TOWARDS END-TO-END NON-AUTOREGRESSIVE SPEECH APPLICATIONS

by

Nanxin Chen

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

February 2022

Abstract

In the speech research community, a very challenging topic researchers are interested in is the **sequence-to-sequence labeling** problem. Speech is a complicated signal, and many tasks aim to assign a sequence of various labels to the signal. Before, the traditional hybrid approach models this problem as a combination of different stages. A separate intermediate label sequence is introduced at each stage, and a component is optimized to model the probabilities. In contrast, end-to-end models have recently become increasingly popular.

Sequence-to-sequence models, a type of end-to-end model, take the sequence as input and predict the target sequence directly, which is more intuitive. They usually predict one label at each time, widely known as **autoregressive models**. Autoregressive models are easy to train and have an excellent theoretical explanation connecting the probability chain rule. This simplicity also results in inefficiency for the inference, particularly with those lengthy output sequences. This becomes a severe problem in reality when the output sequence is particularly long, like a sequence of characters.

ABSTRACT

To speed up the inference procedure, researchers started to be interested in another type of sequence-to-sequence model, known as **non-autoregressive models**. In contrast to the autoregressive models, non-autoregressive models predict the whole sequence within a constant number of iterations. However, non-autoregressive model training is more challenging compared to autoregressive models.

In this dissertation, we propose two different types of non-autoregressive models for speech applications: **mask-based** approach and **noise-based** approach. To demonstrate the effectiveness of the two proposed methods, we explored their usage for two essential topics: **speech recognition** and **speech synthesis**. The two novel directions proposed in this dissertation provide a good tradeoff between performance and decoding speed and are important for the non-autoregressive speech research field. They allow researchers to apply larger sophisticated networks in their research and companies can also use those methods for businesses to provide service with better quality under time and budget constraints. Some of the methods in this dissertation are not limited to speech applications and may facilitate neural network research in other fields, like neural machine translation or image captioning.

Primary Reader and Advisor: Najim Dehak

Secondary Reader: Jesus Villalba Lopez, Hynek Hermansky

Acknowledgments

First of all, I would like to thank Prof. Najim Dehak, my advisor, for his excellent guidance throughout my whole Ph.D. study, on academic research and daily life. Najim always teaches us how to be optimistic about things and become successful working with different people. He made relentless efforts to regularly arrange individual meetings even though he was really busy. His suggestion is always helpful and insightful. I couldn't finish this journey without his open-mindedness. As a student, I was unsure about my research interests, and my thesis topic changed two times during the last two years. We had several conversions about this, and Najim is always supportive.

I also want to thank Dr. Jesus Villalba Lopez, my second advisor. Jesus shows us a great example of becoming a rigorous researcher, and he spent much time working with his students. While he is more interested in methods with clear mathematical foundations, he is very practical in research, running many experiments to get the best performance. He is good at understanding things, and we can all feel his passion for research.

ACKNOWLEDGMENTS

I am also grateful to Dr. Laureano Moro Velazquez and Dr. Piotr Zelasko. They both provided tremendous help for this thesis, and we had terrific cooperation on multiple projects. I learned a lot from their attitudes toward research and writing skills.

I also want to thank my girlfriend Alice, who gave me a lot of valuable suggestions for the presentation. She was my only audience for the presentation practice, and she is always happy to be my audience. I also owe much gratitude to Dr. Yu Zhang, my host in Google Brain, when I was doing an internship there. Yu has much experience and helped me connect with great researchers, like Dr. William Chan, Dr. Heiga Zen, Dr. Ron Weiss, and Dr. Norouzi Mohammad. We had an incredible journey exploring non-autoregressive models for speech synthesis, which is one of the main topics of this dissertation. I also want to thank Dr. Roger Hsiao, Dr. Arnab Ghoshal, and other people from the Apple Siri team, who helped me polish my internship project. It was my honor to present my work to the senior vice president, Craig Federighi, who inspired me to focus on user experience.

Finally, I want to thank all my lab mates and teammates during the JSALT workshop. It is such a great opportunity to meet them and work together, which I will never forget. I also want to mention Cheng-I Lai (Jeff), who graduated from our lab and shared many interesting research ideas with me.

Contents

Abstract **ii**

Acknowledgments **iv**

List of Tables **xi**

List of Figures **xiii**

List of Algorithms **xv**

1 Introduction to Speech Applications and Models **1**

1.1 Mathematical Definition of Speech Recognition 3

1.2 Mathematical Definition of Speech Synthesis 6

1.3 Hybrid Models 8

1.3.1 Hybrid Models for Speech Recognition 9

1.3.2 Hybrid Models for Speech synthesis 10

1.4 End-to-end Models 11

1.4.1 E2E Models for Speech Recognition 12
1.4.1.1 Transformer-based E2E ASR 12
1.4.2 E2E Models for Speech Synthesis 14
1.4.3 Autoregressive Training 15
1.5 Disadvantages of Autoregressive Systems 17
1.6 Towards Non-autoregressive Speech Applications 20
1.6.1 Benefits of Non-Autoregressive Systems 21
1.6.2 Why Non-Autoregressive Systems are challenging to train 23
1.7 Non-Autoregressive Methods Proposal 24
1.7.1 Mask-based Approach 24
1.7.2 Noise-based Approach 26
1.7.3 Relationship between Two Methods 28
1.8 Datasets 29
1.8.1 Speech Recognition 29
1.8.2 Speech Synthesis 30
1.9 Contribution of this Dissertation 30

2 The Literature of Non-autoregressive Systems for Speech Applications 33

2.1 Non-autoregressive ASR 34
2.1.1 Hybrid Models 34
2.1.2 Connectionist Temporal Classification 35

2.1.3 RNN Transducer (RNN-T) 38
2.1.4 Non-autoregressive End-to-end Speech Recognition 39
2.2 Non-autoregressive Text-to-speech 41
2.3 Summary 42

3 Mask-based non-autoregressive Speech Recognition 43
3.1 Conditional masked language model (CMLM) 45
3.2 Factorized masked language model (FMLM) 46
3.2.1 Easy First Decoding 48
3.2.2 Mask-predict 50
3.2.3 Example 52
3.2.4 Output sequence length prediction 54
3.3 Experiments and results 54
3.4 Analysis 60
3.4.1 Ablation Studies of External Language Model 60
3.4.2 Ablation Studies of Beam Search 63
3.4.3 Error Analysis of A-FMLM on CSJ 68
3.5 Summary 68

4 Noise-based non-autoregressive Text-to-Speech 70

4.1 Score matching 71
4.1.1 Diffusion Probabilistic Model 73
4.1.2 Noise Schedule and Conditioning on Noise Level 76
4.2 Network Architecture 82
4.2.1 WaveGrad Vocoder 82
4.2.2 WaveGrad 2: Phoneme-to-Wave model 87
4.2.2.1 Encoder 87
4.2.2.2 Resampling 89
4.2.2.3 Sampling Window 90
4.2.2.4 Hidden Features Augmentation 90
4.3 Noise Schedule 91
4.4 Evaluation 92
4.5 Results 95
4.6 Ablation Studies 97
4.6.1 WaveGrad: Speed-Quality Tradeoff 97
4.6.2 WaveGrad 2: Sampling Window Size 100
4.6.3 WaveGrad 2: Network Size 101
4.6.4 WaveGrad 2: Hidden Features Augmentation 102
4.6.5 WaveGrad 2: Multi-task Learning and Speed-Quality Trade-off 103
4.7 Summary 104

5 Noise-based non-autoregressive ASR **105**

5.1 Align-Refine 106

5.2 Align-Denoise 108

5.2.1 Noise Distribution 110

5.3 Experiments 114

5.3.1 Network Architecture 114

5.3.2 Results 115

5.4 Discussion 120

5.4.1 Alignment mismatch 120

5.4.2 Combining with beam search and external language model 121

5.5 Summary 124

6 Conclusion **126**

Bibliography **131**

List of Tables

1.1 Autoregressive training examples 16

2.1 Examples of CTC decoding 36

3.1 Comparison of baselines, previous work, A-CMLM and A-FMLM on AISHELL 56
3.2 Comparison of baselines, previous work, A-CMLM and A-FMLM on CSJ 58
3.3 Comparison of baselines, previous work, A-CMLM and A-FMLM on WSJ 58
3.4 Comparison of baselines, previous work, A-CMLM and A-FMLM on Tedlium2 60
3.5 Comparison of A-FMLM with and without external LM on AISHELL. 61
3.6 Comparison of A-FMLM with and without external LM on CSJ. . 61
3.7 Comparison of A-CMLM with and without external LM on WSJ. . 62
3.8 Comparison of A-CMLM with and without external LM on Tedlium 2. 62
3.9 Performance comparison on Tedlium2 test with or without beam search 65
3.10 Performance comparison on AISHELL with or without beam search 66
3.11 Performance comparison on CSJ with or without beam search . . 66
3.12 Performance comparison on WSJ test with or without beam search 67

4.1 Mean opinion scores (MOS) of various models and their confidence intervals 96
4.2 Objective and subjective metrics of the WaveGrad Base models. . 99
4.3 Comparison between different sampling window sizes for WaveGrad 2 100
4.4 Comparison between different network sizes for WaveGrad 2 . . . 101

4.5 Impact of augmentation on the learned WaveGrad 2 representations . . . 102
4.6 Impact of multi-task (MT) learning and number of iterations for WaveGrad 2. . . . 104

5.1 Align-Denoise training examples . . . 112
5.2 Word error rates (WERs) and real time factor (RTF) for WSJ (English) . . . 116
5.3 Character error rates (CERs) for Corpus of Spontaneous Japanese (CSJ) . . . 118
5.4 Character error rates (CERs) for AISHELL with Align-Denoise . 118
5.5 Word error rates (WERs) for Tedlium2 . . . 119
5.6 Mis-aligned example for Align-Denoise . . . 121
5.7 Align-Denoise with/without beam search and external language model on WSJ . . . 122
5.8 Align-Denoise with/without beam search and external language model on CSJ . . . 122
5.9 Align-Denoise with/without beam search and external language model on AISHELL . . . 123
5.10 Align-Denoise with/without beam search and external language model on TEDLIUM2 . . . 123

List of Figures

1.1 Speech recognition and synthesis 2
1.2 Encoder-Decoder architecture 12
1.3 Autoregressive model: slow inference 18
1.4 Autoregressive model: mismatch between training and inference 19
1.5 Autoregressive model: unidirectional decoding 20
1.6 Comparison between autoregressive and non-autoregressive systems 21
1.7 Diagram of mask-based approach 25
1.8 Example of mask-based approach decoding for speech recognition 25
1.9 Example of noise-based approach training 26
1.10 Example of noise-based approach decoding 27

3.1 An illustration of the *easy first* inference procedure 51
3.2 Illustration of inference procedure 53
3.3 How the sequence length is predicted for A-CMLM/A-FMLM. . . 55
3.4 How the number of refinement iterations impacts the result without beam search using mask predict 64
3.5 How the number of refinement iterations impacts the real time factor (RTF) without beam search using mask predict 65
3.6 Error analysis of autoregressive and non-autoregressive on different output sequence length bins 69

4.1 A directed graphical model of the diffusion probabilistic model . . 76
4.2 WaveGrad network architecture 83
4.3 A block diagram of the Upsampling Block (UBlock) 84
4.4 A block diagram of the downsampling block (DBlock). 85
4.5 A block diagram of feature-wise linear modulation (FiLM) module 85
4.6 WaveGrad 2 network architecture 88
4.7 Plot of different noise schedules for WaveGrad and WaveGrad 2. . 92

LIST OF FIGURES

5.1 Visualization of the Align-Denoise training 113

List of Algorithms

1 Training procedure for A-FMLM 49

2 Training algorithm for WaveGrad 80

3 Sampling algorithm for WaveGrad 81

Chapter 1

Introduction to Speech Applications and Models

Communication is the procedure of transmitting information, and it is crucial in human-to-human interactions. Speech is a convenient human vocal communication system with language, one of the most natural ways for people to communicate. For example, in comparison, there exist some languages don't have writing system. Those facts may explain why speech is widely adopted for human-computer interaction.

Even though for human beings, it is easy to communicate with speech from a very young age [1], the development of speech technologies for machines already has taken a long time. The communication process includes two stages, receiving and sending. For speech, these are the process of recognition and

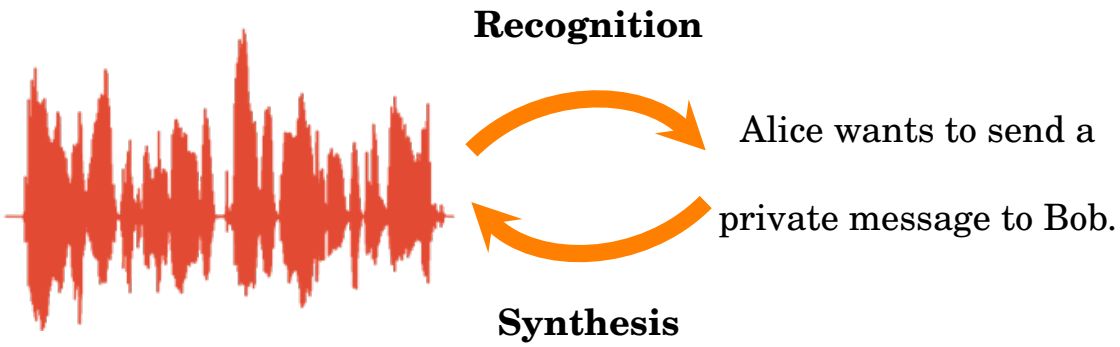

Figure 1.1: Speech recognition and synthesis.

generation, as shown in Figure 1.1.

Speech recognition is the process of recognizing human speech from audio. The verb 'recognize' here indicates extracting the content (text) information from the speech. Getting the text information is the first vital step towards understanding. While the ability to recognize seems natural to humans, it is still challenging for machines to achieve high recognition quality under some instances, for example, noisy environment [2], spontaneous speech [3]. Recognition speed is another concern to apply for large-scale applications.

Speech synthesis is the process of generating audio from text information. This is how the computer gives feedback to humans during the conversation. Recently with the development of deep learning [4–9], many proposed systems are able to generate high fidelity audio which is clear, natural and easy to follow. Since the fidelity has been improved a lot, there are increasing demands to speed up the generation process and fit it into small portable devices.

Those two are the main focus of this dissertation. While the understanding plays an important role in the interaction, the analysis is usually based on the text instead of using the speech directly. Thus it is out of the scope of this thesis. Currently, most successful approaches tackle speech recognition and synthesis by probability and statistical modeling. To help to understand it, it is better to start the introduction with mathematical definitions, which are given in the following sections.

1.1 Mathematical Definition of Speech Recognition

If we denote the speech observation as x and the text sequence as y, then the speech recognition problem is finding the most appropriate text sequence y given the observation x. To represent the observation and the outcome, there are different choices in the literature.

For the speech observation x, common choices include Mel-frequency cepstral coefficients [10] (MFCC), filter bank features (FBANK), perceptual linear predictive [11] (PLP), and raw waveform samples. For the output sequence y, there are different choices, and they are usually chosen empirically based on the corpus. While words may consider intuitively to be the unit of the language, they are not used widely when the lexicon is not included because of

the large dictionary size and difficulties in modeling unseen ones. Instead, end-to-end systems embrace different forms y of word factorizations, such as characters [12], subwords [13].

In the earliest stage of speech recognition research, templates are built for individual words in the dictionary, and the recognition is designed as a procedure of matching between observations and templates using *dynamic time wrapping* [14] (DTW):

$$\arg\max_{k} P(y_k|\mathbf{x}) \tag{1.1}$$

where y_k is a single word from the dictionary and x is a small speech segment. This method is proposed for the *isolated word recognition* and to get individual x, a pre-processing step is needed to find the boundary of individual words. Overall, this method isn't working well for real-life speech because most speech is continuous instead of isolated words. Also, mathematically, recognizing individual words ignores the latent structure of the human language, which is crucial to determine words with similar pronunciation.

The Dragon system [15] introduces Hidden Markov Model (HMM) and modular design, which soon becomes the dominant approach among the speech recognition community. The problem of speech recognition is factorized as the combination of acoustic model, lexicon, language model, and decoder. Different groups of researchers are working on individual components, and the improvement of those components leads to the performance boost of speech recognition

in different ways.

For the continuous speech recognition, the goal is to find the optimal sequence y given the observation x

$$\arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \propto P(\mathbf{x} \mid \mathbf{y}) P(\mathbf{y}) \tag{1.2}$$

where the decomposition follows Bayes' Rule. The first term $P(\mathrm{x} \mid \mathrm{y})$ involves acoustic model and lexicon, while the second term $P(\mathrm{y})$ is modeled by the language model.

Acoustic model builds the connection between input observations x and linguistic units like phonemes. Various statistical models are proposed, including Gaussian Mixture Models (GMM), sub-space GMM [16] and deep neural network [17] (DNN). Lexicon is similar to a dictionary which includes the mapping between phoneme sequences and words. Language model estimates how likely a text sequence y appears and we have seen a shift from adopting N-gram model [18], to recurrent neural network [19] (RNN) and Transformer network [20].

This decomposition has been challenged in recent years with the availability of large amount of labelled data and development of neural machine translation (NMT) models [21] which surpassed the traditional statistical machine translation (SMT) models [22] with a very similar decomposition. Applying similar methodologies to the speech recognition, *end-to-end speech recognition* [23] becomes increasingly popular nowadays. Most end-to-end approach

still embraces a similar decomposition as equation 1.2 where $P(\mathrm{y})$ is the language model trained on external data. However end-to-end approach greatly simplifies the whole process since $P(\mathrm{x} \mid \mathrm{y})$ is estimated directly by the neural network. End-to-end approach leads to state-of-the-art performance on multiple datasets [24].

1.2 Mathematical Definition of Speech Synthesis

The research of speech synthesis can be traced back to eighteen century [25]. A Hungarian pioneering, Wolfgang Ritter von Kempelen, built a mechanical speaking machine by wood to mimic human speech production. It includes a pressure chamber as lungs, a leather tube as the vocal tract and a metal reed as the vocal cords. It can produce different vowels by changing the shape of the tube. The related observations and experiments are published in his book [26].

Later research followed this direction, and different studies focused on the physiology of speech production. Two methods were proposed to simulate the vocal tract resonance characteristic: articulatory synthesis [27, 28] and formant synthesis [29, 30]. Articulatory synthesis models articulation processes in the human vocal tract while formant synthesis uses mathematical models such as additive synthesis to create the signal. The latter one is an example

of *parametric synthesis* which uses parameters to describe the characteristics of the vocal tract without the need for waveform samples. In contrast, a different approach, *concatenative synthesis* utilizes the waveform database to create continuous speech. The main focus is on selecting the proper pre-recorded unit from the database based on the context analysis and concatenating them to get smooth audio. The selection process relies on a large number of resource materials that require many hours of work and are pretty expensive. The concatenation process is crucial for obtaining smooth transitions to avoid acoustic glitches. Even with the proper setup, concatenative synthesis still suffers from coverage issues when the unit is not seen in some context.

In recent years, with the development of statistical models, especially neural networks, the naturalness and expressiveness of statistical parametric synthesis have been greatly improved [4–9]. In contrast, statistical models extract parametric representations of speech from the database and model them by a set of generative models. Those spectral and excitation parameters are estimated based on contextual information. They are typically estimated by a Maximum Likelihood criterion

$$\theta = \arg\max_{\theta} p(\mathbf{x} \mid \mathbf{y}, \theta) \tag{1.3}$$

where θ is a set of parameters, $\mathbf{x}$ is a set of training data, and $\mathbf{y}$ is a set of word sequences or phoneme sequences. To generate the speech, we can either

sample from the model $p(\mathbf{x} \mid \mathbf{y})$ or take the one with maximum probability:

$$\mathbf{x} = \arg\max_{\mathbf{x}} p(\mathbf{x} \mid \mathbf{y}, \theta) \tag{1.4}$$

In comparison, the statistical parametric synthesis can be extended naturally to the unseen context and the quality heavily depends on the expressiveness of the model $p(\mathbf{x} \mid \mathbf{y})$. The statistical parametric synthesis also allows flexibility in changing its voice characteristics and speaking styles by naturally introducing additional parameters. Because of these reasons, statistical parametric synthesis is the most popular approach among the text-to-speech research community, and concatenative synthesis is not discussed in this thesis.

1.3 Hybrid Models

Hybrid models are a combination of multiple models and they were broadly adopted for speech applications. Hybrid models split the whole task into multiple sub-tasks and train different parts separately. The split process usually involves domain knowledge and human experience. Intermediate labels are often required in order to do the sub-task training. By optimizing the performance of the individual component, the combined system usually works reasonably well, specially with a limited amount of training data.

1.3.1 Hybrid Models for Speech Recognition

As mentioned in equation 1.2, the training of hybrid speech recognition system involves the posterior $P(\mathbf{x} \mid \mathbf{y})$ and the prior $P(\mathbf{y})$. The former probability can be further factorized as

$$P(\mathbf{x} \mid \mathbf{y}) = \sum_{\mathbf{q}} P(\mathbf{x}, \mathbf{q} \mid \mathbf{y}) \tag{1.5}$$

$$= \sum_{\mathbf{q}} P(\mathbf{x} \mid \mathbf{q}, \mathbf{y}) P(\mathbf{q} \mid \mathbf{y}) \tag{1.6}$$

$$= \sum_{\mathbf{q}} P(\mathbf{x} \mid \mathbf{q}) P(\mathbf{q} \mid \mathbf{y}) \tag{1.7}$$

where $\mathbf{q}$ is the phone state sequence which builds the connection between word sequence $\mathbf{y}$ and actual sound $\mathbf{x}$. The last equation introduces conditional independence assumption that given the phone states sequence, observation $\mathbf{x}$ and word sequence $\mathbf{y}$ are independent. $P(\mathbf{q} \mid \mathbf{y})$ measures the pronunciation of different words and is usually given by the *Lexicon*. $P(\mathbf{x} \mid \mathbf{y})$ estimates probabilities of acoustics given the phone states which is modelled by the *Acoustic Model*. To combine scores from acoustic model and language model, Weighted Finite State Transducers [31] is typically used. However for decoding, proper weights are still needed to balance these two since they are trained separately on different data.

1.3.2 Hybrid Models for Speech synthesis

The resolution of waveform samples is much higher than the resolution of word or phoneme sequences. Thus in order to generate the waveform, similar to speech recognition, the most popular speech synthesis system relies on an intermediate representation between samples and phoneme sequences. This representation is largely chosen by experience, for example, mel-spectrogram or linear-spectrogram features. By introducing this intermediate feature, text-to-speech is divided into two sub-tasks:

Character-to-features (C2F) predicts linear spectrogram or mel-spectrogram features from given input text/phoneme sequence and speaker identity. Many C2F models have been proposed in the literature, including autoregressive systems such as Tacotron [32], Tacotron 2 [33], and non-autoregressive systems like FastSpeech [34], non-attentive Tacotron [35].

Vocoder generates the waveform given spectrogram features as input. Because of the high resolution of waveforms, for instance, 16000 or 24000 samples per second, in general, the waveform generation is the slowest part in speech synthesis. WaveNet [36] is the first neural model that reaches high quality, and it can be used as a neural vocoder. However, the generation speed of WaveNet is extremely slow due to the autoregressive characteristic and large network architecture. WaveRNN [37] uses much simpler architecture, and the performance is close to the WaveNet. Nevertheless, it is still an autoregressive model,

so the sample needs to be computed one by one.

1.4 End-to-end Models

For hybrid models, errors may accumulate along different sub-tasks since individual components are trained without input errors. Also, labels at different stages are required as the target for the sub-task training, which can be both expensive and time-consuming to get. End-to-end models, in contrast, introduce a standalone model for the task without dividing it into separate problems, which simplifies both training and inference.

For the two important topics discussed in this thesis, **speech recognition** and **speech synthesis**, both of them take a sequence as input and predict another sequence with various lengths. The length of the input sequence and output sequence can be quite different. *Encoder-Decoder sequence-to-sequence* (Seq2Seq) model is proposed in [38] to handle the problem of mismatch length. As shown in Figure 1.2, it consists of the encoder and decoder, which handles input and output sequence separately. Encoder and decoder usually exchange information via the attention mechanism [39] between them. End-to-end sequence-to-sequence models play an essential role in speech applications and have become increasingly popular with the development of computing hardware as well as a large amount of available data.

