Streaming Automatic Speech Recognition with Low Latency and High Accuracy

低遅延かつ高精度 ストリーミング音声認識

A Thesis Submitted to the Department of Computer Science and Communications Engineering,
the Graduate School of Fundamental Science and Engineering of Waseda University
in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission date: July 24th, 2023

Huaibo Zhao
趙　懐博

5121FE01-4

Advisor: Prof. Tetsunori Kobayashi
Research guidance: Research on Perceptual Computing

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 End-to-end speech recognition . . . 4
  - 1.2.1 Attention-based encoder-decoder . . . 5
  - 1.2.2 Connectionist temporal classification (CTC) . . . 5
  - 1.2.3 Transducer . . . 6
- 1.3 Objective . . . 7
- 1.4 Overview . . . 8

2 Existing streaming end-to-end ASR **9**
- 2.1 Introduction . . . 9
- 2.2 Trigger attention-based streaming encoder-decoder (TA-E/D) . . . 10
  - 2.2.1 Streaming encoder with attention masks . . . 11
  - 2.2.2 Trigger attention-based streaming decoder . . . 11
- 2.3 Streaming Transformer-Transducer (T-T) . . . 12
  - 2.3.1 Streaming acoustic encoder with chunk-wise attention masks . . . 13
- 2.4 Contextual block streaming encoder-decoder (CBS-E/D) . . . 14
  - 2.4.1 Contextual block streaming encoder . . . 15
  - 2.4.2 Streaming decoding with block-wise synchronous beam search . . . 16
- 2.5 Contextual block streaming Transducer (CBS-T) . . . 17

3 Mask-CTC-based encoder pre-training for streaming ASR **19**
- 3.1 Introduction . . . 19
- 3.2 Mask-CTC framework . . . 20
- 3.3 Mask-CTC-based pre-training . . . 22
  - 3.3.1 Mask-CTC-based pre-training for TA-E/D . . . 23
  - 3.3.2 Mask-CTC-based pre-training for streaming T-T . . . 25
  - 3.3.3 Mask-CTC-based pre-training for CBS-E/D . . . 26
- 3.4 Speech recognition experiments . . . 26
  - 3.4.1 Experiment of TA-E/D . . . 26
  - 3.4.2 Experiment of streaming T-T . . . 29
  - 3.4.3 Experiment of CBS-E/D . . . 30
  - 3.4.4 Analysis of output token alignments . . . 32
- 3.5 Summary . . . 34

4 Multi-look-ahead architecture for zero look-ahead streaming ASR **35**
4.1 Introduction . . . 35
4.2 Multi-look-ahead streaming ASR . . . 36
4.3 Implementation methods of the multi-look-ahead architecture . . . 38
4.3.1 Gangi model . . . 38
4.3.2 Unity model . . . 40
4.3.3 Bifurcation model . . . 41
4.4 Speech recognition experiments . . . 43
4.4.1 Datasets . . . 43
4.4.2 Model settings . . . 44
4.4.3 Compared models . . . 45
4.4.4 Experimental results . . . 46
4.4.5 Effect of parameter sharing . . . 51
4.5 Summary . . . 53
4.6 Appendix . . . 53

5 Conclusion and future work **57**

Acknowledgements **59**

Bibliography **60**

List of works **67**

List of Tables

3.1 Effectiveness of using Mask-CTC in pre-training of TA-E/D. Existing and enhanced TA-E/D models were compared in terms of word error rates (%) along with encoder latency (ms) on WSJ dataset. 28
3.2 Effectiveness of using Mask-CTC in pre-training of streaming T-T. Existing and enhanced streaming T-T models were compared in terms of word error rates (%) along with encoder latency (ms) on WSJ dataset. 30
3.3 Effectiveness of using Mask-CTC in pre-training of CBS-E/D evaluated on WSJ dataset. 32
3.4 Effectiveness of using Mask-CTC in pre-training of CBS-E/D evaluated on the test set of the TED2 dataset. 32

4.1 Experimental results of multi-look-ahead CBS-T on WSJ dataset. 46
4.2 Experimental results on TED2 dataset with a frame rate of 40ms. 49
4.3 Experimental results on TED2 dataset with a frame rate of 33ms. 49
4.4 Improvements of Gangi model on TED2 dataset with a frame rate of 33ms. . . . 49
4.5 Experimental results of multi-look-ahead CBS-T on CSJ dataset. 51
4.6 Improvements of Gangi model on CSJ dataset. 51
4.7 Bifurcation model performance with various number of shared layers on TED2 dataset with a frame rate of 40ms and a block setting of 8-4-12. 52
4.8 Bifurcation model performance with various number of shared layers on TED2 dataset with a frame rate of 33ms and a block setting of 8-3-12. 53
4.9 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.2. 54
4.10 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.3. 54
4.11 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.4. 54
4.12 Detailed frame-wise delay results of multi-look-ahead CBS-T on CSJ in Table 4.5. 55
4.13 Detailed frame-wise delay results of multi-look-ahead CBS-T on CSJ in Table 4.6. 55
4.14 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.7. 55
4.15 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.8. 56

List of Figures

1.1 Comparisons of non-streaming and streaming ASR. Non-streaming ASR requires the entire speech sequence input to generate text output. While streaming ASR is able to recognize partial sequence (real-time speech input), where look-ahead frames (extra input frames ahead of the current analytic point) are usually used to ensure accuracy. 4

2.1 Overview of the trigger attention-based streaming encoder-decoder model. 10
2.2 Overview of the Transducer framework, consisting of an acoustic encoder, a label encoder and a joint network. 12
2.3 Chunk-wise attention mask in the streaming Transformer-Transducer model represented as a matrix. 13
2.4 The perception range of history frames and look-ahead frames in each self-attention layer of the acoustic encoder in streaming Transformer-Transducer. . . . 14
2.5 Overview of the contextual block streaming (CBS) encoder. In the CBS encoder, each input block consists of history frames (N_l), target frames (N_c) and look-ahead frames (N_r). During encoding, a contextual vector is inherited from the previous block in the previous encoder layer to leverage past contexts. Only features of the target frames are output at the final encoder layer. 16
2.6 Overview of the contextual block streaming Transducer (CBS-T). A CBS encoder is introduced to the Transducer framework as the acoustic encoder to perform streaming speech recognition based on block processing. 18

3.1 Mask-CTC framework with joint CTC and CMLM training objectives. 20
3.2 Two-pass inference in Mask-CTC framework with the utterance "the snow". In the first pass, the CTC output is generated based on the encoded acoustic features, where the posterior probability of each token is calculated. Tokens below the probability threshold are masked. In the second pass, the CMLM decoder is used to resolve the masked tokens with reliable outputs. 21
3.3 Procedure of building enhanced triggered attention-based streaming model (TA-E/D) for low latency and high accuracy streaming ASR: Stage 0 (left) for anticipatory feature learning, Stage 1 (center) for pre-training, and Stage 2 (right) for streaming model learning. 24
3.4 Procedure of Mask-CTC-based pre-training using Transformer-Transducer model. In stage 1, encoder is trained with Mask-CTC framework. In stage 2, Transformer-Transducer model is initialized with pre-trained encoder and fine-tuned with streaming objective. 25

3.5 Output token alignments of non-streaming and streaming Transformer-Transducer models. Color in background represents ground-truth alignment. Solid lines in top, middle, and bottom figures illustrate alignments obtained from non-stream ASR model, streaming model with Mask-CTC-based pre-training, and baseline streaming model, respectively. 34

4.1 Overview of the multi-look-ahead CBS encoder architecture for streaming ASR. The input block consists of N_l history frames, N_c target frames and N_r look-ahead frames. A primary encoder recognizes the target frames utilizing the look-ahead frames, while an auxiliary encoder recognizes the look-ahead frames without further look-ahead. The two encoders operate in parallel and have all the parameters shared. 37

4.2 Encoder structure of the Gangi model. In the Gangi model, the same set of parameters is used for all CBS encoders plotted. The look-ahead frames are recognized as the encoder shifts to the right. 39

4.3 Encoder structure of the Unity model. In the Unity model, the outputs of both target frames and look-ahead frames are generated with one encoding pass. . . . 40

4.4 Encoder structure of the Bifurcation model. Similar to the Unity model, the outputs of both target frames and look-ahead frames are generated with one encoding pass. In the Bifurcation model, the top encoder layers are separated. . . 42

Abstract

In recent years, the deep neural network has emerged as the core technology in automatic speech recognition (ASR). The advent of end-to-end ASR has revolutionized the field by consolidating the traditional components, namely acoustic, pronunciation, and language models, into a single deep neural network. This integration has played a crucial role in simplifying ASR development and enhancing its accuracy. While accuracy remains paramount, the real-time processing capabilities, known as streaming properties, are of utmost importance in applications such as conversational systems and online speech translations. Streaming ASR has the ability to transcribe speech synchronously by generating outputs based on partial speech sequences. However, a major challenge arises in streaming ASR due to the lack of future contexts, which are crucial for accurate predictions in the current recognition frame. This leads to performance degradation.

To address this challenge, we introduce two novel methods for constructing streaming ASR models that achieve both high accuracy and low latency. Our first method involves a Mask-CTC-based pre-training approach aimed at reducing latency while maintaining high accuracy. The Mask-CTC framework combines conditional masked language models (CMLM) and CTC objectives to train an encoder network. This enables the network to extract acoustic feature representations that capture long-term dependencies by anticipating future contexts. By conducting supervised pre-training with the Mask-CTC objective on the streaming ASR encoder, the encoder becomes proficient in capturing more long-term contexts within a limited look-ahead range, thereby reducing latency requirements. We evaluate the effectiveness of this method across various streaming ASR architectures.

The second method we propose is a multi-look-ahead model architecture for streaming ASR, which aims to mitigate latency increments associated with the use of look-ahead frames. Unlike the previous method that focuses on improving feature extraction within a limited look-ahead range, the multi-look-ahead architecture emphasizes recognizing the look-ahead frames without introducing further delay. This approach involves operating two encoders in parallel: a primary encoder that recognizes the target frames using look-ahead frames to achieve high accuracy, and

an auxiliary encoder that conducts early recognition on the look-ahead frames to avoid latency increments. We present various implementation methods for the multi-look-ahead architecture and demonstrate through experimental results on different datasets that this method significantly reduces recognition latency without compromising accuracy.

1 Introduction

1.1 Background

The development of deep neural networks has simplified automatic speech recognition (ASR) by integrating the various components of speech recognition systems into a single sequence-to-sequence network [1, 2, 3]. Attention-based encoder-decoder architecture [2, 3] is one of the major types of architectures for implementation, which contains an encoder module that extracts features and a decoder module that maps the encoder output to the text sequence using the attention mechanism. Such architecture could be implemented with various types of neural networks, including long short-term memory (LSTM) networks [3, 4, 5] and Transformer [6, 7, 8, 9, 10, 11]. Connectionist temporal classification (CTC) is another choice for realizing end-to-end ASR, which generates monotonic alignments between the input and output sequences with Markov assumption and dynamic programming [1]. The combined CTC-attention model is often trained to further improve the model training and inference processes [9, 12, 13]. Other than those, the Transducer [14] framework has been proposed for constructing end-to-end ASR models, incorporating an acoustic encoder, a label encoder and a joint network to achieve high speed in decoding. Transducer models have also achieved high accuracy in speech recognition by utilizing the Transformer neural network [15, 16]. Detailed formulations of each end-to-end ASR architecture are provided in Sect. 1.2.

In addition to recognition accuracy, the streaming property of ASR has garnered considerable research attention [17, 18, 19]. Streaming ASR processes input audio in real-time, enabling synchronous output generation and facilitating various ASR applications, including transcribing input speeches in conversational systems [20, 21, 22]. However, the trade-off between recognition accuracy and latency poses a challenge for streaming ASR, particularly in real-world applications, such as conversational systems, where accurately recognizing all input frames without any delay is crucial. As shown in Fig. 1.1, streaming ASR deals with partial input sequences, where future contexts are usually missing, and therefore, faces challenges in achieving high recognition

accuracy. The example shows that to recognize part of a word, such as the letter "n" within the word "snow", correctly, attending to the rest of the word (i.e., "ow") is of great importance. For such a reason, during streaming speech recognition, look-ahead frames are often used to provide the speech recognizer with more future contexts to achieve high accuracy, but since the look-ahead frames cannot be recognized promptly, it inevitably increases the latency and harms real-time performance. While attention masks can be applied to the look-ahead part to reduce latency in streaming ASR [23, 24, 25], employing them results in a notable reduction in accuracy compared to full context implementations that allow look-ahead [26]. Therefore, developing highly accurate streaming ASR models with low latency requirements has become an area of great interest.

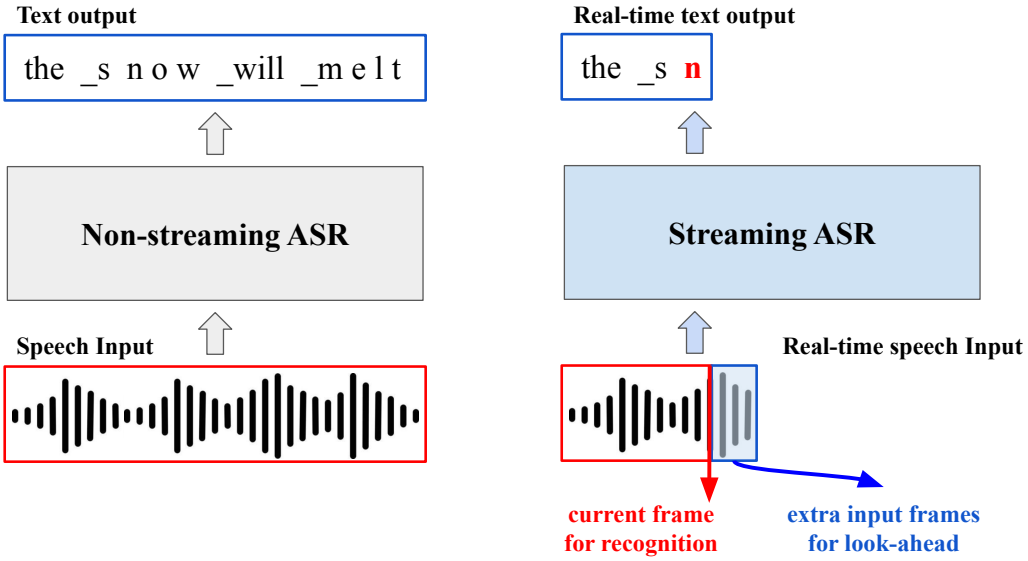

Figure 1.1 Comparisons of non-streaming and streaming ASR. Non-streaming ASR requires the entire speech sequence input to generate text output. While streaming ASR is able to recognize partial sequence (real-time speech input), where look-ahead frames (extra input frames ahead of the current analytic point) are usually used to ensure accuracy.

1.2 End-to-end speech recognition

End-to-end ASR aims to formulate the mapping between input sequence $X = (\mathbf{x}_t \in \mathbb{R}^D | t = 1, ..., T)$ and output sequence $Y = (y_l \in \mathcal{V} | l = 1, ..., L)$. Here, X is a T-length sequence of speech features such as MFCCs and log-Mel filterbanks, where $\mathbf{x}_t$ represents the D-dimensional feature vector at the t-th frame. Y is an L-length sequence of output tokens such as characters, sub-words

and words, where y_l represents the token at the l-th position in the vocabulary $\mathcal{V}$. Various types of architectures for implementing the end-to-end ASR are formulated below.

1.2.1 Attention-based encoder-decoder

The attention-based encoder-decoder architecture processes speech feature sequences through an encoder-decoder module and maps the input speech sequence X to the output text sequences Y. The encoder module, which works as an acoustic model, generates hidden states from the audio sequence and the decoder module finds the alignment between the hidden states and the text sequences by using the attention mechanism [3].

For each output token, the attention-based encoder-decoder architecture estimates the posterior probability in chain rule based on the input sequence X and the previous outputs, which is

$$p(Y|X) = \prod_{l=1}^{L} p(y_l|y_1, ..., y_{l-1}, X) \tag{1.1}$$

The posterior probability (r.h.s of the equation) is calculated with approximation to the attention-based objective function, formulated as follows

$$\mathbf{h}_t = \text{Encoder}(X) \tag{1.2}$$

$$a_{lt} = \text{ContextAttention}(\mathbf{q}_{l-1}, \mathbf{h}_t) \tag{1.3}$$

$$a_{lt} = \text{LocationAttention}(\{a_{lt}\}_{t-1}^{T}, \mathbf{q}_{l-1}, \mathbf{h}_t) \tag{1.4}$$

$$\mathbf{r}_l = \sum_{t=1}^{T} a_{lt}\mathbf{h}_t \tag{1.5}$$

$$p(y_l|y_1, ..., y_{l-1}, X) = \text{Decoder}(\mathbf{r}_l, \mathbf{q}_{l-1}, y_{l-1}) \tag{1.6}$$

Here $\mathbf{h}_t$ stands for the hidden states obtained from the encoder module. While $\mathbf{r}_l$ stands for the token-wise hidden vector derived from $\mathbf{h}_t$ and attention weights a_{lt}. The posterior probability is yielded from the decoder module with $\mathbf{r}_l, \mathbf{q}_{l-1}$ and previous output y_{l-1} [27].

1.2.2 Connectionist temporal classification (CTC)

Connectionist temporal classification (CTC) leverages dynamic programming to efficiently compute a strictly monotonic alignment between sequences [27]. Given the input sequence X,

CTC generates a character sequence of length T using the complete set of alphabets, including a 'blank symbol' $\langle b \rangle$ used for separating repeated letters.

Using this set of labels, CTC constructs augmented token sequences called paths. These paths can be converted into a single sequence by removing blank and repeated labels [28]. When presented with an input speech sequence, CTC introduces a letter sequence Z of length T, which contains labels from the set $\mathcal{V} \cup \langle b \rangle$ and allows for frame-wise predictions.

