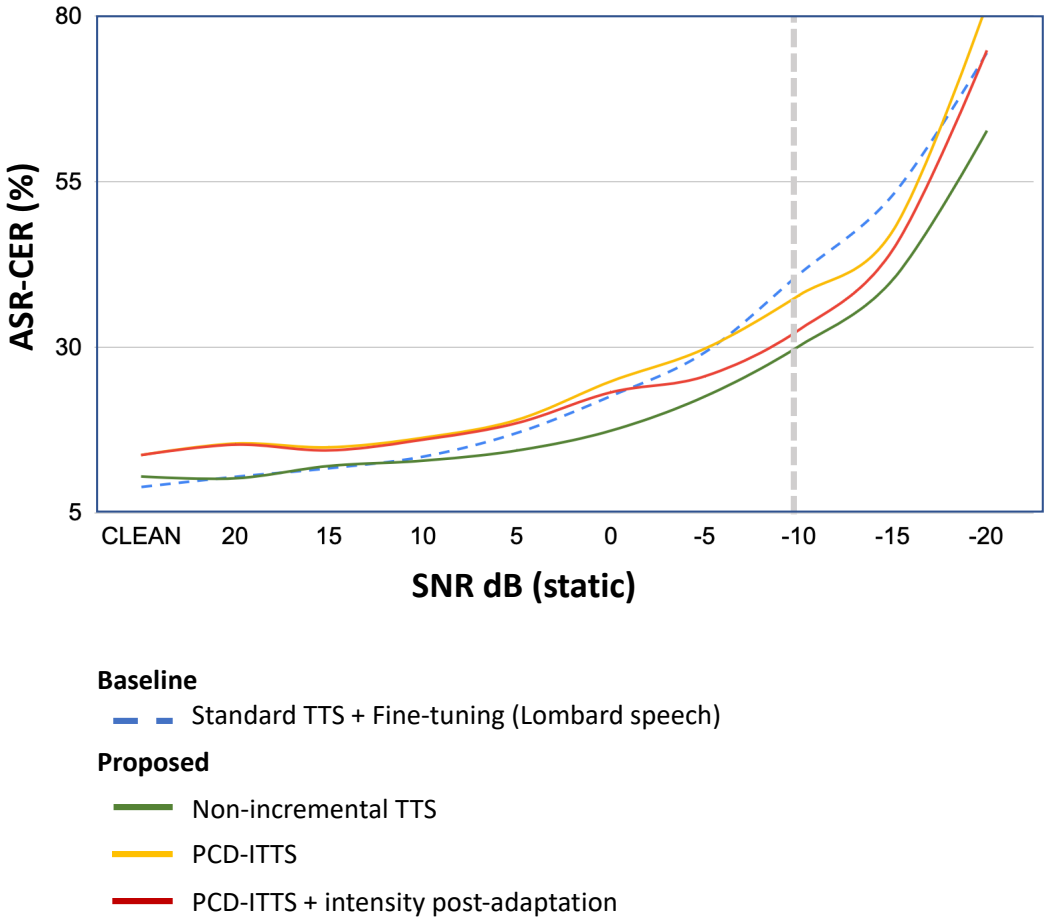


Figure B.2: Hurricane TTS intelligibility at different SNR level.

References

[1] P. B. Denes and E. N. Pinson. *The Speech Chain: The Physics and Biology of Spoken Language*. Science/communication. W.H. Freeman, New York, N.Y, 1993.

[2] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Listening while speaking: Speech chain by deep learning. In *Proc. IEEE ASRU*, pages 301–308, 2017.

[3] Yi Ren, C. Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and T. Liu. FastSpeech 2: Fast and high-quality end-to-end text to speech. *ArXiv*, abs/2006.04558, 2020.

[4] John Locke. Babbling and early speech: Continuity and individual differences. *First Language*, 9:191–205, 01 1989.

[5] Samsul Alam. Babbling: A definition & overview of theories. 1998.

[6] Barbara C. Lust. *Child Language: Acquisition and Growth*. Cambridge Textbooks in Linguistics. Cambridge University Press, 2006.