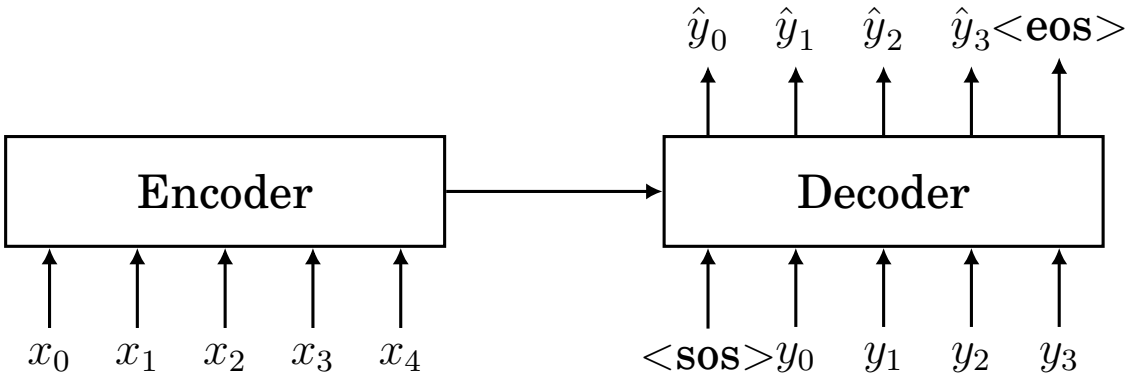

Figure 1.2: Encoder-Decoder architecture.

1.4.1 E2E Models for Speech Recognition

In recent years, end-to-end models have started to catch up with or even surpass the hybrid speech recognition system [24]. End-to-end models directly estimate $P(\mathbf{x} \mid \mathbf{y})$ in equation 1.2 without the need of lexicon. It predicts the probabilities of characters or subwords directly based on the input speech and contextual information. However, language model $P(\mathbf{y})$ is still included for most cases which is trained separately on a large text corpus.

1.4.1.1 Transformer-based E2E ASR

The transformer model is a sequence-to-sequence encoder-decoder model originally introduced for neural machine translation [39]. Transformer applies attention-based modules to learn sequential information instead of recurrent connection employed in the recurrent neural networks. To transfer information

from the encoder to the decoder, it adopts the cross attention similar to the one included in the recurrent-based E2E ASR model.

The main disadvantage of the Transformer-based E2E ASR system is the decoding speed. While the hidden dimension, in general, is much higher than the sequence length in neural machine translation [39]; this observation doesn't hold to speech applications. Speech recognition typically uses Mel-spectrogram features as input, and the input sequence length highly depends on the frame size and frame shift. For example, World Street Journal (WSJ) [40] is a commonly used corpus for speech recognition and the lengthiest audio in the training set includes 2,433 frames when the frame shift is 10ms. For the output sequence, it totally depends on the output tokens used. On the same dataset, the longest training sample includes 253 characters.

A practical solution is to include down-sampling operators in the encoder to reduce the sequence length. Even with the down-sampling in the encoder, the autoregressive systems still suffer from the slow decoding problem since the generation speed always depends on the output sequence length. For instance, for WSJ, 253 characters require 253 predictions since the model needs to predict one character by one character. Each prediction requires one forward pass of the decoder.

1.4.2 E2E Models for Speech Synthesis

End-to-end text-to-speech combines character-to-features with the vocoder, and it is promising to deploy in reality since only one model is needed. However, it is challenging because of two different reasons. First, the mapping between characters/phonemes and intermediate features, usually known as the alignment, is not easy to model. To get the alignment, it requires the participation of speech recognition models. That indicates errors are usually included. During the generation, the alignment is not available which needs to be provided by the model prediction. Second, the resolution of intermediate features and samples are quite different. A lot of upsampling is needed to synthesize the waveform and details are required.

FastSpeech 2 [41] is a popular End-to-End speech synthesis framework. It is conditioned on multiple additional features to address the variations in speech, including pitch, energy and duration. To make the end-to-end training possible, multitask training is also introduced to predict intermediate features and waveform samples simultaneously.

EATS [42] requires less conditioning signals but the training objective is much more complicated, which combines multi-resolution adversarial loss, dynamic time warping loss and length prediction loss. Even though additional information is not extracted during training, the objective becomes difficult to tune and dynamic time warping loss is slow to compute due to the dynamic

programming and non-smooth optimization.

1.4.3 Autoregressive Training

The Sequence-to-sequence model predicts sequences with various lengths. In general, it is challenging to predict the output sequences directly since they have very high complexities. For some instances, like text-to-speech or Mandarin recognition, the output vocabulary size is enormous. In addition to the large vocabulary size, some sequences are quite long. Besides, those two sequences have quite different latent structures: one sequence is a text sequence while another measures the amplitude of the waveform or represents certain features extracted from the samples. The differences between those two signals make the training even more challenging.

To make sequence-to-sequence training possible, autoregressive models are widely adopted for speech applications. For the autoregressive model, only one token is predicted each time instead of multiple tokens, reducing the complexity of the output space.

For the Transformer-based E2E ASR, the decoder takes the text sequence as input. It predicts the next token, based on contextual information from the input text and the input audio. The input sequence is 'shifted' to guarantee that the input and output share the same length. The start-of-sentence token (**sos**) is added before the input sequence and the end-of-sentence token (**eos**) is

appended after the output sequence. Some examples are included in Table 1.1.

Table 1.1: Autoregressive training examples from one sentence. Each training sentence provides multiple training examples with variable length. In this table, three of them are included. Input sequence to the decoder always have the same length as the output sequence. **Bold** words are special tokens indicating the start or the end of the sentence. By adding these two special tokens, input sequence and output sequence is shifted by one position. During inference, tokens are predicted one by one starting from **sos**, shown in the second case.

Input									
sos									
Output									
Alice									
Input									
sos	Alice	wants	to	send	a	private			
Output									
Alice	wants	to	send	a	private	message			
Input									
sos	Alice	wants	to	send	a	private	message	to	Bob
Output									
Alice	wants	to	send	a	private	message	to	Bob	**eos**

For Text-to-Speech (TTS), Wave-Tacotron [43] handles longer-term dependencies autoregressively by conditioning each flow on preceding blocks. Samples within each block are estimated by the flow, which enables parallel training and synthesis. Dependencies between blocks are more difficult to solve since it must synthesize the fine-time structure in the waveform as well as produce a coherent long-term structure like prosody and semantics. Text-to-speech is a multimodal generation problem with many variations like speaking style and phonation modes. Conditioning on the whole history resolves many uncertainties about the fine-time structure and long-term structure, which is

also preserved by the conditional dependence assumption. This makes the autoregressive model training easy and stable. Different blocks still need to be generated sequentially for the inference, which might be inefficient for some real applications.

1.5 Disadvantages of Autoregressive Systems

As discussed in the last section, autoregressive models are widely adopted by the speech community. However, from the previous introduction, there are several clear disadvantages of those autoregressive approaches:

- Slow. Autoregressive systems have to decode token-by-token, which is extremely slow for long utterances. For speech recognition, sentences should be able to be recognized in parallel in many cases, but autoregressive models need to recognize character by character or word by word as shown in Figure 1.3. It becomes even a serious issue for speech synthesis since high-fidelity audio requires a high sampling rate, for example, 16,000 or 24,000 samples per second.

- Mismatch between training and inference. During training, teacher forcing is utilized to speed up the computation. However, during inference,

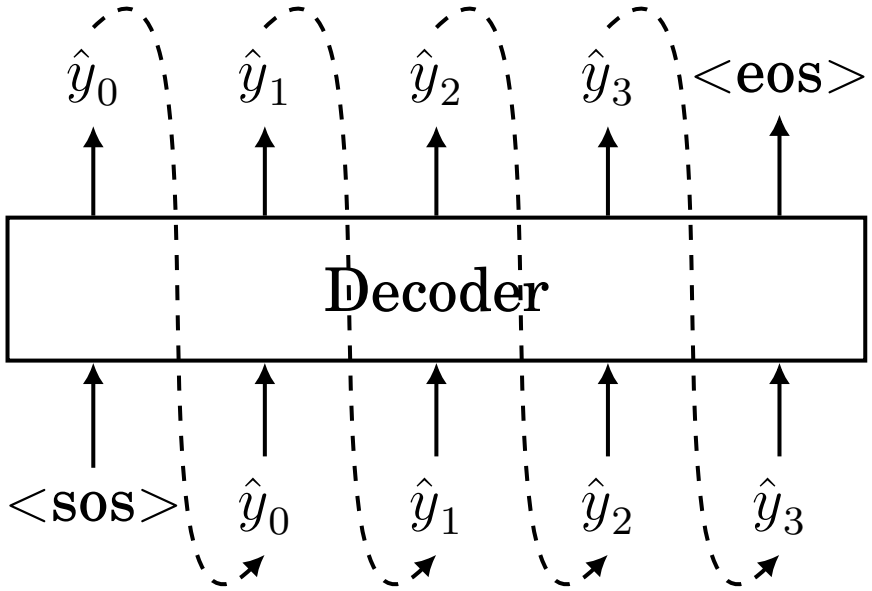

Figure 1.3: Autoregressive model: slow inference. Dashed line represents feeding the network prediction as the decoder input, which indicates one forward pass through the whole decoder.

errors accumulate when the utterance becomes longer. Those decoding mistakes introduce mismatches, which are not covered during training as explained in Figure 1.4. These mismatches make the inference performance unpredictable. A special issue of speech recognition decoding is the usage of beam search, which is not included in the training.

- Uni-directional. Autoregressive systems consider only uni-directional in-

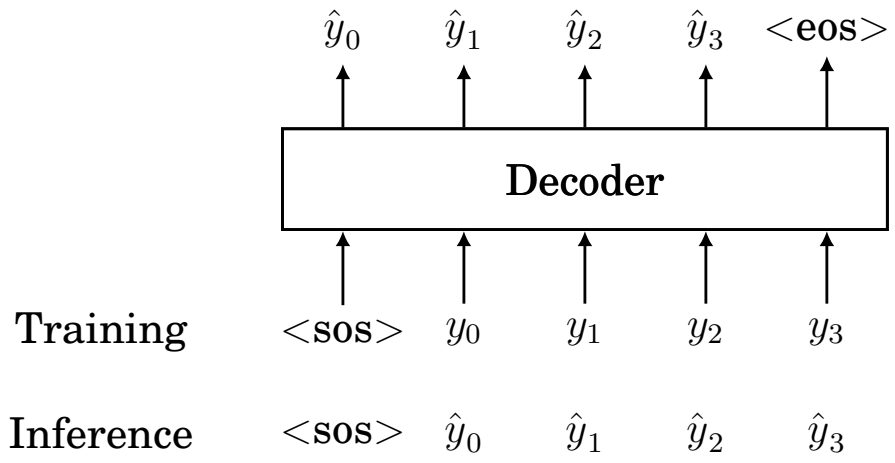

Figure 1.4: Autoregressive model: mismatch between training and inference. For training, ground truth sequence y is used while the network prediction $\hat{y}$ is the only choice for inference. y and $\hat{y}$ can be quite different.

formation from the start of the output sequence to the current position. One example is included in Figure 1.5. This limited information is probably not enough to make decisions. For instance, in speech recognition, uni-directional information introduces high uncertainties, which results in prediction errors. Those errors cannot be fixed when bi-directional information becomes available without beam search.

- Not intuitive and not similar to how the human brain works. Humans cannot recognize some words at the beginning of the recognition, but they can still determine them based on the context. The state-of-the-art autoregressive ASR systems work quite differently and need to determine each token or rely on the beam search.

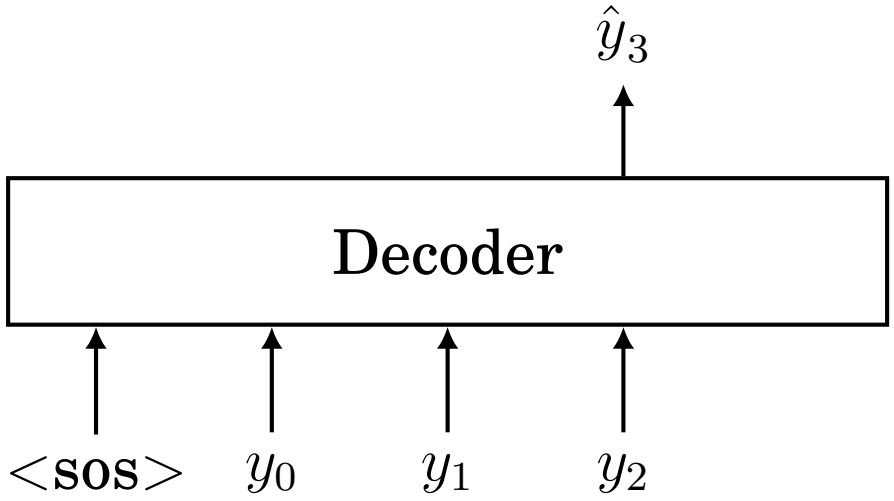

Figure 1.5: Autoregressive model: unidirectional decoding.

1.6 Towards Non-autoregressive Speech Applications

Autoregressive systems are easy to train but slow to decode. To bridge the gap between training and inference, a natural idea is to make the training process more challenging, which should be close to the inference. *Non-autoregressive* models are a different type of model which predicts the whole sequence directly. This improves inference speed by a large margin because a constant number of passes through the decoder is needed, which doesn't depend on the length of the output sequence. To predict the whole sequence, non-autoregressive models also take the whole sequence as input. Intuitively, the input sequence is incomplete, noisy, and contains many mistakes, while the

output sequence is expected to be more accurate than the input sequence. An example of comparison between autoregressive and non-autoregressive models is shown in Figure 1.6.

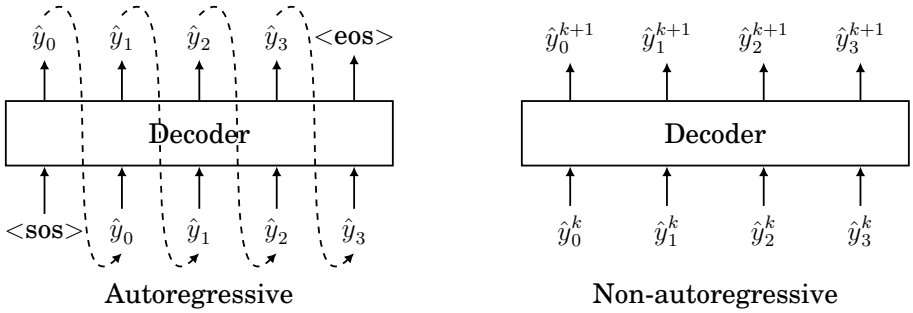

Figure 1.6: Comparison between autoregressive and non-autoregressive systems. They usually share the same encoder, so the encoder is omitted in this figure. For non-autoregressive systems, since the whole sequence is fed into the decoder, there is no need to use special tokens. Instead, it uses multiple iterations to decode the utterance and the result $\hat{\mathbf{y}}^{k+1}$ of $k+1$-th iteration depends on iteration k.

1.6.1 Benefits of Non-Autoregressive Systems

Compared to the autoregressive systems, non-autoregressive systems have the following advantages:

- Fast. Tokens can be predicted in parallel in **certain order** iteratively. So the inference complexity is linear with respect to the input utterance

length. For example, in speech recognition, different sentences can be decoded in parallel. This brings even large benefits to the speech synthesis since each second of audio requires 16,000 to 24,000 samples.

- Similar training and inference objective. In both training and inference, the network make predictions based on **partial** or **some initial** decoding results. This makes training and inference objectives similar. Also, non-autoregressive systems are able to avoid the post-processing stages such as beam search, which unifies training and inference.

- Bi-directional contextual information is taken into consideration. Non-autoregressive systems decode tokens simultaneously, which introduce bi-directional information. Errors made in the early stage can be resolved in the later iterations.

- More intuitive and similar to the human brain. Different non-autoregressive systems have different similarities, which are covered in the following sections.

1.6.2 Why Non-Autoregressive Systems are challenging to train

Non-autoregressive systems introduce strong conditional independence assumptions. Usually, results decoded in the same iteration are assumed to be conditional independent. This assumption is too strong in general. For instance, if nearby tokens are decoded together, the network may not have enough contextual information to decide them. One extreme example is that there might be multiple correct decoding sequences with the same length due to the ambiguity introduced by sub-word decomposition. That indicates it is better to decode nearby tokens in **different iterations**. Thus, multiple iterations might be helpful.

Another challenge is to decide the sequence length in advance. Insertion-based non-autoregressive systems [44, 45] can dynamically grow the canvas size, while the masking-based approaches require predicting the output sequence length in advance before the first decoding iteration. One possibility is to set the output sequence length to something related to the input sequence length. In such cases, the length of the final output is dynamically changed based on the predictions.

1.7 Non-Autoregressive Methods Proposal

In this dissertation, two different non-autoregressive directions are proposed to address concerns in the last section: mask-based and noise-based. These two directions are explored in other field, like machine translation [46], image generation [47]. It is the first time to be used for non-autoregressive speech applications.

1.7.1 Mask-based Approach

The mask-based approach introduces a special token ⟨MASK⟩ to indicate that the position hasn't been decided yet. During training, ground truth sequences are masked randomly, and the model is optimized to predict the original unmasked tokens. The inference starts from a sequence with all ⟨MASK⟩ s and gradually recovers the whole sequence based on **partial results**. The procedure of training and inference is illustrated in Figure 1.7.

One inference example is shown in Figure 1.8. The decoding is finished within $K = 3$ iterations, which indicates three passes through the decoder. The encoder only needs to be computed once.

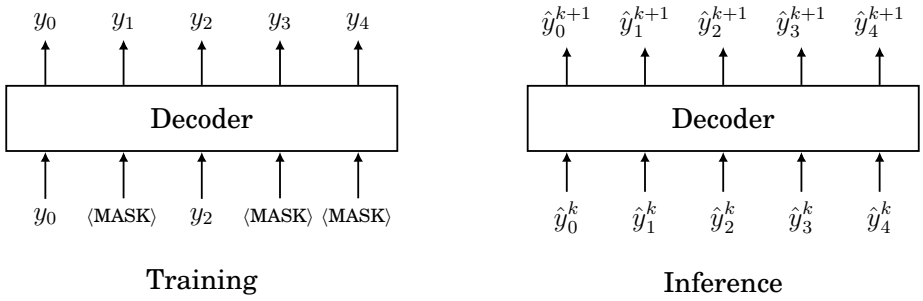

Figure 1.7: Diagram of mask-based approach. ⟨MASK⟩ is a special token indicates that the corresponding position is not decided yet. Inference starts from $\hat{\mathrm{y}}^0$ which contains only ⟨MASK⟩ .

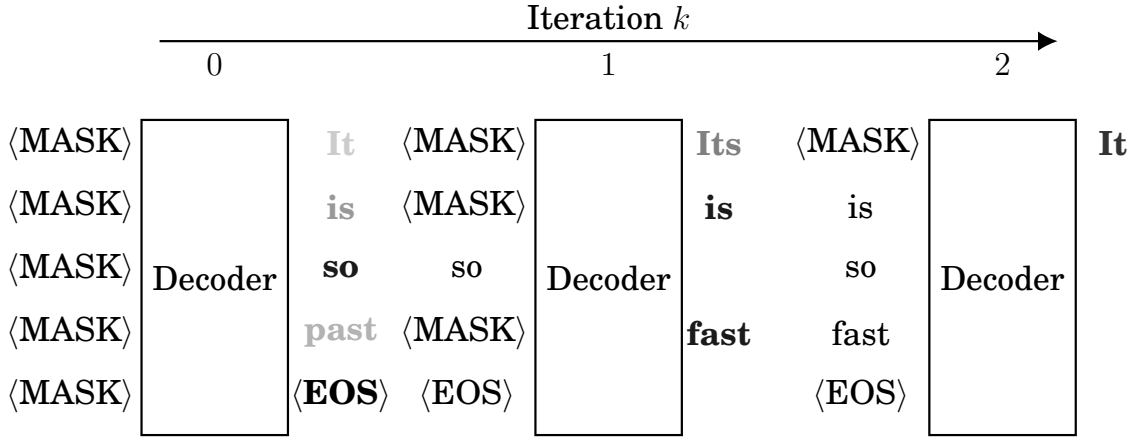

Figure 1.8: Example of mask-based approach decoding with $K = 3$ iterations for speech recognition. Shade indicates confidences from the model prediction.

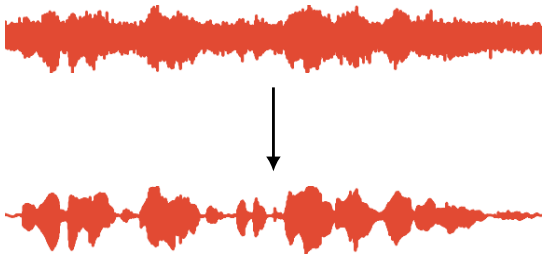

Figure 1.9: Example of noise-based approach training for speech synthesis.

1.7.2 Noise-based Approach

The noise-based approach adds noise to the ground truth sequence, and the network is learned to predict the original uncorrupted signal. Different amounts of noise are added during the training, so the model is expected to learn the denoising process with any portion of noise. During inference, starting from an **initial signal** with a certain amount of noise, the network refines the signal iterative to reach high quality.

Network training for noise-based speech synthesis approach is shown in Figure 1.9. Noise is injected into the clean waveform, and the model is trained to recover it from the corrupted one. This denoising process requires an understanding of the global structure.

Inference example is included in Figure 1.10. It starts from the pure Gaussian noise ($n = 0$), and the network denoises the input signal. The decoding finishes within $K = 6$ iterations.

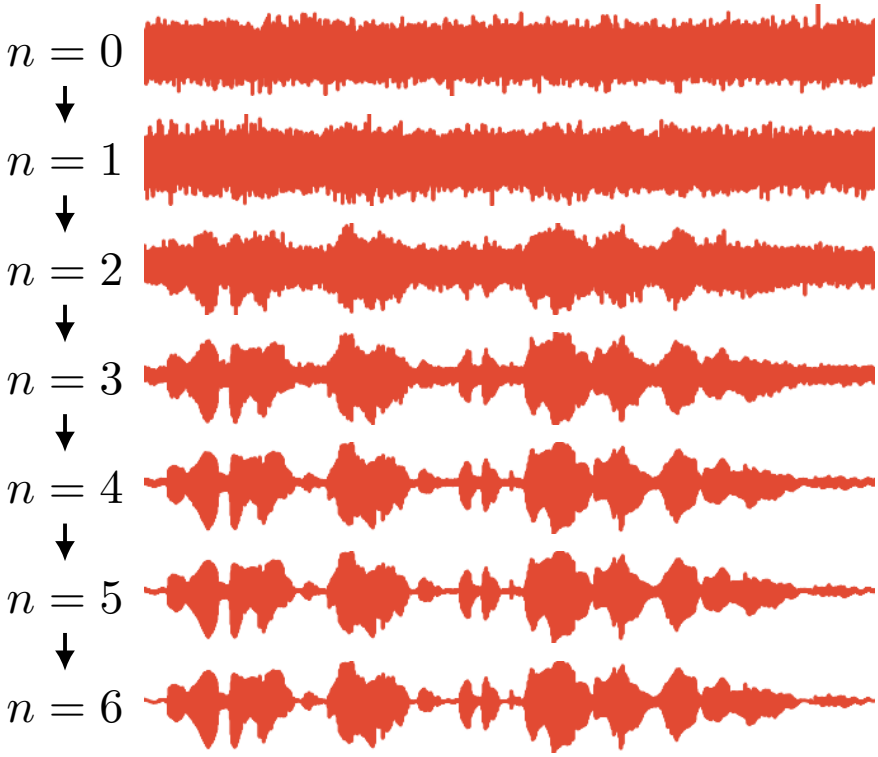

Figure 1.10: Example of noise-based approach decoding with $K = 6$ iterations.