The formulations of CTC are shown as follows.

$$p(Y|X) = \sum_{Z} p(Y|Z, X)p(Z|X) \tag{1.7}$$

Since CTC assumes that the probabilities of the outputs at different timestamps are conditionally independent, the equation can be further simplified into

$$p(Y|X) = \sum_{Z} p(Y|Z)p(Z|X) \tag{1.8}$$

Applying Bayes' rule yields

$$p(Y|X) = \sum_{Z} p(z_t|z_{t-1}, Y)p(z_t|X)\frac{p(Y)}{p(Z)} \tag{1.9}$$

In the given context, $p(z_t|z_{t-1}, Y)$ represents the frame-wise posterior, $p(z_t|X)$ represents the transition probability, and $p(Y)$ represents a letter model. CTC efficiently computes the posterior distribution using the forward-backward algorithm implemented via dynamic programming based on this objective function [27, 28].

1.2.3 Transducer

A Transducer-based ASR model comprises three components: an acoustic encoder, a label encoder, and a joint network. The acoustic encoder generates acoustic features for the input speech, the label encoder handles the features of the previous output token sequence, and the joint network combines these features to predict the next output token.

Similar to CTC, the Transducer model captures token alignment for the input at each timestamp. However, the inclusion of the label encoder empowers the Transducer to make predictions based on previous token outputs, thereby strengthening the conditional dependency among the output tokens.

Given an input to a time index t, the output probability of the l-th token is formulated as follows:

$$\mathbf{h}_t^{\mathsf{AE}} = \text{AcousticEncoder}(\mathbf{X}_{1:t}), \tag{1.10}$$

$$\mathbf{h}_{l-1}^{\mathsf{LE}} = \text{LabelEncoder}(Y_{1:l-1}), \tag{1.11}$$

$$\mathbf{h} = \text{Tanh}(\text{Linear}(\mathbf{h}_t^{\mathsf{AE}}) + \text{Linear}(\mathbf{h}_{l-1}^{\mathsf{LE}})), \tag{1.12}$$

$$P(y_u|y_{1:l-1}, \mathbf{x}_{1:t}) = \text{SoftMax}(\mathbf{h}). \tag{1.13}$$

First, the acoustic encoder embeds the input sequence $\mathbf{x}_{1:t}$ into vector $\mathbf{h}_t^{\mathsf{AE}}$ (Eq. (1.10)). Meanwhile, the label encoder generates $\mathbf{h}_{l-1}^{\mathsf{LE}}$ from the previous output token sequence $y_{1:l-1}$ (Eq. (1.11)). The two outputs are then sent to the joint network, projected to the same dimension, and added up (Eq. (1.12)). Finally, the output probabilities against tokens in a vocabulary $\mathcal{V}$ are calculated based on the previous result (Eq. (1.13)).

1.3 Objective

The primary objective of this thesis is to develop streaming end-to-end ASR models that offer both high accuracy and low latency. Specifically, our goal is to address the trade-off between recognition accuracy and look-ahead delay increments. To achieve this objective, we propose two distinct approaches that target different aspects of reducing latency in streaming ASR.

Our first approach revolves around enhancing the encoder network to enable implicit modeling of long-term dependencies and capturing global contexts. This enhancement facilitates achieving high recognition accuracy even with limited look-ahead lengths. On the other hand, our second approach focuses on eliminating the recognition delay caused by the use of look-ahead by enabling early recognition of the look-ahead frames. We explore various implementation methods to effectively and efficiently realize this proposal.

We examine the proposed methods on various end-to-end ASR architectures mentioned above. By presenting these proposals and the experimental results in this thesis, we aim to make contributions to the application of streaming ASR across diverse industries.

1.4 Overview

The structure of this thesis is as follows: Chapter 2 provides an overview of various existing streaming end-to-end ASR models that serve as baselines for our work. These include the triggered attention-based encoder-decoder model (TA-E/D), streaming Transformer Transducer (T-T) with chunk-wise attention mask, contextual block streaming encoder-decoder (CBS-E/D), and contextual block streaming Transducer (CBS-T). This chapter focuses on the streaming processing techniques employed in these models and highlights their unique characteristics.

Chapter 3 introduces our first proposal: the Mask-CTC-based pre-training method for achieving high accuracy and low latency in streaming ASR. We begin by presenting the Mask-CTC framework and its advantages, followed by a detailed explanation of our proposed pre-training method. The chapter concludes with experimental results that validate the effectiveness of the method across various model types and datasets. Additionally, we discuss the latency reduction effects by measuring the output spike timing of the streaming ASR models.

In Chapter 4, we present our second proposal: multi-look-ahead streaming ASR. We introduce the general concept behind this proposal, which involves utilizing two encoders with different look-ahead settings operating in parallel to mitigate latency increments. Various implementation methods are discussed, and experimental results are presented, including recognition accuracy and real-time frame-wise latency analysis, to examine the impact of latency reduction.

Finally, Chapter 5 provides a comprehensive review of the thesis and discusses future applications based on the proposed methods.

2 Existing streaming end-to-end ASR

2.1 Introduction

The ability to process audio input in real-time and compute outputs synchronously makes the streaming property highly desirable for various real-world applications. However, incorporating the streaming property into end-to-end ASR models presents several challenges.

As discussed in Sect. 1.2, end-to-end ASR architectures include attention-based encoder-decoder, CTC, and Transducer models, with Transformer-based neural networks delivering state-of-the-art performance. Streaming processing poses difficulties for both the encoder and decoder modules in attention-based encoder-decoder networks. In the encoder module, the self-attention layers of the Transformer network require the entire input sequence to initiate computation. Similarly, in the decoder module, the Transformer layers depend on the complete sequence of acoustic features from the encoder to begin text generation.

Compared to the encoder-decoder architecture, achieving the streaming property is more feasible in the Transducer framework, as it naturally enables streaming decoding. However, when a Transformer neural network is utilized as the acoustic encoder in a Transducer model, it faces the same challenge of requiring global context due to the self-attention layers.

Considerable research has been dedicated to achieving streaming properties in both encoder-decoder and Transducer ASR models [16, 19, 29, 30, 31, 32, 33]. Dealing with the global context requirement of self-attention layers in the encoder involves two primary solutions: applying attention masks [24, 25, 16] and implementing block processing [31, 32]. For streaming decoding in the encoder-decoder model, various algorithms have been proposed, such as the trigger mechanism [19] and block boundary detection [33].

In this thesis, we have selected four existing streaming ASR models as our baselines. These include the Trigger attention-based streaming encoder-decoder [19], Streaming Transformer-Transducer [16], Contextual block streaming encoder-decoder [33] and Contextual block streaming Transducer [32, 34]. The chosen baseline models cover both encoder-decoder and Transducer

architectures, as well as attention-mask-based and block processing streaming methods. By competing with these baselines, we aim to provide a comprehensive and robust validation of the proposals presented in this thesis.

2.2 Trigger attention-based streaming encoder-decoder (TA-E/D)

The overview of the TA-E/D model is provided in Fig. 2.1. Based on the attention-based encoder-decoder architecture, the TA-E/D model utilizes Transformer neural network in both encoder and decoder modules. An additional CTC module is also included for hybrid CTC/attention training and joint decoding [27]. To enable streaming processing in the entire ASR model, attention masks are introduced to the encoder module to realize streaming feature extraction, whereas a trigger attention mechanism is applied to the decoder module to enable streaming decoding, with the CTC module serving as the trigger generator.

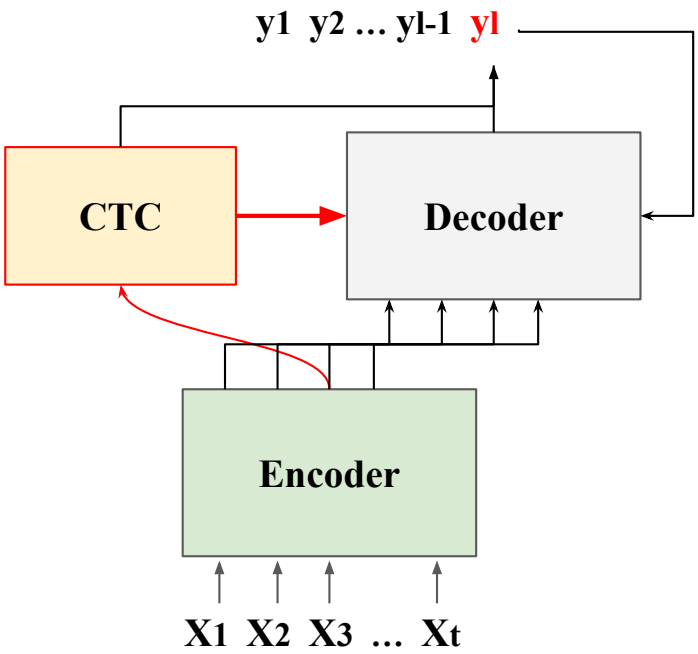

Figure 2.1 Overview of the trigger attention-based streaming encoder-decoder model.

2.2.1 Streaming encoder with attention masks

Following the conventional Transformer-based ASR system [8, 9], the encoder module contains two convolutional neural network (CNN) layers and an N stack of self-attention layers. The requirement of global attention in the self-attention layers hinders the ASR model's ability to perform stream processing. To address this limitation, attention masks [24, 25] are employed to regulate the past and future contexts attended by the self-attention mechanism. The conventional self-attention layer calculates attention scores using the scaled dot-product. In this work, we introduce a span mask W of fixed size, which is added to the result of the scaled dot-product prior to the softmax operation, as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}W\right)V, \tag{2.1}$$

where $Q \in \mathbb{R}^{n_q \times d_q}$, $K \in \mathbb{R}^{n_k \times d_k}$, and $V \in \mathbb{R}^{n_v \times d_v}$ denote query, key, and value matrices respectively, with n_* as sequence length and d_* as feature dimension. Since a controllable latency of the system is aimed for, restrictions are only set on the range of the future contexts, while attending to the past contexts is always allowed.

2.2.2 Trigger attention-based streaming decoder

For introducing the streaming property into the decoder module, a triggered attention mechanism was adopted in the decoding. The encoder transforms an input feature sequence into an encoded feature sequence, while the CTC module computes spike-like posteriors (triggers) based on the encoder outputs. These spike-like posteriors allow us to dynamically divide pause-delimited speech utterances into smaller sub-sequences according to the input.

At the beginning of each sub-sequence, the attention mechanism is triggered, and the encoded feature frames following the trigger event, along with some look-ahead frames, are decoded. This approach, denoted as triggered attention [19], enables frame-synchronous decoding.

To facilitate the triggered attention-based decoder, we need to establish alignment between the encoded feature sequence H and the output label sequence Y. This alignment information is crucial for training the decoder effectively. In practice, we utilize Viterbi decoding with a CTC-based trigger network to generate the alignment information. This alignment is then used

to condition the attention-based decoder solely on the past encoded frames and past outputs, ensuring accurate decoding. The formulation is shown as follows.

$$P_{\text{ta}}(Y|H) = \prod_{l=1}^{L} P(y_l|y_{1:l-1}, \mathbf{h}_{1:n_l}), \tag{2.2}$$

where n_l denotes the first occurring position of the symbol y_l in the CTC alignment. This formulation is specifically designed for streaming ASR and offers the advantage of not requiring explicit feature extraction on future inputs.

During the training phase, we initialize all modules with a pre-trained non-streaming Transformer model, and then fine-tune them to adapt to the streaming scenario. In particular, the encoder module is trained with attention spans as described in Sect. 2.2.1. During the run-time phase, joint CTC-attention decoding is performed, combining the CTC module and the attention mechanism to generate accurate transcriptions in real-time.

2.3 Streaming Transformer-Transducer (T-T)

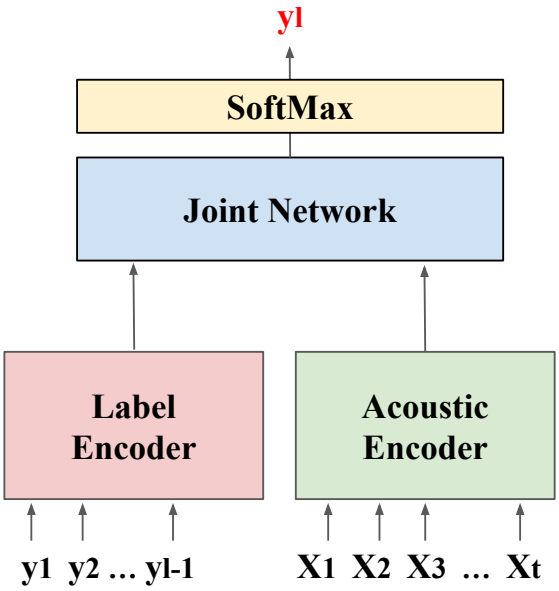

Figure 2.2 Overview of the Transducer framework, consisting of an acoustic encoder, a label encoder and a joint network.

Another type of streaming ASR model is the streaming Transformer-Transducer, which is based

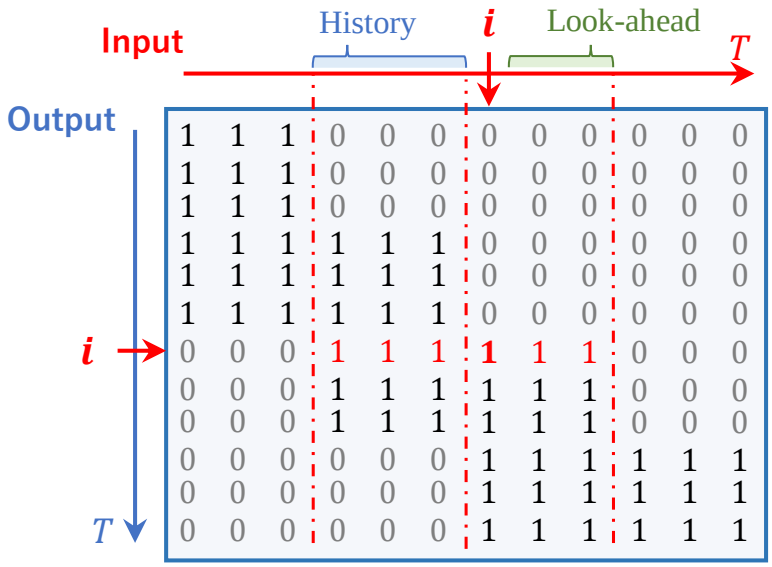

Figure 2.3 Chunk-wise attention mask in the streaming Transformer-Transducer model represented as a matrix.

on the Transducer framework, as shown in Fig. 2.2. Compared to encoder-decoder architecture, the Transducer framework possesses the advantage of natural streaming decoding, as the Transducer framework predicts the current symbol for each input frame based on only the past output tokens.

Various neural network types can be applied to implement the acoustic and label encoders [4, 14, 29]. In the work of [16], Transformer [6] is applied to the acoustic encoder to achieve high accuracy and LSTM [4] for the label encoder in consideration of the model size control. To enable streaming processing in the Transformer-based acoustic encoder, chunk-wise attention masks are applied to the self-attention layers, which will be introduced in details below.

2.3.1 Streaming acoustic encoder with chunk-wise attention masks

Similar to the attention mask in 2.2.1, the chunk-wise attention mask in the T-T model is directly applied to the attention calculation. Different from the former, chunk-wise attention mask is designed to group the input frames into chunks.

In Fig. 2.3, the chunk-wise attention mask is represented as a matrix used to compute the attention weights in the self-attention mechanism. In this matrix, frames with a value of 1 are used in the calculation of attention weights, while frames with a value of 0 are not utilized.

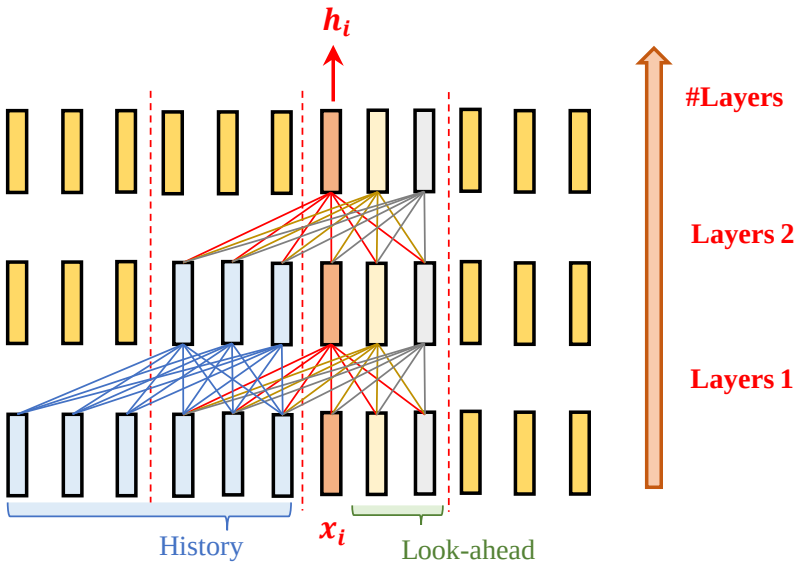

Figure 2.4 The perception range of history frames and look-ahead frames in each self-attention layer of the acoustic encoder in streaming Transformer-Transducer.

With the chunk-wise attention mask, the input sequence is divided into fixed-width chunks (sub-sequences), and attention weights are computed for each frame within the chunk. When two frames are within the same chunk, they can attend to each other. However, if two frames are in different chunks, regardless of the offset, the frame on the left cannot attend to the frame on the right. Thus, the look-ahead range is controlled by the width of the chunk.

When stacking Transformer layers, as shown in Fig. 2.4, the range of past frames increases with the number of layers, but the look-ahead is always limited to the chunk size.

When processing each input frame, the self-attention layer can only attend to the frames within the current chunk and the previous chunk, which limits the look-ahead range to the chunk size. When multiple Transformer layers are applied to the acoustic encoder, the reception range for past frames grows by the number of layers. In contrast, the look-ahead range is always limited to the chunk size, regardless of the layers.

2.4 Contextual block streaming encoder-decoder (CBS-E/D)

The previous streaming ASR models adopted attention masks in the self-attention layers to limit the range of accessible future contexts in model training and inference. Meanwhile, leverage

block processing is another approach to enabling streaming processing. By dividing the input sequence into small blocks of frames, streaming processing can be realized as the input block shifts over the entire sequence. In this case, attention masks are not required, as there are no more accessible frames outside the input block.

