Incorporating Linguistic Cues for End-to-End Speech Recognition

End-to−End音声認識における言語的な手がかりの利用

February 2024

Yosuke HIGUCHI

樋口 陽祐

Incorporating Linguistic Cues for End-to-End Speech Recognition

End-to−End音声認識における言語的な手がかりの利用

February 2024

Waseda University
Graduate School of Fundamental Science and Engineering
Department of Computer Science and Communications Engineering
Research on Perceptual Computing

Yosuke HIGUCHI

樋口 陽祐

Incorporating Linguistic Cues for End-to-End Speech Recognition

by

Yosuke HIGUCHI

Submitted to the Department of Computer Science and Communications Engineering
in February 2024, in partial fulfillment of the
requirements for the degree of
Doctor of Engineering

Abstract

When humans listen to speech, they do not simply process the raw sound waves; instead, they actively engage with layers of linguistic knowledge — semantic, syntactic, and contextual — to accurately identify which words are spoken. This dissertation presents a series of studies focused on the role of such *top-down linguistic cues* in performing automatic speech recognition (ASR), with the primary goal of enhancing the capabilities of end-to-end systems. End-to-end ASR aims to achieve direct conversion from speech to text by training a single deep neural network model using pairs of speech utterances and their corresponding transcriptions. However, the modality gap between audio input and text output poses a significant challenge in capturing linguistic information directly from speech, which is essential for generating accurate textual output. In addressing this challenge, this dissertation introduces four distinct yet interrelated approaches, each designed to effectively extract and manage linguistic information within end-to-end ASR models.

The first study explores methods to hierarchically increase the abstraction level in linguistic outputs, aiming to efficiently learn representations for sparse word-level units. In end-to-end ASR, models are expected to implicitly learn representations conducive to word prediction. However, this requires an extensive amount of training data to overcome the substantial abstraction gap between input acoustic signals and output linguistic tokens. To facilitate word-level representation learning, I first develop a hierarchical conditional model. The proposed model is trained by auxiliary connectionist temporal classification (CTC) losses applied to intermediate layers, where the vocabulary size of each target subword sequence is gradually increased as the layers become close to the word-level output. Each level of sequence prediction is explicitly conditioned on the sequences predicted at previous levels, enabling the model to progressively construct word-level representations by considering a hierarchy of linguistic structures. Then, I enhance the generation capability of the hierarchical model by employing a refinement mechanism at each stage of intermediate prediction. This mechanism repeatedly uses the shared model parameters to refine its intermediate representations, which leads to an improvement in the overall performance of the model. Finally, I design an efficient pseudo-labeling-based algorithm for the proposed hierarchical model, which utilizes audio-only data within a semi-supervised learning framework to further boost the model performance.

The second study involves incorporating the concept of masked language modeling into end-to-end ASR, with the goal of augmenting the model's ability to capture long-range linguistic contexts. To this end, I first propose joint training and decoding strategies that synergize CTC with masked language modeling. To mitigate the limitation of CTC in explicitly modeling dependencies between output tokens, contextual information derived from masked language modeling is used to enhance the performance of CTC-based ASR. The non-autoregressive nature shared by both CTC and masked language modeling also enables fast inference without compromising recognition accuracy. In addition to end-to-end ASR, the proposed framework shows promise for application to end-to-end speech translation tasks, adeptly handling semantic information vital for producing translated sequences. Then, I establish that the adoption of the masked language model mechanism offers benefits to other end-to-end ASR models. The proposed model architecture augments the transducer-based model by injecting explicit contextual linguistic cues into the speech-encoding process, which is shown to push the limits of prior state-of-the-art results. Lastly, I demonstrate that masked language modeling is advantageous in acquiring representations beneficial for streaming end-to-end ASR models, allowing for the extraction of anticipated linguistic contexts from constrained speech input.

The third study investigates the application of pre-trained masked language models (e.g., BERT) in end-to-end ASR, utilizing their versatile linguistic knowledge to guide the generation process of linguistic sequences. In the field of natural language processing, large-scale pre-training using vast amounts of text data has greatly advanced language models' ability to learn diverse aspects of linguistic information. Such capabilities are expected to enhance end-to-end ASR systems by empowering models to effectively interpret complex linguistic elements, which is important for choosing words that are both grammatically correct and contextually appropriate. To harness this potential, I first present a novel end-to-end ASR formulation that explicitly conditions BERT's contextualized word embeddings on the ASR process, adapting BERT for the CTC-based training and inference framework. By seamlessly infusing BERT knowledge into audio information, the proposed model improves over conventional approaches. Additionally, I demonstrate its potential application in end-to-end spoken language understanding tasks, which typically require more abstract linguistic processing. Subsequently, I introduce an extension of the proposed BERT-based model by implementing an additional transducer-based decoder. The decoder is trained using a vocabulary suitable for ASR training, aiming to bridge the gap between the text processed in end-to-end ASR and BERT. This is shown crucial as these models utilize distinct vocabularies and exhibit different text formats and styles, including variations in punctuation usage.

The last study further delves into the integration of pre-trained language models in end-to-end ASR, with a specific emphasis on larger-sized and controllable language models. Modern large language models (e.g., ChatGPT) are capable of performing a wide range of linguistic tasks within zero-shot learning, guided by precise instructions or prompts to direct the text-generation process toward the desired task. I explore using this zero-shot capability inherent in large language models to enhance end-to-end ASR. The proposed approach involves guiding a large language model to perform zero-shot grammatical error correction, thereby extracting linguistic information that contributes to improving ASR performance. The linguistic knowledge drawn from the large language model is then used to trigger the ASR decoding process, along with acoustic information, for achieving accurate sequence generation.