1.7.3 Relationship between Two Methods

Mask-based and noise-based are two different ways to introduce the disturbances on the target sequence:

- Mask-based approach introduces special ⟨MASK⟩ token, which can be considered as a discrete noise
- Noise-based approach directly adds perturbations, which can be both continuous and discrete

Recovering from these disturbances helps the network to model the data distribution or conditional distribution.

In both cases, the network predicts the directions to higher log-likelihood regions. The mask-based model predicts the discrete edits (substitutions) on the discrete sequence. Those edits are directions to modify the discrete sequence to regions of higher log-likelihood, analogous to the gradient. The noise-based approach introduces perturbations to broaden the support of the data distribution, and the target is closely related to the gradient.

The major difference between the two proposed methods is the percentage of sequence updated during each iteration. Mask-based models tend to **determine** some tokens, and only some tokens are updated in each iteration. The number of changes they made usually depends on the number of iterations. For instance, if four iterations schedule is adopted, roughly one-quarter of the

predictions are selected after one pass. Noise-based models update the whole sequence every time to get a less noisy one. This matches the training process where different noise levels are introduced to sample noisy sequences.

1.8 Datasets

Comprehensive studies are conducted on the proposed methods. To verify their effectiveness, we rely on multiple datasets widely adopted by the academic or industry.

1.8.1 Speech Recognition

For the speech recognition, all systems are tested on four different datasets: 150-hours AISHELL dataset [48], 581-hours Corpus of Spontaneous Japanese (CSJ) dataset [49], 81-hours Wall Street Journal (WSJ) dataset [40] and 210-hours Tedlium2 dataset [50]. For the language, AISHELL includes Mandarin, while CSJ contains Japanese. Both WSJ and Tedlium2 are English datasets. Tedlium2 comes from real TED talks, which are noisier. The diversity of those corpora provides more convincing conclusions.

1.8.2 Speech Synthesis

For the speech synthesis task, we used Google's proprietary speech dataset consisting of 385 hours of high-quality English speech from 84 professional voice talents for training models. A female speaker in the training dataset is chosen for testing, who had also been used in previous studies [33, 51–53]. The test set included 1,000 sentences that cover different varieties of phonemes.

1.9 Contribution of this Dissertation

In the previous sections, we identified several disadvantages of autoregressive systems and the barrier of switching to the non-autoregressive models. Those are the topics covered in this dissertation, which makes the following contributions:

1. We propose mask-based speech recognition approaches, which include Conditional Masked Language Model (CMLM) and Factorized Masked Language Model (FMLM).

   (a) CMLM is inspired by the previous work [46], and it is the first attempt of non-autoregressive end-to-end speech recognition.

   (b) FMLM is proposed based on observations, and it reduces the number of required iterations.

   (c) They reach similar performance as the autoregressive baseline on Mandarin dataset while the performance is slightly worse on Japanese and English dataset.

2. We propose noise-based speech synthesis approaches which are non-autoregressive and support a dynamic trade-off between fidelity and inference speed.

   (a) WaveGrad is a vocoder that generates waveform samples from the input mel-spectrogram features.

   (b) WaveGrad 2 is a phone-to-wave model which is the combination of WaveGrad decoder with Tacotron-2 [33] based encoder.

   (c) Both outperform other non-autoregressive baselines and bridge the performance gap between autoregressive systems and non-autoregressive systems. The mean opinion score difference is less than **0.1**.

3. We adapted the noise-based approach idea to the speech recognition and proposed Align-Denoise, which favors a single pass decoding and can be considered as an extension to the existing CTC model.

   (a) The method significantly speeds up both the training and inference with no performance drop.

The methods proposed in this paper could be applied to lots of resource-limited scenarios, like smartphones, tablets, voice assistants. They can reduce power

consumption by reducing the inference time and providing better latencies.

Chapter 2

The Literature of Non-autoregressive Systems for Speech Applications

Fast inference is a major concern for deploying neural network-based models to real-world applications. In recent years, non-autoregressive systems have attracted more and more attention from the speech research community due to the fast and straightforward inference scheme. This chapter briefly reviews some previous essential work to better understand the literature of non-autoregressive systems for speech applications.

2.1 Non-autoregressive ASR

2.1.1 Hybrid Models

The hybrid speech recognition system introduced in Chapter 1.3.1 can be considered as a non-autoregressive system if both the acoustic model and language model are non-autoregressive. For example, the combination of deep neural network acoustic model [17] and n-gram language model [54] is non-autoregressive.

Common choice of acoustic model includes deep neural network (DNN) [17], time-delay neural network [55], long short term memory (LSTM) [56], bidirectional long short term memory (BLSTM) [57], Transformer [58], etc. Different architectures have different receptive fields, which impacts the accuracy of the acoustic unit prediction. For example, BLSTM outperforms LSTM by a large margin [57] since it provides bidirectional information which uses future context information.

The language model instead models how likely one text sequence appears. *n-gram* model is commonly used for many speech recognition applications, and different approaches [59, 60] have been proposed to improve it. Language model also benefits from advanced neural architecture, like recurrent neural network [61] (RNN), long short term memory [62] (LSTM), gated recurrent unit [63] (GRU), Transformer [64], etc. However, all those advanced models

are autoregressive: they condition on all or part of the previous tokens to predict the probability of the next token. The autoregressive characteristic makes the inference slow since tokens are predicted one by one.

The dependencies between tokens are handled by the Weighted Finite State Transducers [31] (WFST). Scores from ASR components, like acoustic model, lexicon, language model, are unified in the WFST framework, which provides flexibility for the implementation. For the inference, to find the optimal path from the WFST, usually, the beam search process is used to greedily maintain a certain number of most possible paths in the repository.

In summary, the hybrid model relies on a powerful language model to achieve high quality, but they are usually autoregressive. The beam search process introduces extra complexities which make the inference process slow. Also, it results in discrepancies between training and inference, which are harmful for optimization.

2.1.2 Connectionist Temporal Classification

For the end-to-end model, a real question is how to handle the mapping between frames and tokens. Commonly, multiple frames correspond to the same token, and the number of frames varies a lot due to different speeds, speakers, etc. Connectionist Temporal Classification [65] (CTC) is a fundamental approach designed for the end-to-end model, and it is also the cornerstone of

more advanced non-autoregressive systems.

CTC is proposed to handle the noisy and unsegmented input signal, for example, speech. Speech is continuous, and the boundary between different words or sub-words is usually not available. CTC is a general neural network output whose objective function does not require any duration or segmentation information.

CTC learns a direct mapping from the input frames to the vocabulary in addition to a special token ϵ. This special blank token ϵ represents observing a 'blank' or no label. To handle different alignment from various possible durations of the same word, CTC introduces a many-to-one mapping $\mathcal{B}$ between frame predictions and label sequence by removing all blank and repeated tokens from the prediction. Several examples are included in Table 2.1.

Table 2.1: Examples of CTC decoding. Mapping function $\mathcal{B}$ removes all blank and repeated tokens from the model prediction. From the second and the third case, to differentiate between 'll' and 'l', one ϵ is needed in the middle.

Model Prediction	**Result**
`hhhhellll_lo`	`hello`
`hhe__lll_llo`	`hello`
`hhe__llllllo`	`helo`

Given the many-to-one mapping $\mathcal{B}$, CTC maximizes the probability of all appropriate alignment π. One alignment π is appropriate if $\mathcal{B}(\pi) = y$. In other words, CTC optimizes the probability of all possible paths which are mapped to

the label sequence y after applying the mapping $\mathcal{B}$. By marginalizing over all possible alignments π, CTC does not require any segmentation and duration information during training.

To model the probability of individual alignment π given the input sequence x, CTC uses conditional independence assumption:

$$P(\boldsymbol{\pi} \mid \mathbf{x}) = \prod_{t} P(\boldsymbol{\pi}_t \mid \mathbf{x}) \tag{2.1}$$

So basically prediction of each frame π_t is independent conditioning on the input sequence x. The objective function of CTC is

$$\begin{aligned} \max P(\mathbf{y} \mid \mathbf{x}) &= \max \sum_{\boldsymbol{\pi}:\mathcal{B}(\boldsymbol{\pi})=\mathbf{y}} P(\boldsymbol{\pi}|\mathbf{x}) \\ &= \max \sum_{\boldsymbol{\pi}:\mathcal{B}(\boldsymbol{\pi})=\mathbf{y}} \prod_{t} P(\boldsymbol{\pi}_t \mid \mathbf{x}) \end{aligned} \tag{2.2}$$

During training, given the label sequence y, dynamic programming known as the forward backward algorithm [65] can efficiently optimize the probability of all possible paths. However the optimization for inference is not feasible and it usually relies on the approximation. One simple and practical solution is to use the greedy decoding

$$\boldsymbol{\pi}_t = \arg\max P(\boldsymbol{\pi}_t \mid \mathbf{x}) \tag{2.3}$$

and then the prediction can be computed accordingly using $\mathcal{B}$ mapping

$$\mathbf{y} = \mathcal{B}(\boldsymbol{\pi}) \tag{2.4}$$

The principal issue of CTC is the absence of dependence between predictions, as shown in Equation 2.1. To incorporate the external language model and beam search, the conditional independence assumption is usually artificially broken by the inclusion of a language model. This results in a discrepancy between how models are trained and tested. In reality, CTC models exhibit fast decoding but relatively poor performance [66].

2.1.3 RNN Transducer (RNN-T)

CTC makes a strong conditional independence assumption, and it requires that the output length be smaller than the input length. RNN Transducer [67] (RNN-T) elegantly solves both problems associated with CTC by introducing two sub-networks:

- **Predictor** takes previous outputs and produces features in an autoregressive manner that can be used for predicting the following output.
- **Joiner** combines outputs from both encoder and predictor to predict the next token.

By allowing multiple outputs for each input, RNN-T outperforms CTC in many experiment results [66].

However, RNN-T makes both training and inference slow by using the autoregressive predictor. Since the predictor requires the history, outputs can

not be estimated simultaneously, limiting its usage for real applications. Also, the memory usage of RNN-T is quite large due to the computation of weights between each input frame and each output.

2.1.4 Non-autoregressive End-to-end Speech Recognition

Non-autoregressive systems are proposed to speed up the inference process. For the hybrid system, the bottleneck of the inference usually comes from the beam search and the external language model. Even though both acoustic and language models can be non-autoregressive, the beam search still slows down the inference procedure and is not powerful enough to handle the dependencies between tokens.

A more exciting direction for speech recognition is the *end-to-end non-autoregressive models*. End-to-end non-autoregressive models predict characters or subwords directly, which does not require the participation of any other models. By adopting powerful advanced models, we further show it is possible to avoid the beam search during decoding.

Conditional Masked Language Model (CMLM) and Factorized Masked Language Model (FMLM) [68] are the first attempts on the end-to-end non-autoregressive speech recognition, which is covered in Chapter 3. CMLM

is inspired by the previous work [46] on Neural Machine Translation, and we explored its usage on the speech recognition task. We further proposed a novel FMLM to bridge the gap between training and inference based on the observation. Our work verifies the possibility to make the non-autoregressive system work with speech recognition, and we show similar or slightly worse performance on different datasets.

Mask-CTC [69] is an extension to the proposed CMLM. The training procedure is the same as CMLM, but the CTC greedy decoding result is adopted as the initialization for inference. Mask-CTC proposes different criteria to select some predictions, which are refined in the following iterations.

There is another direction of non-autoregressive speech recognition which is based on CTC, including Imputer [70], Align-Refiner [71]. Those models work directly on CTC alignment, and they need to apply the mapping $\mathcal{B}$ at the end of the decoding. Imputer [70] applies a similar conditional masked language model (CMLM) objective on the CTC alignment, and fixed block-based decoding order is proposed. Align-Refiner [71] relies on the CTC greedy decoding, and it optimizes the decoder (refiner) to reduce the word error rate after refining multiple steps.

2.2 Non-autoregressive Text-to-speech

Two popular directions of non-autoregressive models have been explored: The flow-based approach and the GAN-based approach.

Flow-based models [72–77] is a generative model which is constructed by a sequence of invertible transformations. The model explicitly learns the data distribution $p(\mathbf{x})$ by leveraging normalizing flow [78, 79] and therefore the loss function is simply the negative log-likelihood. Normalizing flow maps proposed simple distribution $p(\mathbf{z})$ to the data distribution $p(\mathbf{x})$ via a series of invertible transformations such as invertible 1 by 1 convolution [80], coupling-based flows [72]. Those transformations require to be invertible in order to estimate the likelihood, and their Jacobian determinants need to be easy to compute for efficiency. Those require specialized architectures, which limit the expressive power.

Generative Adversarial Network [53,81–87] (GAN) based approach uses the combination of two sub-networks. The first sub-network, *generator*, maps the simple distribution $p(\mathbf{z})$ to the data distribution we are trying to model. The second sub-network, *discriminator*, differentiates between real samples and samples from the generator network. The discriminator provides the training signal to the generator during the training and the generator is optimized to improve the quality to fool the discriminator. Text-to-speech is a multi-

modal problem and there usually exists multiple possibilities corresponding to the same input sequence. GAN is a good fit for the speech synthesis problem because it differentiates distributions instead. However, GAN-based approaches usually require additional losses, for example, losses on different resolutions [53] or STFT-based losses [42]. Also, GAN training is not stable without proper hyperparameters.

2.3 Summary

With the development of deep learning, the performance of many speech applications gets considerable improvement. Based on that, more and more researchers and engineers have started to focus on improving the current state-of-the-art autoregressive systems. Non-autoregressive is a very promising direction, which involves more parallel computation. This dissertation proposes two novel methods to improve the existing non-autoregressive speech applications.

Chapter 3

Mask-based non-autoregressive Speech Recognition

The first attempt of non-autoregressive speech recognition combines a state-of-the-art autoregressive ASR system with mask-based training. The autoregressive system discussed in this dissertation is the joint training of connectionist temporal classification and attention [88]. This chapter starts with the introduction of autoregressive system and walks through the process to make the autoregressive system non-autoregressive by introducing the mask-based idea. Two variations are discussed for the network training, which are named conditional masked language model (CMLM) and factorized masked language model (FMLM). Experiments include results on different datasets and ablations related to the external language model and beam search, which

are broadly used for autoregressive systems.

The Mask-based non-autoregressive model is built on the base of the current autoregressive end-to-end speech recognition system. A general sequence-to-sequence autoregressive model consists of an encoder and decoder. The encoder takes the t-th input frame speech features $\mathbf{x}_t$ like the log Mel filter bank coefficients as input and produces the corresponding hidden representations

$$\mathbf{h}_t = \mathbf{h}_t(\mathbf{x}) \tag{3.1}$$

The decoder predicts a next token $\mathbf{y}_l$ based on the previous history $\mathbf{y}_{<l}$ and all hidden representations $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \cdots)$:

$$P(\mathbf{y}_l \mid \mathbf{y}_{<l}, \mathbf{x}) = P_{\mathsf{dec}}(\mathbf{y}_l \mid \mathbf{y}_{<l}, f_l(\mathbf{h})) \tag{3.2}$$

where f is a l-dependent function on all hidden representations $\mathbf{h}$. A common choice for f is an attention mechanism, which can be considered to be a weighted combination of all hidden representations:

$$f_l^{\mathsf{att}}(\mathbf{h}) = \sum_t w_{l,t}\mathbf{h}_t \tag{3.3}$$

where t enumerates all possible hidden representations in $\mathbf{h}$. The weight $w_{l,t}$ is usually determined by a similarity between the decoder hidden state at l and hidden representation $\mathbf{h}_t$.

3.1 Conditional masked language model (CMLM)

The first training framework that we considered is inspired by the work from non-autoregressive neural machine translation [46]. The idea is to replace $\mathbf{y}_{<l}$ with partial decoding results we got from previous computations. A new token $\langle \text{MASK} \rangle$ is introduced for training and decoding, similar to the idea of BERT [89]. Let L_{M} and L_{U} be the sets of masked and unmasked tokens, respectively. The posterior of the masked tokens given the unmasked tokens and the input speech is approximated by,

$$P(\mathbf{y}_{L_{\text{M}}} \mid \mathbf{y}_{L_{\text{U}}}, \mathbf{x}) = \prod_{l \in L_{\text{M}}} P_{\mathsf{dec}}(\mathbf{y}_l \mid \mathbf{y}_{L_{\text{U}}}, f_l(\mathbf{h})) \ . \tag{3.4}$$

where the conditional independence assumption still holds like equation 2.1 for CTC. However, the main difference is the introduction of the partition L_{M} and L_{U}. In the case of CTC, the unmasked part L_{U} is empty, so all tokens are conditionally independent. In our case, we carefully choose the split and adopt predictions with high confidence. By embracing an iterative decoding strategy, we gradually increase the size of the set of unmasked tokens L_{U} and decrease the size of the masked part L_{M} by selecting predictions, starting from the empty set L_{U} and the whole set L_{M}.

During training, some random tokens are replaced by this special $\langle \text{MASK} \rangle$ token.

The network is asked to predict the original unmasked tokens based on input speech and context. The total number of mask tokens is randomly sampled from a uniform distribution of the whole utterance length. Ground truth tokens are randomly selected to be replaced with this ⟨MASK⟩ token. Intuitively, when we mask more tokens, the model will rely more on the input speech. In contrast, if we mask fewer tokens, the model will leverage more the language context, similarly to a language model. This combines the advantages of both speech recognition and language modeling. With the assumption that predictions of masked tokens are conditionally independent given unmasked tokens, masked tokens can be estimated simultaneously as the product in equation (3.4).

Because of the connection between this objective function and mask-based pre-training [89], we named this approach Audio-Conditional Masked Language Model (A-CMLM).

3.2 Factorized masked language model (FMLM)

During the training of A-CMLM, ground truth tokens at L_{U} in (3.4) are provided to predict the masked part. However, during inference, none of those tokens are given. Thus the model needs to predict without any context infor-

mation. This mismatch might be harmful to the final performance.

Inspired by [90, 91], we formalize the idea to mitigate the training and inference mismatch as follows. Let $Z_i \subset [0, 1, ..., L-1]$ be a length-$(K+1)$ subsequences of indices such that

$$\begin{aligned} Z_0 &= \emptyset \\ Z_K &= [0, 1, ..., L-1] \\ \forall i \quad Z_i &\subset Z_{i+1} \end{aligned} \tag{3.5}$$

where L is the output sequence length and K is the number of iterations. For both training and inference, the objective can be expressed as

$$P(\mathbf{y} \mid \mathbf{h}) = \prod_{i=1}^{K} \prod_{l \in Z_i \cap \overline{Z_{i-1}}} P_{\mathsf{dec}}(\mathbf{y}_l \mid \mathbf{y}_{Z_{i-1}}, f_l(\mathbf{h})) \tag{3.6}$$

where $\overline{Z}_{i-1}$ are the set of indices not included in Z_{i-1}, $Z_i \cap \overline{Z}_{i-1}$ are the indices for decoding in iteration i. For example, to decode an utterance of length 5 with 3 iterations, one possibility is:

$$\begin{aligned} Z_0 &= \emptyset \\ Z_1 &= 0 \\ Z_2 &= 0, 1, 2 \\ Z_3 &= 0, 1, 2, 3, 4 \end{aligned} \tag{3.7}$$

Similar to A-CMLM, $\langle$MASK$\rangle$ tokens are added to decoder inputs when corresponding tokens are not decided yet. A special case is the autoregressive case: $K = L$ and $Z_i = [0, 1, ..., i-1]$.

For the inference, one can only rely on the confidence from the model predictions. Ideally, Z_i should be decided based on confidence scores for the training to match the inference case. During training, we sort all posteriors from iteration $i-1$ and choose the most confident ones. The size of Z_i is also sampled from the uniform distribution between 0 and L to support different possibilities for decoding. To speed up, we set $K = 2$ during training so that the optimization objective can also be written as

$$\begin{aligned} P(\mathbf{y} \mid \mathbf{h}) = & \prod_{l \notin Z_1} P_{\mathsf{dec}}(\mathbf{y}_l \mid \mathbf{y}_{Z_1}, f_l(\mathbf{h})) \\ & \times \prod_{j \in Z_1} P_{\mathsf{dec}}(\mathbf{y}_j \mid f_j(\mathbf{h})) \end{aligned} \tag{3.8}$$

Comparing with equation (3.4), A-CMLM training only includes first term if $Z_1 = L_{\mathrm{U}}$ and doesn't optimize the probability $\prod_{j \in L_{\mathrm{U}}} P_{\mathsf{dec}}(\mathbf{y}_j \mid f_j(\mathbf{h}))$ for the first iteration. However, during inference, some explicit factorization is still needed based on the confidences.

Pseudo-code of the A-FMLM algorithm can be found in Algorithm 1.

3.2.1 Easy First Decoding

During inference, a multi-iteration process is proposed. Autoregressive models typically use left-to-right decoding. In [46], they propose mask-predict for non-autoregressive neural machine translation. In comparison, we propose a novel, simple decoding method for non-autoregressive speech recogni-

input : minibatch size n, dataset D, encoder network f_{enc}, decoder network f_{dec}

output: Posterior P

Sample $X = \mathbf{x}_1, ..., \mathbf{x}_n$, $Y = \mathbf{y}_1, ..., \mathbf{y}_n$ from D;

$\mathbf{h} = f_{\text{enc}}(\mathbf{x})$;

$\hat{\mathbf{y}}^{(0)}[:] = \langle \textbf{MASK} \rangle$;

$P(\mathbf{y}^1|\mathbf{h}) = f_{\text{dec}}(\hat{\mathbf{y}}^{(0)}, \mathbf{h})$;

$\text{mask} = zeros(n, \text{max\_length})$;

$\hat{\mathbf{y}}^{(1)}[:] = \langle \textbf{MASK} \rangle$;

for i=1,...,n **do**

$\text{probs} = P_i(\mathbf{y}^1|\mathbf{h})$;

$\text{indices} = \text{argsort}(\text{probs}.max(-1))$;

$Z_i \sim \text{Uniform}(1, length(\text{probs}))$;

$\text{mask}[i, \text{indices}[Z_i :]] = 1$;

$\hat{\mathbf{y}}^{(1)}[i, \text{indices}[Z_i :]] = y[i, \text{indices}[Z_i :]]$;

end

$P(\mathbf{y}^2|\mathbf{y}^1, \mathbf{h}) = f_{dec}(\hat{\mathbf{y}}^{(1)}, \mathbf{h})$;

$P = \text{mask} * P(\mathbf{y}^1|\mathbf{h}) + (1 - \text{mask}) * P(\mathbf{y}^2|\mathbf{y}^1, \mathbf{h})$;

Algorithm 1: Training procedure for A-FMLM

tion: easy first.

The idea of this strategy is to predict the most confident tokens first, similar to easy-first parsing [92]. In the first iteration, the decoder is fed with predictions $\hat{\mathbf{y}}_l^{(0)} = \langle \text{MASK} \rangle$ tokens for all l since we do not have any partial results. After getting decoding results $P(\mathbf{y}_l^1|.)$[1], we keep those most confident ones and update them in $\mathbf{y}_l$:

$$\hat{\mathbf{y}}_l^{(1)} = \begin{cases} \arg\max_V P(\mathbf{y}_l^1|.) & l \in \text{largest}_C(\max_V P(\mathbf{y}^1|.)) \\ \hat{\mathbf{y}}_l^{(0)} & \text{otherwise} \end{cases} \tag{3.9}$$

where V is the vocabulary, $C = \lceil L/K \rceil$ is the number of largest predictions that we keep. Conditioned on this new $\hat{\mathbf{y}}^{(1)}$, the network is required to make new predictions if there are still masked tokens. One example is shown in Figure 3.1.

3.2.2 Mask-predict

This is studied in [46]. Similarly to Section 3.2.1, we start with $\hat{\mathbf{y}}_t^{(0)} = \langle \text{MASK} \rangle$. In each iteration k, we check the posterior probability of the most probable token for each output t (i.e., $\max_V P(\mathbf{y}_t^k|.)$) and use this probability as a confidence score to replace least confident ones in an utterance by $\langle \text{MASK} \rangle$ tokens. The number of masked tokens in an utterance is $\lceil L * (1 - k/K) \rceil$ for k-th itera-

[1] We omit the dependencies of the posterior to keep the notation uncluttered.