CBS-E/D [33] was proposed as a streaming ASR based on block-processing, following the attention-based encoder-decoder architecture. For streaming encoding, a contextual block streaming (CBS) encoder [32] was adopted. For decoding, a block boundary detection algorithm was proposed to enable block-wise synchronous beam search.

2.4.1 Contextual block streaming encoder

The structure of the CBS encoder is shown in Fig. 2.5, which utilizes block processing with a context inheritance mechanism to enable streaming feature extraction. During streaming speech recognition, the speech input is segmented into blocks containing history, target, and look-ahead frames with the numbers of N_l, N_c, and N_r. Streaming encoding can be conducted by processing each single block, however, the lack of global contexts may hurt the recognition performance of each local block. In order to capture long-term contexts from the past blocks, the context inheritance mechanism is introduced by utilizing an additional contextual embedding vector, which is computed in each encoder layer for the input block and passed on to the next layer to be used in the processing of the next input block. In such a way, the contexts in the past blocks can be used for the current block processing efficiently with low computational costs.

When an input block is passed on to the encoder layer, the CBS encoder processes the target frames for the output with future contexts provided by the look-ahead frames, as well as history contexts provided by history frames and the contextual embedding vector inherited from the previous block. Since only the target frames can be recognized in one encoding pass, latency is induced accordingly with the number of look-ahead frames. The block settings (i.e., N_l-N_c-N_r) of the CBS encoder can be easily adjusted to control the look-ahead latency of the streaming ASR. The block-wise streaming encoding can be formulated as follows:

$$Z_b, \mathbf{c}_b = \text{BlockEncoder}(X_b, \mathbf{c}_{b-1}) \tag{2.3}$$

where the b-th input block X_b with $|X_b| = N_l + N_c + N_r$ and a contextual vector from the previous

block $\mathbf{c}_{b-1}$ are processed to output the acoustic feature Z_b and current contextual vector $\mathbf{c}_b$.

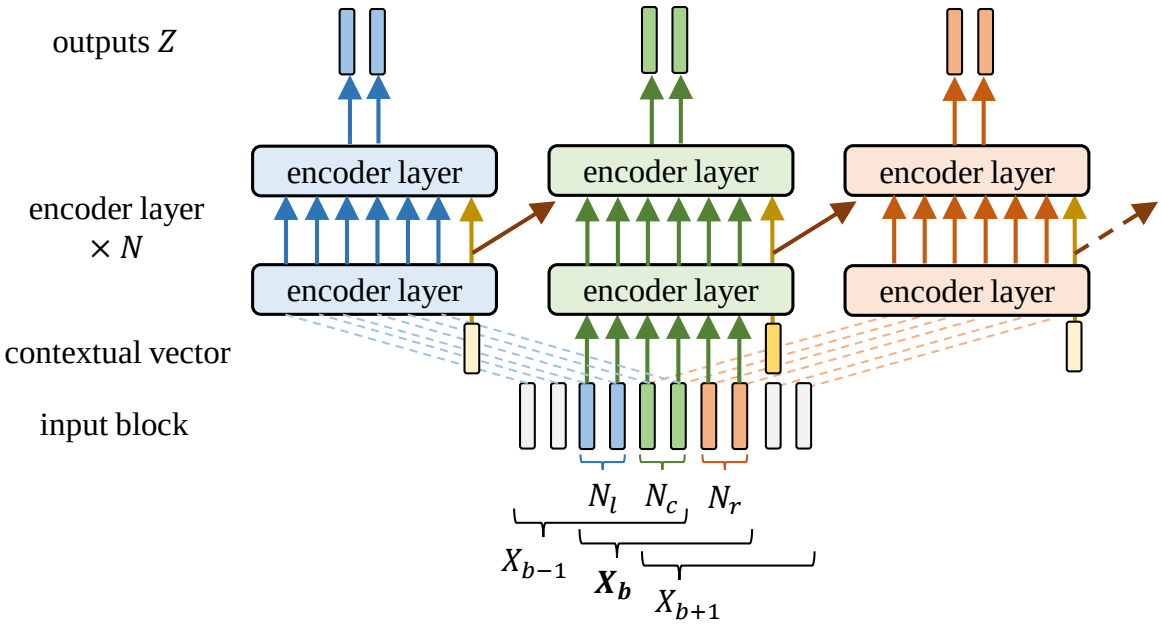

Figure 2.5 Overview of the contextual block streaming (CBS) encoder. In the CBS encoder, each input block consists of history frames (N_l), target frames (N_c) and look-ahead frames (N_r). During encoding, a contextual vector is inherited from the previous block in the previous encoder layer to leverage past contexts. Only features of the target frames are output at the final encoder layer.

2.4.2 Streaming decoding with block-wise synchronous beam search

Although the CBS encoder manages to conduct acoustic feature extraction in a streaming fashion, the ASR model still faces the challenge of enabling streaming decoding in the Transformer decoder. Similar to the trigger attention mechanism introduced in Sect. 2.2.2, a block boundary detection (BBD) algorithm is proposed to detect the boundary for decoding in a streaming fashion.

The objective of the BBD algorithm is to find the length of a hypothesis supported by the encoded block. During the decoding process, if a hypothesis exceeds the length that can be accommodated by the available encoded data, it becomes unreliable. In such situations, the decoder often encounters two typical errors [35], including prematurely predicting the "<eos>" token (representing sentence ending) and predicting a repeated token. The former is caused as

the attention mechanism reaches the end of inadequate encoder blocks, while the latter happens when the model attends to a position that has already been attended to. The BBD algorithm is designed based on such observations and the block boundary I_b is determined by comparing the reliability scores of each hypothesis on-the-fly.

Once the block boundary I_b is determined, block-wise synchronous beam search can be conducted to yield the outputs. The score of the partial hypothesis $y_{0:i}$ during streaming beam search decoding can be formulated as follows:

$$\alpha(y_{0:i}, Z_{1:B}) \approx \sum_{b=1}^{B} \sum_{j=I_{b-1}+1}^{I_b} \log p(y_i | y_{0:j-1}, Z_{1:b}). \tag{2.4}$$

where where y_0 is the start-of-sequence token. I_b denotes the index boundary of the b-th input block derived from the BBD algorithm.

2.5 Contextual block streaming Transducer (CBS-T)

Incorporating both CBS encoder and the BBD algorithm, the CBS-E/D model manages to enable both streaming encoding and decoding. However, the computational cost of the BBD algorithm largely impedes the real-world application of the ASR model. Based on such a shortcoming of the CBS-E/D model, a CBS-T model was proposed [34] utilizing the CBS encoder as the acoustic encoder of a Transducer framework to realize streaming feature extraction and decoding. Compared to the CBS-E/D model, the CBS-T model is well-suited for streaming ASR scenarios, as it achieves significant computational complexity reduction. Meanwhile, the graceful block processing implemented in the CBS encoder persists in the CBS-T model, supporting the achievement of high recognition accuracy with high-level acoustic feature extraction on each input block.

The structure of the CBS-T model is shown by Fig. 2.6, where the acoustic feature output possesses the length of the number of target frames (N_c). The Transducer framework further conducts frame-wise decoding with such acoustic features. The recognition latency of the CBS-T model is induced by the look-ahead frame setting of the input blocks.

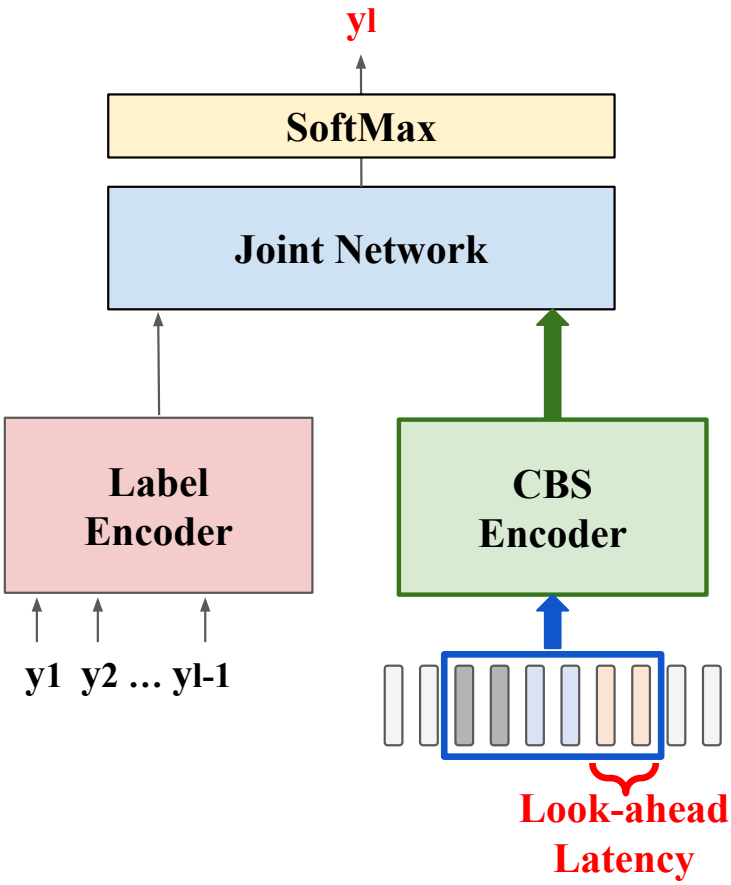

Figure 2.6 Overview of the contextual block streaming Transducer (CBS-T). A CBS encoder is introduced to the Transducer framework as the acoustic encoder to perform streaming speech recognition based on block processing.

3 Mask-CTC-based encoder pre-training for streaming ASR

3.1 Introduction

The aim of this chapter is to introduce our first proposed method for constructing streaming ASR models with low latency and high accuracy. As shown in the previous chapter, there are many types of existing streaming ASR models, with various architectures and streaming algorithms. However, one consistent characteristic among them is that to achieve high accuracy, the encoder needs to attend to look-ahead frames in order to acquire future contexts and support the predictions in the current recognition frames. For streaming ASR models like TA-E/D and T-T, the look-ahead is determined by the attention masks applied to the self-attention layers in the encoder. In the case of CBS-E/D and CBS-T, the look-ahead range is adjusted using the block setting parameter, denoted as N_l. Increasing the look-ahead range improves recognition accuracy but increases latency. Conversely, reducing the look-ahead range limits the global contexts available to the encoder, resulting in a degradation of accuracy.

To address this issue, a logical approach to reducing the latency requirements of streaming ASR models is to develop encoder modules that can anticipate the future, taking into account dependencies on long-term future contexts. If the encoder can model these dependencies during current processing, the number of required look-ahead frames can be minimized, enabling high accuracy with low latency settings. Many approaches have been proposed to construct streaming ASR models with encoders that consider long-term dependencies [36, 37]. In the work of [36], contrastive learning [38] was applied to bridge the gap between the acoustic features generated by streaming and non-streaming modes in order to reduce the latency requirements. In [37], dynamic latency settings were applied to the encoder training to enable the streaming ASR to consider long-term dependencies while operating in low latency mode. Similarly, our proposal is also focused on improving the feature representations generated by the encoder to achieve high accuracy with low latency settings.

This chapter is structured as follows. In Sect. 3.2, the Mask-CTC framework, which is the key

of this proposed approach, is introduced. In Sect. 3.3, the encoder pre-training method based on Mask-CTC is proposed. Experiment settings and results are summarized in Sect. 3.4 and Sect. 3.5 concludes the chapter.

3.2 Mask-CTC framework

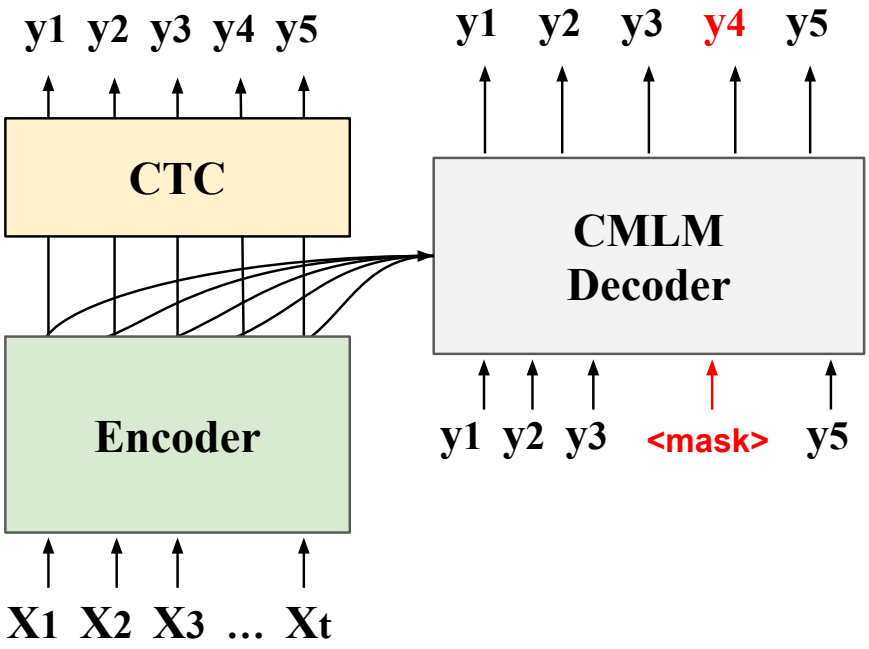

Figure 3.1 Mask-CTC framework with joint CTC and CMLM training objectives.

Mask-CTC [11, 39] is a framework for learning feature representations to achieve high accuracy and fast inference speed, and is the cornerstone of this proposal.

As shown in Fig. 3.1, the Mask-CTC framework follows the hybrid CTC/attention encoder-decoder architecture. The training of the Mask-CTC model contains two objectives: the CTC objective and the conditional masked language model (CMLM) objective [40, 41]. Specifically, during training, a portion of the ground truth data is randomly substituted with a mask token. This is done to train the Transformer decoder as a conditional masked language model (CMLM), which is responsible for resolving the masked tokens Y_{mask} based on observed tokens Y_{obs} and encoder sequence X.

The CMLM decoder is formulated as follows:

$$P_{\text{cmlm}}(Y_{\text{mask}}|Y_{\text{obs}}, X) = \prod_{y \in Y_{\text{mask}}} P_{\text{cmlm}}(y|Y_{\text{obs}}, X). \tag{3.1}$$

During inference, the Mask-CTC model leverages a two-pass approach, as shown in Fig. 3.2. In the first pass, the CTC output is used to generate an initial inference output from the encoded speech feature sequence. In this step, tokens that have lower confidence are masked and then refined in the second pass through mask prediction performed by the Transformer decoder.

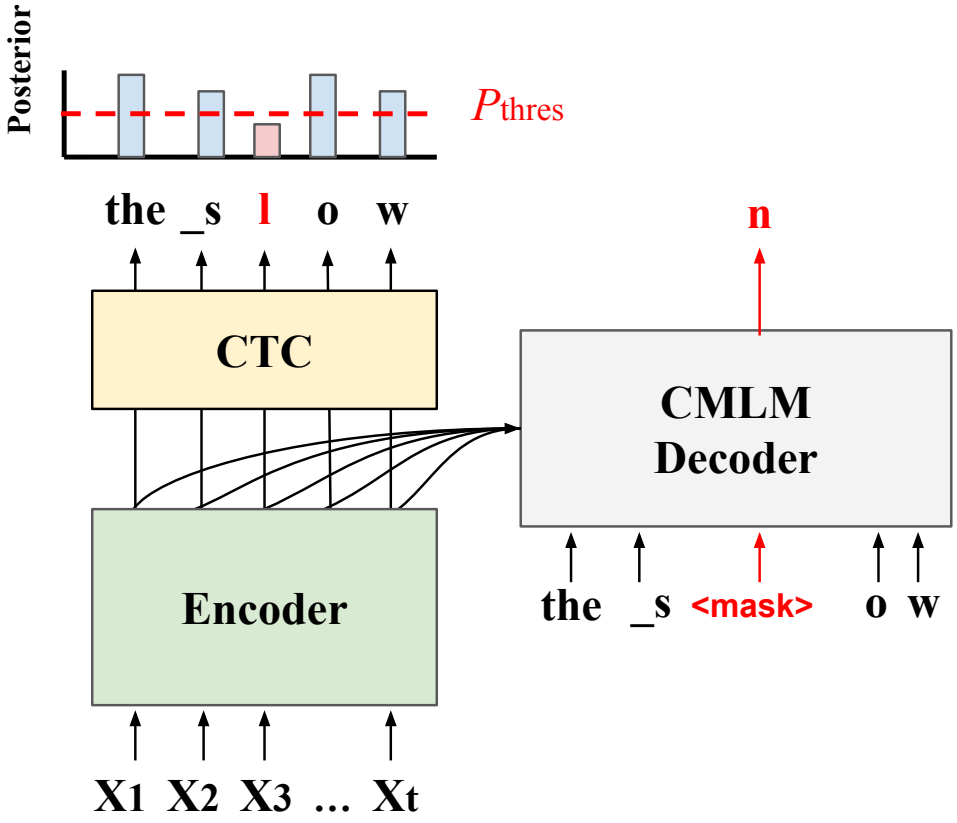

Figure 3.2 Two-pass inference in Mask-CTC framework with the utterance "the snow". In the first pass, the CTC output is generated based on the encoded acoustic features, where the posterior probability of each token is calculated. Tokens below the probability threshold are masked. In the second pass, the CMLM decoder is used to resolve the masked tokens with reliable outputs.

Using both CTC and CMLM objectives during training, the encoder-CTC network of the Mask-CTC model is improved by incorporating contextual information captured in the CMLM decoder. Specifically, through CMLM decoder objective, where the bidirectional dependencies

among output tokens are modelled to resolve masked tokens, the encoder is also reinforced to consider long-term dependencies while generating the acoustic feature sequence. As the encoder model is enhanced to extract features from future contexts more effectively, the CTC model also facilitates better alignment between the encoded feature sequence and the output label sequence. This improvement results in comparable accuracy in the generated output when compared to autoregressive decoding outcomes. When focusing solely on CTC prediction, the framework outperforms greedy CTC prediction in hybrid CTC-attention models and achieves similar results to autoregressive Transformer decoding.