[7] M. Badian, E. Appel, D. Palm, W. Rupp, W. Sittig, and K. Taeuber. Standardized mental stress in healthy volunteers induced by delayed auditory feedback (DAF). *European Journal of Clinical Pharmacology*, 16:171–176, 1979.

[8] Kazutaka Kurihara and Koji Tsukada. SpeechJammer: a system utilizing artificial speech disturbance with delayed auditory feedback. *CoRR*, abs/1202.6106, 2012.

[9] Joseph S. Perkell, Melanie Matthies, Harlan Lane, Frank H. Guenther, Reiner Wilhelms-Tricarico, Jane Wozniak, and Peter Guiod. Speech motor control: Acoustic goals, saturation effects, auditory feedback and internal models. *Speech Communication*, 22:227–250, 1997.

[10] Albert Postma. Detection of errors during speech production: a review of speech monitoring models. *Cognition*, 77(2):97–132, 2000.

[11] Harlan Lane and Bernard Tranel. The Lombard sign and the role of hearing in speech. *Journal of Speech and Hearing Research*, 14(4):677–709, 1971.

[12] Maëva Garnier, Nathalie Henrich, and Danièle Dubois. Influence of sound immersion and communicative interaction on the Lombard effect. *J. Speech Lang. Hear. Res.*, 53(3):588–608, 2010.

[13] Tomasz Letowski, Tom Frank, and Jane Caravella. Acoustical properties of speech produced in noise presented through supra-aural earphones. *Ear and hearing*, 14:332–338, 1993.

[14] Gopala Krishna Anumanchipalli, Prasanna Kumar Muthukumar, Udhyakumar Nallasamy, Alok Parlikar, Alan W Black, and Brian Langner. Improving speech synthesis for noisy environments. In *Proc. ISCA Workshop on Speech Synthesis*, 2010.

[15] Lauren Stowe and Edward Golob. Evidence that the Lombard effect is frequency-specific in humans. *The Journal of the Acoustical Society of America*, 135:640–647, January 2014.

[16] Theda Heinks-Maldonado and John Houde. Compensatory responses to brief perturbations of speech amplitude. *Acoustics Research Letters Online*, 6, July 2005.

[17] Jay Bauer, Jay Mittal, Charles Larson, and Timothy Hain. Vocal responses to unanticipated perturbations in voice loudness feedback: An automatic mechanism for stabilizing voice amplitude. *The Journal of the Acoustical Society of America*, 119:2363–71, May 2006.

[18] Kristen R Anderson Foery. *Triggering the Lombard effect: Examining automatic thresholds*. PhD thesis, University of Colorado at Boulder, 2008.

[19] N. Dixon and H. Maxey. Terminal analog synthesis of continuous speech using the diphone method of segment assembly. *IEEE Transactions on Audio and Electroacoustics*, 16(1):40–50, 1968.

[20] J. Olive. Rule synthesis of speech from dyadic units. In *ICASSP '77. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 568–570, 1977.

[21] K. Tokuda, T. Kobayashi, and S. Imai. Speech parameter generation from hmm using dynamic features. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 660–663 vol.1, 1995.

[22] Takayoshi Yoshimura, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis. volume J83-D-II, 01 1999.

[23] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech*, pages 4006–4010, 2017.

[24] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis, and Yonghui Wu. Natural TTS synthesis by conditioning Wavenet on MEL spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783, 2018.

[25] Mingjian Chen, Xu Tan, Yi Ren, Jin Xu, Hao Sun, Sheng Zhao, and Tao Qin. MultiSpeech: Multi-speaker text to speech with transformer. In *Proc. Interspeech*, pages 4024–4028, 2020.

[26] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proc. AAAI Conference on Artificial Intelligence*, pages 6706–6713, 2019.

[27] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. FastSpeech: Fast, robust and controllable text to speech. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, pages 3165–3174, 2019.

[28] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[29] B. H. Juang and L. R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[30] Mark Gales, Steve Young, et al. The application of hidden Markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304, 2008.

[31] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of ICML*, pages 369–376, Pittsburgh, Pennsylvania, USA, 2006.