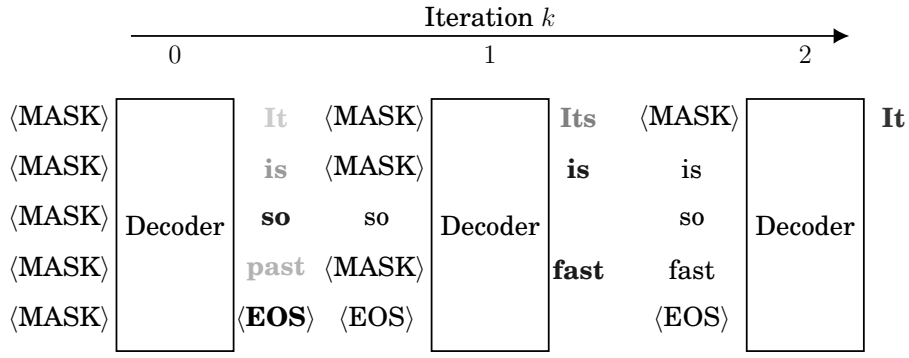

Figure 3.1: An illustration of the *easy first* inference procedure. In the first iteration, all input tokens are ⟨MASK⟩. Shade here presents the certainties from network outputs. For example, token "so" is confident enough in the first iteration to be decided and it will never change in the future for easy first decoding.

tion:

$$\hat{\mathbf{y}}_t^{(k)} = \begin{cases} \langle \text{MASK} \rangle & t \in \text{smallest}_C(\max_V P(\mathbf{y}_t^k|.)) \\ \arg\max_V P(\mathbf{y}_t^k|.) & \text{otherwise} \end{cases} \tag{3.10}$$

where $C = \lceil L * (1 - k/K) \rceil$. For instance, if $K = 10$, we mask 90% tokens in the first iteration, 80% in second and so on. After getting prediction results, we update all tokens previously masked in $\hat{\mathbf{y}}^{(k-1)}$:

$$P(\mathbf{y}_t^k|.) = \begin{cases} P(\mathbf{y}_t^k|.) & \hat{\mathbf{y}}_t^{(k-1)} = \langle \text{MASK} \rangle \\ P(\mathbf{y}_t^{k-1}|.) & \text{otherwise} \end{cases} \tag{3.11}$$

The difference between mask-predict and easy first is that mask-predict will accept all decisions but it reverts decisions made earlier if it is less confident. Easy first is more conservative and it gradually adopts decisions with the high-

est confidence. For both strategies, predictions become more and more accurate since it can utilize context information from both future and past directions. This is achieved by replacing input $\mathbf{y}_{<t}$ with all $\mathbf{y}_t$ since left-to-right decoding is no longer necessary.

3.2.3 Example

As shown in Figure 3.2, part (a) shows easy first and part (b) demonstrates mask-predict. In this example sequence length is 4 but after adding ⟨EOS⟩ token to the end of the sequence we have $L = 5$ and $K = 3$. When $K = 2$ the two strategies become the same. In the first iteration, the network is inputted with all ⟨MASK⟩. Top $\lceil 5/3 \rceil = 2$ tokens get kept in each iteration, and based on partial results network predicts again on all rest ⟨MASK⟩ tokens.

For easy first, it always ranks confidence from the last iteration and then keeps top-2 confident predictions. Based on partial results, it will complete the rest.

For mask-predict, it maintains confidence scores from multiple iterations. It chooses the least confident ones from all scores to mask. The last iteration chooses to change its previous prediction of “so” because its confidence is less than other predictions from the second iteration.

The left-to-right inference procedure can be considered as a particular case when $K = L$ and instead of taking the most confident one, the prediction of the

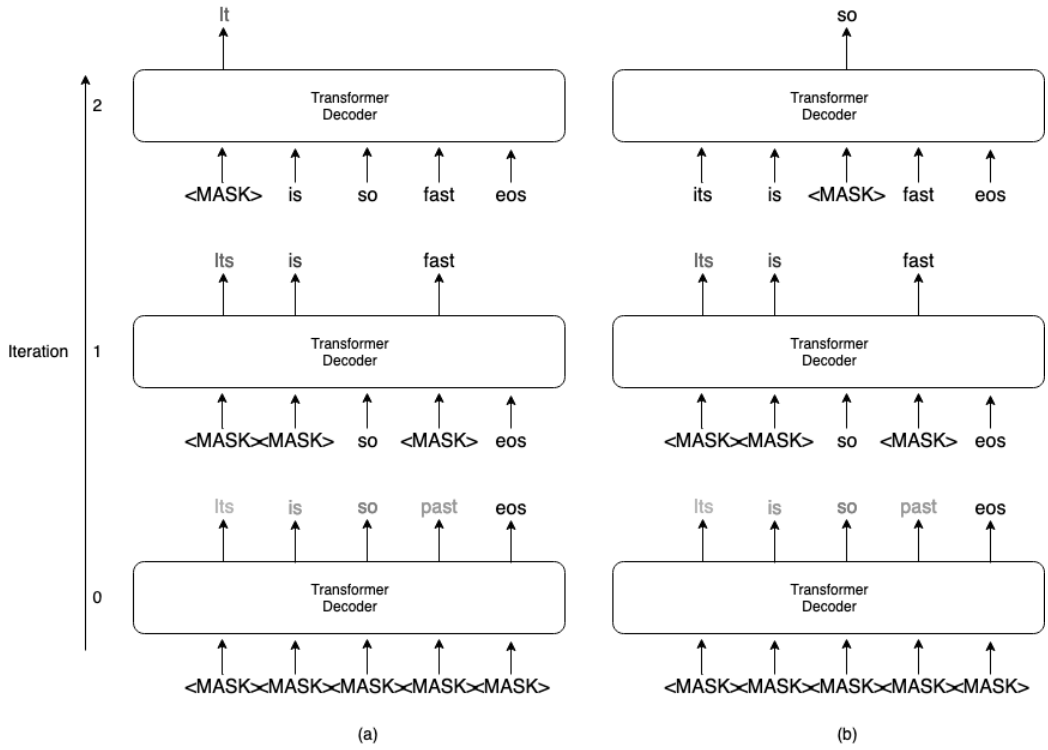

Figure 3.2: Illustration of inference procedure. To predict the whole sequence with $K = 3$ passes, initially, the network is fed with all ⟨MASK⟩ tokens. Shade here presents the certainties from network outputs. Part (a) shows the easy first process. Since token “so” is confident enough in the first iteration to be decided, it will never change in the future. Part (b) shows the mask-predict process. The last iteration goes back to the word ”so” because it is less confident in the first iteration than other predictions in other iterations.

next token is always adopted.

3.2.4 Output sequence length prediction

In [46], they introduced a special token ⟨LENGTH⟩ in input to predict output sequence length in neural machine translation. For ASR, length prediction is more difficult since the output length varies much more w.r.t. the speech duration. In this dissertation, a more straightforward approach is proposed: we asked the network to predict end-of-sequence token ⟨EOS⟩ at the end of the sequence as shown in Fig. 3.2.

During inference, we still need to specify the initial length. We manually set it to a constant value depending on the input length for the first iteration. So the first input sequence to the decoder contains only ⟨MASK⟩ tokens, and the length depends on the input audio length. After that we estimate the sequence length from the first iteration ⟨EOS⟩ prediction. Specifically, we look for the location of the first ⟨EOS⟩ token and truncate the sequence after that. One example is included in Figure 3.3.

3.3 Experiments and results

In all experiments, the encoder included 12 blocks with convolutional layers at the beginning for down-sampling. The decoder consisted of 6 blocks,

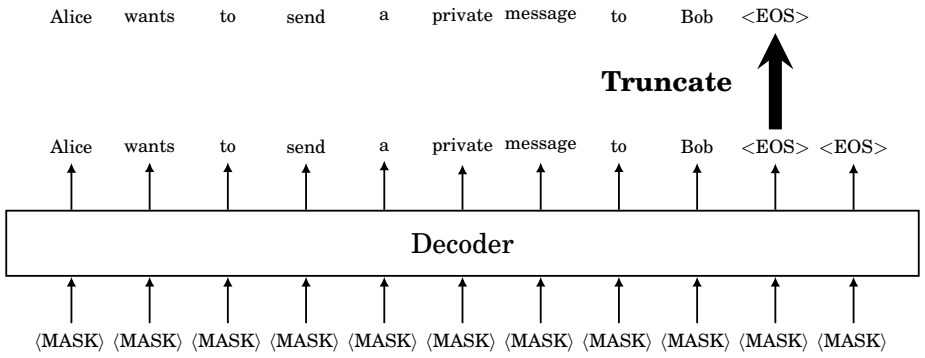

Figure 3.3: How the sequence length is predicted for A-CMLM/A-FMLM. Initially we estimate the length based on the input sequence length. After first iteration prediction, we truncate the sequence according to the position of the first ⟨EOS⟩ token.

and each block included four attention heads, 256 hidden units, and 2048 feed-forward units. The network was trained for 50/100/300/150 epochs with 1/2/1/1 NVIDIA GTX1080TI GPUs for AISHELL/CSJ/WSJ/Tedlium2. In comparison to autoregressive baselines, we observed that, in general, non-autoregressive models take more iterations to converge. Warmup [39] is used for the first 25,000 steps. The real-time factor is measured on the same machine with Intel(R) Xeon(R) CPU E5-2640 v3 CPU for decoding. For AISHELL/CSJ/WSJ, we used a character-based model while byte pair encoding [93] is adopted for Tedlium2. For all experiments, we perform post-processing based on beam search and language model shallow fusion on top of our proposed methods to make fair comparisons with the conventional autoregressive methods. The

same language model is used for both proposed non-autoregressive systems and the autoregressive baseline.

The results of AISHELL are given in Table 3.1. For A-CMLM, no improvement was observed for more than three decoding iterations. For A-FMLM, experiments show that one decoding iteration is enough to get the best performance. All decoding methods result in performance very close to state-of-the-art autoregressive models. Especially A-FMLM matched the performance of autoregressive baseline, but real-time factor reduced from 1.44 to 0.22, which is around **7x speedup**. The reason is that our non-autoregressive systems only perform major decoder computation a constant number of times K, compared to the autoregressive model, which depends on the length of output sequence L. It also outperformed two different hybrid systems in Kaldi by 22% and 11% relative, respectively. From our observations, the computation time for the proposed method is dominated by the beam search, so the real-time factor is similar for a different number of iterations.

Table 3.1: Comparison of baselines, previous work, A-CMLM and A-FMLM on AISHELL. Easy first is used for all proposals since it shows better performance from experiment results.

System	**Dev CER(↓)**	**Test CER(↓)**	**Real Time Factor(↓)**
Baseline			
Autoregressive Transformer	**6.0**	**6.7**	1.44
Kaldi nnet3	-	8.6	-
Kaldi chain	-	7.5	-
Previous work			

CAT [94]	-	6.3	-
Unsupervised Pre-training [95]	-	6.7	-
A-CMLM + Mask-predict (K=3) [46]	6.4	7.2	0.24
Proposal			
A-CMLM (K=1)	6.8	7.6	0.22
A-CMLM (K=3)	6.4	7.1	0.22
A-FMLM (K=1)	**6.2**	**6.7**	0.28
A-FMLM (K=2)	**6.2**	6.8	0.22
A-FMLM (K=3)	**6.2**	6.8	0.32

CSJ results are given in Table 3.2. Here, we observed a larger difference between non-autoregressive and autoregressive models. Multiple iterations of different decoding strategies are not helping to improve. Still, the proposed A-FMLM outperformed A-CMLM with up to **11x** speedup compared to the autoregressive baseline.

We also compare with our implementation of Imputer [70]–another non-autoregressive model. To make the model size comparable, we use eight blocks, including four attention heads, 256 hidden units, and 2048 feed-forward units. Since Imputer estimates the alignment, we changed the down-sampling rate from 4 to 2 to fit it into the memory. In comparison, the results from Imputer model are slightly better than the ones of our models, but the real-time factor is almost two times larger. Imputer depends on the input frame length, while our model depends on the output token level. Since there are many more frames than output tokens, the proposed model's inference is faster and easier.

Table 3.2: Comparison of baselines, previous work, A-CMLM and A-FMLM on CSJ.

System	**Eval1 CER(↓)**	**Eval2 CER(↓)**	**Eval3 CER(↓)**	**Real Time Factor(↓)**
Baseline				
Autoregressive Transformer	**5.9**	**4.1**	**4.6**	9.50
Kaldi	7.5	6.3	6.9	-
Other non-autoregressive methods				
CTC	8.0	5.4	7.2	1.47
Imputer (DP) our implementation	7.0	4.7	5.9	1.48
Proposal				
A-CMLM (K=1)	8.8	6.7	7.4	0.94
A-CMLM (K=3)	9.3	7.0	8.3	0.98
A-FMLM (K=1)	**7.1**	**5.2**	**6.2**	**0.88**
A-FMLM (K=2)	7.3	5.3	**6.2**	**0.88**

Performance of our systems on Wall Street Journal (WSJ) is given in Table 3.3. For WSJ, we tried both subwords and characters-based tokenization. Overall, using characters gives a better word error rate than using subwords. This can be explained by a small training set, and characters work better for out-of-vocabulary words. A-CMLM slightly outperforms A-FMLM on this dataset, and overall autoregressive baseline is still better.

Table 3.3: Comparison of baselines, previous work, A-CMLM and A-FMLM on WSJ. For char-based and BPE-based models, different external language models are used for beam search.

System	**dev93 WER(↓)**	**eval92 WER(↓)**
Baseline		

Autoregressive Transformer (char)	**7.8**	**5.3**
Kaldi		
Other non-autoregressive methods		
CTC [69]	22.2	17.9
Proposal (char)		
A-CMLM (K=1)	11.0	8.0
A-CMLM (K=3)	17.8	14.7
A-FMLM (K=1)	13.6	10.2
A-FMLM (K=2)	13.6	10.2
Proposal (BPE, 100)		
A-CMLM (K=1)	12.9	10.9
A-CMLM (K=3)	14.6	17.1
A-FMLM (K=1)	13.7	11.4
A-FMLM (K=2)	13.7	11.4

A-CMLM also get 13.0/10.7 WER on Tedlium2 as reported in Table 3.4, which is slightly better than A-FMLM. The difference is about one percent difference. All results are slightly worse than the autoregressive baseline.

Table 3.4: Comparison of baselines, previous work, A-CMLM and A-FMLM on Tedlium2.

System	**dev WER(↓)**	**test WER(↓)**
Baseline		
Autoregressive Transformer (BPE, 500)	11.7	9.9
+ beam search(40)	10.4	9.0
+ external language model	9.3	8.1
Kaldi	7.5	6.3
Other non-autoregressive methods		
CTC [96]	-	16.6
Proposal (BPE, 500)		
A-CMLM (K=1)	13.0	10.7
A-CMLM (K=3)	14.6	12.0
A-FMLM (K=1)	14.1	11.9
A-FMLM (K=2)	14.1	11.9

3.4 Analysis

3.4.1 Ablation Studies of External Language Model

In this section, we analyze the contribution of the external language model. In contrast to neural machine translation, the state-of-the-art end-to-end autoregressive model still includes an external language model to get the best performance. For a fair comparison, we also used the same external LM for the proposed non-autoregressive models. Table 3.5, Table 3.6, Table 3.7 and

Table 3.8 show the performance comparison.

From the results, the contribution of the external language model depends on the amount of text data. The external language model indicates a separately trained language model, which is trained on a text corpus. For some corpus like AISHELL and CSJ, additional text data are not available. For Tedlium2 and WSJ, some additional text data is not included in the ASR training pairs.

When the additional text corpus is not available, the performance difference is not significant. From Table 3.5, it improves the performance on AISHELL while the performance is similar on CSJ shown in Table 3.6. The difference between the two datasets can explain this. AISHELL has less training data and utterance length is shorter, which provides less contextual information. CSJ includes a larger training set, and utterance length is longer in comparison.

Table 3.5: Comparison of A-FMLM with and without external LM on AISHELL.

System	**Dev CER**	**Test CER**
A-FMLM (K=1) with LM	**6.2**	**6.7**
A-FMLM (K=1) without LM	6.9	7.9

Table 3.6: Comparison of A-FMLM with and without external LM on CSJ.

System	**Eval1 CER**	**Eval2 CER**	**Eval3 CER**
A-FMLM (K=1) with LM	**7.1**	**5.2**	**6.2**
A-FMLM (K=1) without LM	7.4	5.2	5.9

When additional text data can be used to train the external language model, the performance difference is much more significant. For WSJ, as shown in Table 3.7, the word error rate almost increases by 100% when the external LM is not used. From our observation, the language model helps detect the boundary of words, leading to a large word error rate improvement on this dataset. For Tedlium2 results in Table 3.8, external LM improves the performance by 4% absolutely.

Overall, the external language model improves beam search performance, especially when additional text data is available. The proposed non-autoregressive models also benefit from the development of the language model.

Table 3.7: Comparison of A-CMLM with and without external LM on WSJ.

System	**dev93 CER**	**dev93 WER**	**eval92 WER**	**eval92 WER**
A-CMLM (K=1) with LM	**4.0**	**11.0**	**2.9**	**8.0**
A-CMLM (K=1) without LM	5.9	21.6	4.8	19.1

Table 3.8: Comparison of A-CMLM with and without external LM on Tedlium 2.

System	**dev WER**	**Test WER**
A-CMLM (K=1) with LM	**13.0**	**10.7**
A-CMLM (K=1) without LM	17.0	14.8

3.4.2 Ablation Studies of Beam Search

In this part, we are exploring how beam search improves decoding performance. For non-autoregressive speech recognition models, some tokens need to be predicted without enough contextual information. Especially for tokens that are predicted in the first iteration, they are estimated solely on the input audio. This results in large uncertainties at the beginning of the inference when the context is not available.

There are two different ways to overcome this problem. One solution is to use the beam search and external language model to combine the context information from the text. Predictions made by the model are purely conditioned on the input speech, but beam search reduces uncertainties by searching over the whole sequence. The external model further combines the contextual text information, which leads to further improvement. This is adopted for all previous experiments.

Another solution is to make some errors first and then correct them relying on the noisy context. Intuitively, this requires correcting mistakes made before and a large number of iterations. This suggests mask predict is preferred in this scenario and more iterations are needed. Experiments are conducted using CMLM model with mask predict decoding on Tedlium2 test. To make the comparison fair, external language models are not used for all experiments in this section. The results are shown in Figure 3.4.

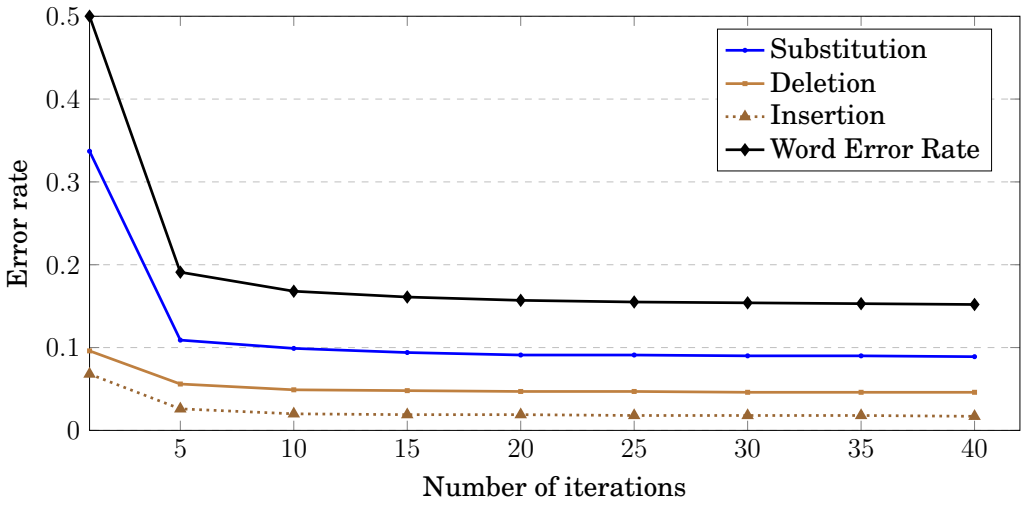

Figure 3.4: How the number of refinement iterations impacts the result without beam search using mask predict

From the graph, it can be observed that beam search is not necessary, but it greatly reduces the number of iterations needed. Without beam search, more iterations, in general, improve the performance by up to 40 iterations. Substitution errors are reduced from 33.7% to 9.1% by using 40 iterations. Instead, insertion and deletion errors are pretty stable when the number of iterations increases. This suggests that position predictions are quite accurate for the trained models, but there are substitution errors at the beginning of decoding. Mask predict decoding is able to correct errors which results in continuous improvement.

The real-time factor (RTF) for different number of iterations is given in Figure 3.5. Clearly, the RTF is linear with respect to the number of steps when

the beam search is not used. This matches one's expectations.

Final results for Tedlium2 are summarized in Table 3.9. Clearly, more iterations without beam search are three times faster than the beam search approach, and the performance difference is relatively small (about 0.4, 2.6% relatively). Comparing to Table 3.8, one can observe that external language model is much more important than beam search on this dataset.

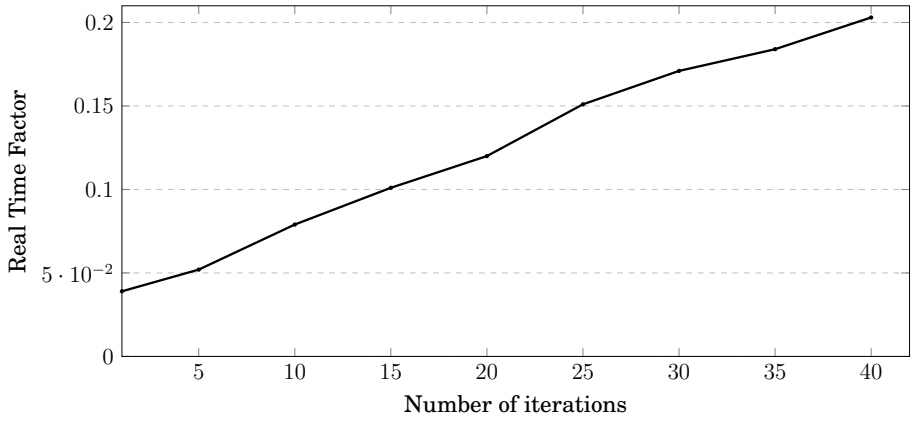

Figure 3.5: How the number of refinement iterations impacts the real time factor (RTF) without beam search using mask predict

Table 3.9: Performance comparison on Tedlium2 with or without beam search.

System	Sub	Del	Ins	Err	RTF
A-CMLM					
Without beam search (40 iters)	**8.9**	4.6	1.7	15.2	**0.2**
With beam search (1 iter)	9.4	**4.0**	**1.5**	**14.8**	0.6

AISHELL results are presented in Table 3.10. For this dataset, no improvements are observed beyond one iteration even with mask predict because of the

short output sequence length. The difference between with beam search and without beam search is about 0.4, and beam search is twice slower.

Table 3.10: Performance comparison on AISHELL with or without beam search.

System	**Dev CER(↓)**	**Test CER(↓)**	RTF
A-CMLM			
Without beam search (1 iter)	7.4	8.4	**0.03**
With beam search (1 iter)	**7.0**	**8.0**	0.08

Similar trends are observed on CSJ shown in Table 3.11. Overall the performance differences between with and without beam search are minimal when a sufficient number of iterations are used. With 35 iterations, the real-time factor reduced from 0.41 to 0.21.

Table 3.11: Performance comparison on CSJ with or without beam search.

System	**Eval1 CER(↓)**	**Eval2 CER(↓)**	**Eval3 CER(↓)**	RTF
A-CMLM				
Without beam search (35 iters)	**9.8**	7.6	8.0	**0.21**
With beam search (1 iter)	**9.8**	**7.5**	**7.9**	0.41

Finally, WSJ results are pretty interesting, included in Table 3.12. Without an external language model, beam search on network predictions improved a little on character error rate, making the word error rate much worse. Beam search failed to detect the boundary of words such that word error rate isn't improved. In contrast, the refinement introduced by mask predict can utilize

contextual information to make the right decisions.