3.3 Mask-CTC-based pre-training

Through Mask-CTC training, the encoder network is enhanced to conduct feature extraction that considers long-term dependencies of the output tokens, which leads to performance improvements in CTC inference with non-streaming settings. Such feature extraction characteristics can be of great importance in streaming settings as well. In order to introduce the enhanced feature extraction capability of the encoder network in Mask-CTC model into a streaming ASR model, we propose a Mask-CTC-based encoder pre-training method, containing two steps as follows:

- **Stage 1 (Feature representation learning):** The Mask-CTC model is pre-trained to obtain an encoder network that can consider long-term dependencies and anticipate future information.

- **Stage 2 (Streaming ASR training):** The pre-trained Mask-CTC model is exploited to initialize the streaming ASR models. For Transducer-based streaming ASR models, the acoustic encoders are initialized with the Mask-CTC encoder. For streaming ASR models with hybrid CTC/attention encoder-decoder architecture, both the Mask-CTC encoder and CTC networks are used to initialize the corresponding components.

To examine the general effectiveness of the proposed method, this 2-step training method is designed to suit different types of streaming ASR models introduced in Sect. 2, including attention-based encoder-decoder models: TA-E/D and CSB-E/D, as well as Transducer-based

models: streaming T-T. However, due to the specific training procedures as well as the characteristics of each model, some specifications have been made in the 2-step training method.

3.3.1 Mask-CTC-based pre-training for TA-E/D

Firstly, for the TA-E/D model, since the streaming ASR model itself requires a pre-training stage to be constructed, the 2-step method is expanded into the following 3-steps, as shown in Fig. 3.3:

- **Stage 0 (Feature representation learning)** aims to pre-train a Mask-CTC model to obtain an encoder-CTC network that can consider long-term dependencies and anticipate future information.
- **Stage 1 (Pre-training for reliable streaming ASR)** aims to acquire a pre-trained model that enables the efficient development of low-latency streaming ASR models in the subsequent stage. In this stage, a CTC-attention model is constructed, which utilizes autoregressive Transformer-based decoding. This approach deviates from the conventional method as it initializes the encoder and CTC modules with those from Mask-CTC to enhance feature representations.
- **Stage 2 (Streaming ASR model learning)** aims to construct the triggered attention-based streaming model using the reliable alignment information and initial parameters, all of which are transferred from the pre-trained CTC-attention model built in Stage 1.

It should be noted that the impacts of Mask-CTC are directly utilized during the pre-training phase (Stage 1) and indirectly transferred to the final streaming ASR model during its construction phase (Stage 2). Attempts were made to initialize Mask-CTC components directly in the creation of the streaming ASR model, bypassing Stage 1. However, this approach proved unsuitable for autoregressive decoding, resulting in inferior performance compared to the conventional model.

In Stage 1, a pre-training process is performed for the TA-E/D model using a CTC-attention model. This model consists of a Transformer encoder module, a CTC module, and an autoregressive Transformer decoder module. In the proposed method, the encoder and CTC modules

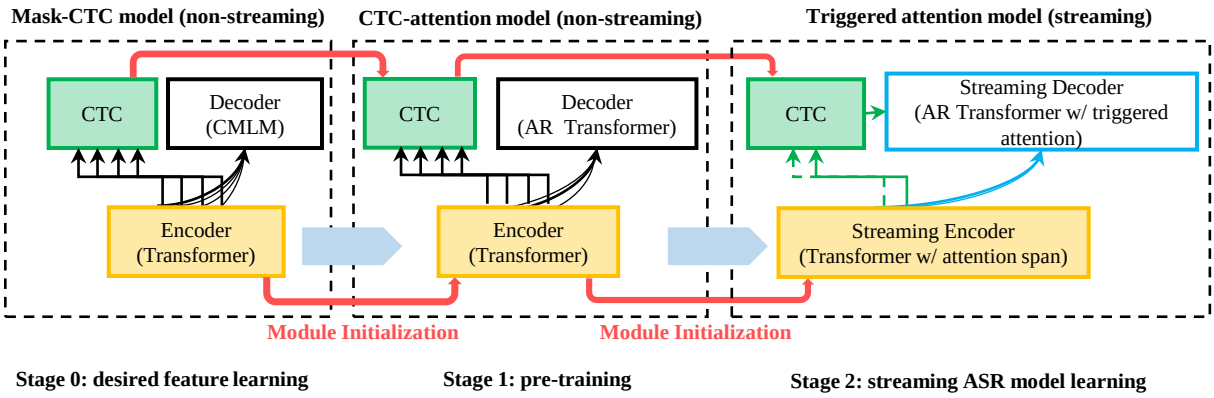

Figure 3.3 Procedure of building enhanced triggered attention-based streaming model (TA-E/D) for low latency and high accuracy streaming ASR: Stage 0 (left) for anticipatory feature learning, Stage 1 (center) for pre-training, and Stage 2 (right) for streaming model learning.

are initialized with the corresponding modules obtained from Mask-CTC in Stage 0. These initialized modules, along with the autoregressive Transformer decoder, are trained using the hybrid CTC-attention objective. The enhanced Transformer encoder, thanks to the initialization from Mask-CTC, is expected to produce more accurate CTC outputs compared to the Transformer encoder in the traditional CTC-attention model. Since Mask-CTC is originally designed for non-autoregressive and non-streaming ASR, Stage 1 remains in a non-streaming manner. The focus here is to obtain the initial values for the encoder and CTC modules that constitute the triggered attention-based streaming mechanism, as well as the reliable alignments required to train the decoder module.

During Stage 2, a triggered attention-based streaming model is constructed, which deviates from the conventional approach. The key difference is that an enhanced CTC-attention model, built in Stage 1, is utilized to generate alignment information and serves as a pre-trained model for the streaming model. As streaming ASR is designed for real-time audio processing, it requires a high capability for feature representation extraction, even when only a limited number of look-ahead frames are available to achieve low latency.

The pre-trained CTC-attention model incorporates the advantages of Mask-CTC in extracting better features from future contexts. Consequently, it is expected to generate reliable alignments even with low latency and provide improved guidance to the triggered attention mechanism. Moreover, training a streaming model using such a pre-trained model is anticipated to result

in more accurate recognition compared to existing streaming ASR models, particularly when operating at low latency.

3.3.2 Mask-CTC-based pre-training for streaming T-T

For streaming T-T, the Mask-CTC-based encoder pre-training procedure is shown in Fig. 3.4, where the pre-trained Mask-CTC encoder network is initialized in the acoustic encoder of the streaming T-T model.

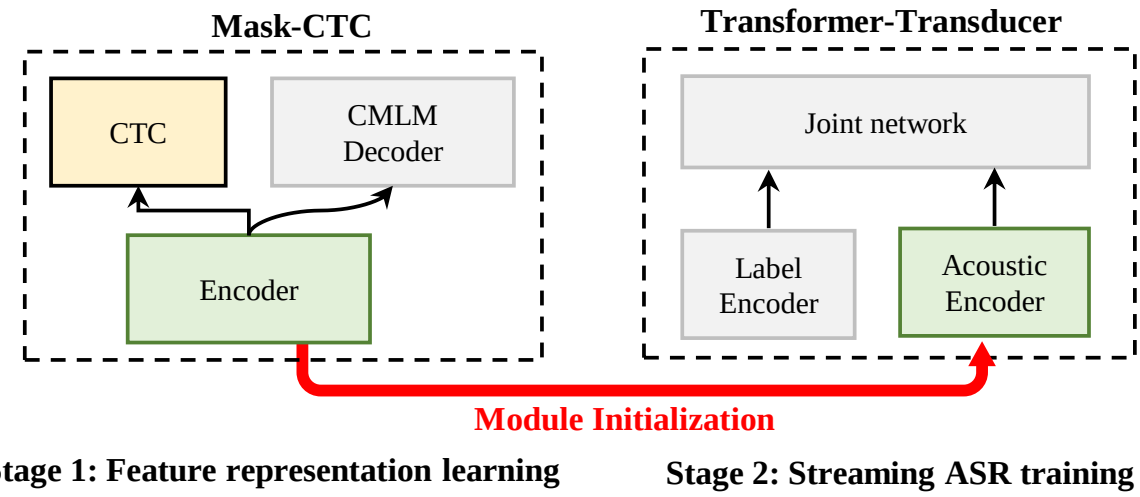

Figure 3.4 Procedure of Mask-CTC-based pre-training using Transformer-Transducer model. In stage 1, encoder is trained with Mask-CTC framework. In stage 2, Transformer-Transducer model is initialized with pre-trained encoder and fine-tuned with streaming objective.

The Transformer-based acoustic encoder in the streaming T-T is implemented with chunk-wise attention masks (Sect. 2.3.1) to limit the perception range in each self-attention layer. However, the attention mask is not applied to the self-attention layers in the Mask-CTC encoder. By setting the attention to global during pre-training phase, the Mask-CTC encoder is provided with more long-range contexts and achieves higher efficiency in the feature representation pre-training. When initialized in the streaming T-T model, the attention masks are applied and the encoder layer adapts to the new setting through the training stage of the streaming ASR model. In such a way, we expect to inherit the characteristics of Mask-CTC to a streaming ASR model to capture long-term contextual information in maximum and reduce the latency dependency.

3.3.3 Mask-CTC-based pre-training for CBS-E/D

The pre-training method for CBS-E/D differs from the method for TA-E/D as no pre-training is required for the training of a conventional CBS-E/D model. Therefore, the Mask-CTC-based pre-training method for CBS-E/D consists of only stage 1 (Feature representation learning) and stage 2 (Streaming ASR training). The Mask-CTC encoder network in stage 1 is set to full-context and both encoder and CTC networks from the Mask-CTC models are initialized in the CBS-E/D model with block settings applied in the encoder.

3.4 Speech recognition experiments

Speech recognition experiments were conducted using using ESPnet [42] and ESPnet2 [43], to examine the effectiveness of the proposed Mask-CTC-based pre-training method on each streaming ASR model type. The recognition accuracy was evaluated based on word error rates (WERs) without the usage of any external language model. Recognition latency was calculated as the delay caused by the look-ahead frames. Experimental results for three different types of streaming ASR models: TA-E/D, streaming T-T and CBS-E/D, are reported.

3.4.1 Experiment of TA-E/D

Datasets

In the experiments of TA-E/D, the models were trained using the 81-hour Wall Street Journal (WSJ) corpus [44], which includes English utterances from read news articles. We used the standard test sets *Eval92* and *Dev93* from the WSJ corpus for model evaluations. As input speech features, we used 80-dimensional log-Mel spectral energies plus three extra pitch features using [45]. We used characters (i.e., Latin alphabets) for tokenizing output texts.

Model settings

All the models were constructed using the same Transformer-based encoder-decoder architecture as in [9], where the encoder and decoder modules contained 12 and 6 self-attention layers, respectively. Each self-attention layer was implemented with 256 hidden units, 2048 feed-forward

inner dimensions, and 4 attention heads. For training non-streaming models, the learning rate was set to 10.0 and the training converged at around 50 through 100 epochs. For training streaming models, the learning rate was set to 1.0 and the number of training epochs was set to 120. The final models were obtained by averaging model parameters from the last five snapshots. For decoding with CTC only, we used the best path decoding algorithm [46], where a greedy output is obtained by suppressing repeated tokens and removing blank symbols. For joint CTC-triggered attention decoding, we utilized the frame-synchronous one-pass decoding algorithm [26], which integrated triggered attention decoding with the frame-synchronous prefix beam search algorithm from [47]. The CTC-attention weight for the joint decoding was set to 0.5 and the beam search was conducted with the beam size of 10.

Compared models

we investigated the effectiveness of the proposed triggered attention-based streaming ASR model, which was constructed based on Mask-CTC-based feature learning in Stage 2. The evaluation of streaming performance focused on two key aspects: recognition accuracy and latency. To facilitate this evaluation, we compared the proposed streaming ASR model with the following models:

- **TA-E/D**: Conventional triggered attention-based streaming ASR model trained with the CTC-attention objective [26].
- **Enhanced TA-E/D**: Enhanced triggered attention-based streaming model, whose feature representation learning implicitly utilizes Mask-CTC.

Experimental results

Table 3.1 shows the experimental results. This table lists the WERs (%) of the conventional and enhanced TA-E/D along with the encoder latency values (ms) computed on the WSJ dataset. This result demonstrates that the proposed enhancement of the TA-E/D model reduced the WERs of the existing model (e.g., 2.1% to 8.6%), regardless of the encoder latency values. In addition, the results demonstrate that the proposed pre-training can achieve low latency processing while

Table 3.1 Effectiveness of using Mask-CTC in pre-training of TA-E/D. Existing and enhanced TA-E/D models were compared in terms of word error rates (%) along with encoder latency (ms) on WSJ dataset.

		WER [%] (↓)	
Model	**Latency** [ms]	eval92	dev93
TA-E/D	160	28.2	34.5
	320	22.4	27.5
	480	18.9	24.1
	640	17.0	20.2
Enhanced TA-E/D	160	21.3	25.9
	320	15.5	19.5
	480	14.3	19.1
	640	**14.1**	**18.1**

maintaining recognition accuracy in the triggered attention-based streaming ASR.

The results presented in Table 3.1 demonstrate that, in both systems, the word error rates (WERs) increased as the encoder latency values decreased. It is important to note that the degradation in recognition accuracy was primarily due to the limited number of future contexts provided to the encoder for feature extraction.

In the case of the conventional streaming ASR model, the change in latency had a significant impact on recognition accuracy, resulting in performance degradation. However, when the proposed learning method was applied, this performance degradation was mitigated. Specifically, when the encoder latency value was reduced from 640 ms to 320 ms, the enhanced model maintained almost the same accuracy level, with only a slight degradation of approximately 1.4%. In contrast, the existing model experienced a 7.3% increase in WER.

Moreover, the proposed learning method achieved higher recognition accuracy with lower latency compared to the existing models. For instance, the performance of the proposed model with a latency of 320 ms (e.g., 15.5% for eval92 and 19.5% for dev93) outperformed the performance of the existing model with a latency of 640 ms (e.g., 17.0% for eval92 and 20.2% for dev93). This increased resilience towards latency reduction indicates that our proposal contributes to better feature extraction and alignment generation, even with a reduced amount of future context from

the input sequence.

However, it should be noted that the proposed method still exhibited a sharp decline in performance when the latency value decreased from 320 ms to 160 ms, suggesting the presence of a minimum latency requirement below which performance is significantly compromised.

3.4.2 Experiment of streaming T-T

Datasets

The experiments of streaming T-T adopted the same dataset: the 81-hour WSJ corpus, as in the previous subsection. We used SentencePiece [48] to construct a 80 sub-word vocabulary.

Model settings

For the streaming T-T model, the acoustic encoder was implemented with 12 Transformer encoder layers and a single LSTM layer for the label encoder. Each self-attention layer was implemented with 256 hidden units, 2048 feed-forward inner dimensions, and 4 attention heads. For streaming feature extraction, a chunk-wise attention mask was implemented and applied to the encoder layers as in [16]. The latency value was calculated as the product of the maximum look-ahead range (i.e., chunk size $-$ 1) and a frame rate of 40ms. All the models were trained by 150 epochs, and the final models were obtained by averaging the snapshots of the 10 epochs of the minimal loss. For Transducer decoding, a beam search was conducted with the beam size of 10.

Compared models

We compared the streaming ASR models including **Streaming T-T**, an existing streaming Transformer-Transducer model [16], and **Enhanced streaming T-T**, an enhanced streaming Transformer-Transducer, whose acoustic encoder was initialized from a pre-trained Mask-CTC model with an identical encoder architecture. Various latency settings were carried out in the experiment, including a non-streaming (latency is ∞) T-T model, serving as the upper-bound of this experiment.

Table 3.2 Effectiveness of using Mask-CTC in pre-training of streaming T-T. Existing and enhanced streaming T-T models were compared in terms of word error rates (%) along with encoder latency (ms) on WSJ dataset.

		WER [%] (↓)	
Model	**Latency** [ms]	eval92	dev93
Streaming T-T	120	19.5	23.3
	160	16.8	20.9
	200	**15.1**	**18.9**
	∞	14.7	17.3
Enhanced streaming T-T	120	16.6	20.8
	160	15.0	19.0
	200	**14.8**	**18.5**

Experimental results

The experimental results of streaming T-T are summarized in Table 3.2. Non-streaming T-T was used as upper bounds in the experiments.

Results in Table 3.2 shows that for streaming T-T, the enhanced models outperformed the baseline models by achieving lower WERs under all latency settings, suggesting the accuracy enhancements introduced by the Mask-CTC-based pre-training method. Furthermore, 40ms latency reduction was reached for streaming T-T, while achieving better or equal recognition accuracy than the baseline models. For instance, the enhanced streaming T-T with 120ms latency achieved lower WERs (16.6% for eval92 and 20.8% for dev93) than the WERs of the baseline with 160ms latency (16.8% for eval92 and 20.9% for dev93). The enhanced model with 200ms even achieved comparable accuracy to the non-streaming upper bound for eval92.

3.4.3 Experiment of CBS-E/D

Datasets

In the experiment of CBS-E/D, the models were trained and evaluated using both the WSJ dataset, and the TED-LIUM2 (TED2) [49] dataset, which contains 207h English spontaneous speech. For the output tokens, we used SentencePiece [48] to construct a 80 subword vocabulary

for WSJ and a 500 subword vocabulary for TED2, respectively. For robust model training, we applied SpecAugment [50] to the input data.

Model settings

For WSJ experiments, the CBS-E/D model consisted of 6 Conformer encoder layers [7] and 6 Transformer decoder layers. The input block settings followed N_l as 8, N_c as 4, and N_r varying from 0 to 6. The latency for CBS-E/D was calculated as the product of the maximum look-ahead range in the block (i.e., $N_c + N_r - 1$) and a frame rate of 40ms. For TED2 experiments, the CBS-E/D model consisted of 12 Conformer encoder layers and 6 Transformer decoder layers. The N_r was set to 6. For the pre-trained Mask-CTC model, the encoder was constructed with the identical setting as the target streaming model. The CMLM decoder was built with 6 Transformer decoder layers.