[32] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of ICML*, pages 1764–1772, Beijing, China, 2014.

[33] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of ICASSP*, pages 4960–4964, Shanghai, China, 2016.

[34] L. Dong, S. Xu, and B. Xu. Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *Proc. ICASSP*, pages 5884–5888, 2018.

[35] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660*, 2019.

[36] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Machine speech chain. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:976–989, 2020.

[37] Junichi Yamagishi, Takao Kobayashi, Yuji Nakano, Katsumi Ogata, and Juri Isogai. Analysis of speaker adaptation algorithms for HMM-based speech synthesis and a constrained SMAPLR adaptation algorithm. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17:66 – 83, February 2009.

[38] Tuomo Raitio, Antti Suni, Martti Vainio, and Paavo Alku. Analysis of HMM-based lombard speech synthesis. In *Proc. Interspeech*, pages 2781–2784, 2011.

[39] Dipjyoti Paul, Muhammed P.V. Shifas, Yannis Pantazis, and Yannis Stylianou. Enhancing speech intelligibility in text-to-speech synthesis using speaking style conversion. In *Proc. Interspeech*, pages 1361–1365, 2020.

[40] Jagadeesh Balam, Jocelyn Huang, Vitaly Lavrukhin, Slyne Deng, Somshubra Majumdar, and Boris Ginsburg. Improving noise robustness of an end-to-end neural model for automatic speech recognition, 10 2020.

[41] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[42] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[44] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[45] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv*, abs/1211.3711, 2012.

[46] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Machine speech chain with one-shot speaker adaptation. In *Proc. of Interspeech*, pages 887–891, 2018.

[47] Alex Graves. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Studies in Computational Intelligence, pages 5–13. Springer, Berlin, 2012.

[48] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *Proc. ICASSP*, pages 4945–4949, 2016.

[49] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu. Deep Speaker: an end-to-end neural speaker embedding system. *CoRR*, abs/1705.02304, 2017.

[50] Genichiro Kikui, Eiichiro Sumita, Toshiyuki Takezawa, and Seiichi Yamamoto. Creating corpora for speech-to-speech translation. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[51] Murat Saraclar, Michael Riley, Enrico Bocchieri, and Vincent Goffin. Towards automatic closed captioning : Low latency real time broadcast news transcription. In *Proc. ICSLP*, 2002.

[52] Hasim Sak, Murat Saraclar, and Tunga Gungor. On-the-fly lattice rescoring for real-time automatic speech recognition. In *Proc. Interspeech*, pages 2450–2453, 2010.

[53] Chandrashekhar Lavania and Jeff Bilmes. Reducing total latency in online real-time inference and decoding via combined context window and model smoothing latencies. In *Proc. ICASSP*, pages 2791–2795, 2017.

[54] Vijayaditya Peddinti, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur. Low latency acoustic modeling using temporal convolution and LSTMs. *IEEE Signal Processing Letters*, 25(3):373–377, 2018.

[55] Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems*, pages 5067–5075, 2016.

[56] Sashi Novitasari, Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Sequence-to-sequence learning via attention transfer for incremental speech recognition. In *Proc. Interspeech*, pages 3835–3839, 2019.

[57] Sashi Novitasari, Sakriani Sakti, and Satoshi Nakamura. Neural incremental speech recognition toward real-time machine speech translation. *IEICE Transactions on Information and Systems*, E104.D(12):2195–2208, 2021.

[58] Tara N. Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, Chung-Cheng Chiu, David Garcia, Alex Gruenstein, Ke Hu, Anjuli Kannan, Qiao Liang, Ian McGraw, Cal Peyser, Rohit Prabhavalkar, Golan Pundak, David Rybach, Yuan Shangguan, Yash Sheth, Trevor Strohman, Mirkó Visontai, Yonghui Wu, Yu Zhang, and Ding Zhao. A streaming on-device end-to-end model surpassing server-side conventional model quality and latency. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6059–6063, 2020.