Table 3.12: Performance comparison on WSJ with or without beam search.

System	**dev93 CER/WER(↓)**	**eval92 CER/WER(↓)**	RTF
A-CMLM			
Without beam search (40 iters)	5.9/19.2	4.8/16.6	0.44
With beam search (1 iter)	6.2/21.6	5.0/19.1	0.73

In conclusion, beam search dramatically reduces the number of steps required to get a reasonable word error rate. With beam search, only one iteration is enough as reported by the previous results. Without beam search, mask predict can still get good WER performance by using enough iterations. Those results clearly reveal that mask predict with a sufficient number of iterations is preferred over beam search. Without an external language model, beam search alone doesn't improve the performance much and makes the inference process much slower. They are suitable for different applications. When the response time is sensitive, and the decoder network is not very large, mask predict approach without beam search is a good fit for the performance and efficiency tradeoff. When the performance is the primary concern, beam search with an external language model still provides the best performance and faster inference speed than autoregressive models.

3.4.3 Error Analysis of A-FMLM on CSJ

To analyze the performance difference between our autoregressive and non-autoregressive systems, we plot the relationship between output sequence length and character error rate (CER) in Fig. 3.6. The most considerable performance difference happens when the output sequence is short (less than ten tokens). This is reasonable since context is quite limited for non-autoregressive systems to simultaneously make predictions. However, this can be easily fixed by doing left-to-right autoregressive decoding for such short sequences.

Overall, the performance gap exists on all bins but relatively stable, which suggests that our non-autoregressive systems can handle both long sequences and short sequences.

3.5 Summary

This section explores two novel models: conditional masked language model (CMLM) and factorized masked language model (FMLM). Autoregressive models rely on beam search and external language models to achieve the best performance. The proposed models also benefit from the external language model from experiments, especially when additional unpaired text data is available. The beam search could be introduced to reduce the number of iterations. Only a single iteration is required to achieve the best performance.

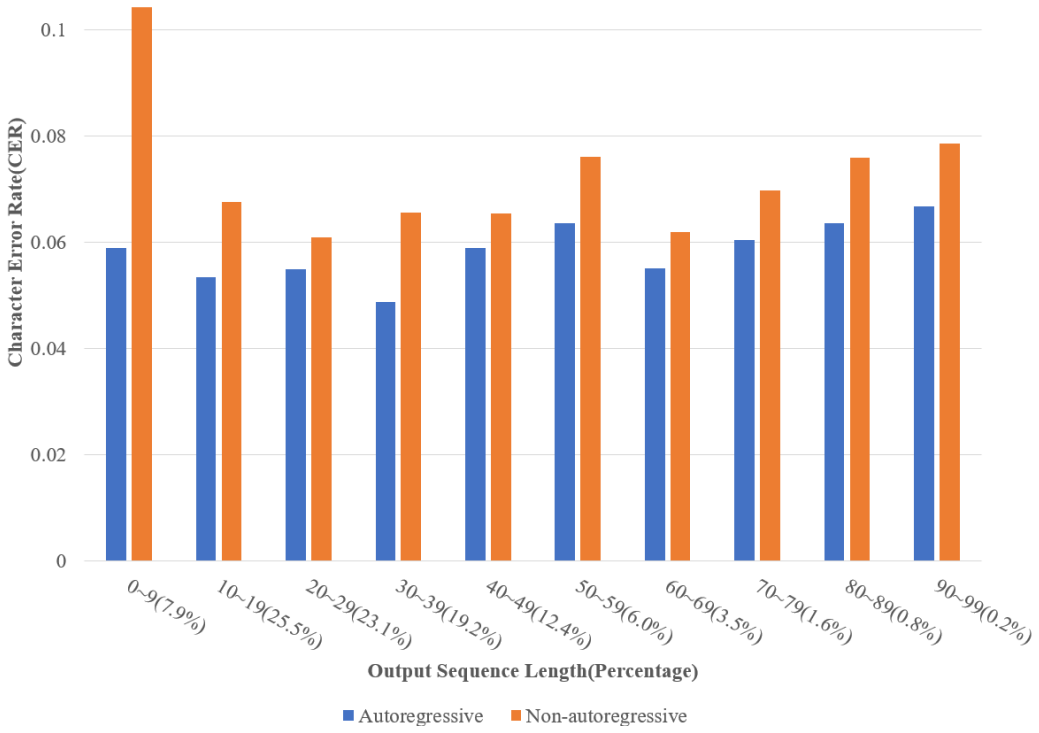

Figure 3.6: Error analysis of autoregressive and non-autoregressive on different output sequence length bins. Percentage indicates the ratio of data in this bin.

Chapter 4

Noise-based non-autoregressive Text-to-Speech

While the mask-based approach works well for speech recognition, some initial experiment results suggest that this approach doesn't work for speech synthesis. This is due to the characteristics of speech synthesis. Speech synthesis is a one-to-mapping problem: multiple possible speech sequences can correspond to the same text sequence due to variations in speech, such as pitch, duration, sound volume, and prosody. It is challenging to model the long-range dependencies between samples and determine some samples simultaneously in the early iteration when the contextual information is not provided. Instead, we proposed another approach that is based on the iterative refinement from the denoising perspective. The proposed methods, WaveGrad and WaveGrad 2,

are built on the idea of score matching [97, 98] and diffusion probabilistic models [99], which enables flexible inference schedules with a different number of iterations for various applications.

This chapter starts with a brief introduction to the general concept of score matching. The diffusion probabilistic model is built on the idea of score matching, and it relies on discrete steps. In this dissertation, a continuous training objective is proposed, which supports a flexible number of steps during generation. Two different models are discussed, WaveGrad and WaveGrad 2, that generate waveform from mel-spectrogram features and phoneme sequence. Various ablation studies reveal the important factors of model training and how the number of generation steps impacts the quality.

4.1 Score matching

We begin with a brief review of the Stein score function, Langevin dynamics, and score matching. The Stein score function [97] is the gradient of the data log-density $\log p(\mathbf{x})$ with respect to the datapoint $\mathbf{x}$:

$$s(\mathbf{x}) = \nabla_x \log p(\mathbf{x}). \tag{4.1}$$

Given the Stein score function $s(\cdot)$, one can draw samples from the corresponding density, $\tilde{\mathbf{x}} \sim p(\mathbf{x})$, via Langevin dynamics, which can be interpreted

as stochastic gradient ascent in the data space:

$$\tilde{\mathbf{x}}_{i+1} = \tilde{\mathbf{x}}_i + \frac{\eta}{2} s(\tilde{\mathbf{x}}_i) + \sqrt{\eta}\, \mathbf{z}_i, \tag{4.2}$$

where $\eta > 0$ is the step size, $\mathbf{z}_i \sim \mathcal{N}(0, I)$, and I denotes an identity matrix.

A generative model can be built by training a neural network to learn the Stein score function directly, using Langevin dynamics for inference. This approach, known as score matching [97, 98], has seen success in image [100, 101] and shape generation [102]. The denoising score matching objective [98] takes the form:

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}\, \mathbb{E}_{\tilde{\mathbf{x}} \sim q(\tilde{\mathbf{x}} | \mathbf{x})} \left[\left\| s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q(\tilde{\mathbf{x}} \mid \mathbf{x}) \right\|_2^2 \right], \tag{4.3}$$

where $p(\cdot)$ is the data distribution, and $q(\cdot)$ is a noise distribution.

Recently, a weighted denoising score matching objective is proposed in [100], in which data is perturbed with different levels of Gaussian noise, and the score function $s_\theta(\tilde{\mathbf{x}}, \sigma)$ is conditioned on σ, the standard deviation of the noise used:

$$\sum_{\sigma \in S} \lambda(\sigma)\, \mathbb{E}_{\mathbf{y} \sim p(\mathbf{x})}\, \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma)} \left[\left\| s_\theta(\tilde{\mathbf{x}}, \sigma) - \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} \right\|_2^2 \right], \tag{4.4}$$

where S is a set of standard deviation values that one perturbs the data with, and $\lambda(\sigma)$ is a weighting function for different σ. WaveGrad and WaveGrad 2 are a variant of this approach applied to learning *conditional* generative models of the form $p(\mathrm{x} \mid \mathrm{y})$. The proposed approaches learn the gradient of the data density, and use a sampler similar to Langevin dynamics for inference.

The denoising score matching framework relies on a noise distribution to provide support for learning the gradient of the data log density (i.e., q in Equation 4.3, and $\mathcal{N}(\cdot, \sigma)$ in Equation 4.4). The choice of the noise distribution is critical for achieving high-quality samples [101]. WaveGrad and WaveGrad 2 rely on the diffusion model framework [47, 99] to generate the noise distribution used to learn the score function.

4.1.1 Diffusion Probabilistic Model

In Denoising Diffusion Probabilistic Models [47], the authors made the observation that diffusion probabilistic models [99] and score matching objectives [98, 100, 101] are closely related. As such, we will first introduce WaveGrad as a diffusion probabilistic model [99].

We adapt the diffusion model setup from [47], from unconditional image generation to conditional raw audio waveform generation. WaveGrad and WaveGrad 2 model the conditional distribution $p_\theta(\mathbf{x}_0 \mid \mathbf{y})$ where $\mathbf{x}_0$ is the waveform and $\mathbf{y}$ is the conditioning features corresponding to $\mathbf{x}_0$, such as phoneme sequence, linguistic features derived from the corresponding text, ground-truth Mel-spectrogram extracted from $\mathbf{x}_0$, or acoustic features predicted by a Tacotron-style text-to-speech synthesis model [33]):

$$p_\theta(\mathbf{x}_0 \mid \mathbf{y}) \coloneqq \int p_\theta(\mathbf{x}_{0:N} \mid \mathbf{y}) \, \mathrm{d}\mathbf{x}_{1:N}, \tag{4.5}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_N$ is a series of latent variables, each of which are of the same dimension as the data $\mathbf{x}_0$, and N is the number of latent variables (iterations). The generative distribution $p_\theta(\mathbf{x}_{0:N} \mid \mathbf{y})$ is called the denoising process (or reverse process), and is defined through the Markov chain:

$$p_\theta(\mathbf{x}_{0:N} \mid \mathbf{y}) \coloneqq p(\mathbf{x}_N) \prod_{n=1}^{N} p_\theta(\mathbf{x}_{n-1} \mid \mathbf{x}_n, \mathbf{y}), \tag{4.6}$$

where each iteration is modelled as a Gaussian transition:

$$p_\theta(\mathbf{x}_{n-1} \mid \mathbf{x}_n, \mathbf{y}) \coloneqq \mathcal{N}\left(\mathbf{x}_{n-1}; \mu_\theta(\mathbf{x}_n, n, \mathbf{y}), \Sigma_\theta(\mathbf{x}_n, n, \mathbf{y})\right), \tag{4.7}$$

starting from Gaussian white noise $p(\mathbf{x}_N) = \mathcal{N}\left(\mathbf{x}_N; 0, I\right)$. The approximate posterior $q(\mathbf{x}_{1:N} \mid \mathbf{x}_0)$ is called the diffusion process (or forward process), and is defined through the Markov chain:

$$q(\mathbf{x}_{1:N} \mid \mathbf{x}_0) \coloneqq \prod_{n=1}^{N} q(\mathbf{x}_n \mid \mathbf{x}_{n-1}), \tag{4.8}$$

where each iteration adds Gaussian noise:

$$q(\mathbf{x}_n \mid \mathbf{x}_{n-1}) \coloneqq \mathcal{N}\left(\mathbf{x}_n; \sqrt{(1-\beta_n)}\,\mathbf{x}_{n-1}, \beta_n I\right), \tag{4.9}$$

under some (fixed constant) noise schedule $\beta_1, \ldots, \beta_N$. We emphasize the property observed by [47], the diffusion process can be computed for any step n in a closed form:

$$q(\mathbf{x}_n \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_n; \sqrt{\bar{\alpha}_n}\,\mathbf{x}_0, (1-\bar{\alpha}_n) I\right), \tag{4.10}$$

where $\alpha_n \coloneqq 1 - \beta_n$ and $\bar{\alpha}_n \coloneqq \prod_{s=1}^{n} \alpha_s$. During training, we can optimize for the variational lower-bound on the log-likelihood (upper-bound on the negative log-likelihood):

$$-\log p_\theta(\mathbf{x}_0 \mid \mathbf{y}) \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_0 \mid \mathbf{y})}{q(\mathbf{x}_{1:N} \mid \mathbf{x}_0)} \right] \tag{4.11}$$

$$= \mathbb{E}_q \left[-\log p(\mathbf{x}_N) - \sum_{n=1}^{N} \log \frac{p_\theta(\mathbf{x}_{n-1} \mid \mathbf{x}_n, \mathbf{y})}{q(\mathbf{x}_n \mid \mathbf{x}_{n-1})} \right]. \tag{4.12}$$

Following Equation 4.12, it would be natural to parameterize a neural network to model the mean μ_θ and variance Σ_θ of the Gaussian distribution in Equation 4.7, which would make it possible to directly optimize the KL-divergence with Monte Carlo estimates. However, [47] found it beneficial to simply set Σ_θ to a constant following the β_n schedule, and to reparameterize the neural network to model ϵ_θ, predicting the noise $\epsilon \sim \mathcal{N}(0, I)$ instead of μ_θ. Under this reparameterization, the loss can be written as:

$$\mathbb{E}_{n,\epsilon} \left[C \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_n}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_n}\, \epsilon, \mathbf{y}, n \right) \right\|_2^2 \right], \tag{4.13}$$

where

$$C = \frac{\beta_n^2}{2\sigma_n^2 \alpha_n (1 - \bar{\alpha}_n)}. \tag{4.14}$$

In practice, [47] also found it beneficial to drop the C term, and thus leading to a weighted variational lower bound of the log-likelihood. As noted by [47], ϵ_θ can be interpreted as the score function or the gradient of the data density, and thus the objective resembles score matching as in Equation 4.4 [100]. Additionally

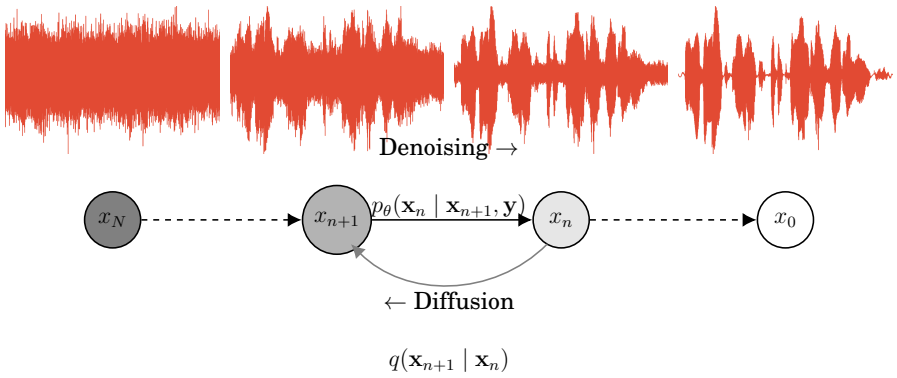

Figure 4.1: A directed graphical model of the diffusion probabilistic model, adapted from [47]. The diffusion process $q(\mathbf{x}_{n+1} \mid \mathbf{x}_n)$ iteratively adds Gaussian noise to the signal starting from the waveform $\mathbf{x}_0$; while the denoising process $p_\theta(\mathbf{x}_n \mid \mathbf{x}_{n+1}, \mathbf{y})$ iteratively removes noise from the signal starting from Gaussian noise $\mathbf{x}_N$.

under the parameterization of [47], ϵ_θ conditions on the discrete index n, we will discuss this further in the text. We also found that substituting the original L_2 distance metric with L_1 offers better training stability. Figure 4.1 visualizes the diffusion and denoising processes.

4.1.2 Noise Schedule and Conditioning on Noise Level

In the score matching setup, [100, 101] noted the importance of the noise distribution during training. The noise distribution is crucial since it pro-

vides support for model gradient distribution. The diffusion framework can be viewed as a specific way to provide support to score matching. In the diffusion framework, the noise schedule is parameterized by $\beta_1, \ldots, \beta_N$, as described in the previous section. This is typically determined via some hyperparameter heuristic, e.g., a linear decay schedule [47]. We found the choice of the noise schedule to be critical towards achieving high fidelity audio in our experiments, especially when trying to minimize the number of inference iterations N to make inference efficient. A schedule with superfluous noise may result in a model unable to recover the high-frequency detail of the waveform, while a schedule with too little noise may result in a model that converges poorly during inference. [101] provides some insights about how to tune the noise schedule under the framework of score matching. We will connect some of these insights and apply them to WaveGrad under the diffusion framework.

Another closely related problem is determining the hyperparameter N, the number of diffusion/denoising steps. A large N would equip the model with more computational capacity and may improve sample quality. However, using a small N would result in faster inference and lower computational costs. [100] adopted $N = 10$ to generate 32×32 images, while [47] used 1,000 iterations to generate high resolution 256×256 images. In our case, WaveGrad generates audio sampled at 24 kHz, containing 24,000 samples per second.

We found that tuning both the noise schedule and N in conjunction with

each other was critical to attaining high fidelity audio, especially when N is small. If these hyperparameters are poorly tuned, the training sampling procedure may provide deficient support for the distribution. Consequently, our sampler may converge poorly during inference when the sampling trajectory encounters regions that deviate from the conditions seen during training. However, tuning these hyperparameters can be costly due to the large search space, as a large number of models need to be trained and evaluated. We make empirical observations and discuss this in more details in Section 5.4.

We address some of the issues above in our WaveGrad implementation. First, compared to the diffusion probabilistic model from [47], we reparameterize the model from conditioning on the discrete iteration index n to conditioning on the continuous noise level $\bar{\alpha}$. This reparameterization allows us (1) not to rely on discrete indices; (2) to enable the model to directly condition on the amount of noise. The loss becomes

$$\mathbb{E}_{\bar{\alpha},\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_n}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_n}\,\epsilon, \mathbf{y}, \sqrt{\bar{\alpha}}\right)\right\|_1\right], \tag{4.15}$$

A similar approach was also used in the score matching framework [100, 101], wherein they conditioned on the noise variance.

There is one minor technical issue we must resolve in this approach. In the diffusion probabilistic model training procedure conditioned on the discrete iteration index (Equation 4.13), we would sample $n \sim \mathrm{Uniform}(\{1, \ldots, N\})$, and then proceed to compute its corresponding α_n. In the case of the directly con-

ditioning on the continuous noise, we need to determine a sampling procedure that directly samples $\bar{\alpha}$. Recall that $\bar{\alpha}_n \coloneqq \prod_s^n (1-\beta_s) \in [0,1]$, while we could simply just sample from the uniform distribution $\bar{\alpha} \sim \text{Uniform}(0,1)$, however, we found this to provide poor empirical results. We instead use a simple hierarchical sampling method that mimics the discrete sampling strategy. We first define a noise schedule with S iterations and compute all its $\sqrt{\bar{\alpha}_s}$:

$$l_0 = 1, \qquad l_s = \sqrt{\prod_{i=1}^{s}(1-\beta_s)}. \tag{4.16}$$

We first sample a segment $s \sim U(\{1,\dots,S\})$, which provides a segment (l_{s-1}, l_s), and then sample from this segment uniformly to compute $\sqrt{\bar{\alpha}}$. The complete WaveGrad training algorithm with our sampling procedure is illustrated in Algorithm 2.

One benefit of the proposed WaveGrad model is that it needs to be trained only once, yet inference can be run over a large space of trajectories without retraining. To be more specific, once we train a model, we can use an arbitrary different number of N iterations during inference, making it possible to explicitly trade-off between inference computation and output quality using the same model. This also makes fast hyperparameter search possible, as we will illustrate in Section 5.4. The complete WaveGrad inference algorithm is explained in Algorithm 3 which is a gradient-based sampler akin to Langevin dynamics.

input: data distribution $q(\mathbf{x}_0)$, conditioning signal $\mathbf{y}$, predefined schedule l with S steps, network ϵ_θ

repeat

$\mathbf{x}_0 \sim q(\mathbf{x}_0)$;

$s \sim \text{Uniform}(\{1, \dots, S\})$;

$\sqrt{\bar{\alpha}} \sim \text{Uniform}(l_{s-1}, l_s)$;

$\epsilon \sim \mathcal{N}(0, I)$;

Take gradient descent step on;

$$\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}}\,\epsilon, \mathbf{y}, \sqrt{\bar{\alpha}}) \right\|_1$$

until *converged*;

Algorithm 2: Training algorithm for WaveGrad. WaveGrad directly conditions on the continuous noise $\sqrt{\bar{\alpha}}$, and l is from a predefined noise schedule.

input : conditioning features y

output: data sample $\mathbf{x}_0$, inference schedule α

$\mathbf{x}_N \sim \mathcal{N}(0, I)$;

for $n = N, \ldots, 1$ **do**

$z \sim \mathcal{N}(0, I)$ if $n > 1$, else $z = 0$;

$\mathbf{x}_{n-1} = \frac{1}{\sqrt{\alpha_n}} \left(\mathbf{x}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}} \epsilon_\theta(\mathbf{x}_n, \mathbf{y}, \sqrt{\bar{\alpha}_n}) \right) + \sigma_n z$;

end

return $\mathbf{x}_0$;

Algorithm 3: Sampling algorithm for WaveGrad. WaveGrad follows a gradient-based sampler similar to Langevin dynamics to generate samples [47]. σ_n is the noise term in Langevin dynamics introduced in iteration n.

4.2 Network Architecture

4.2.1 WaveGrad Vocoder

A vocoder predicts waveform samples conditioning on the input linear or Mel-spectrogram features. Because of the high resolution of the waveform (e.g., 16kHz or 24kHz), the vocoder is usually the bottleneck part of the text-to-speech system. We start our non-autoregressive model from the vocoder.

To convert the Mel-spectrogram signal (80 Hz) into raw audio (24 kHz), five upsampling blocks (UBlock) are applied to gradually upsample the temporal dimension by factors of 5, 5, 3, 2, 2, with the number of channels of 512, 512, 256, 128, 128 respectively. Additionally, one convolutional layer is added before and after these blocks.

The UBlock is illustrated in Figure 4.3. Each UBlock includes two residual blocks [103]. Neural audio generation models often use large receptive field [4, 53, 85]. The dilation factors of four convolutional layers are 1, 2, 4, 8 for the first three UBlocks and 1, 2, 1, 2 for the rest. Upsampling is carried out by repeating the nearest input. For the large model, we use 1, 2, 4, 8 for all UBlocks.

As an iterative approach, the network prediction is also conditioned on noisy waveform $\sqrt{\bar{\alpha}_n}\, x_0 + \sqrt{1-\bar{\alpha}_n}\epsilon$. Downsampling blocks (DBlock), illustrated in Figure 4.4, are introduced to downsample the temporal dimension of the noisy

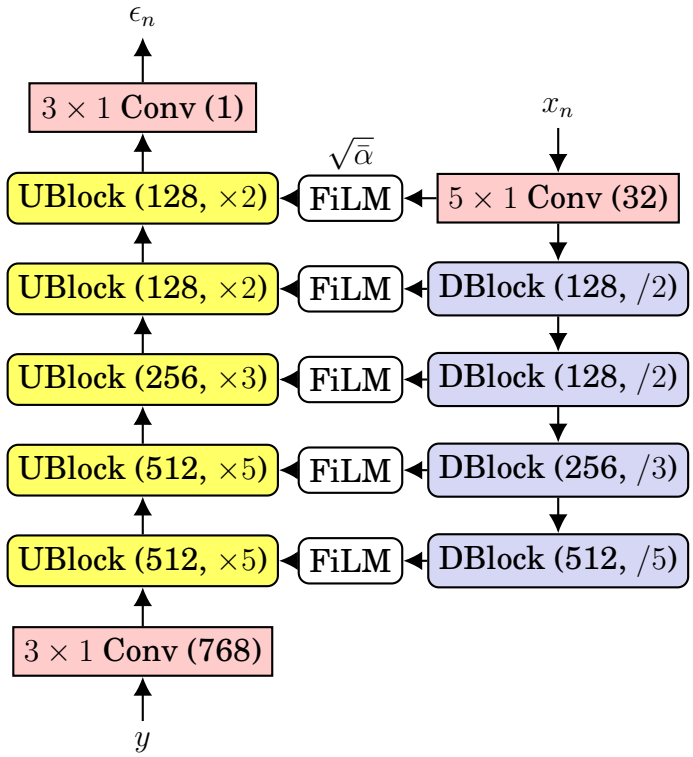

Figure 4.2: WaveGrad network architecture. The inputs consists of the mel-spectrogram conditioning signal y, the noisy waveform generated from the previous iteration x_n, and the noise level $\sqrt{\bar{\alpha}}$. The model produces ϵ_n at each iteration, which can be interpreted as the direction to update x_n.