All the models were trained by 150 epochs, and the final models were obtained by averaging the snapshots of the 10 epochs of the best accuracy. For decoding, a beam search was conducted with a beam size of 10 for all.

Compared models

We compared the streaming ASR models including **CBS-E/D**, an existing contextual block streaming encoder-decoder model [33], and **Enhanced CBS-E/D**, whose encoder and CTC modules were initialized from a pre-trained Mask-CTC model with an identical encoder architecture.

Experimental results

The experimental results of CBS-E/D are summarized in Table 3.3 and Table 3.4 for WSJ and TED2 datasets, respectively. CBS-E/D models with 1240ms latency were used as upper bounds in the experiments.

The results on WSJ show that for CBS-ASR, the enhanced models achieved lower WERs than the baselines under all latency settings, confirming the effectiveness of the Mask-CTC-based pre-training method. The enhanced CBS-ASR model with 280ms latency managed to achieve the same level of accuracy as the baseline model with 360ms latency, suggesting an 80ms latency

Table 3.3 Effectiveness of using Mask-CTC in pre-training of CBS-E/D evaluated on WSJ dataset.

		WER [%] (↓)	
Model	**Latency** [ms]	eval92	dev93
CBS-E/D	200	14.4	18.1
	280	13.2	16.2
	360	**12.9**	**16.1**
	1240	11.2	14.2
Enhanced CBS-E/D	200	13.5	17.2
	280	12.9	**16.0**
	360	**12.2**	16.1

Table 3.4 Effectiveness of using Mask-CTC in pre-training of CBS-E/D evaluated on the test set of the TED2 dataset.

Model	**Latency** [ms]	**WER**[%] (↓)
CBS-E/D	280	11.3
	1240	9.8
Enhanced CBS-E/D	280	11.1

reduction. Such results demonstrated that our method contributed to the construction of streaming ASR models with low latency and high accuracy.

For TED2 dataset, the enhanced CBS-ASR model also achieved 0.2 percentage point of WER reduction compared to the baseline model, which proves the general effectiveness of the proposed method regardless of the dataset.

3.4.4 Analysis of output token alignments

As an additional experiment, we also examined the effect of the Mask-CTC-based pre-training method by studying the output token spikes of some baseline and enhanced streaming ASR models.

According to the findings of [51], the streaming model aims to shift token boundaries towards

the future side to gather more contextual information. However, this approach causes a delay in the occurrence of posterior probability spikes for output tokens compared to non-streaming models. On the other hand, if the encoder network learns to anticipate future information and incorporates it into the feature representations, the output tokens can be confirmed earlier, potentially resolving the issue of token boundary shifting in certain cases. To investigate this further, we conducted measurements to compare the delay in spike occurrences between streaming models and alignments obtained from a non-streaming model. We anticipate that the use of the Mask-CTC-based pre-training method will help reduce this delay.

We performed measurements on the dev93 validation set of WSJ. Specifically, we evaluated the baseline and enhanced models using a latency setting of 200ms for the streaming T-T architecture. To assess their performance, we compared the alignments generated by these models with those obtained from a non-streaming T-T model. The alignments were derived from the output of the joint network. Additionally, for CBS-E/D, we set the latency to 200ms and compared the token boundaries of the output between the baseline and enhanced models. The ASR alignments were obtained from the CTC predictions of CBS-E/D in the same manner as [51] and the reference alignments were obtained with the Montreal Forced Aligner [52].

Figure 3.5 illustrates one example of output token alignments given by streaming T-T. Here, the color in the background represents the reference alignment to the speech input. The non-streaming ASR (top) managed to predict accurate token alignments. However, the baseline streaming ASR (bottom) showed a significant delay in the alignments, indicating token boundary shifting due to the lack of contexts. Meanwhile, our enhanced streaming ASR (middle), with a Mask-CTC-based pre-trained encoder network, largely improved the alignments of the streaming ASR. We calculated the average output delay reduction across the dev93 validation set for both streaming T-T and CBS-E/D. For streaming T-T, the spike output delay was reduced by 44ms, and for CBS-E/D, 46ms. Such results help us to understand the knowledge learned from the Mask-CTC-based pre-training method and the reason for the latency reduction capability.

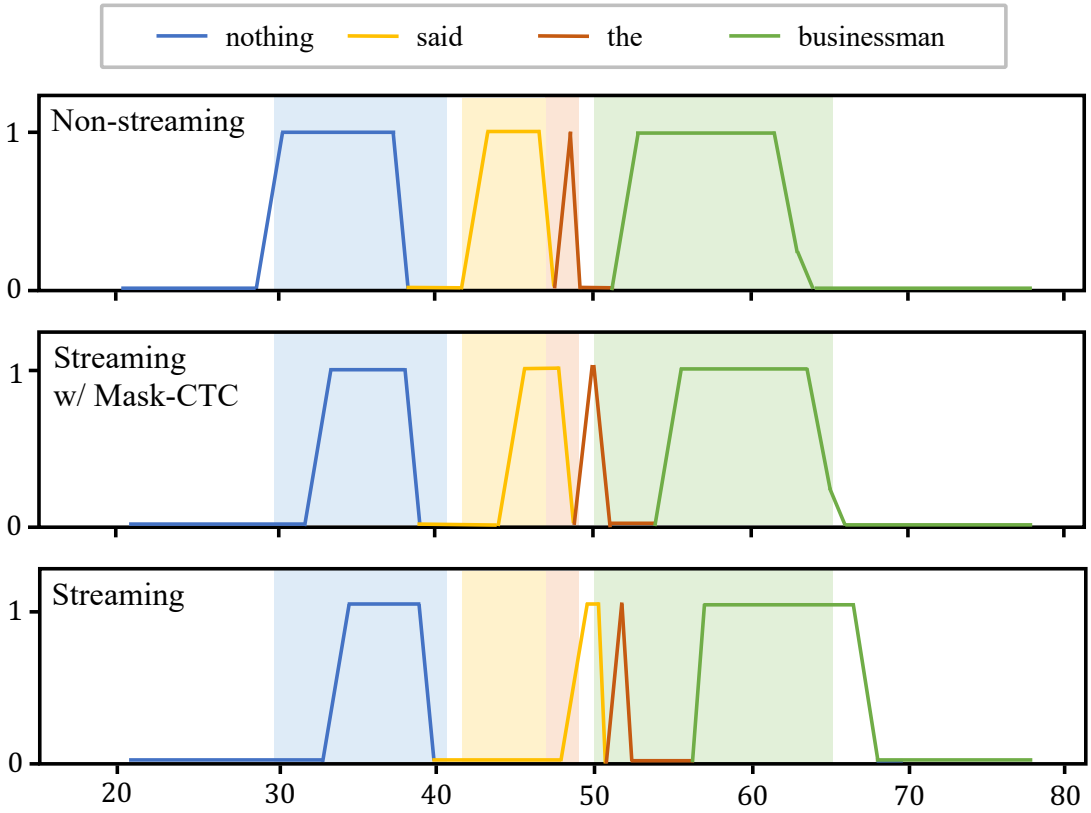

Figure 3.5 Output token alignments of non-streaming and streaming Transformer-Transducer models. Color in background represents ground-truth alignment. Solid lines in top, middle, and bottom figures illustrate alignments obtained from non-stream ASR model, streaming model with Mask-CTC-based pre-training, and baseline streaming model, respectively.

3.5 Summary

In this chapter, we presented the Mask-CTC-based encoder pre-training method, aimed at improving the encoder feature representation of streaming ASR models for achieving high accuracy with minimal latency. We conducted speech recognition experiments on various streaming ASR models, namely TA-E/D, streaming T-T, and CBS-E/D, using both the WSJ and TED2 datasets to assess the overall effectiveness of the proposed method.

The experimental results showcased the effectiveness of the Mask-CTC-based pre-training, regardless of the underlying model architecture. Moreover, through the analysis of output token alignments, we visually demonstrated the reduction in latency achieved by the proposed method.

4 Multi-look-ahead architecture for zero look-ahead streaming ASR

4.1 Introduction

In this chapter, we propose a streaming ASR model with zero look-ahead latency to meet the requirements for applications in conversational systems.

As human beings, we possess the remarkable ability to infer a speaker's intention during a conversation, even in the midst of their utterance, and prepare our subsequent actions accordingly. This capability enables us to respond with impeccable timing, at times without waiting for the completion of the speaker's utterance, resulting in a fluid and natural conversation. To imbue conversational systems with this same functionality, it is crucial for the system's speech recognizer to accurately transcribe the input speech instantaneously, without any delays.

By employing look-ahead techniques, streaming ASR can attain impressive levels of accuracy. However, the drawback of utilizing look-ahead is the resulting latency, which significantly increases the delay in speech recognition. The objective of this research is to design and develop a highly accurate speech recognizer that operates without the need for any look-ahead. This entails creating a system that can achieve exceptional accuracy in real-time, without introducing any additional delays.

One approach to achieve high accuracy without any additional delay is the multi-latency approach, which combines a short look-ahead ASR and a long look-ahead one [53, 54, 55, 56]. In [53], the beam search results of a low-latency encoder that utilizes a short look-ahead are corrected by utilizing a high-latency encoder, which employs a long look-ahead. In [55], a second-pass non-streaming recognition is conducted to refine the first-pass streaming outputs. One common characteristic among them is the adoption of a cascaded configuration, where high-latency, high-precision recognizers are utilized to compensate for the results obtained from low-latency recognizers.

In a similar vein, the system proposed in this thesis is a multi-latency Automatic Speech Recognition (ASR) system that integrates both a high-latency, high-accuracy encoder and a low-

latency encoder. However, what sets this system apart is its unique approach of operating both encoders in parallel and utilizing the CBS encoder [32, 33] as the foundational system. The CBS encoder has a strong compatibility with multi-latency architecture, further enhancing the system's capabilities. The proposed method incorporates a decoder that predominantly utilizes the output from the primary encoder, which incorporates look-ahead frames. Meanwhile, the look-ahead frames, which the primary encoder cannot generate output features for, is recognized by the auxiliary encoder, which operates without any look-ahead. Consequently, the entire system forms a recognizer that operates with utmost precision and efficiency, without causing look-ahead latency.

The chapter is structured as follows. In Sect. 4.2, the proposed multi-look-ahead streaming ASR architecture is introduced. Various implementation methods for the system are explained in Sect. 4.3. Sect. 4.4 reports the experimental results of the proposal and Sect. 4.5 concludes the chapter.

4.2 Multi-look-ahead streaming ASR

Fig. 4.1 shows the proposed multi-look-ahead architecture. As shown in Fig. 4.1, the proposed streaming ASR system features the simultaneous operation of two encoders. The primary encoder captures precise recognition results for the target frames by utilizing look-ahead frames. Meanwhile, the auxiliary encoder focuses specifically on recognizing the look-ahead portion of the primary encoder, employing a zero look-ahead approach to prevent any increase in latency. The proposed multi-look-ahead architecture is based on the CBS encoder. Therefore, our proposal can be applied to both CBS-E/D and CBS-T streaming ASR models. However, considering the real-time performance of the streaming ASR in application, we adopted the CBS-T model in our experiments, which outperforms CBS-E/D in terms of computational speed.

Algorithm 1 demonstrate the working system of the proposed method, where $\langle/s\rangle$ denotes the end-of-utterance (EoU) speech event. In this scenario, the proposed system is expected to detect the EoU output without delay. The system contains two CBS encoders with the same block size, where the primary encoder is configured with N_l history frames, N_c target frames, and N_r look-ahead frames, and the auxiliary encoder is configured with $N_l + N_c$ history frames, N_r

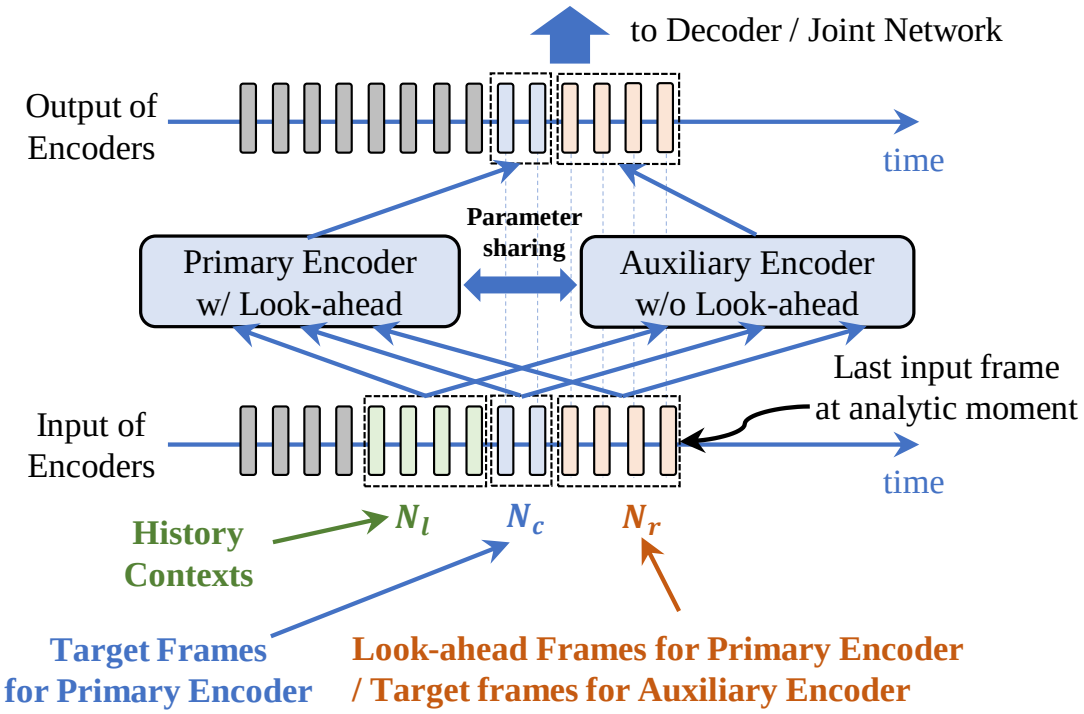

Figure 4.1 Overview of the multi-look-ahead CBS encoder architecture for streaming ASR. The input block consists of N_l history frames, N_c target frames and N_r look-ahead frames. A primary encoder recognizes the target frames utilizing the look-ahead frames, while an auxiliary encoder recognizes the look-ahead frames without further look-ahead. The two encoders operate in parallel and have all the parameters shared.

target frames and 0 look-ahead frames. When an input block is passed on to the streaming ASR, the primary encoder processes the target frames of the input block and outputs acoustic feature $\mathbf{z}_c$. Simultaneously, the auxiliary encoder processes the look-ahead frames of the input block and yields acoustic feature $\mathbf{z}_r$. Beam search is first conducted with feature $\mathbf{z}_c$ and the previous hypothesis $\mathbf{y}$ to extend $\mathbf{y}$ with the recognition results of the target frames in this block. Based on the extended hypothesis, beam search is further conducted with feature $\mathbf{z}_r$ to obtain a recognition result $\mathbf{y}_r$ of all input frames including the look-ahead portion. The speech recognition process will be terminated if $\mathbf{y}_r$ contains the EoU token $\langle/s\rangle$. Otherwise, the beam search results for the auxiliary encoder output are discarded, and streaming ASR moves on to the next input block.

In the case of single latency streaming ASR, when the frame containing the $\langle/s\rangle$ token is input to the speech recognizer, it will first be treated as a look-ahead frame and hence, cannot be recognized promptly and the EoU detection will be delayed. On the other hand, in the proposed

multiple latency streaming ASR, once the frame containing the $\langle/s\rangle$ token is passed on to the speech recognizer, it will first be recognized by the auxiliary encoder and EoU can be detected without delay.

Algorithm 1 Multiple latency streaming ASR

// Primary encoder block setting: (N_l, N_c, N_r)
// Auxiliary encoder block setting: $(N_l + N_c, N_r, 0)$
$T_B = N_l + N_c + N_r$
$\mathbf{y} \leftarrow \emptyset$
for $t = T_B$ to T by T_B **do**
    $\mathbf{z}_c$ = PrimaryEncoder($X[t - T_B, t]$)
    $\mathbf{z}_r$ = AuxiliaryEncoder($X[t - T_B, t]$)
    $\mathbf{y} \leftarrow BeamSearch(\mathbf{y}, \mathbf{z}_c)$
    $\mathbf{y}_r \leftarrow BeamSearch(\mathbf{y}, \mathbf{z}_r)$
    if $\langle/s\rangle$ in $\mathbf{y}_r$ **then** ▷ appropriate timing for ending
        break
    end if
end for
return $\mathbf{y}_r$ ▷ output final recognition result

4.3 Implementation methods of the multi-look-ahead architecture

To realize the proposed multi-look-ahead CBS encoder architecture in Sect. 4.2, three different implementation methods are proposed, namely Gangi model, Unity model and Bifurcation model.

4.3.1 Gangi model

Gangi ("雁木") is a term in Japanese architecture that refers to a specific type of decorative wooden ornamentation found at the ridge of a traditional hipped roof. Gangi consists of a paralleled series of identical wooden boards or tiles placed vertically along the ridge of the roof. Our first implementation method is named as the Gangi model due to its similarity to the Gangi structure, as plotted in Fig. 4.2.