[59] Bo Li, Shuo-yiin Chang, Tara N. Sainath, Ruoming Pang, Yanzhang He, Trevor Strohman, and Yonghui Wu. Towards fast and accurate streaming end-to-end asr. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6069–6073, 2020.

[60] Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe. Streaming transformer ASR with blockwise synchronous beam search. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 22–29, 2021.

[61] Timo Baumann and David Schlangen. Evaluating prosodic processing for incremental speech synthesis. In *Proc. Interspeech*, pages 438–441, 2012.

[62] Timo Baumann. Decision tree usage for incremental parametric speech synthesis. In *Proc. ICASSP*, pages 3819–3823, 2014.

[63] Mael Pouget, Thomas Hueber, Gerard Bailly, and Timo Baumann. HMM training strategy for incremental speech synthesis. In *Proc. Interspeech*, page 1201–1205, 2015.

[64] Tomoya Yanagita, Sakriani Sakti, and Satoshi Nakamura. Incremental TTS for Japanese language. In *Proc. Interspeech 2018*, pages 902–906, 2018.

[65] Tomoya Yanagita, Sakriani Sakti, and Satoshi Nakamura. Neural iTTS: Toward synthesizing speech in real-time with end-to-end neural text-to-speech framework. In *Proc. ISCA Speech Synthesis Workshop (SSW)*, pages 183–188, 2019.

[66] Mingbo Ma, Baigong Zheng, Kaibo Liu, Renjie Zheng, Hairong Liu, Kainan Peng, Kenneth Church, and Liang Huang. Incremental text-to-speech synthesis with prefix-to-prefix framework, 2019.

[67] Tara N Sainath, Chung-Cheng Chiu, Rohit Prabhavalkar, Anjuli Kannan, Yonghui Wu, Patrick Nguyen, and ZhiJeng Chen. Improving the performance of online neural transducer models. In *Proc. ICASSP*, pages 5864–5868, 2018.

[68] Douglas B Paul and Janet M Baker. The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362, 1992.

[69] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Multi-scale alignment and contextual history for attention mechanism in sequence-to-sequence model. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 648–655. IEEE, 2018.

[70] Jeff Donahue, Sander Dieleman, Mikołaj Bińkowski, Erich Elsen, and Karen Simonyan. End-to-end adversarial text-to-speech. In *Proc. ICLR*, 2021.

[71] Martin Cooke, C Mayo, and Cassia Valentini-Botinhao. Intelligibility-enhancing speech modifications: the Hurricane Challenge. In *Proc. Interspeech*, pages 3552–3556, 2013.

[72] Jan Rennies, Henning Schepker, Cassia Valentini-Botinhao, and Martin Cooke. Intelligibility-enhancing speech modifications — the Hurricane Challenge 2.0. In *Proc. Interspeech*, pages 1341–1345, 2020.

[73] Vincent Aubanel and Martin Cooke. Information-preserving temporal reallocation of speech in the presence of fluctuating maskers. In *Interspeech*, pages 3592–3596, 2013.

[74] Daniel Erro, Tudor-Catalin Zorila, Yannis Stylianou, Eva Navas, and Inma Hernáez. Statistical synthesizer with embedded prosodic and spectral modifications to generate highly intelligible speech in noise. In *INTERSPEECH*, pages 3557–3561, 2013.

[75] Reiko Takou, Nobumasa Seiyama, and Atsushi Imai. Improvement of speech intelligibility by reallocation of spectral energy. In *INTERSPEECH*, pages 3605–3607, 2013.

[76] Elizabeth Godoy and Yannis Stylianou. Increasing speech intelligibility via spectral shaping with frequency warping and dynamic range compression plus transient enhancement. In *INTERSPEECH*, pages 3572–3576, 2013.

[77] Henning Schepker, Jan Rennies, and Simon Doclo. Speech-in-noise enhancement using amplification and dynamic range compression controlled by the speech intelligibility index. *The Journal of the Acoustical Society of America*, 138(5), 2015.

[78] Felicitas Bederna, Henning Schepker, Christian Rollwage, Simon Doclo, Arne Pusch, Jörg Bitzer, and Jan Rennies. Adaptive compressive onset-enhancement for improved speech intelligibility in noise and reverberation. In *Proc. Interspeech*, pages 1351–1355, 2020.