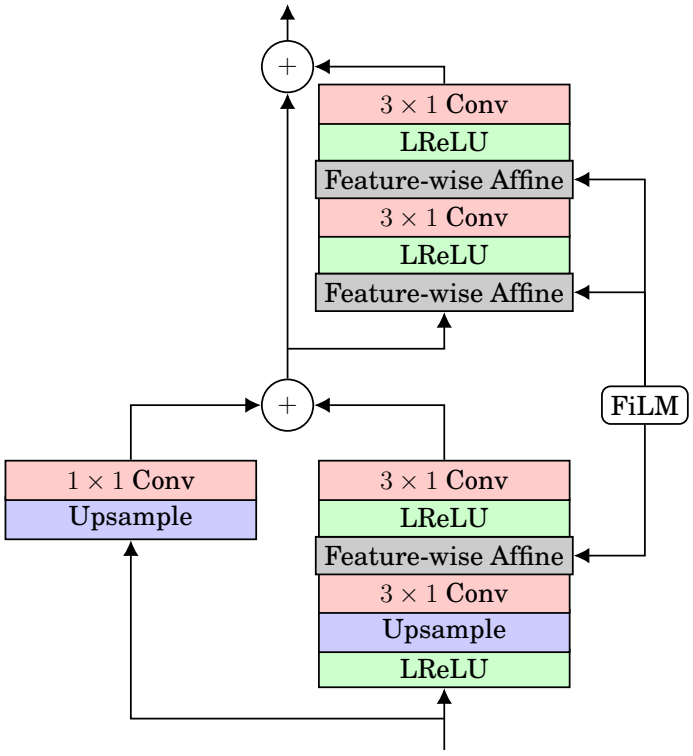

Figure 4.3: A block diagram of the Upsampling Block (UBlock). We upsample the signal modulated with information from the FiLM module.

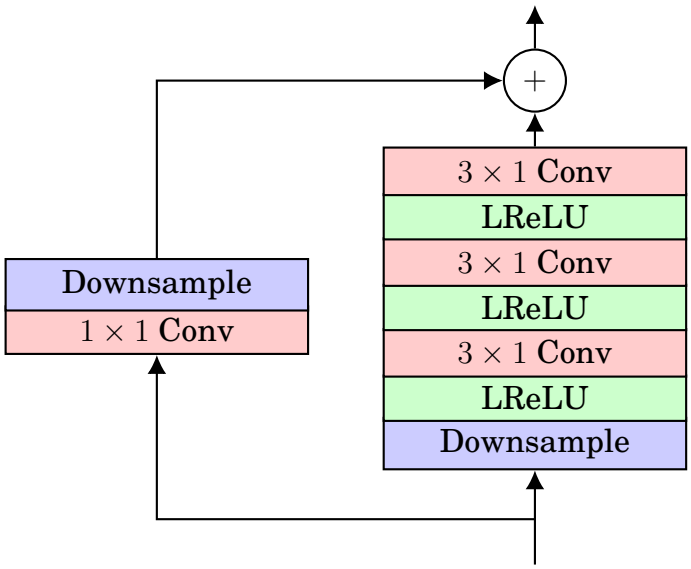

Figure 4.4: A block diagram of the downsampling block (DBlock).

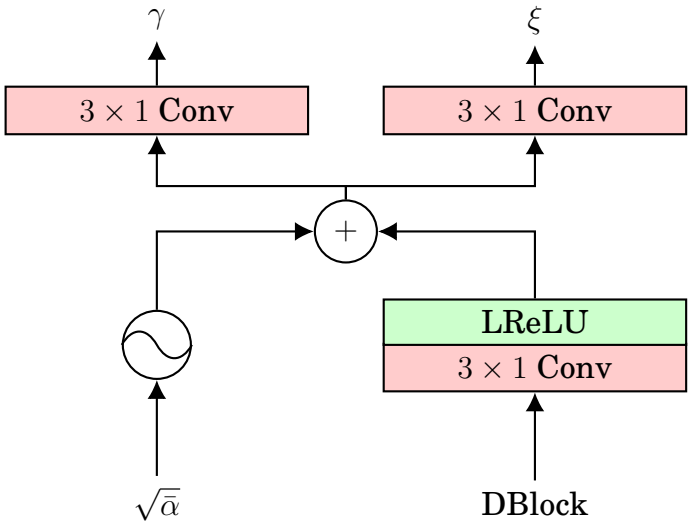

Figure 4.5: A block diagram of feature-wise linear modulation (FiLM) module. We condition on the noise level $\sqrt{\bar{\alpha}}$ of diffusion/denoising process, and pass it to a positional encoding function.

waveform. The DBlock is similar to UBlock, except that only one residual block is included. The dilation factors are 1, 2, 4 in the main branch. Downsampling is carried out by convolution with strides. Orthogonal initialization [104] is used for all UBlocks and DBlocks.

The feature-wise linear modulation (FiLM) [105] module combines information from both noisy waveform and input Mel-spectrogram. We also represent the iteration index n, which indicates the input waveform's noise level, using Transformer-style sinusoidal positional embeddings [39]. To condition on the noise level directly, n is replaced by $\sqrt{\bar{\alpha}}$ and a linear scale $C = 5000$ is applied. The FiLM module produces both scale and bias vectors given inputs, which are used in a UBlock for feature-wise affine transformation as

$$\gamma(D, \sqrt{\bar{\alpha}}) \odot U + \xi(D, \sqrt{\bar{\alpha}}), \tag{4.17}$$

where γ and ξ correspond to the scaling and shift vectors from the FiLM module, D is the output from corresponding DBlock, U is an intermediate output in the UBlock, and $\odot$ denotes the Hadamard product.

An overview of the FiLM module is illustrated in Figure 4.5. The structure is inspired by spatially-adaptive denormalization [106]. However, batch normalization [107] is not applied in our work since each minibatch contains samples with different levels of noise. Batch statistics are not accurate since they are heavily dependent on sampled noise levels. Experiment results also verified our assumption that models trained with batch normalization generate

low-quality audio.

4.2.2 WaveGrad 2: Phoneme-to-Wave model

We further proposed WaveGrad 2, a more end-to-end model which directly estimates waveform from phoneme sequence. WaveGrad 2 includes three modules illustrated in Figure 4.6:

- The encoder takes a phoneme sequence as input and extracts abstract hidden representations from the input context.
- The resampling layer changes the resolution of the encoder output to match the output waveform time scale, quantized into 10ms segments (similar to typical Mel-spectrogram features). This is achieved by conditioning on the target duration during training. Durations predicted by the duration predictor module are utilized during inference.
- The WaveGrad decoder as introduced in Section 4.2.1 predicts the raw waveform by refining the noisy waveform iteratively. In each iteration, the decoder gradually refines the signal and adds fine-grained details.

4.2.2.1 Encoder

The design of the encoder follows that of Tacotron 2 [33]. Phoneme tokens are used as inputs, with silence tokens inserted at word boundaries. An end-

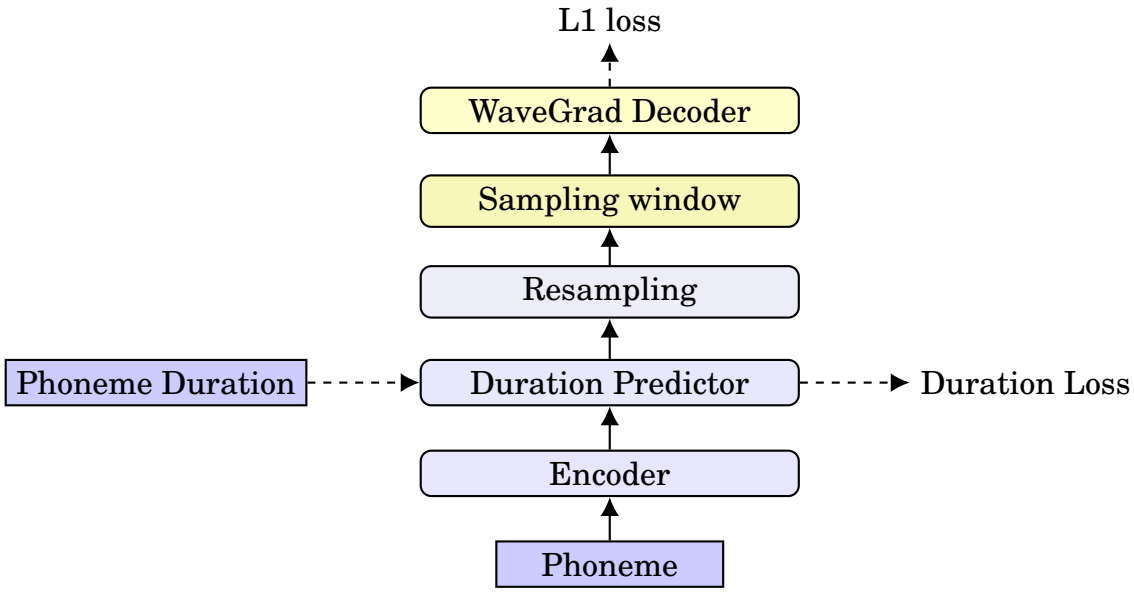

Figure 4.6: WaveGrad 2 network architecture. The inputs consists of the phoneme sequence. Dashed lines indicates computation only performed during training.

of-sequence token is added after each sentence. Tokens are first converted into learned embeddings, which are then passed through 3 convolution layers with dropout [108] and batch normalization layer [109]. Finally, long-term contextual information is modeled by passing the output through a single bi-directional long short-term memory (LSTM) layer with ZoneOut regularization [110].

4.2.2.2 Resampling

The length of the output waveform sequence is very different from the length of encoder representations. In Tacotron 2 [33], this is resolved by the attention mechanism. To make the structure non-autoregressive and speed-up inference, we adopt the Gaussian upsampling introduced in the non-attentive Tacotron [35]. Instead of repeating each token according to its duration, Gaussian upsampling predicts the duration and influence range simultaneously. These parameters are used in the attention weights computation, which purely relies on the predicted position. The ground truth duration is used instead during training, and an additional mean square loss is measured to train the duration predictor. This is labeled as **Duration Loss** in Figure 4.6. Ground truth duration is not needed during inference and predicted duration is adopted instead.

4.2.2.3 Sampling Window

Since the waveform resolution is very high (24,000 samples per second in our case), it is not feasible to compute the loss on all waveform samples in an utterance because of the high computation cost and memory constraints. Instead, after learning the representations on the whole input sequence, we sample a small segment to synthesize the waveform. Due to the introduction of the resampling layer, the encoder representations and waveform samples are already aligned. Random segments are sampled individually in each minibatch and the corresponding waveform segment is extracted based on the upsampling rate (300 in our setup). The entire encoder sequence (after resampling) is used during inference, introducing a small mismatch between training and inference.

4.2.2.4 Hidden Features Augmentation

We explored applying a variant of SpecAugment [111] to the conditioning input to the decoder (the resampled encoder output). The augmentation is applied to the learned hidden representations instead of the spectrograms. This can be viewed as a form of correlated block dropout. Thirty-two consecutive frames were randomly selected to be masked and we applied it twice. The intuition is that the WaveGrad decoder can recover the masked part by conditioning the contextual information. This enforces the encoder to learn robust

representations which include more context information.

4.3 Noise Schedule

We tested different noise schedules during the training of WaveGrad and WaveGrad 2 models. For 1000 and 50 iterations, we set the forward process variances to constants increasing linearly from β_1 to β_N, defined as Linear(β_1, β_N, N). We used Linear(1×10^{-4}, 0.005, 1000) for 1000 iterations and Linear(1×10^{-4}, 0.05, 50) for 50 iterations. For 25 iteration, a different Fibonacci-based schedule was adopted (referred to as Fibonacci(N)):

$$\begin{gathered} \beta_0 = 1 \times 10^{-6} \quad \beta_1 = 2 \times 10^{-6} \\ \beta_n = \beta_{n-1} + \beta_{n-2} \quad \forall n \geq 2. \end{gathered} \tag{4.18}$$

When a fixed schedule was used during training, the same schedule was used during inference. We found that a mismatch in the noise schedule degraded performance. Different noise schedules and corresponding $\sqrt{\bar{\alpha}}$ are plotted in Figure 7.

To sample the noise level $\sqrt{\bar{\alpha}}$, we set the maximal iteration S to 1000 and precompute l_1 to l_S from Linear(1×10^{-6}, 0.01, 1000). Then each $\sqrt{\bar{\alpha}}$ is sampled by sampling s first and then sample from uniform distribution between l_s and l_{s+1}. Unlike the base fixed schedule, our models support using a different schedule during inference; thus "Manual" schedule was also explored to demonstrate

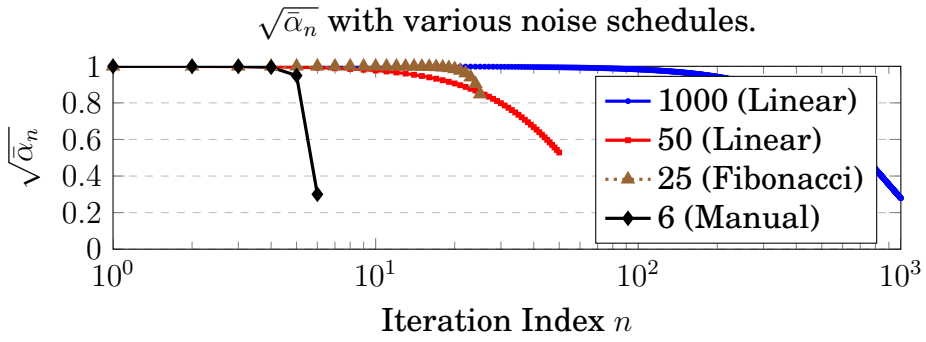

Figure 4.7: Plot of different noise schedules for WaveGrad and WaveGrad 2.

the possibilities with WaveGrad vocoder. For example, the 6-iteration inference schedule was explored by sweeping the βs over following nine possibilities:

$$\begin{gathered} 1 \times 10^{-6}, 1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1} \\ 2 \times 10^{-6}, 2 \times 10^{-5}, 2 \times 10^{-4}, 2 \times 10^{-3}, 2 \times 10^{-2}, 2 \times 10^{-1} \\ \ldots \\ 9 \times 10^{-6}, 9 \times 10^{-5}, 9 \times 10^{-4}, 9 \times 10^{-3}, 9 \times 10^{-2}, 9 \times 10^{-1} \end{gathered} \tag{4.19}$$

where each line contains one possible schedule.

Again, we did not need to train individual models for such hyper-parameter tuning. Here we used Log-mel spectrogram mean squared error metrics (LS-MSE) as a metric for tuning.

4.4 Evaluation

The following vocoders were used as baselines in the experiment of vocoders: (1) WaveRNN [52] conditioned on Mel-spectrograms predicted by a Tacotron 2

model in teacher-forcing mode following [33]; The model used a single long short-term memory (LSTM) layer with 1,024 hidden units, five convolutional layers with 512 channels as the conditioning stack to process the Mel-spectrogram features, and a 10-component mixture of logistic distributions [112] as its output layer, generating 16-bit samples at 24 kHz. It had 18M parameters and was trained for 1M steps. Preliminary experiments indicated that further reducing the number of units in the LSTM layer hurts performance. (2) Parallel WaveGAN [85] with 1.57M parameters, trained for 1M steps. (3) MelGAN [83] with 3.22M parameters, trained for 4M steps. (4) Multi-band MelGAN [84] with 2.27M parameters, trained for 1M steps. (5) GAN-TTS [53] with 21.4M parameters, trained for 1M steps.

All models were trained using a proprietary speech dataset consisted of 385 hours of high quality English speech from 84 professional voice talents. Following the original papers, Parallel WaveGAN, MelGAN, and Multi-band MelGAN were conditioned on the Mel-spectrograms computed from ground truth audio during training. They were trained using a publicly available implementation at `https://github.com/kan-bayashi/ParallelWaveGAN`. Note that the hyper-parameters of these baseline models were not fully optimized for this dataset. Conditioning Mel-spectrograms for the test set were predicted using a Tacotron 2 model, which were passed to these models to synthesize audio signals. Note that the Tacotron 2 model was identical to the one used to predict

Mel-spectrograms for training WaveRNN and GAN-TTS models.

For the WaveGrad vocoder, two network size variations were explored: Base and Large. The WaveGrad Base model took 24 frames corresponding to 0.3 second audio as input during training. Our network is fully convolutional, non-autoregressive and highly parallelizable. We set the batch size to 256 and models were trained on Google Tensor Processing Unit (TPU) v2 Pods for 12 hours. The base model contained 15 million parameters. For the WaveGrad Large model, we repeated each UBlock/DBlock twice, one with upsampling/downsample and another without. Each training sample included 60 frames corresponding to a 0.75 second audio segment. We kept the same batch size and trained the model on TPU v3 Pods. The WaveGrad Large model contained 23 million parameters.

We include Wave-Tacotron [43] as the phoneme-to-wave baseline to compare with the proposed WaveGrad 2 model. For Wave-Tacotron and the proposed WaveGrad 2 models, we used a subset of the training set that included all the test speaker's audio. This subset included 39 hours of speech. Preliminary results suggested that WaveGrad 2 trained on a single-speaker dataset gave better performance, especially when the network size was small.

We report subjective listening test results rating speech naturalness on a 5-point Mean Opinion Score (MOS) scale to compare these models. Subjects were asked to rate the naturalness of each stimulus after listening to it. Fol-

lowing previous studies, a five-point Likert scale score (1: Bad, 2: Poor, 3:Fair, 4: Good, 5: Excellent) was adopted with rating increments of 0.5. Each subject was allowed evaluate up to six stimuli. MOS is reported by the average and standard deviation.

4.5 Results

Subjective evaluation results are summarized in Table 4.1. Models conditioned on discrete indices followed the formulation from Equation 4.13, and models conditioned on continuous noise level followed the formulation from Equation 4.15. For vocoders, WaveGrad models matched the performance of the autoregressive WaveRNN baseline and outperformed the non-autoregressive baselines. Although increasing the model size slightly improved naturalness, the difference was not statistically significant. Using six iterations, the WaveGrad Base model achieved a real-time factor (RTF) of 0.2 on an NVIDIA V100 GPU while still achieving a MOS above 4.4. As a comparison, the WaveRNN model achieved an RTF of 20.1 on the same GPU, 100 times slower.

For the phoneme-to-wave models, WaveGrad 2 models almost matched the performance of the autoregressive Tacotron 2 + WaveRNN baseline and outperformed other baselines with non-autoregressive vocoders. When the number of iterations reduces from 1000 to 50, the mean opinion score is still pretty good,

which almost matches the performance of the combination of Tacotron 2 and GAN-TTS, outperforms the Wave-Tacotron model. The phoneme sequence is much weaker than the Mel spectrogram features as the conditioning signal, which explains why we cannot further reduce the number of steps for inference.

Table 4.1: Mean opinion scores (MOS) of various models and their confidence intervals. All models except WaveRNN are non-autoregressive. WaveGrad, Parallel WaveGAN, MelGAN, and Multi-band MelGAN were conditioned on the Mel-spectrograms computed from ground truth audio during training. WaveRNN and GAN-TTS used predicted features for training. **MT**: Multi-task training.

Model	**MOS** (↑)
Tacotron 2 + WaveRNN	4.49 ± 0.04
Tacotron 2 + Parallel WaveGAN	3.92 ± 0.05
Tacotron 2 + MelGAN	3.95 ± 0.06
Tacotron 2 + Multi-band MelGAN	4.10 ± 0.05
Tacotron 2 + GAN-TTS	4.34 ± 0.04
Tacotron 2 + WaveGrad	
Base (6 iterations, continuous noise levels)	4.41 ± 0.03
Base (1,000 iterations, discrete indices)	4.47 ± 0.04
Large (1,000 iterations, discrete indices)	4.51 ± 0.04
Wave-Tacotron	4.08 ± 0.06
WaveGrad 2	
Large (1000 iterations) + MT + SpecAug	4.43 ± 0.05
Large (50 iterations) + MT	4.32 ± 0.05
Ground Truth	4.58 ± 0.05

4.6 Ablation Studies

Different experiments are conducted to analyze the impact of different hyperparameters. We first conduct experiments on the inference schedule with WaveGrad vocoder and demonstrate how WaveGrad provides the trade-off between fidelity and speed naturally. Then various hyperparameters are explored for the phoneme-to-wave WaveGrad 2 model.

4.6.1 WaveGrad: Speed-Quality Tradeoff

To understand the impact of different noise schedules and to reduce the number of iterations in the noise schedule from 1,000, we explored different noise schedules using fewer iterations. We found that a well-behaved inference schedule should satisfy two conditions:

1. The KL-divergence $D_{\mathrm{KL}}\left(q(y_N \mid y_0) \parallel \mathcal{N}(0, I)\right)$ between y_N and standard normal distribution $\mathcal{N}(0, I)$ needs to be small. Large KL-divergence introduces mismatches between training and inference. To make the KL-divergence small, some βs need to be large enough.
2. β should start with small values. This provides the model training with fine granularity details, which we found crucial for reducing background static noise.

In this section, all the experiments were conducted with the WaveGrad Base

model. Both objective and subjective evaluation results are reported. The objective evaluation metrics include

1. Log-mel spectrogram mean squared error metrics (LS-MSE), computed using 50 ms window length and 6.25 ms frame shift;
2. Mel cepstral distance (MCD) [113], a similar MSE metric computed using 13-dimensional mel frequency cepstral coefficient features;
3. F_0 Frame Error (FFE) [114], combining Gross Pitch Error and Voicing Decision metrics to measure the signal proportion whose estimated pitch differs from ground truth.

Since the ground truth waveform is required to compute objective evaluation metrics, we report results using ground truth Mel-spectrograms as conditioning features. We used a validation set of 50 utterances for objective evaluation, including audio samples from multiple speakers.

The test set included 1,000 sentences. Subjects were asked to rate the naturalness of each stimulus after listening to it. Following previous studies, a five-point Likert scale score (1: Bad, 2: Poor, 3: Fair, 4: Good, 5: Excellent) was adopted with rating increments of 0.5. Each subject was allowed to evaluate up to six stimuli. Test stimuli were randomly chosen and presented for each subject. Each stimulus was presented to a subject in isolation and was evaluated by one subject. The subjects were paid and native speakers of English living in the United States. They were requested to use headphones in a quiet

room.

We experimented with different noise schedules and number of iterations. These models were trained with conditioning on the discrete index. Subjective and quantitative evaluation results are in Table 4.2.

We also performed a detailed study on the WaveGrad model conditioned on the continuous noise level in the bottom part of Table 4.2. Compared to the model conditioned on the discrete index with a fixed training schedule (top of Table 4.2), conditioning on the continuous noise level generalized better, especially if the number of iterations was small. It can be seen from Table 4.2 that degradation with the model with six iterations was not significant. The model with six iterations achieved real time factor (RTF) = 0.2 on an NVIDIA V100 GPU and RTF = 1.5 on an Intel Xeon CPU (16 cores, 2.3GHz). As we did not optimize the inference code, further speed-ups are likely possible.

Table 4.2: Objective and subjective metrics of the WaveGrad Base models. When conditioning on the discrete index, a separate model needs to be trained for each noise schedule. In contrast, a single model can be used with different noise schedules when directly conditioning the noise level. This variant yields high fidelity samples using as few as six iterations.