In the Gangi model, the system contains N_r/N_c auxiliary encoders in parallel, each of which shares exactly the same structure and parameters as the primary encoder. The length of target

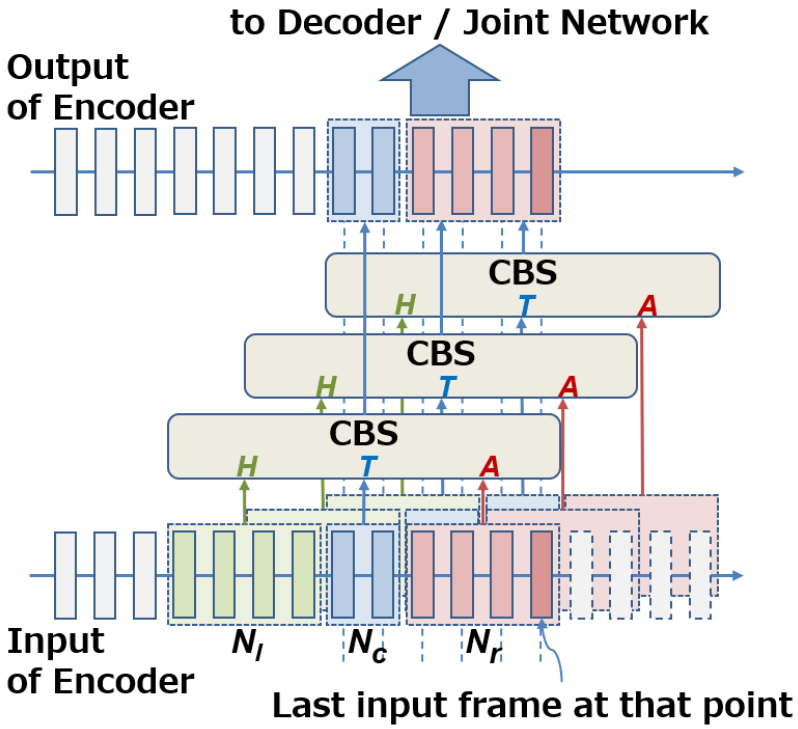

Figure 4.2 Encoder structure of the Gangi model. In the Gangi model, the same set of parameters is used for all CBS encoders plotted. The look-ahead frames are recognized as the encoder shifts to the right.

frames (N_c) in the input block of each encoder is consistent, while the length of look-ahead frames (N_l) varies for each encoder. To recognize the look-ahead frames, the auxiliary encoder shifts forward (to the right side) to place the look-ahead frames as its target frames. Specifically, the i-th auxiliary encoder uses the input of the primary encoder shifted forward by $i \times N_c$ frames. Consequently, the look-ahead frames for the i-th auxiliary encoder are shortened by $i \times N_c$ frames.

As shown in Fig. 4.2, when N_r/N_c equals 2, the system contains 2 auxiliary encoders. The primary encoder (CBS at bottom) extracts features from the N_c target frames using all N_l look-ahead frames. The first auxiliary encoder (CBS in middle) is shifted to the right by N_c frames and takes the first half of the look-ahead frames as its target frames using the second half as look-ahead frames. Blank frame padding is applied since the look-ahead frame is shortened. The second auxiliary encoder (CBS on top) is further shifted to the right and takes the second part of the look-ahead portion as its target frames, while no look-ahead is attended to (all blank frame

padding). In such a way, both target and look-ahead frames of the input block are recognized utilizing $N_r/N_c + 1$ encoding passes.

In the training phase, the primary encoder and all the auxiliary encoders are simultaneously trained by masking the last N_c, $2 \times N_c$, $\cdots$, $(N_r/N_c) \times N_c$ look-ahead frames with a certain probability. Therefore, all the encoders are expected to be sufficiently trained and capable of achieving high accuracy during inference. However, the increment of computational cost remains a limitation of the Gangi model.

4.3.2 Unity model

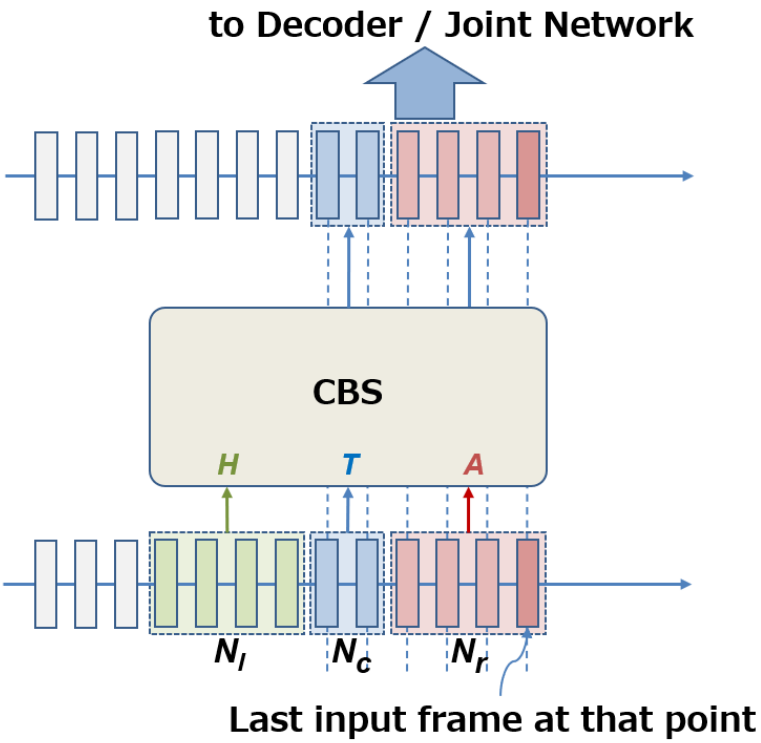

Figure 4.3 Encoder structure of the Unity model. In the Unity model, the outputs of both target frames and look-ahead frames are generated with one encoding pass.

The Unity model is proposed to remedy the high computational cost required in the Gangi model. As shown in Fig. 4.3, a single network is utilized to recognize both target frames and look-ahead frames in one encoding pass. With the block setting of N_l-N_c-N_r , the model is trained with a multi-task objective, where the main task is set to recognize N_c only and the auxiliary task

is to recognize N_c and N_r together. The loss function can be formulated as follows:

$$\mathcal{L}_{\text{unity}} = \mathcal{L}_{\text{main}} + \lambda \mathcal{L}_{\text{aux}}, \tag{4.1}$$

where the losses for the main task and for the auxiliary task are denoted by $\mathcal{L}_{\text{main}}$ and $\mathcal{L}_{\text{aux}}$, respectively. A weight λ is introduced to control the parameter updating rate with respect to the auxiliary task.

During inference, the CBS encoder is able to output acoustic features for both N_c and N_r in one encoding pass. The results for N_c are treated as the primary encoder outputs. The results for N_r are regarded as the auxiliary encoder outputs.

Compared to the Gangi model, whose computational cost grows as the look-ahead frame length extends, the computation in Unity model is always consistent and therefore, is expected to achieve higher inference speed. On the other hand, the training of a Unity model is more challenging than the Gangi model due to a large number of output frames. The difficulty in training increases as the length of N_r is extended in the block setting.

4.3.3 Bifurcation model

The Bifurcation model was proposed as an improvement to the Unity model. The training of the Unity model faces challenges when the two tasks possess large difference. Specifically, when the number of look-ahead frames N_r is significantly larger than the number of target frames N_c, it poses difficulty to generate accurate features for both tasks with all the parameters shared. Therefore, to improve the performance of the Unity model, partial parameter sharing mechanism was applied between the primary encoder and the auxiliary encoder in the Bifurcation model. As shown in Fig. 4.4, the two encoders share parameters in the first layers and use separated parameters in the last layers. We limit the parameter-sharing mechanism to only the first half of the layers (i.e., share the first 6 layers when 12 layers are in total). The remaining half of the layers, however, operate independently, ensuring a clear separation.

By adopting this approach, the common layers effectively extract features that are utilized by both the primary and auxiliary encoders. These shared features then proceed to the latter half of the layers, where they undergo separate processing: the primary encoding exclusively handles the

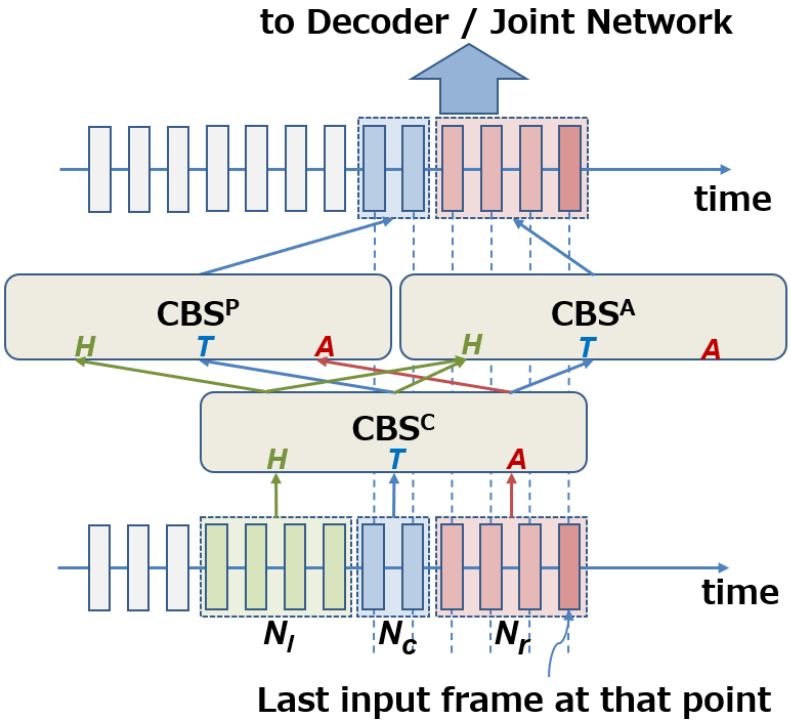

Figure 4.4 Encoder structure of the Bifurcation model. Similar to the Unity model, the outputs of both target frames and look-ahead frames are generated with one encoding pass. In the Bifurcation model, the top encoder layers are separated.

target frames, while the auxiliary encoding is dedicated to the look-ahead frames. This segregation enables the last layers to focus on generating accurate outputs for their respective tasks, preventing any potential decline in performance while leveraging the shared features obtained from the initial layers.

Similar to Unity model, during model training, we incorporate loss functions of both primary encoding and auxiliary encoding. The loss of primary encoding is used to update the common encoder layers, the primary-encoder-only layers as well as the label encoder and the joint network. Similarly, the loss of auxiliary encoding updates the common encoder layers, the auxiliary-encoder-only layers, and other Transducer components. We also introduce the hyper-parameter λ to the loss of auxiliary encoding such that the model training can be focused more on the primary encoding training, as the primary encoder contributes to most part of the output results.

4.4 Speech recognition experiments

To assess the accuracy and latency reduction performance of the proposed multi-look-ahead streaming ASR, we conducted speech recognition experiments utilizing ESPnet2 [42, 43]. For the evaluation of recognition accuracy, we used word error rates (WERs) and character error rates (CERs). Additionally, we introduced a novel **frame-wise delay** metric to evaluate the latency reduction performance of the multi-look-ahead models. For single look-ahead models (i.e., conventional CBS-T models), the frame-wise delay was composed of the look-ahead delay, block-wise encoding time, and block-wise decoding time, where the look-ahead delay was induced by the look-ahead frames and the target frames in the block setting ($(N_r + N_c/2) \times FrameRate$). For multi-look-ahead models, since the look-ahead portions were recognized without delay, the frame-wise delay was only induced by the target frames, the block-wise encoding time, and the block-wise decoding time. For all the models under the block setting of 8-3-12 with a frame rate of 33ms, the inference process was conducted using 32 cores of Intel(R) Xeon(R) Gold 6148 CPU operating at the speed of 2.40GHz. We recorded the frame-wise delay measurements under the same inference conditions and calculated the 50th percentile (P50) and 90th percentile (P90) results by averaging them over 10 repetitions. In the experimental results section, we report the overall frame-wise delay result, which is the sum of the look-ahead delay, block-wise encoding time, and block-wise decoding time. Detailed information on the composition of the frame-wise delay results can be found in the appendix section of this chapter.

All three implementation methods were examined and compared with each other. Additionally, we investigated to find the optimal sharing structure in the Bifurcation model by varying the number of shared encoder layers between the primary and auxiliary encoder.

4.4.1 Datasets

We conducted streaming ASR experiments on both English and Japanese datasets. For English datasets, we adopted both the 81-hour WSJ [44] dataset, and the 207-hour TED-LIUM2 (TED2) [49] dataset. For Japanese dataset, we used the 581-hour Corpus of Spontaneous Japanese (CSJ) [57] dataset.

For the output tokens, we used SentencePiece [48] to construct a 80 sub-word vocabulary for

WSJ and a 500 sub-word vocabulary for TED2, respectively. For CSJ dataset, we constructed a 500 sub-word vocabulary based on only *Katakana* characters in Japanese. For robust model training, we applied SpecAugment [50] to the input data. For the experiments of WSJ, the frame rate was set to 32ms, following the default settings of ESPNet2. Similarly, for the TED2 experiments, the frame rate was initially set to 40ms, following the recipe's settings. We also conducted experiments with a frame rate of 33ms for both TED2 and CSJ datasets.

For evaluation on the WSJ dataset, we used the averaged word error rates (WER) (averaged over *dev93* and *eval92* of WSJ). For TED2, we used the WER result on standard *test* set. For CSJ, we reported the averaged character error rates (CER) (averaged over *eval1*, *eval2* and *eval3* of CSJ).

4.4.2 Model settings

The CBS-T model was used in all experiments. In the experiments of the WSJ dataset, the CBS-T models were constructed with a CBS encoder with 6 Conformer [7] layers for the acoustic encoder and one layer of long short-term memory (LSTM) network [4] for the label encoder. For the TED2 dataset, the CBS encoders were constructed with 12 Conformer layers. For the CSJ dataset, CBS encoders with both 12 layers and 18 layers were examined. The label encoders were configured identically to the settings for the WSJ dataset. A CTC [46] objective was introduced to the training on the TED2 and CSJ dataset for better performance.

In the experiments of WSJ, three CBS blocks configurations were examined, with the history frame number N_l as 8, the future frame number N_r as 12 and the target frame number N_c varying among 3,4 and 6 (i.e., 8-3-12, 8-4-12 and 8-6-12 in the format of N_l-N_c-N_r).

For Japanese ASR models trained on CSJ dataset, in order to suit the requirements of their applications in the conversational system, where the block shift (i.e., target length) should be under 100ms, we set the target frame number N_c as 3 (i.e., 8-3-12 in the format of N_l-N_c-N_r). With the frame rate of 33ms, we managed to control the block shift to around 100ms.

For TED2 experiments, we used both 8-4-12 block setting with 40ms frame rate and 8-3-12 block setting with 33ms frame rate (identical to CSJ setting).

All the models were trained by 150 epochs for WSJ and 25 epochs for both TED2 and CSJ. The

final models were obtained by averaging the snapshots of the 10 epochs with minimal losses. An auxiliary CTC loss with a weight of 0.3 was added to the experiments of TED2 and CSJ for better training effectiveness following the ESPNet2 recipes. For the multi-look-ahead CBS-T models, the weight λ for auxiliary encoder training was set to 0.2. For decoding, modified adaptive expansion beam search [58] was conducted with a beam size of 10 for all.

4.4.3 Compared models

The following CBS-T models were evaluated in the experiments.

- **Single**: Conventional single-look-ahead CBS-T model. Always access look-ahead frames, which guarantees high recognition accuracy but induces large frame-wise delay.
- **Multi-look-ahead (Gangi)**: Proposed multi-look-ahead CBS-T model implemented with the Gangi method.
- **Multi-look-ahead (Unity)**: Proposed multi-look-ahead CBS-T model implemented with the Unity method.
- **Multi-look-ahead (Bif)**: Proposed multi-look-ahead CBS-T model implemented with the Bifurcation method. By default, the parameter sharing was conducted on the first half of the encoder layers. For example, when each CBS encoder consists of 12 layers, the first 6 layers are shared and the top 6 layers are separated, resulting in a total number of 18 layers in the encoder architecture.

  The increase in the total number of layers is unavoidable in the Bifurcation model. However, to maintain consistency in the overall layer count, the number of layers within each encoder can be decreased. For instance, if each encoder consists of 8 layers and the first 4 layers are shared, the Bifurcation model would have a total of 12 layers, aligning with the Single baseline model. Experiments were performed to assess the impact of the total number of layers.

Table 4.1 Experimental results of multi-look-ahead CBS-T on WSJ dataset.

			Frame-wise Delay	
Model	**Block**	**WER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Single	8-3-0	15.7	**82.7**	**86.0**
Single	8-3-12	13.4	469.8	473.6
Multi-look-ahead (Gangi)	8-3-12	13.6	174.8	185.1
Multi-look-ahead (Unity)	8-3-12	14.1	102.3	107.7
Multi-look-ahead (Bif)	8-3-12	**12.8**	118.9	122.3
Single	8-4-0	15.4	**86.5**	**89.7**
Single	8-4-12	13.4	468.4	472.5
Multi-look-ahead (Gangi)	8-4-12	13.4	193.5	207.1
Multi-look-ahead (Unity)	8-4-12	13.8	104.7	109.3
Multi-look-ahead (Bif)	8-4-12	**13.1**	120.6	124.7
Single	8-6-0	15.6	**87.4**	**94.0**
Single	8-6-12	**13.0**	467.4	508.8
Multi-look-ahead (Gangi)	8-6-12	**13.0**	125.0	140.2
Multi-look-ahead (Unity)	8-6-12	13.5	106.4	112.4
Multi-look-ahead (Bif)	8-6-12	**13.0**	123.2	127.7

4.4.4 Experimental results

Results on WSJ dataset

The experimental results on WSJ dataset are summarized in Table 4.1. The proposed method was examined under three different CBS block settings, including 8-3-12, 8-4-12 and 8-6-12, to show the general effectiveness. Results of Single baseline models with no look-ahead were provided as the lower bounds of accuracy. First, we compare our proposal with the baseline models. When comparing the proposed multi-look-ahead models to the single-look-ahead models, we can see that across all block settings, the Gangi model and the Bifurcation model managed to maintain the recognition accuracy, while the Unity model suffered from performance degradation. On the other hand, the Unity model achieved the lowest numbers on frame-wise delay among the proposed methods, with up to 460ms reduction when compared to the single-look-ahead model. Although with heavier computational cost, both the Gangi model and the Bifurcation

model managed to reduce the frame-wise delay largely, with up to 300ms reduction compared to the single-look-ahead baseline. Such results demonstrated the effectiveness of the proposed multi-look-ahead in terms of reducing look-ahead latency without any performance degradation.