[79] Antti Suni, Tuomo Raitio, Martti Vainio, and Paavo Alku. The GlottHMM speech synthesis entry for Blizzard Challenge 2010. In *Blizzard Challenge 2010 Workshop*, 2010.

[80] Antti Suni, Reima Karhila, Tuomo Raitio, Mikko Kurimo, Martti Vainio, and Paavo Alku. Lombard modified text-to-speech synthesis for improved

intelligibility: Submission for the Hurricane Challenge 2013. In *Proc. Interspeech 2013*, pages 3562–3566, 2013.

[81] Qiong Hu, Tobias Bleisch, Petko Petkov, Tuomo Raitio, Erik Marchi, and Varun Lakshminarasimhan. Whispered and Lombard neural speech synthesis. In *Proc. IEEE SLT*, 2021.

[82] Zolzaya Byambadorj, Ryota Nishimura, Altangerel Ayush, Kengo Ohta, and Norihide Kitaoka. Text-to-speech system for low-resource language using cross-lingual transfer learning and data augmentation. *EURASIP J. Audio Speech Music Process.*, 2021(1), dec 2021.

[83] Guoguo Chen, Xingyu Na, Yongqing Wang, Zhiyong Yan, Junbo Zhang, Sifan Ma, and Yujun Wang. Data augmentation for children's speech recognition - the Ethiopian system for the SLT 2021 Children Speech Recognition Challenge. *ArXiv*, abs/2011.04547, 2020.

[84] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal Forced Aligner: Trainable text-speech alignment using Kaldi. In *Proc. Interspeech*, pages 498–502, 2017.

[85] David Pearce and Hans Gnter Hirsch. The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy condition. In *Proc. ICSLP*, pages 29–32, 2000.

[86] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011.

[87] Hans Bosker and Martin Cooke. Enhanced amplitude modulations contribute to the Lombard intelligibility benefit: Evidence from the Nijmegen corpus of Lombard speech. *The Journal of the Acoustical Society of America*, 147, 01 2020.

[88] Takaaki Saeki, Shinnosuke Takamichi, and Hiroshi Saruwatari. Incremental text-to-speech synthesis using pseudo lookahead with large pretrained language model. *IEEE Signal Processing Letters*, 28:857–861, 2021.

[89] Sashi Novitasari, Andros Tjandra, Tomoya Yanagita, Sakriani Sakti, and Satoshi Nakamura. Incremental machine speech chain towards enabling listening while speaking in real-time. In *Proc. Interspeech 2020*, pages 4372–4376, 2020.

[90] Danni Liu, Changhan Wang, Hongyu Gong, Xutai Ma, Yun Tang, and Juan Pino. Incremental speech synthesis for speech-to-speech translation. *arXiv*, 2021.

[91] Martin Cooke, Catherine Mayo, and Cassia Valentini-Botinhao. Hurricane natural speech corpus, [sound]. `https://doi.org/10.7488/ds/140`, 2013.

[92] Christian Benoît, Martine Grice, and Valérie Hazan. The SUS test: A method for the assessment of text-to-speech synthesis intelligibility using semantically unpredictable sentences. *Speech Communication*, 18(4):381–392, 1996.

[93] Thuanvan Ngo, Rieko Kubo, and Masato Akagi. Increasing speech intelligibility and naturalness in noise based on concepts of modulation spectrum and modulation transfer function. *Speech Communication*, 135:11–24, 2021.

[94] J.-C. Junqua, S. Fincke, and K. Field. The Lombard effect: a reflex to better communicate with others in noise. In *Proc. ICASSP 1999*, volume 4, pages 2083–2086 vol.4, 1999.

[95] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.