Iterations (schedule)	**LS-MSE (↓)**	**MCD (↓)**	**FFE (↓)**	**MOS (↑)**
WaveGrad conditioned on discrete index				
25 (Fibonacci)	283	3.93	3.3%	3.86 ± 0.05
50 (Linear $(1 \times 10^{-4}, 0.05)$)	181	3.13	3.1%	4.42 ± 0.04
1000 (Linear $(1 \times 10^{-4}, 0.005)$)	116	2.85	3.2%	4.47 ± 0.04
WaveGrad conditioned on continuous noise level				

6 (Manual)	217	3.38	2.8%	4.41 ± 0.04
25 (Fibonacci)	185	3.33	2.8%	4.44 ± 0.04
50 (Linear ($1 \times 10^{-4}, 0.05$))	177	3.23	2.7%	4.43 ± 0.04
1000 (Linear ($1 \times 10^{-4}, 0.005$))	106	2.85	3.0%	4.46 ± 0.03

4.6.2 WaveGrad 2: Sampling Window Size

Memory usage is a major concern for end-to-end training. Long sequences corresponding to multi-second utterances may not fit into memory since the main computation bottleneck comes from the WaveGrad decoder, which operates at the waveform sample rate. To make training efficient, we sample a small segment from the resampled encoder representation and train the decoder network using this segment instead of the whole sequence.

Two different window sizes were explored: 64 and 256 frames, corresponding to 0.8 and 3.2 seconds of speech, respectively. The results are shown in Table 4.3. The use of the large window gave better MOS compared to the small window. In all following experiments, we use the large window for training.

Table 4.3: Comparison between different sampling window sizes for WaveGrad 2. All models use 1,000 iterations for inference.

Model	Window Size	MOS (↑)
Encoder(512) + WaveGrad(Base)	0.8 sec	3.80 ± 0.07
Encoder(512) + WaveGrad(Base)	3.2 sec	3.88 ± 0.07

4.6.3 WaveGrad 2: Network Size

We carried out ablations using different network sizes. The encoder only needs to be computed once, thus increasing the hidden dimension has a small impact on the inference speed. On the other hand, the WaveGrad decoder needs to be executed multiple times depending on the number of iterations.

Subjective evaluation results are presented in Table 4.4. It can be seen from the table that the larger encoder size increased the number of parameters by a large margin and led to a slight quality improvement. However, the improvement was smaller compared to using a larger WaveGrad decoder, indicating that having a larger decoder is crucial.

Table 4.4: Comparison between different network sizes for WaveGrad 2. All models use 1000 iterations for inference.

Model	Model size	MOS (↑)
Encoder(512) + WaveGrad(Base)	37M	3.88 ± 0.07

Encoder(512) + WaveGrad(Large)	40M	4.19 ± 0.06
Encoder(2048) + WaveGrad(Base)	188M	4.05 ± 0.07
Encoder(2048) + WaveGrad(Large)	193M	4.37 ± 0.05

Table 4.5: Impact of augmentation on the learned representations. All models use 1000 iterations for inference.

Model	**SpecAug**	**MOS (↑)**
Encoder(2048) + WaveGrad(Large)	N	4.37 ± 0.05
Encoder(2048) + WaveGrad(Large)	Y	4.40 ± 0.05

4.6.4 WaveGrad 2: Hidden Features Augmentation

We explored applying a variant of SpecAugment [111] to the conditioning input to the decoder (the resampled encoder output). The augmentation is applied to the learned hidden representations instead of the spectrograms. This can be viewed as a form of correlated block dropout. Thirty-two consecutive frames were randomly selected to be masked and we applied it twice. The

intuition is that the WaveGrad decoder can recover the masked part by conditioning the contextual information. This enforces the encoder to learn robust representations which include more context information. Results are shown in Table 4.5. We did not observe large improvements with this regularization.

4.6.5 WaveGrad 2: Multi-task Learning and Speed-Quality Tradeoff

Inspired by FastSpeech 2s [41], we explored leveraging Mel-spectrogram features to enhance encoder training. The encoder is encouraged to extract representations that can directly predict the spectrogram features. We added a separate Mel-spectrogram decoder after the resampling layer to predict the Mel-spectrogram features. This decoder included one upsampling block and the mean squared error (MSE) was measured as an additional loss on the whole sequence. During inference, we dropped this decoder similar to FastSpeech 2s [41].

As shown in Table 4.6, there was no significant performance difference with multi-task training. This suggests that multi-task learning is not beneficial for the end-to-end generation. We also explored reducing the number of iterations from 1000 to 50 and found a small performance degradation (about 0.07 points).

Table 4.6: Impact of multi-task (MT) learning and number of iterations for WaveGrad 2.

Model	**MT**	**Iter**	**MOS (↑)**
Encoder(2048) + WaveGrad(Large)	N	1000	4.37 ± 0.05
Encoder(2048) + WaveGrad(Large)	Y	1000	4.39 ± 0.05
Encoder(2048) + WaveGrad(Large)	Y	50	4.32 ± 0.05

4.7 Summary

This section proposed two non-autoregressive text-to-speech models based on iterative refinement: WaveGrad and WaveGrad 2. WaveGrad is a vocoder model which relies on the predicted mel-spectrogram features given by another separately trained model. WaveGrad 2 is a phoneme-to-wave model which generates waveform directly from the input phoneme sequence. Both WaveGrad and WaveGrad 2 offer a flexible trade-off between quality and speed by adjusting the number of refinement steps during inference.

Chapter 5

Noise-based non-autoregressive ASR

The noise-based non-autoregressive method has enormous successes for both spectrogram-to-wave and phoneme-to-wave generation. Inspired by those approaches, a natural question is how to apply a similar denoising idea to the topic of speech recognition. In this chapter, a novel noise-based non-autoregressive speech recognition approach, Align-Denoise, is proposed. Align-Denoise is built on the idea of denoising autoencoder [115] and previous work Align-Refine [71]. This chapter starts with the introduction of previous work, Align-Refine. It then discusses how Align-Denoise is designed to speed up the training procedure of Align-Refine. Align-Denoise reduces the training complexities of Align-Refine and requires only a single iteration to reach comparable performance.

Ablation studies discuss some phenomena observed during training and how external language model or beam search improve Align-Denoise's performance.

5.1 Align-Refine

Here we first review the previous approach, Align-Refine [71]. The Align-Refine model can be considered as an extension to the classical model Connectionist Temporal Classification (CTC) [65]. CTC overall simplifies the speech recognition decoding by assuming conditional independence between predictions,

$$p(\mathbf{a} \mid \mathbf{x}) = \prod_{t=0}^{T-1} p(\mathbf{a}_t \mid \mathbf{x}) \tag{5.1}$$

where T is the number of frames, $\mathbf{x}$ are the features of the input audio, $\mathbf{a}$ are the alignments that can be mapped to the final result $\mathbf{y}$. To measure the likelihood efficiently and exactly, CTC uses dynamic programming to marginalize over all possible alignments. However, this independence assumption is too strong for speech recognition since it ignores the dependence between tokens at different positions. Consequently, dependencies related to the language model cannot be easily learned by CTC training.

To alleviate the conditional independence assumption, a common solution is to provide an additional conditional signal which provides *a priori* alignment

information:

$$p(\mathbf{a} \mid \mathbf{x}, \hat{\mathbf{a}}) = \prod_{t=0}^{T-1} p(\mathbf{a}_t \mid \mathbf{x}, \hat{\mathbf{a}}) \tag{5.2}$$

where $\hat{\mathbf{a}}$ contains alignment related information. Previous works explored different forms of $\hat{\mathbf{a}}$. Imputer [70] uses a partial result as the conditional signal, and the unfinished positions are labeled by the special token MASK. In comparison, Align-Refine and this work condition the likelihood on noisy alignment $\hat{\mathbf{a}}$ which may provide more information than the partial result.

Align-Refine optimizes a non-causal decoder to refine the initial decoding result from the encoder. Instead of training separate models for each step, Align-Refine applies the same decoder iteratively on the previous predicted alignment, starting from the encoder's prediction. During training, this recursive refinement has been unfolded for K iterations but there are no gradient flows between different iterations. Thus Align-Refine requires K forward passes through the decoder but during back-propagation, they are treated independently.

Align-Refine can be viewed as a special denoising autoencoder [115]. The goal of the denoising autoencoder is to learn the mapping from the noisy sample to the clean one:

$$\mathbb{E}_{\mathbf{a}\sim p(\mathbf{a})}\, \mathbb{E}_{\tilde{\mathbf{a}}\sim q(\tilde{\mathbf{a}}|\mathbf{a})} \left[\left\| f_{\mathsf{dec}}(\mathbf{a} \mid \mathbf{x}, \tilde{\mathbf{a}}) - \mathbf{a} \right\|_2^2 \right] \tag{5.3}$$

to optimize the network f_{dec} where $q(\tilde{\mathbf{a}} \mid \mathbf{a})$ is the noise distribution. For the case of ASR, additional conditional signal x is available and the CTC objective

is used instead:

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\,\mathbb{E}_{\tilde{\mathbf{a}}\sim q(\tilde{\mathbf{a}}|\mathbf{x},\mathbf{a})}\,L_{\mathsf{CTC}}\left[f_{\mathsf{dec}}(\mathbf{a}\mid\mathbf{x},\tilde{\mathbf{a}}),\mathbf{y}\right]. \tag{5.4}$$

The noise distribution in Align-Refine has been simplified to the form $q(\tilde{\mathbf{a}} \mid \mathbf{x})$ and is a collection of sequences $\{f_{\mathsf{enc}}(\mathbf{x}),\ f_{\mathsf{dec}}(f_{\mathsf{enc}}(\mathbf{x})),\ ...,\ f^{K}_{\mathsf{dec}}(f_{\mathsf{enc}}(\mathbf{x}))\}$ where $f_{\mathsf{enc}}(\mathbf{x})$ is the decoding result from the encoder, $f^{i}_{\mathsf{dec}}(f_{\mathsf{enc}}(\mathbf{x}))$ is the result after applying the decoder i times.

5.2 Align-Denoise

The noise distribution $q(\tilde{\mathbf{a}} \mid \mathbf{a},\mathbf{x})$ used in Align-Refine requires K forward passes through the decoder. This increases the training time by a large margin, and it is impossible to increase the number of iterations K to an arbitrary number since it is limited by the GPU memory. Another important factor is that the ground-truth alignment a is not used in the noise distribution. In this case, we may not have enough training samples to cover the space near a if the initial proposal is far away from a.

Align-Denoise speeds up the training process of Align-Refine by incorporating the ground-truth alignment a. Instead of taking a fixed number of steps from the initial proposal generated by the encoder, Align-Denoise samples alignments that combine the ground-truth alignment and the initial proposal.

The initial proposal $\mathbf{a}_{\mathsf{enc}}$ is generated by the encoder using greedy decoding,

$$\mathbf{a}_{\mathsf{enc}} = \arg\max f_{\mathsf{enc}}(\mathbf{a} \mid \mathbf{x}) \tag{5.5}$$

which can be estimated efficiently by taking argmax separately at each position due to the conditional independence assumption.

The posterior alignment provided by the encoder given the ground truth transcription y is

$$P(\mathbf{a}_t = k \mid \mathbf{x}, \mathbf{y}, f_{\mathsf{enc}}) = \sum_{\mathbf{a}:\phi(\mathbf{a})=\mathbf{y}\ \mathbf{a}_t=k} f_{\mathsf{enc}}(\mathbf{a} \mid \mathbf{x}) \tag{5.6}$$

where ϕ is the mapping function from CTC alignment to the label sequence. The posterior can be estimated by a dynamic programming algorithm known as the forward-backward algorithm [116]. This posterior is the probability of $\mathbf{a}_t = k$ among all acceptable alignments and the condition of acceptable alignments is $\phi(\mathbf{a}) = \mathbf{y}$. Alternatively, it can be viewed as a weighted finite-state acceptor (WFSA) that represents all possible alignments with frame-level label likelihoods.

The ground-truth alignment can be derived from the posterior by

$$\mathbf{a}_{\mathsf{gt}} = g(\mathbf{x}, \mathbf{y}, f_{\mathsf{enc}}) = \arg\max_k P(\mathbf{a}_t = k \mid \mathbf{x}, \mathbf{y}, f_{\mathsf{enc}}) \tag{5.7}$$

During training, noisy alignments ã are sampled from the special noise distribution following our alignment policy. Similar to Align-Refine, the label posterior is computed by marginalizing over all possible alignments a which

correspond to the target label sequence,

$$P(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{a}}, f_{\mathsf{dec}}) = \sum_{\mathbf{a}:\phi(\mathbf{a})=\mathbf{y}} f_{\mathsf{dec}}(\mathbf{a} \mid \mathbf{x}, \tilde{\mathbf{a}}) \tag{5.8}$$

$$= \sum_{\mathbf{a}:\phi(\mathbf{a})=\mathbf{y}} \prod_{t=0}^{T-1} f_{\mathsf{dec}}(\mathbf{a}_t \mid \mathbf{x}, \tilde{\mathbf{a}}) \tag{5.9}$$

The factorization of $\prod_{t=0}^{T-1} f_{\mathsf{dec}}(\mathbf{a}_t \mid .)$ is due to the conditional independence assumption, which results in non-autoregressive decoding.

Finally, the encoder and decoder are jointly trained by minimizing this modified CTC loss,

$$L_{\text{CTC}} = - \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p(\mathbf{x},\mathbf{y})} \mathbb{E}_{\mathbf{a}_{\text{gt}}=g(\mathbf{x},\mathbf{y},f_{\mathsf{enc}})}$$

$$\mathbb{E}_{\tilde{\mathbf{a}}\sim q(\tilde{\mathbf{a}}|\mathbf{x},\mathbf{a}_{\text{gt}})} \left[\log P(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{a}}, f_{\mathsf{dec}})\right] \tag{5.10}$$

where $p(\mathbf{x}, \mathbf{y})$ is the data distribution, $\mathbf{a}_{\text{gt}}$ are the ground truth alignments obtained by (5.7); and $q(\tilde{\mathbf{a}} \mid \mathbf{a}_{\text{gt}}, \mathbf{x})$ is the noisy distribution that generates noisy alignments given the ground truth alignments.

5.2.1 Noise Distribution

The key idea behind Align-Denoise is to design an alignment policy that can generate noisy transcripts, similar to the intermediate results encountered during inference. Given that the input sequence is discrete, we add noise to the posterior instead and sample from it.

If the greedy decoding $\mathbf{a}_{\mathsf{enc}}$, obtained from the encoder in (5.5), is adopted as the initialization, our goal is to find a mapping from $\mathbf{a}_{\mathsf{enc}}$ to the true alignment $\mathbf{a}_{\mathsf{gt}}$. This is achieved by adding the noise to the encoder posterior given ground truth labels,

$$P_{\mathsf{gt}}(\mathbf{a}_t = k) = P(\mathbf{a}_t = k \mid \mathbf{x}, \mathbf{y}, f_{\mathsf{enc}}) \tag{5.11}$$

given by (5.6). From now on, we will denote $P_{\mathsf{gt}}(\mathbf{a}_t = k)$ as *ground truth posterior*. The errors made by the encoder correspond to the set of indices $r \in R$ such that

$$\mathbf{a}_{\mathsf{enc},r} \neq \mathbf{a}_{\mathsf{gt},r} \tag{5.12}$$

To sample the noisy alignment $\tilde{\mathbf{a}}$, we keep the correct predictions from the greedy encoder result consistent and sample the rest frames:

$$V_t(k) \sim \mathcal{N}(\sqrt{\alpha} * P_{\mathsf{gt}}(\mathbf{a}_t = k), (1-\alpha) * \sigma_t^2(k) * \mathbb{1}t \in R) \tag{5.13}$$

$$\tilde{\mathbf{a}}_t = \arg\max_k V_t \tag{5.14}$$

where α is sampled from $[0, 1]$ to control the global noise level like Chapter 4, V_t is the sampled per-frame distribution of labels. Intuitively, we choose frames based on the correctness of the initial proposal and sample frame-level labels based on combined uncertainties σ_t from CTC decoding and forward-backward algorithm. For a specific position, the amount of noise in sampling comes directly from model's uncertainty.

To mimic the error made by the encoder greedy decoding, the noise variance

is defined as the combination of both greedy result and true posterior,

$$\sigma_t^2(k) = \max(P_{\mathsf{gt}}(\mathbf{a}_t = k), \lambda \times P_{\mathsf{enc}}(\mathbf{a}_t = k)), \tag{5.15}$$

with

$$P_{\mathsf{enc}}(\mathbf{a}_t) = f_{\mathsf{enc}}(\mathbf{a}_t \mid \mathbf{x}). \tag{5.16}$$

From now on, we will denote $P_{\mathsf{enc}}(\mathbf{a}_t)$ as the *encoder posterior*. λ controls the weight of the greedy decoding result. When λ is large, the input is mainly determined by the encoder result so that it may contain more errors. When λ is small, the input is close to the ground-truth alignment which includes fewer mistakes.

Figure 5.1 includes visualization of the proposed Align-Denoise training process. CTC losses are measured on both initial proposal and decoder prediction following other previous work [71, 88].

Table 5.1: Training samples generated by our policy. The noise distribution is a Gaussian distribution whose variance is controlled by the confidence from the encoder initial proposal (enc) and posterior from the forward-backward algorithm. Training samples inherit some predictions errors from the initial proposal.

posterior	______SSOO_ MR.. WANNNG TEL__LS PEOPLLE HEE IIS FFIFFTYYY___
enc	______SSOO_ MR.. WIGNNNTTELLLLS PEOPLLE HE IIS FFIFFTYYY____
sample 1	______SSOO_ MR.. WINNNG TELL_LS PEOPLLE HE IIS FFIFFTYYY____
sample 2	______SSOO_ MR.. WANNNT_TELL_LS PEOPLLE HE' IIS FFIFFTYYY___
sample 3	______SSOO_ MR.. WAHNNGTTEL_ELS PEOPLLE HE IIS FFIFFTYYY____
sample 4	______SSOO_ MR.. WANNNG_TELLLLS PEOPLLE HE IIS FFIFFTYYY____
sample 5	______SSOO_ MR.. WENNNGTTELLLLS PEOPLLE HE IIS FFIFFTYYY____

Table 5.1 demonstrates how our policy generates different training samples.

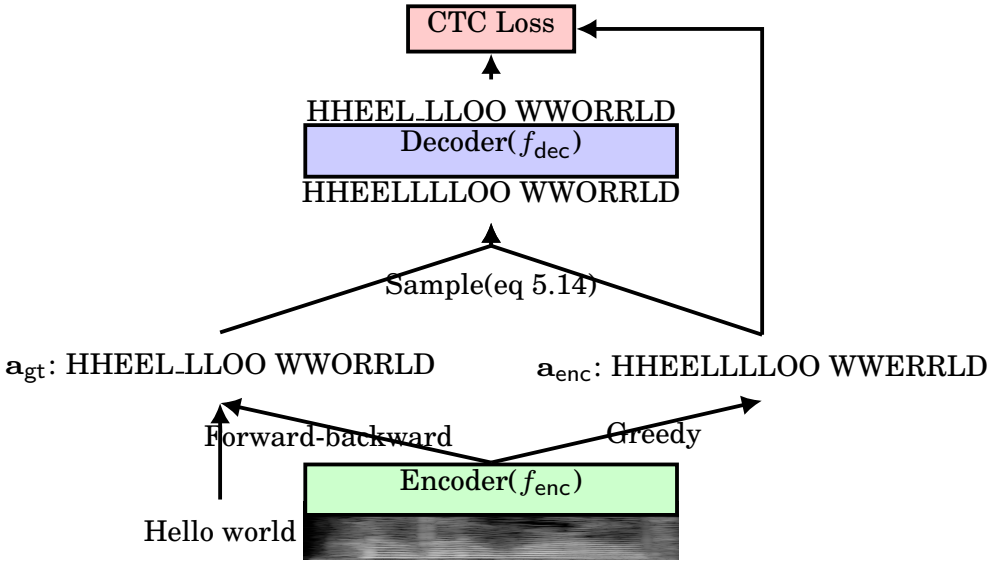

Figure 5.1: A visualization of the Align-Denoise training. Two signals are estimated from the encoder output: the initial proposal $\mathbf{a}_{enc}$ from greedy decoding and the ground-truth alignment $\mathbf{a}_{gt}$ from the forward-backward algorithm. The noisy transcript is sampled by mixing these two sequences. The decoder, a non-causal Transformer decoder, conditions on this and the encoder to improve the alignment. The connection between encoder and decoder is omitted.

The encoder greedy result makes several mistakes, for example, the words 'Wang' and 'tell'. The training examples keep some of them, and the model is optimized to correct.

5.3 Experiments

To evaluate the effectiveness of the proposed model, we conduct speech recognition experiments to compare different encoder-decoder models. The performance of the proposed model is evaluated based on character error rates (CERs) and word error rates (WERs) without relying on external language models and beam search unless specified.

All models are tested on the 150-hours AISHELL dataset [48], 581-hours Corpus of Spontaneous Japanese dataset [49], 81-hours Wall Street Journal (WSJ) dataset [40] and 210-hours Tedlium2 dataset [50]. For the network inputs, we use 80 Mel-scale filterbank coefficients with three-dimensional pitch features and apply SpecAugment [111] during model training. Our experiment setups match previous works [70, 71].

5.3.1 Network Architecture

For all experiments, we adapt the same encoder-decoder architecture as Mask CTC [117] and Align-Refine [71], which consists of Transformer self-attention layers with four attention heads, 256 hidden units, and 2048 feed-forward inner dimension size. The encoder includes 12 self-attention layers with convolutional layers that downsample the input features by a factor of 4. The decoder contains six non-causal self-attention layers.

We set the dropout rate to 0.1 and the weight of the initial proposal loss to 0.3. Each minibatch includes 48 sequences, and gradients are accumulated from 8 batches. The model is trained for 100/100/500/300 epochs for AISHELL, CSJ, WSJ and Tedlium2 respectively with a standard inverse square-root learning rate schedule and a linear warmup of 25000 steps.

5.3.2 Results

Table 5.2 shows the results for WSJ based on WERs and real-time factors (RTFs) that were measured for decoding the eval92 subset on a single CPU thread. By comparing the results for non-autoregressive models, we can see that the proposed method outperforms all other non-autoregressive baselines using just a single iteration. $\lambda = 0.3$ gives the best result, whereas the performance of $\lambda = 0.1$ is worse in comparison. When λ is small, the input alignment is too close to the ground truth, giving the network limited opportunities to improve it. However, during inference, the initial proposal contains more mistakes. Such mismatch can explain the reason why a small λ is not preferred, which is verified by the result.

Since we use the same architecture, the real-time factor of Align-Denoise matches Align-Refine with $k = 1$ iteration. Compared to the Align-Refine, the proposed Align-Denoise also speeds up the training. Align-Refine requires $K = 4$ forward passes through the decoder, which consists of 6 transformer

blocks. The proposed approach requires just a single forward pass through the decoder since noisy alignments are sampled from frame-level Gaussian distributions instead of using the greedy results from multiple forward passes. The performance overall is better than the autoregressive baseline with beam search. However adding external language model helps a lot since it introduces additional training data.

Table 5.2: Word error rates (WERs) and real time factor (RTF) for WSJ (English).

Model	**Iterations**	**dev93(↓)**	**eval92(↓)**	**RTF(↓)**
Autoregressive				
CTC-attention [88]	L	14.3	11.8	0.97
+ beam search	L	13.8	11.6	4.62
+ external LM	L	7.8	5.3	5.72
Align-Denoise				
$\lambda = 0.5$	1	13.8	11.3	0.05
$\lambda = 0.3$	1	**13.5**	**10.8**	0.05
$\lambda = 0.1$	1	14.4	11.4	0.05
Previous work				
CTC [69]	-	22.2	17.9	0.03
CTC [70]	-	–	15.2	–
Imputer (IM) [70]	8	–	16.5	–
Imputer (DP) [70]	8	–	12.7	–
Mask CTC [69]	1	15.7	12.5	0.07
Mask CTC [69]	10	15.5	12.2	0.07
Align-Refine [71]	1	14.1	11.6	0.05
Align-Refine [71]	5	13.7	11.4	0.07

Our results on CSJ are demonstrated in Table 5.3. With a single iteration, the proposed model, Align-Denoise, outperforms all other systems including the autoregressive transformer baseline with external language model and

beam search. Both external language model and beam search are not used for Align-Denoise, which greatly speeds up the inference process.