Furthermore, we compare the performance between the Gangi model and the Bifurcation model. When comparing the recognition accuracy results of the Bifurcation model and the Gangi model, we can see that better performance was achieved by the Bifurcation model, with even higher recognition accuracy than the baseline models under 8-3-12 and 8-4-12 block settings. Such results showed that the Bifurcation model not only is capable of maintaining the accuracy of the single latency models but also provides varieties in the look-ahead settings during training that can improve the model training performances [37]. Meanwhile, the Bifurcation model achieved up to 80ms lower frame-wise delay compared to the Gangi models, due to relatively lower computational costs.

Results on TED2 dataset

The experimental results on TED2 dataset are summarized in Table 4.2, Table 4.3 and Table 4.4. As a larger dataset for spontaneous speech, experimental results on TED2 help examining the performance of our method in a case close to the applications in conversational systems. The results are majorly following two different settings: 8-4-12 CBS setting with 40ms frame rate and 8-3-12 CBS setting with 33ms frame rate, where results of the former are recorded in Table 4.2 and the latter in Table 4.3 and Table 4.4.

From the results in Table 4.2 we can see that our proposed methods performed similarly to the results on WSJ dataset, showing the general effectiveness of the proposal. The Gangi model and the Bifurcation model achieved the same recognition accuracy with the baseline model, while the Unity model suffered from accuracy degradation. The lowest frame-wise delay was achieved by the Unity model, followed by the Bifurcation model and the Gangi model.

Table 4.3 shows the experimental results when the block setting was adjusted to 8-3-12 and frame rate set to 33ms. In this table, a column of *Total layers* was added, indicating the total number of layers in the encoder. When the Single baseline model contains 12 layers in the encoder, both the Gangi model and the Unity model contain 12 layers as well. However, for the Bifurcation

model with default setting, the number of total layers become 18 layers, indicating the increments of total number of parameters. In this table, we also included the result of Bifurcation model with 12 total encoder layers (i.e., 8 layers for each encoder) for the performance comparison.

From Table 4.3 we can see that the Gangi model managed to maintain the same recognition accuracy as the baseline model. However, due to the large number of model shifting required (4 times in the case of 8-3-12), the frame-wise delay was largely affected and did not show significant reduction compared to the baseline model. On the other hand, the Unity model managed to keep low frame-wise delay but suffered from accuracy degradation with 0.8% higher WER result. For the Bifurcation model with 18 encoder layers in total, the recognition accuracy was maintained and the frame-wise delay was kept to a low level. However, for the Bifurcation model with 12 encoder layers in total, 0.4% WER increment occurred, indicating the insufficiency of layers in each encoder to maintain high accuracy.

To address the large computational cost in the Gangi model under the block setting of 8-3-12, we considered adopting the block setting of 8-5-10 and operating the model with 3 frame shifting. With the CBS block setting of 8-5-10, only two times of model shifting are required in the Gangi model, which largely reduces the computational costs. When operating the model with 8-5-10 block setting in a block shift of 3 frames, it is expected to achieve the same effect of 8-3-12 but with lower computational costs. The results of such setting are shown in Table 4.4, where we can see that with the CBS block setting of 8-5-10, the model even achieved higher recognition accuracy than the 8-3-12 model and maintained the same accuracy when operated under 3 frame shifting. Furthermore, the Gangi model with block setting of 8-5-10 achieved up to 150ms lower frame-wise delay compared to the Gangi model with block setting of 8-3-12. Such results showed that the issue of high computational cost in the Gangi model can be effectively remedied by adjusting the block settings and frame shifting.

Results on CSJ dataset

The experimental results on CSJ dataset are summarized in Table 4.5 and Table 4.6. As a comprehensive Japanese spontaneous speech dataset, the results on the CSJ dataset provide insights into the performance of a production streaming ASR model within a Japanese conversational

Table 4.2 Experimental results on TED2 dataset with a frame rate of 40ms.

				Frame-wise Delay	
Model	**Block**	**Total layers**	**WER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Single	8-4-12	12	**10.8**	594.3	602.8
Multi-look-ahead (Gangi)	8-4-12	12	**10.8**	214.6	224.0
Multi-look-ahead (Unity)	8-4-12	12	11.4	**124.2**	**138.2**
Multi-look-ahead (Bif)	8-4-12	18	**10.8**	149.3	177.7

Table 4.3 Experimental results on TED2 dataset with a frame rate of 33ms.

				Frame-wise Delay	
Model	**Block**	**Total layers**	**WER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Single	8-3-0	12	13.3	117.4	122.4
Single	8-3-12	12	**11.5**	517.6	523.1
Multi-look-ahead (Gangi)	8-3-12	12	**11.5**	318.3	335.9
Multi-look-ahead (Unity)	8-3-12	12	12.3	133.3	150.0
Multi-look-ahead (Bif)	8-3-12	18	**11.5**	181.1	199.8
Multi-look-ahead (Bif)*	8-3-12	12	11.9	163.1	176.5

system.

The upper part of Table 4.5 contains the experimental results with the identical settings as in Table 4.3, where the Single baseline model with 12 total encoder layers was compared with the Gangi model of 12 total encoder layers, the Unity model of 12 total encoder layers, and Bifurcation models with 18 and 12 total encoder layers, respectively. The Gangi model achieved the same recognition accuracy as the baseline model but did not manage to improve the frame-

Table 4.4 Improvements of Gangi model on TED2 dataset with a frame rate of 33ms.

				Frame-wise Delay	
Model	**Block**	**Frame shift**	**WER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Multi-look-ahead (Gangi)	8-3-12	3	11.5	318.3	335.9
Multi-look-ahead (Gangi)	8-5-10	5	**11.3**	175.3	248.2
Multi-look-ahead (Gangi)	8-5-10	3	**11.3**	179.8	250.2

wise delay significantly, due to the increasing computational costs. The Unity model showed severe accuracy degradation, with even higher CER results than the lower bound baseline model with no look-ahead. For the Bifurcation model with 18 encoder layers in total, the recognition accuracy was maintained and the frame-wise delay was kept to a low level. However, for the Bifurcation model with 12 encoder layers in total, 0.2% CER increment occurred, indicating the insufficiency of layers in each encoder to maintain high accuracy. Based on the results in the upper part of Table 4.5 and the results in Table 4.3, we can see that the proposed method performed consistently in both English and Japanese streaming ASR tasks.

We also conducted experiments with 18 encoder layers, as shown in the lower part of Table 4.5. Here we can see that for all the evaluated models, no improvements in CER were achieved when increasing the encoder layers from 12 to 18. The results indicate that the model's performance has reached saturation point when using 12 encoder layers. As a result, the Bifurcation model, with a total of 18 encoder layers, achieved the same performance as the baseline model, which also had 18 encoder layers. This demonstrates that the enhancements achieved by the Bifurcation model are not solely attributed to a higher number of parameters.

Similar to Table 4.4 in the TED2 experiments, we examined the improvements of the Gangi model by adjusting the CBS block setting to 8-5-10 and shifting the block by 3 frames. As shown in Table 4.6, with the block setting of 8-5-10 and shifting by 3 frames, the Gangi model achieved the same accuracy with the Gangi model under 8-3-12 block setting, while achieving lower frame-wise delay due to the reduction of computational costs.

In conclusion, our experiments on three datasets encompassing different speech styles and languages demonstrate the effectiveness of the proposed multi-look-ahead method in reducing recognition latency caused by the use of look-ahead frames, while maintaining the same accuracy as the baseline model. Among the three implementation methods tested, the Gangi model and the Bifurcation model exhibited superior performance. Notably, the Gangi model can be further enhanced with lower computational costs by adjusting the block settings. In the case of the Bifurcation model, it was found that increasing the total number of encoder layers was necessary for optimal performance. However, in these experiments, we only utilized half of the total layers in each encoder as shared layers. By incorporating more shared layers, the total number of layers

Table 4.5 Experimental results of multi-look-ahead CBS-T on CSJ dataset.

				Frame-wise Delay	
Model	**Block**	**Total layers**	**CER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Single	8-3-0	12	5.8	**111.5**	**121.0**
Single	8-3-12	12	**4.3**	513.5	523.7
Multi-look-ahead (Gangi)	8-3-12	12	**4.3**	321.8	342.0
Multi-look-ahead (Unity)	8-3-12	12	7.0	138.3	155.6
Multi-look-ahead (Bif)	8-3-12	18	**4.3**	181.5	203.6
Multi-look-ahead (Bif)*	8-3-12	12	4.5	159.1	177.3
Single	8-3-12	18	**4.3**	538.3	555.2
Multi-look-ahead (Gangi)	8-3-12	18	**4.3**	336.5	352.3
Multi-look-ahead (Unity)	8-3-12	18	7.0	183.2	205.4

Table 4.6 Improvements of Gangi model on CSJ dataset.

				Frame-wise Delay	
Model	**Block**	**Frame shift**	**WER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Multi-look-ahead (Gangi)	8-3-12	3	4.3	321.8	342.0
Multi-look-ahead (Gangi)	8-5-10	5	4.3	203.6	256.0
Multi-look-ahead (Gangi)	8-5-10	3	4.3	201.9	257.5

can be reduced. As a result, the next section presents additional experimental results that focus on exploring the impact of parameter sharing in the Bifurcation models.

4.4.5 Effect of parameter sharing

To investigate how the number of shared layers affects the Bifurcation model performance, we conducted experiments on the TED2 dataset using the Bifurcation model with different numbers of shared layers. Similar to the TED2 experiments in the previous section, the results were provided based on two different settings: CBS block setting of 8-4-12 with a frame rate of 40ms; and CBS block setting of 8-3-12 with a frame rate of 33ms, presented in Table 4.7 and Table 4.8, respectively. In each table, the experimental results of Bifurcation models with shared layer values of 12, 9, 6, 3, and 0 are reported. When the number of shared layers is 12, it is equivalent

Table 4.7 Bifurcation model performance with various number of shared layers on TED2 dataset with a frame rate of 40ms and a block setting of 8-4-12.

				Frame-wise Delay	
Model	**Shared layers**	**#params**	**WER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Multi-look-ahead (Bif)	12	33.72M	11.4	**124.2**	**138.2**
Multi-look-ahead (Bif)	9	41.42M	10.9	140.7	169.3
Multi-look-ahead (Bif)	6	49.13M	**10.8**	149.3	177.7
Multi-look-ahead (Bif)	3	56.84M	11.0	153.2	179.7
Multi-look-ahead (Bif)	0	64.55M	11.4	172.4	182.5

to the Gangi model. The Bifurcation models discussed in the previous experiments were set with the number of shared layers as 6. The total number of parameters in each model is shown in the column *#params*.

From the results we can see that decreasing the number of shared layers resulted in increased frame-wise delay due to the additional computational cost in separate layers of the bifurcate encoder structure. On the other hand, the WER did not improve linearly with the computational cost increments. Instead, in Table 4.7, we observed a constant improvement in accuracy as we reduced the number of shared layers from 12 to 6. Further reduction to 3 shared layers or no shared layers resulted in accuracy degradation, with a 0.2% and 0.6% WER increment compared to the result with 6 shared layers, respectively.

In Table 4.8, the best WER result was achieved by both the model with 9 shared layers and the model with 6 shared layers. Accuracy degradation happened when reducing the number of shared layer to 3 and 0, highlighting the importance of shared layers in maintaining consistency between the output features of the two encoders. Compared to the model with 6 shared layers, the model with 9 shared layers contained fewer parameters and achieved lower frame-wise delay while maintaining the same recognition accuracy. Therefore, the setting of 9 shared layers is optimal in this specific case.

Table 4.8 Bifurcation model performance with various number of shared layers on TED2 dataset with a frame rate of 33ms and a block setting of 8-3-12.

				Frame-wise Delay	
Model	**shared layers**	**#params**	**WER** [%](↓)	**P50** [ms](↓)	**P90** [ms](↓)
Multi-look-ahead (Bif)	12	33.76M	12.3	**133.3**	**150.0**
Multi-look-ahead (Bif)	9	41.49M	**11.5**	155.6	174.1
Multi-look-ahead (Bif)	6	49.21M	**11.5**	181.1	199.8
Multi-look-ahead (Bif)	3	56.93M	11.8	208.0	226.4
Multi-look-ahead (Bif)	0	64.65M	11.9	229.1	248.9

4.5 Summary

In this chapter, we introduce a novel multi-look-ahead architecture based on the CBS encoder, which effectively eliminates the look-ahead latency associated with using look-ahead frames in the input blocks. To realize this architecture, we present three implementation methods: Gangi, Unity, and Bifurcation.

Through experimentation on English and Japanese datasets, we demonstrate that our proposed multi-look-ahead streaming ASR system maintains high recognition accuracy without incurring any look-ahead latency. Among the three implementation methods, the Bifurcation model presented the best performance and highest suitability for real-world applications.

4.6 Appendix

In this section, the detailed results of the measured frame-wise delay for each experiment are reported. In each table, **TG** denotes delay caused by target input frames. **LH** denotes delay caused by look-ahead input frames. **Enc** presents the encoding processing time and **Dec** presents the decoding time using beam search.

Table 4.9 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.2.

		Frame-wise Delay									
		P50 [ms](↓)					**P90** [ms](↓)				
Model	**Block**	TG	LH	Enc	Dec	**Total**	TG	LH	Enc	Dec	**Total**
Single	8-4-12	80	480	18.2	16.1	594.3	80	480	21.2	21.6	602.8
MLA (Gangi)	8-4-12	80	0	111.3	23.3	214.6	80	0	113.4	30.6	224.0
MLA (Unity)	8-4-12	80	0	18.4	25.8	124.2	80	0	23.0	35.2	138.2
MLA (Bif)	8-4-12	80	0	44.2	25.1	149.3	80	0	64.7	33.0	177.7

Table 4.10 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.3.

		Frame-wise Delay									
		P50 [ms](↓)					**P90** [ms](↓)				
Model	**Block**	TG	LH	Enc	Dec	**Total**	TG	LH	Enc	Dec	**Total**
Single	8-3-0	49.5	0	52.0	15.9	117.4	49.5	0	49.5	23.4	122.4
Single	8-3-12	49.5	396	52.5	19.6	517.6	49.5	396	53.8	23.8	523.1
MLA (Gangi)	8-3-12	49.5	0	241.8	27.0	318.3	49.5	0	254.5	31.9	335.9
MLA (Unity)	8-3-12	49.5	0	58.5	25.3	133.3	49.5	0	66.7	33.8	150.0
MLA (Bif)	8-3-12	49.5	0	105.8	25.8	181.1	49.5	0	78.5	35.9	199.8
MLA (Bif)*	8-3-12	49.5	0	89.3	24.3	163.1	49.5	0	96.2	30.8	176.5

Table 4.11 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.4.

		Frame-wise Delay									
		P50 [ms](↓)					**P90** [ms](↓)				
Model	**Block**	TG	LH	Enc	Dec	**Total**	TG	LH	Enc	Dec	**Total**
MLA (Gangi)	8-3-12	49.5	0	241.8	27.0	318.3	49.5	0	254.5	31.9	335.9
MLA (Gangi)	8-5-10	49.5	0	101.0	24.8	175.3	49.5	0	166.9	31.8	248.2
MLA (Gangi) + 3-shift	8-5-10	49.5	0	106.3	24.0	179.8	49.5	0	169.2	31.5	250.2

Table 4.12 Detailed frame-wise delay results of multi-look-ahead CBS-T on CSJ in Table 4.5.

		Frame-wise Delay									
		P50 [ms](↓)					**P90** [ms](↓)				
Model	**Block**	TG	LH	Enc	Dec	**Total**	TG	LH	Enc	Dec	**Total**
Single	8-3-0	49.5	0	46.8	15.2	111.5	49.5	0	45.4	26.1	121.0
Single	8-3-12	49.5	396	48.6	19.4	513.5	49.5	396	49.7	28.5	523.7
MLA (Gangi)	8-3-12	49.5	0	242.9	29.4	321.8	49.5	0	254.4	38.1	342.0
MLA (Unity)	8-3-12	49.5	0	64.4	24.4	138.3	49.5	0	70.8	35.3	155.6
MLA (Bif)	8-3-12	49.5	0	107.3	24.7	181.5	49.5	0	119.2	34.9	203.6
MLA (Bif)*	8-3-12	49.5	0	84.3	25.3	159.1	49.5	0	94.6	33.2	177.3
Single 18	8-3-12	49.5	396	73.1	19.7	538.3	49.5	396	81.9	27.8	555.2
MLA (Gangi) 18	8-3-12	49.5	0	259.2	27.8	336.5	49.5	0	37.9	264.9	352.3
MLA (Unity) 18	8-3-12	49.5	0	106.5	27.2	183.2	49.5	0	38.3	117.6	205.4

Table 4.13 Detailed frame-wise delay results of multi-look-ahead CBS-T on CSJ in Table 4.6.

		Frame-wise Delay									
		P50 [ms](↓)					**P90** [ms](↓)				
Model	**Block**	TG	LH	Enc	Dec	**Total**	TG	LH	Enc	Dec	**Total**
MLA (Gangi)	8-3-12	49.5	0	242.9	29.4	321.8	49.5	0	254.4	38.1	342.0
MLA (Gangi)	8-5-10	49.5	0	127.8	26.3	203.6	49.5	0	174.3		32.2 256.0
MLA (Gangi) + 3-shift	8-5-10	49.5	0	125.6	26.8	201.9	49.5	0	176.5	31.5	257.5

Table 4.14 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.7.

	Frame-wise Delay									
	P50 [ms](↓)					**P90** [ms](↓)				
Shared layers	TG	LH	Enc	Dec	**Total**	TG	LH	Enc	Dec	**Total**
12	80	0	18.4	25.8	124.2	80	0	23.0	35.2	138.2
9	80	0	34.9	25.8	140.7	80	0	54.5	34.8	169.3
6	80	0	44.2	25.1	149.3	80	0	64.7	33.0	177.7
3	80	0	46.9	26.3	153.2	80	0	65.5	34.2	179.7
0	80	0	66.5	25.9	172.4	80	0	69.0	33.5	182.5

Table 4.15 Detailed frame-wise delay results of multi-look-ahead CBS-T on TED2 in Table 4.8.