[96] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

List of publications

Publications

Journal Paper (peer-reviewed)

1. *Neural Incremental Speech Recognition Toward Real-Time Machine Speech Translation*
   Sashi Novitasari, Sakriani Sakti, Satoshi Nakamura
   IEICE Transaction on Information and Systems, Vol. E104-D, No. 12, pp. 2198-2208, December 2021
   (Related to Chapter 4)

2. *A Machine Speech Chain Approach for Dynamically Adaptive Lombard TTS in Static and Dynamic Noise Environments*
   Sashi Novitasari, Sakriani Sakti, Satoshi Nakamura
   IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol. 30, pp. 2673-2688, August 2022
   (Related to Chapter 5 and Chapter 6)

International Conference Paper (peer-reviewed)

1. *Simultaneous Speech-to-speech Translation System with Transformer-based Incremental ASR, MT, and TTS*
   Ryo Fukuda, **Sashi Novitasari**, Yui Oka, Yasumasa Kano, Yuki Yano, Yuka Ko, Hirotaka Tokuyama, Kosuke Doi, Tomoya Yanagita, Sakriani Sakti, Katsuhito Sudoh, Satoshi Nakamura
   Proc. Oriental COCOSDA 2021, pp. 186-192, November 2021

(Related to Chapter 4 and Chapter 6)

2. *Dynamically Adaptive Machine Speech Chain Inference for TTS in Noisy Environment: Listen and Speak Louder*
   Sashi Novitasari, Sakriani Sakti, Satoshi Nakamura
   Proc. INTERSPEECH 2021, pp. 4124-4128, September 2021
   (Related to Chapter 5)

3. *Incremental Machine Speech Chain Towards Enabling Listening while Speaking in Real-time*
   Sashi Novitasari, Andros Tjandra, Tomoya Yanagita, Sakriani Sakti, Satoshi Nakamura
   Proc. INTERSPEECH 2020, pp. 4372-4376, October 2020
   (Related to Chapter 4)

4. *Cross-Lingual Machine Speech Chain for Javanese, Sundanese, Balinese, and Bataks Speech Recognition and Synthesis*
   Sashi Novitasari, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   Proc. SLTU-CCURL 2020, pp. 131-138, May 2020
   (Related to Chapter 3)

5. *Sequence-to-Sequence Learning via Attention Transfer for Incremental Speech Recognition*
   Sashi Novitasari, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   Proc. INTERSPEECH 2019, pp. 3835-3839, September 2019
   (Related to Chapter 4)

6. *Rude-Words Detection for Indonesian Speech Using Support Vector Machine*
   Sashi Novitasari, Dessi Puji Lestari, Sakriani Sakti, Ayu Purwarianti
   Proc. IALP 2018, pp. 19-24, November 2018

7. *Multi-Modal Multi-Task Deep Learning For Speaker And Emotion Recognition Of TV-Series Data*
   Sashi Novitasari, Quoc Truong Do, Sakriani Sakti, Dessi Lestari, Satoshi Nakamura
   Proc. Oriental COCOSDA 2018, pp. 37-42, May 2018

8. *Construction of English-French Multimodal Affective Conversational Corpus from TV Dramas*
   Sashi Novitasari, Quoc Truong Do, Sakriani Sakti, Dessi Lestari, Satoshi Nakamura
   Proc. LREC 2018, pp. 2958-2962, May 2018

Domestic Conference Paper

1. *Improving Intelligibility of Synthesized Speech in Noisy Condition with Dynamically Adaptive Machine Speech Chain*
   Sashi Novitasari, Sakriani Sakti, Satoshi Nakamura
   SIG-SLP 2021, December 2021
   (Related to Chapter 5)

2. *Real-time Neural Machine Speech Chain*
   Sashi Novitasari, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   Spring Meeting of the Acoustical Society of Japan (ASJ), March 2021
   (Related to Chapter 4)

3. *Neural Incremental Speech Recognition through Attention Transfer*
   Sashi Novitasari, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   26th Conference on Natural Language Processing (NLP2020), March 2020
   (Related to Chapter 4)

4. *Speaker and Emotion Recognition ofTV-Series Data using Multimodal and Multitask Deep Learning*
   Sashi Novitasari, Quoc Truong Do, Sakriani Sakti, Dessi Lestari, Satoshi Nakamura
   25th Conference on Natural Language Processing (NLP2019), March 2019