Table 5.3: Character error rates (CERs) for Corpus of Spontaneous Japanese (CSJ).

Model	**Iterations**	**Eval 1(↓)**	**Eval 2(↓)**	**Eval 3(↓)**
Autoregressive				
CTC-attention [88]	L	6.7	4.7	5.0
+ beam search(20)	L	6.3	4.3	4.6
+ external LM	L	5.9	4.1	4.6
Align-Denoise				
$\lambda = 0.3$	1	**5.4**	**4.0**	**4.2**
Previous work				
Kaldi	-	7.5	6.3	6.9
CTC	1	8.0	5.4	7.2

On AISHELL, the results of Align-Denoise are reported in the Table 5.4. Align-Denoise matches autoregressive baseline without beam search and external language model, while it is slightly worse when beam search and external language model is included. Considering the fact that Align-Denoise is non-autoregressive and requires only a single iteration, the decoding speed is much faster.

Table 5.4: Character error rates (CERs) for AISHELL with Align-Denoise.

Model	**Iterations**	**Dev CER**	**Test CER(↓)**
Autoregressive			
CTC-attention [88]	L	5.7	6.2
+ beam search(10)	L	**5.2**	**5.7**
+ external language model	L	**5.2**	**5.7**
Kaldi nnet3	-	-	8.6
Kaldi chain	-	-	7.5
Align-Denoise			

$\lambda = 0.3$	1	**5.7**	**6.2**

Finally, results of Tedlium2 are given in Table 5.5. Align-Denoise outperforms autoregressive baseline when the beam search and external language model are not used. However, Align-Denoies is slightly worse than the autoregressive baseline with beam search setting beam size to 40. This suggests that for this noisy dataset based on TED talks, beam search is still helpful in reducing uncertainties. The external language model further enhances the performance of autoregressive models. Incorporating an external language model to Align-Denoise can be achieved by introducing extra complexities during inference, going from simple greedy decoding to more complicated methods like FST decoding. This slows down the whole decoding process, especially compared to the greedy decoding, where predictions for individual frames can be estimated in parallel. The performance comparison is included in the following ablation studies.

Table 5.5: Word error rates (WERs) for Tedlium2.

Model	**Iterations**	**Dev(↓)**	**Eval(↓)**
Autoregressive			
CTC-attention [88]	L	11.7	9.9
+ beam search(40)	L	10.4	9.0
+ external language model	L	9.3	8.1
Align-Denoise			
$\lambda = 0.3$	1	10.7	9.8
Previous work			

Kaldi	-	9.0	9.0
CTC [96]	1	-	16.6

5.4 Discussion

5.4.1 Alignment mismatch

There exist some cases where a_{enc} and a_{gt} are mis-aligned. One example is given in Table 5.6. In the initial proposal from the encoder, the last 'S' in the word *prices* has been repeated twice – which does not change the decoding result. However, it makes the alignment mismatched and there is not enough character space to recognize the word *thing*. The only way to correct this is to remove the extra 'S' and move the whole segment to the left. Similar problems have been observed in other non-autoregressive approaches [69] that predict tokens directly instead of the alignment. Reducing the downsampling rate in the encoder could mitigate this kind of error at the cost of decoding speed.

Table 5.6: Mis-aligned example for Align-Denoise. This is part of the sentence and the original sentence is **the meteoric rise in prices is the worst thing that could happen to the collectible car hobby.**

$\mathbf{a}_{\text{enc}}$	PPRICESS ISS THE WORSTTHAN THAT COUD HAVEPENN
$\mathbf{a}_{\text{gt}}$	PPRICES IS THE WORST THING THAT COULD HAP_PEN

5.4.2 Combining with beam search and external language model

The proposed model, Align-Denoise, already conditions on some contextual information. However, it is still possible to combine the Align-Denoise system with the beam search and external language model to gain further improvement since it is usually trained on some additional text corpus. But this makes the inference process much slower. Before the inference process makes predictions simultaneously and after that nearby tokens are merged according to the CTC mapping rules. After, beam search and language model require left-to-right search which involves a lot of sequential operations.

Overall, consistent improvements are observed on most datasets. On WSJ, as shown in Table 5.7, no improvements are observed with only beam search. Instead, beam search makes the inference process seven times slower. When we used the word-based language model combined, we got about 1.5% absolute

improvement, which is promising. But the real-time factor increased to 1.0, which is 20 times slower than the simple parallel decoding strategy.

Table 5.7: Align-Denoise with/without beam search and external language model on WSJ.

Model	**dev93 WER**	**eval92 WER**	**RTF**
Autoregressive			
CTC-attention [88]	14.3	11.8	0.97
+ beam search	13.8	11.6	4.62
+ external LM	7.8	5.3	5.72
Align-Denoise			
$\lambda = 0.3$	13.5	10.8	0.05
+ Beam search	13.5	10.8	0.37
+ external LM	12.1	9.2	1.06

CSJ results are shown in Table 5.8. No significant improvements are observed on this dataset when beam search is used with an external language model. It is also 20 times slower than the original Align-Denoise decoding for the running time.

Table 5.8: Align-Denoise with/without beam search and external language model on CSJ.

Model	**Eval 1 CER**	**Eval 2 CER**	**Eval 3 CER**	**RTF**
Autoregressive				
CTC-attention [88]	6.7	4.7	5.0	0.21
+ beam search(20)	6.3	4.3	4.6	3.86
+ external LM	5.9	4.1	4.6	4.46
Align-Denoise				
$\lambda = 0.3$	5.4	4.0	4.2	0.05
+ Beam search	5.4	3.9	4.2	0.46
+ external LM	5.3	3.9	4.2	0.96

In Table 5.9, the performance improvement for AISHELL is quite considerable while the real-time factor is five times slower. Similarly, without an external language model, no improvements are observed.

Table 5.9: Align-Denoise with/without beam search and external language model on AISHELL.

Model	**dev CER**	**test CER**	**RTF**
Autoregressive			
CTC-attention [88]	5.7	6.2	0.10
+ beam search(10)	**5.2**	**5.7**	0.73
+ external language model	**5.2**	**5.7**	1.44
Align-Denoise			
$\lambda = 0.3$	5.7	6.2	0.07
+ Beam search	5.7	6.2	0.14
+ external LM	5.5	6.0	0.33

For TEDLIUM2, the improvements with beam search and external language model is up to 8% relatively in Table 5.10 but the inference speed is 250 times slower. This is mainly due to the long output sequence and large language model.

Table 5.10: Align-Denoise with/without beam search and external language model on TEDLIUM2.

Model	**Dev WER**	**Eval WER**	**RTF**
Autoregressive			
CTC-attention [88]	11.7	9.9	0.30
+ beam search(40)	10.4	9.0	12.11
+ external LM	9.3	8.1	24.50
Align-Denoise			
$\lambda = 0.3$	10.7	9.8	0.05
+ Beam search	10.8	9.8	1.60

+ external LM	9.9	9.0	12.98

In summary, Align-Denoise doesn't benefit from beam search alone. However, adding beam search and external language model are always beneficial for the proposed non-autoregressive Align-Denoise system. Sometimes the improvement is quite large but the decoding speed is always much slower. The real-time factor for Align-Denoise is quite consistent and less than 0.07 while adding those makes it 5-250 times slower depending on the dataset. Overall, Align-Denoise is a good tradeoff between performance and decoding speed while still benefits from the development of beam search and language model research.

5.5 Summary

Align-Denoise is a non-autoregressive speech recognition model which resolves the performance gap between autoregressive baseline and non-autoregressive methods. Align-Denoise uses only a single iteration for the decoding, which matches the autoregressive baseline with a much faster speed. The introduction of an external language model improves both autoregressive baseline and Align-Denoise, while the improvement on Align-Denoise is more limited. While the combination of the non-autoregressive and external language models is an

interesting area to explore in the future, it makes the whole process much slower, as observed in the discussion section. The speed deterioration makes it less attractive, so it is not discussed in the dissertation.

Chapter 6

Conclusion

This dissertation investigates the possibility to apply **non-autoregressive end-to-end** approaches to two critical speech applications: speech recognition and speech synthesis. Non-autoregressive approaches provide a new direction to speed up the state-of-the-art end-to-end systems by reducing the number of iterations through the decoder to constant, without much degradation in performance. Two novel directions are discussed in this dissertation.

The first direction is based on masked language models like BERT [89]. Two new non-autoregressive training frameworks are proposed, which we call **A-CMLM** and **A-FMLM**, which is inspired by the previous work on neural machine translation [46]. A-CMLM applies a conditional language model to speech recognition. Besides decoding strategies like left-to-right, mask-predict, a new decoding strategy is proposed. Based on the connection with a classi-

cal dependency parsing [92], this decoding strategy is named *easy first*. Inspired from easy first, a new training framework, A-FMLM, is further proposed, which utilizes factorization to bridge the gap between training and inference. In experiments, in comparison to classical left-to-right order these two show great speedup with reasonable performance on four different corpora. From ablation studies, one can observe that the external language model is very helpful, especially when it is trained on an additional text corpus.

The second direction utilizes a denoising objective to refine the initial result with errors.This dissertation explores this direction in both speech recognition and speech synthesis. For speech recognition, Align-Denoise is proposed, which uses a single iteration to reach high performance, without the need for beam search and external language model. Align-Denoise reaches a similar performance as the autoregressive baseline without an external language model, and it also benefits from the introduction of the external language model. Align-Denoise achieves 20x real-time generation, up to 480x faster than the autoregressive baseline with the external language model.

For text-to-speech, two different tasks are studied in this dissertation: vocoder and phoneme-to-wave. The most popular TTS approach uses two stages to synthesize the audio. In the first stage, the phoneme sequence is transformed into Mel-spectrogram features. In the second stage, a vocoder is trained and used to predict waveform samples from the Mel-spectrogram features. Usu-

ally, the vocoder is the challenging and slowest part. WaveGrad is a novel neural vocoder that supports a different number of steps to generate high-fidelity audio. Our experiments demonstrated that only six iterations are required to generate high-quality samples. We further extend the vocoder to the full phoneme-to-wave model, WaveGrad 2, which generates waveform directly from the phoneme sequence. WaveGrad 2 dramatically reduces the performance gap between autoregressive and non-autoregressive systems and is able to match the state-of-the-art autoregressive system with enough iterations.

To sum up, here are the major contributions of this dissertation:

- A new masking-based framework is proposed for speech recognition, which includes two novel methods, Audio-Conditional Masked Language Model (A-CMLM) and Audio-Factorized Masked Language Model (A-FMLM).
- A-CMLM is the first attempt of non-autoregressive end-to-end speech recognition and achieves similar performance on AISHELL, slightly worse performance on TEDLIUM2, CSJ, and WSJ.
- A-FMLM is proposed based on observations, and it reduces the number of required iterations.
- A new possibility is explored, which uses a large number of refinement steps to avoid the beam search, and acceptable performance is observed with large speedup

- Noise-based speech synthesis approaches which are non-autoregressive and support a dynamic trade-off between fidelity and inference speed.

- WaveGrad is a vocoder that generates waveform samples from the input mel-spectrogram features. WaveGrad starts from Gaussian white noise and iteratively updates the signal via a gradient-based sampler conditioned on the mel-spectrogram. WaveGrad is non-autoregressive, and requires only a constant number of generation steps during inference. Experiments reveal that the model can generate high-fidelity audio samples using as few as six iterations. WaveGrad is simple to train, and implicitly optimizes for the weighted variational lower-bound of the log-likelihood. The empirical experiments demonstrated WaveGrad to generate high fidelity audio samples matching a strong autoregressive baseline.

- WaveGrad 2 is a phone-to-wave model which is the combination of WaveGrad decoder with Tacotron-2 [33] based encoder. The generation procedure provides a trade-off between fidelity and speed by varying the number of refinement steps. Experiments demonstrate that WaveGrad 2 is capable of generating high fidelity audio comparable to solid baselines. Ablation studies exploring different model configurations found that increased model size is the most important factor in determining WaveGrad 2 synthesis quality.

- the noise-based approach idea is adapted to the speech recognition and proposed Align-Denoise, which favors a single pass decoding and can be considered as an extension to the existing CTC model. Align-Denoise reaches comparable results with baselines on multiple datasets without beam search and external language model, which gives a very consistent real-time factor less than 0.07.

These claims have been validated by empirical results on several data sets, including both public available ones and proprietary ones.

Bibliography

[1] J. Sachs and J. Devin, "Young children's use of age-appropriate speech styles in social interaction and role-playing," *Journal of child language*, vol. 3, no. 1, pp. 81–98, 1976.

[2] Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, and B. Schuller, "Deep learning for environmentally robust speech recognition: An overview of recent developments," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 5, pp. 1–28, 2018.

[3] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj *et al.*, "Chime-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings," in *CHiME 2020-6th International Workshop on Speech Processing in Everyday Environments*, 2020.

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet:

A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[5] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. C. Courville, and Y. Bengio, "Char2Wav: End-to-End Speech Synthesis," in *ICLR*, 2017.

[6] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis," in *INTERSPEECH*, 2017.

[7] F. Biadsy, R. J. Weiss, P. J. Moreno, D. Kanevsky, and Y. Jia, "Parrotron: An End-to-End Speech-to-Speech Conversion Model and its Applications to Hearing-Impaired Speech and Speech Separation," in *INTERSPEECH*, 2019.

[8] Y. Jia, R. J. Weiss, F. Biadsy, W. Macherey, M. Johnson, Z. Chen, and Y. Wu, "Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model," in *INTERSPEECH*, 2019.

[9] S. Vasquez and M. Lewis, "MelNet: A Generative Model for Audio in the Frequency Domain," *arXiv preprint arXiv:1906.01083*, 2019.

[10] P. Mermelstein, "Distance measures for speech recognition, psychologi-

cal and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[11] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[12] T. Hori, J. Cho, and S. Watanabe, "End-to-end speech recognition with word-based rnn language models," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 389–396.

[13] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," in *INTERSPEECH*, 2018.

[14] T. K. Vintsyuk, "Speech discrimination by dynamic programming," *Cybernetics*, vol. 4, no. 1, pp. 52–57, 1968.

[15] J. Baker, "The dragon system–an overview," *IEEE Transactions on Acoustics, speech, and signal Processing*, vol. 23, no. 1, pp. 24–29, 1975.

[16] D. Povey, L. Burget, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. K. Goel, M. Karafiát, A. Rastrow *et al.*, "Subspace gaussian mixture models for speech recognition," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 4330–4333.

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

[18] J. T. Goodman, "A bit of progress in language modeling," *Computer Speech & Language*, vol. 15, no. 4, pp. 403–434, 2001.

[19] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, "Rnnlm-recurrent neural network language modeling toolkit," in *Proc. of the 2011 ASRU Workshop*, 2011, pp. 196–201.

[20] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language modeling with deep transformers," *Proc. Interspeech 2019*, pp. 3905–3909, 2019.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *international conference on learning representations*, 2014.

[22] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens *et al.*, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th annual meeting of the association for computational linguistics com-*

panion volume proceedings of the demo and poster sessions, 2007, pp. 177–180.

[23] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N.-E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "ESPnet: End-to-end speech processing toolkit," *Proc. Interspeech 2018*, pp. 2207–2211, 2018.

[24] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on transformer vs rnn in speech applications," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 449–456.

[25] H. Dudley and T. H. Tarnoczy, "The speaking machine of wolfgang von kempelen," *The Journal of the Acoustical Society of America*, vol. 22, no. 2, pp. 151–166, 1950.

[26] W. von Kempelen, *Le mécanisme de la parole, suivi de la déscription d'une machine parlante*. Imprimé chez B. Bauer, 1791.

[27] C. Scully, "Articulatory synthesis," in *Speech production and speech modelling*. Springer, 1990, pp. 151–186.

[28] C. H. Shadle and R. I. Damper, "Prospects for articulatory synthesis: A

position paper," in *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*, 2001.

[29] R. Carlson, B. Granström, and I. Karlsson, "Experiments with voice modelling in speech synthesis," *Speech communication*, vol. 10, no. 5-6, pp. 481–489, 1991.

[30] R. H. Mannell, "Formant diphone parameter extraction utilising a labelled single-speaker database," in *Fifth International Conference on Spoken Language Processing*, 1998.

[31] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[32] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *Proc. Interspeech 2017*, pp. 4006–4010, 2017.

[33] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

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

[35] J. Shen, Y. Jia, M. Chrzanowski, Y. Zhang, I. Elias, H. Zen, and Y. Wu, "Non-attentive tacotron: Robust and controllable neural tts synthesis including unsupervised duration modeling," *arXiv preprint arXiv:2010.04301*, 2020.

[36] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[37] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2410–2419.

[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *NIPS*, 2017.

[40] D. B. Paul and J. M. Baker, "The design for the wall street journal-based CSR corpus," in *Proceedings of Workshop on Speech and Natural Language*, 1992.

[41] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=piLPYqxtWuA

[42] J. Donahue, S. Dieleman, M. Binkowski, E. Elsen, and K. Simonyan, "End-to-end adversarial text-to-speech," in *International Conference on Learning Representations*, 2021.

[43] R. J. Weiss, R. Skerry-Ryan, E. Battenberg, S. Mariooryad, and D. P. Kingma, "Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5679–5683.

[44] L. Ruis, M. Stern, J. Proskurnia, and W. Chan, "Insertion-Deletion Transformer," in *EMNLP: Workshop of Neural Generation and Translation*, 2019.

[45] W. Chan, N. Kitaev, K. Guu, M. Stern, and J. Uszkoreit, "KER-

MIT: generative insertion-based modeling for sequences," *CoRR*, vol. abs/1906.01604, 2019.

[46] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-Predict: Parallel Decoding of Conditional Masked Language Models," in *EMNLP*, 2019.

[47] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[48] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*. IEEE, 2017, pp. 1–5.

[49] K. Maekawa, "Corpus of spontaneous japanese: Its design and evaluation," in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[50] A. Rousseau, P. Deléglise, Y. Esteve *et al.*, "Enhancing the ted-lium corpus with selected data for language modeling and more ted talks." in *LREC*, 2014, pp. 3935–3939.

[51] A. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," in *ICML*, 2016.

[52] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2410–2419.

[53] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High fidelity speech synthesis with adversarial networks," in *International Conference on Learning Representations*, 2020.

[54] T. R. Niesler and P. C. Woodland, "A variable-length category-based n-gram language model," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1. IEEE, 1996, pp. 164–167.

[55] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural networks*, vol. 3, no. 1, pp. 23–43, 1990.

[56] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proc.Interspeech*, pp. 338–342, 01 2014.

[57] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 273–278.

[58] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6874–6878.

[59] A. Pauls and D. Klein, "Faster and smaller n-gram language models," in *Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 258–267.

[60] M. Suzuki, N. Itoh, T. Nagano, G. Kurata, and S. Thomas, "Improvements to n-gram language model using text generated from neural language model," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7245–7249.

[61] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[62] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for

language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[63] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[64] J. Vig and Y. Belinkov, "Analyzing the structure of attention in a transformer language model," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019, pp. 63–76.

[65] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[66] E. Battenberg, J. Chen, R. Child, A. Coates, Y. G. Y. Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, "Exploring neural transducers for end-to-end speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 206–213.

[67] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, 2012.

[68] N. Chen, S. Watanabe, J. Villalba, and N. Dehak, "Non-Autoregressive Transformer Automatic Speech Recognition," in *arXiv*, 2019.

[69] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," *Proc. Interspeech 2020*, pp. 3655–3659, 2020.

[70] C. Saharia, G. E. Hinton, M. Norouzi, N. Jaitly, and W. Chan, "Imputer: Sequence modelling via imputation and dynamic programming," in *ICML*, 2020.

[71] E. A. Chi, J. Salazar, and K. Kirchhoff, "Align-refine: Non-autoregressive speech recognition via iterative realignment," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online, Jun. 2021, pp. 1920–1927.

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

[73] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg *et al.*, "Parallel wavenet:

Fast high-fidelity speech synthesis," in *International conference on machine learning*. PMLR, 2018, pp. 3918–3926.

[74] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," in *International Conference on Learning Representations*, 2018.

[75] S. Kim, S.-G. Lee, J. Song, J. Kim, and S. Yoon, "Flowavenet: A generative flow for raw audio," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3370–3378.

[76] H. Kim, H. Lee, W. H. Kang, S. J. Cheon, B. J. Choi, and N. S. Kim, "Wavenode: A continuous normalizing flow for speech synthesis," *arXiv preprint arXiv:2006.04598*, 2020.

[77] N.-Q. Wu and Z.-H. Ling, "Waveffjord: Ffjord-based vocoder for statistical parametric speech synthesis," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7214–7218.

[78] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.

[79] L. Dinh, D. Krueger, and Y. Bengio, "NICE: non-linear independent com-

ponents estimation," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[80] D. P. Kingma and P. Dhariwal, "Glow: generative flow with invertible 1x1 convolutions," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 10 236–10 245.

[81] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations*, 2018.

[82] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2018.

[83] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Zhen Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

[84] G. Yang, S. Yang, K. Liu, P. Fang, W. Chen, and L. Xie, "Multi-band melgan: Faster waveform generation for high-quality text-to-speech," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 492–498.

[85] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6199–6203.

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

[87] O. McCarthy and Z. Ahmed, "Hooligan: Robust, high quality neural vocoding," *arXiv preprint arXiv:2008.02493*, 2020.

[88] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 4835–4839.

[89] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[90] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V.

Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[91] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[92] Y. Goldberg and M. Elhadad, "An efficient algorithm for easy-first non-directional dependency parsing," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 742–750.

[93] Y. Shibata, T. Kida, S. Fukamachi, M. Takeda, A. Shinohara, T. Shinohara, and S. Arikawa, "Byte pair encoding: A text compression scheme that accelerates pattern matching," Technical Report DOI-TR-161, Department of Informatics, Kyushu University, Tech. Rep., 1999.

[94] K. An, H. Xiang, and Z. Ou, "Cat: A ctc-crf based asr toolkit bridging the hybrid and the end-to-end approaches towards data efficiency and low latency," *Proc. Interspeech*, pp. 566–570, 2020.

[95] P. Ramachandran, P. Liu, and Q. Le, "Unsupervised pretraining for se-

quence to sequence learning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, Sep. 2017, pp. 383–391.

[96] Y. Higuchi, H. Inaguma, S. Watanabe, T. Ogawa, and T. Kobayashi, "Improved mask-ctc for non-autoregressive end-to-end asr," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8363–8367.

[97] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching." *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[98] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.

[99] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *ICML*, 2015.

[100] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

[101] ——, "Improved techniques for training score-based generative models,"

Advances in neural information processing systems, vol. 33, pp. 12 438–12 448, 2020.

[102] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely, and B. Hariharan, "Learning gradient fields for shape generation," *ECCV*, 2020.

[103] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[104] M. A. Saxe, L. J. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *international conference on learning representations*, 2014.

[105] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio, "Feature-wise transformations," *Distill*, vol. 3, no. 7, p. e11, 2018.

[106] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic Image Synthesis with Spatially-Adaptive Normalization," in *CVPR*, 2019.

[107] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ICML*, 2015.

[108] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfit-

ting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[109] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[110] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. C. Courville, and C. J. Pal, "Zoneout: Regularizing rnns by randomly preserving hidden activations." in *ICLR (Poster)*, 2017.

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

[112] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *international conference on learning representations*, 2017.

[113] R. Kubichek, "Mel-Cepstral Distance Measure for Objective Speech Quality Assessment," in *IEEE PACRIM*, 1993.

[114] W. Chu and A. Alwan, "Reducing F0 Frame Error of F0 Tracking Algo-

rithms under Noisy Conditions with an Unvoiced/Voiced Classification Frontend," in *ICASSP*, 2009.

[115] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[116] L. E. Baum *et al.*, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes," *Inequalities*, vol. 3, no. 1, pp. 1–8, 1972.

[117] S. Karita, N. E. Y. Soplin, S. Watanabe, M. Delcroix, A. Ogawa, and T. Nakatani, "Improving Transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," in *Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019.