	Frame-wise Delay									
	P50 [ms](↓)					**P90** [ms](↓)				
Shared layers	TG	LH	Enc	Dec	**Total**	TG	LH	Enc	Dec	**Total**
12	49.5	0	58.5	25.3	133.3	49.5	0	66.7	33.8	150.0
9	49.5	0	79.4	26.7	155.6	49.5	0	91.0	33.6	174.1
6	49.5	0	105.8	25.8	181.1	49.5	0	78.5	35.9	199.8
3	49.5	0	132.8	25.7	208.0	49.5	0	143.2	33.7	226.4
0	49.5	0	153.3	26.3	229.1	49.5	0	163.2	36.2	248.9

5 Conclusion and future work

The objective of this thesis was to construct streaming ASR models with low latency and high accuracy. Conventional streaming ASR suffers from the trade-off between recognition accuracy and the latency introduced by look-ahead frames, which are utilized to enable accurate recognition. In order to reduce the streaming ASR latency while keeping the accuracy, we proposed two different approaches, including enhancing encoder representations with Mask-CTC-based pre-training method; and avoiding look-ahead latency with Multi-look-ahead streaming architecture.

In Chapter 2, we introduced various types of existing streaming ASR models that we adopted as baseline models in this thesis. For streaming ASR utilizing attention masks, the trigger attention-based encoder-decoder streaming ASR (TA-E/D) and the streaming Transformer-Transducer (T-T) were introduced. For streaming ASR based on block processing, we covered CBS-E/D and CBS-T, both of which utilize the contextual block streaming encoder.

In Chapter 3, we introduced the first proposal: Mask-CTC-based encoder pre-training method for achieving low latency and high accuracy in streaming speech recognition. By pre-training the encoder networks of the streaming ASR models with the Mask-CTC framework, we expected to transfer the capability of considering long-term dependencies into the streaming encoder and therefore, reduce the latency requirements. Experimental results showed the effectiveness of the method on various model architectures, including TA-E/D, streaming T-T and CBS-E/D. The enhanced models managed to achieve higher accuracy with lower latency settings. Furthermore, by studying the output spike timings of the streaming models, we discovered that more precise alignments of the input and output sequences are learnt by the pre-training, which contributes to the latency reduction in streaming ASR.

In Chapter 4, we proposed a multi-look-ahead streaming ASR architecture to avoid the latency induced by the look-ahead frames in streaming ASR. Our work was based on the CBS encoder, where look-ahead frames are included in each input block and cannot be recognized promptly within the current block recognition. To remedy this issue, we proposed a multi-look-ahead architecture with two encoders operating in parallel. where a primary encoder generates accurate

outputs utilizing look-ahead frames, and the auxiliary encoder recognizes the look-ahead portion of the primary encoder without look-ahead. We studied various methods to implement the proposed system, including the Gangi method (shifting the network to perform as different encoders), the Unity method (generating both encoders' outputs in one encoding pass) as well as the Bifurcation method to improve the training performance of the Unity model. Experimental results on both English and Japanese datasets have shown that the proposed system yields equal recognition accuracy than the baseline models while maintaining zero look-ahead.

The future work of this thesis is listed as follows.

- **In-depth study of the Bifurcation multi-look-ahead model**: As one of the implementation methods for the multi-look-ahead streaming ASR, the bifurcation model achieves high recognition results with separate encoder layers for each latency mode. However, such a model structure results in higher number of total parameters in the streaming ASR, which might not be considered a fair comparisons to other implementation methods. On the other hand, other techniques can be utilized to realize the bifurcate architecture without adding encoder layers. For instance, adapters have been actively utilized in the field of automatic speech recognition [59, 60, 61]. In the work of [61], adapters were applied to the encoder layers of an ASR model to enable the learning of new tasks, which might also contribute to the training of multi-look-ahead streaming ASR when considering different latency modes as different training tasks. We expect that by utilizing adapters in the encoder layers, the additional layers can be pruned and the number of parameters can stay consistent among different methods.

- **Application to conversational systems**: As the motivation of the proposed multi-look-ahead architecture, conversational systems require the streaming ASR model to transcribe the input speech accurately and promptly to achieve rhythmic conversations. Therefore, our ultimate goal is to integrate the multi-look-ahead streaming ASR into the development of conversational systems. Prior work [62] has been conducted to utilize the multi-look-ahead method for response time estimation of the conversational system. In the future work, we aim to optimize the application of the multi-look-ahead streaming ASR and contribute to a rhythmic and natural conversational system.

Acknowledgements

I would like to express my heartfelt gratitude to Professor Tetsunori Kobayashi for his unwavering guidance and support throughout the entirety of my research journey. Not only did Professor Kobayashi provide invaluable assistance in compiling this thesis, but his guidance spanned across various aspects of my research during my undergraduate and graduate studies. His meticulous and insightful advice has given me a daily opportunity to reflect upon the essence of being a researcher.

I am also deeply appreciative of Professor Tetsuji Ogawa for his invaluable contributions. From Professor Ogawa, I have not only received guidance on research direction but also invaluable expertise in delivering presentations, crafting scholarly papers, and cultivating the mindset of a dedicated researcher. Furthermore, during moments of research stagnation, Professor Ogawa graciously devoted his time to listen and offer guidance. It is without a doubt that, without Professor Ogawa's unwavering support, my research experience would not have been as enriching and rewarding as it has been.

I would also like to extend my heartfelt appreciation to Yosuke Higuchi for his invaluable guidance in delving into the captivating realm of automatic speech recognition. His expertise and inspiration in shaping research topics have played a pivotal role in my journey. It is through his expert advice that I have gained a profound comprehension of the research content and have been able to make significant strides in my own research accomplishments.

I am sincerely grateful to everyone in the Kobayashi-Ogawa Laboratory. I deeply appreciate the senior and junior members in constructive discussions on a regular basis.

Lastly, I would like to express my heartfelt gratitude to my parents and close friends who have supported me both financially and emotionally throughout my research journey.

Bibliography

[1] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proc. ICML*, 2014.

[2] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Proc. NeurIPS*, 2015.

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

[4] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[5] Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Unidirectional neural network architectures for end-to-end automatic speech recognition. In *Proc. INTERSPEECH*, 2019.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NeurIPS*, pages 5998–6008, 2017.

[7] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented Transformer for speech recognition. In *Proc. INTERSPEECH*, pages 5036–5040, 2020.

[8] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. *Proc. ICASSP*, pages 5884–5888, 2018.

[9] Shigeki Karita, Nelson Yalta, Shinji 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 *Proc. INTERSPEECH*, 2019.

[10] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Rwth asr systems for librispeech: Hybrid vs attention - w/o data augmentation. In *Proc. Interspeech*, 2019.

[11] Yosuke Higuchi, Hirofumi Inaguma, Shinji Watanabe, Tetsuji Ogawa, and Tetsunori Kobayashi. Improved mask-ctc for non-autoregressive end-to-end asr. *Proc. ICASSP*, pages 8363–8367, 2020.

[12] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Yalta, Ryuichi Yamamoto, Xiao fei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. A comparative study on transformer vs rnn in speech applications. *Proc. ASRU*, pages 449–456, 2019.

[13] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Katya Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art speech recognition with sequence-to-sequence models. In *Proc. ICASSP*, pages 4774–4778, 2018.

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

[15] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. *Proc. ICASSP*, pages 7829–7833, 2020.

[16] Xie Chen, Yuehua Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. *Proc. ICASSP*, pages 5904–5908, 2021.

[17] Chung-Cheng Chiu and Colin Raffel. Monotonic chunkwise attention. *ArXiv*, abs/1712.05382, 2017.

[18] Tara N. Sainath, Chung-Cheng Chiu, Rohit Prabhavalkar, Anjuli Kannan, Yonghui Wu, Patrick Nguyen, and Zhijeng Chen. Improving the performance of online neural transducer models. *Proc. ICASSP*, pages 5864–5868, 2017.

[19] Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Triggered attention for end-to-end speech recognition. In *Proc. ICASSP*, pages 5666–5670, 2019.

[20] Matt Shannon, Gabor Simko, Shuo yiin Chang, and Carolina Parada. Improved end-of-query detection for streaming speech recognition. In *Proc. INTERSPEECH*, 2017.

[21] Shuo yiin Chang, Rohit Prabhavalkar, Yanzhang He, Tara N. Sainath, and Gabor Simko. Joint endpointing and decoding with end-to-end models. *Proc. ICASSP*, pages 5626–5630, 2019.

[22] Shuo yiin Chang, Bo Li, Tara N. Sainath, Chaoyang Zhang, Trevor Strohman, Qiao Liang, and Yanzhang He. Turn-taking prediction for natural conversational speech. In *Proc. INTERSPEECH*, 2022.

[23] Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur. A time-restricted self-attention layer for asr. *Proc. ICASSP*, pages 5874–5878, 2018.

[24] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proc. ACL*, 2019.

[25] Xuankai Chang, Aswin Shanmugam Subramanian, Pengcheng Guo, Shinji Watanabe, Yuya Fujita, and Motoi Omachi. End-to-end ASR with adaptive span self-attention. In *Proc. INTERSPEECH*, 2020.

[26] Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Streaming automatic speech recognition with the transformer model. *Proc. ICASSP*, pages 6074–6078, 2020.

[27] Shinji Watanabe, T. Hori, Suyoun Kim, J. Hershey, and T. Hayashi. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11:1240–1253, 2017.

[28] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *Proc. ICML*, 2006.

[29] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Álvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo yiin Chang, Kanishka Rao, and Alexander Gruenstein. Streaming end-to-end speech recognition for mobile devices. *Proc. ICASSP*, pages 6381–6385, 2018.

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

[31] Chung-Cheng Chiu and Colin Raffel. Monotonic chunkwise attention. *ArXiv*, abs/1712.05382, 2018.

[32] Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. Transformer asr with contextual block processing. *Proc. ASRU*, pages 427–433, 2019.

[33] Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe. Streaming transformer asr with blockwise synchronous beam search. *Proc. SLT*, pages 22–29, 2020.

[34] Huaibo Zhao, Shinya Fujie, Tetsuji Ogawa, Jin Sakuma, Yusuke Kida, and Tetsunori Kobayashi. Conversation-oriented asr with multi-look-ahead cbs architecture. *ArXiv*, abs/2211.00858, 2022.

[35] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

[36] Yuting Yang, Yuke Li, and Binbin Du. Enhancing the unified streaming and non-streaming model with contrastive learning. *ArXiv*, abs/2306.00755, 2023.

[37] J. Sun, Guiping Zhong, Dinghao Zhou, and Baoxiang Li. Dynamic latency for ctc-based streaming automatic speech recognition with emformer. *ArXiv*, abs/2203.15613, 2022.

[38] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *ArXiv*, abs/2005.10242, 2020.

[39] Yosuke Higuchi, Shinji Watanabe, Nanxin Chen, Tetsuji Ogawa, and Tetsunori Kobayashi. Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict. In *Proc. INTERSPEECH*, pages 3655–3659, 2020.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proc. NAACL-HLT*, 2019.

[41] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proc. EMNLP-IJCNLP*, pages 6114–6123, 2019.

[42] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. ESPnet: End-to-end speech processing toolkit. In *Proc. INTERSPEECH*, pages 2207–2211, 2018.

[43] Florian Boyer, Yusuke Shinohara, Takaaki Ishii, Hirofumi Inaguma, and Shinji Watanabe. A study of Transducer based end-to-end ASR with ESPnet: Architecture, auxiliary loss and decoding strategies. *Proc. ASRU*, pages 16–23, 2021.

[44] Douglas B Paul and Janet Baker. The design for the wall street journal-based CSR corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York*, 1992.

[45] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The Kaldi speech recognition toolkit. In *Proc. ASRU*, 2011.

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

[47] Andrew L. Maas, Awni Y. Hannun, Dan Jurafsky, and A. Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *ArXiv*, abs/1408.2873, 2014.

[48] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[49] A. Rousseau, P. Deléglise, and Y. Estève. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *Proc. LREC*, 2014.

[50] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Dogus Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *ArXiv*, abs/1904.08779, 2019.

[51] Hirofumi Inaguma and Tatsuya Kawahara. Alignment knowledge distillation for online streaming attention-based speech recognition. *ArXiv*, abs/2103.00422, 2021.

[52] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, 2017.

[53] Jay Mahadeokar, Yangyang Shi, Ke Li, Duc Le, Jiedan Zhu, Vikas Chandra, Ozlem Kalinli, and Michael L. Seltzer. Streaming parallel transducer beam search with fast-slow cascaded encoders. *ArXiv*, abs/2203.15773, 2022.

[54] Arun Narayanan, Tara N. Sainath, Ruoming Pang, Jiahui Yu, Chung-Cheng Chiu, Rohit Prabhavalkar, Ehsan Variani, and Trevor Strohman. Cascaded encoders for unifying streaming and non-streaming asr. *Proc. ICASSP*, pages 5629–5633, 2021.

[55] Tara N. Sainath, Yanzhang He, Arun Narayanan, Rami Botros, Ruoming Pang, David Rybach, Cyril Allauzen, Ehsan Variani, James Qin, Quoc-Nam Le-The, Shuo yiin Chang, Bo Li, Anmol Gulati, Jiahui Yu, Chung-Cheng Chiu, Diamantino Caseiro, Wei Li, Qiao Liang, and Pat Rondon. An efficient streaming non-recurrent on-device end-to-end model with improvements to rare-word modeling. In *Proc. INTERSPEECH*, 2021.

[56] Yangyang Shi, Varun K. Nagaraja, Chunyang Wu, Jay Mahadeokar, Duc Le, Rohit Prabhavalkar, Alex Xiao, Ching feng Yeh, Julian Chan, Christian Fuegen, Ozlem Kalinli, and

Michael L. Seltzer. Dynamic encoder transducer: A flexible solution for trading off accuracy for latency. *ArXiv*, abs/2104.02176, 2021.

[57] Kikuo Maekawa. Corpus of spontaneous japanese : Its design and evaluation. 2003.

[58] Juntae Kim, Yoonhan Lee, and Eesung Kim. Accelerating rnn transducer inference via adaptive expansion search. *IEEE Signal Processing Letters*, 27:2019–2023, 2020.

[59] Bethan Thomas, Samuel Kessler, and Salah Karout. Efficient adapter transfer of self-supervised speech models for automatic speech recognition. *Proc. ICASSP*, pages 7102–7106, 2022.

[60] Samuel Kessler, Bethan Thomas, and Salah Karout. An adapter based pre-training for efficient and scalable self-supervised speech representation learning. *Proc. ICASSP*, pages 3179–3183, 2021.

[61] Steven Vander Eeckt and Hugo Van hamme. Using adapters to overcome catastrophic forgetting in end-to-end automatic speech recognition. *Proc. ICASSP*, 2022.

[62] Jin Sakuma, Shinya Fujie, Huaibo Zhao, Tetsuji Ogawa, and Tetsunori Kobayashi. Improving the response timing estimation for spoken dialogue systems by reducing the effect of speech recognition delay,. *Proc. Interspeech*, 2023 (to appear).

List of works

International conferences

- Huaibo Zhao, Yosuke Higuchi, Yusuke Kida, Tetsuji Ogawa, Tetsunori Kobayashi, "Mask-CTC-based Encoder Pre-training for Streaming End-to-End Speech Recognition," in *Proceedings of the 2023 31st European Signal Processing Conference (EUSIPCO 2023)*, September 2023. (to appear)

- Jin Sakuma, Shinya Fujie, Huaibo Zhao, Tetsuji Ogawa, Tetsunori Kobayashi, "Improving the response timing estimation for spoken dialogue systems by reducing the effect of speech recognition delay," in *Proceedings of 24th Annual Conference of the International Speech Communication Association (INTERSPEECH 2023)*, August 2023. (to appear)

- Huaibo Zhao, Shinya Fujie, Tetsuji Ogawa, Jin Sakuma, Yusuke Kida, Tetsunori Kobayashi, "Conversation-oriented ASR with Multi-look-ahead CBS Architecture," in *Proceedings of 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023)*, pages 1-5, June 2023.

- Huaibo Zhao, Yosuke Higuchi, Tetsuji Ogawa, Tetsunori Kobayashi, "An Investigation of Enhancing CTC Model for Triggered Attention-based Streaming ASR," in *Proceedings of 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2021)*, pages 477–483, December 2021.

Domestic conferences

- Huaibo Zhao, Shinya Fujie, Tetsuji Ogawa, Tetsunori Kobayashi, "An investigation on constructing Multi-look-ahead Contextual Block Streaming Transducer," 日本音響学会研究発表会講演論文集 *(ASJ)*, September 2023. (to appear)

- Huaibo Zhao, Shinya Fujie, Tetsuji Ogawa, Jin Sakuma, Yusuke Kida, Tetsunori Kobayashi,

"Multiple Latency CBS Streaming ASR for Conversational Systems," 情報処理学会研究報告 *(SLP)*, vol.2023-SLP-146, no.9, pages 1-6, February 2023.

- 趙懐博, 樋口陽祐, 木田祐介, 小川哲司, 小林哲則, "Tranducer 型ストリーミング音声認識における Mask-CTC を用いた事前学習," 情報処理学会研究報告 *(SLP)*, vol.2022-SLP-142, no.61, pages 1-6, June 2022.
- 趙懐博, 樋口陽祐, 小川哲司, 小林哲則, "Triggered attention 型ストリーミング音声認識における Mask-CTC を用いた事前学習," 情報処理学会研究報告（*SLP*）, vol.2021-SLP-138, pages 1-6, October 2021.

Awards

- Yamashita SIG Research Award, from Information Processing Society of Japan (IPSJ), 2023. (Recommended Candidate)
- CCE Department Award, from Department of Communications and Computer Engineering, Waseda University, September 2021.