Defense Committee

Tetsunori Kobayashi	Professor, Faculty of Science and Engineering, Waseda University
Tetsuji Ogawa	Professor, Faculty of Science and Engineering, Waseda University
Daisuke Kawahara	Professor, Faculty of Science and Engineering, Waseda University
Shinji Watanabe	Associate Professor, Language Technologies Institute, Carnegie Mellon University

Acknowledgments

First of all, I would like to thank all the members of my thesis committee: Professors Tetsunori Kobayashi, Tetsuji Ogawa, Daisuke Kawahara, and Shinji Watanabe. Their invaluable guidance and insights have been instrumental in shaping the success of my defense.

I would like to express my deepest and most sincere gratitude to Professor Tetsunori Kobayashi, whose insightful guidance and profound wisdom have been fundamental to the success of my Ph.D. journey. "Research is what you enjoy," Kobayashi-sensei always told us, the guiding principle that significantly influenced my approach to academic research. Although the early stages of my undergraduate research were challenging, and I was sometimes intimidated by his strict teaching style, I have gradually discovered a deep sense of enjoyment in my research, particularly as I neared the end of my Ph.D. studies. It has been a great honor to see Kobayashi-sensei genuinely engaged and enjoying our discussions about my Ph.D. research.

I wish to extend my deepest appreciation to Professor Tetsuji Ogawa for his invaluable guidance and support throughout my academic pursuits. When I first joined the lab as a fourth-year university student in 2018, it was Ogawa-sensei who sparked my interest in speech recognition research. Over the past five years, leading up to the completion of my Ph.D., my initial curiosity has evolved into profound engagement with the field. His continuous guidance has been crucial in my development as a researcher, covering various skills such as conducting research, engaging in academic environments, creating presentations, and writing papers. I am also thankful to him for providing consultations on my career-related concerns, generously dedicating his valuable time despite his busy schedule.

I would also like to express my heartfelt gratitude to Professor Shinji Watanabe at Carnegie Mellon University (CMU) for his substantial contributions to my research. His influence was the critical factor in my decision to pursue a doctoral degree, and throughout my Ph.D. studies, he has provided extensive mentorship and insightful advice, enabling me to embark on a fruitful academic journey. My initial inspiration for Ph.D. research emerged during a visit to Johns Hopkins University (JHU), where I developed a strong interest in end-to-end speech recognition under the expert guidance of Watanabe-sensei. He kindly invited me for a follow-up visit to CMU, which significantly contributed to forming the fundamental core of my Ph.D. research. Furthermore, he generously continued our collaboration beyond my visiting periods, leading to numerous achievements that encompassed not only my primary research focus but also areas outside my expertise. Watanabe-sensei's broad insights, extending beyond speech recognition, along with his astute guidance, have been instrumental in the completion of this dissertation. His genuine approach to scientific research will continue to serve as a pivotal guide in forming my future career as a researcher.

Gratefully, Watanabe-sensei also provided me with diverse opportunities to connect with leading researchers in the field, many of which led to collaborations and internships that have shaped my Ph.D. research. I would like to thank all the members of the Center for Language and Speech Processing (CLSP) at JHU for their warm hospitality and support during my visit. In particular, I would like to extend my heartfelt thanks to Nanxin Chen (currently at Google) for his invaluable collaboration. Likewise, my experience at Watanabe's Audio and Voice Laboratory (WAVLab) at CMU was exceptionally rewarding, thanks to the exciting collaborations with Brian Yan, Siddhant Arora, and other lab members. I cannot thank enough all the contributors in the ESPnet open-source community. The tools and resources they have provided have been crucial in advancing my research. A special acknowledgment is due to Hirofumi Inaguma (currently at Meta), whose advice on end-to-end speech recognition has been extraordinarily helpful.

In the beginning of my Ph.D. studies, I was extremely fortunate to do an internship at Mitsubishi Electric Research Laboratories (MERL). I am deeply thankful to my mentors and collaborators at MERL: Takaaki Hori (currently at Apple), Niko Moritz (currently at Meta), and Jonathan Le Roux. Their exceptional expertise and insights in the field of speech processing were invaluable and played a crucial role in my growth as a researcher. Despite the challenges posed by remote work during the pandemic, the internship was an amazing and pivotal experience that broadened my research interests, especially in the area of semi-supervised learning. The part of my dissertation focusing on semi-supervised speech recognition has been built upon the research accomplishments attained during our collaboration at MERL.

During my second year of Ph.D. studies, I had the remarkable opportunity to intern at Google. I am grateful to all the members of the speech recognition team at the New York office. My time there was marked by stimulating and dynamic discussions, with each researcher enthusiastically exchanging ideas both within and across research groups. Such a collaborative environment greatly enriched my experience as an industrial researcher. I further would like to extend my deep appreciation to my mentors and collaborators: Andrew Rosenberg, Yuan Wang, Murali Karthick Baskar, and Bhuvana Ramabhadran. Working with them was immensely rewarding, allowing us to blend our research expertise to achieve innovative outcomes, which I have been fortunate to include as a major part of my dissertation.

I would also like to thank all the members of our Perceptual Computing Laboratory and Media Intelligence Laboratory for their insightful and constructive discussions on a daily basis. Particularly, I am thankful to Secretary Madoka Kuniyasu for her invaluable assistance with administrative tasks and Huaibo Zhao, Keita Karube, and Nahomi Kusunoki for their excellent collaboration and support in various aspects of my research.

I extend my gratitude to the funds and fellowships that have supported my research endeavors and living expenses throughout my Ph.D. pursuit. I was honored to receive the JSPS Research Fellowship for Young Scientists (DC1), grant number JP21J23495, which provided support from April 2021 to March 2024. Additionally, my research work was further supported by the JST ACT-X project, under grant number JPMJAX210J, from October 2021 to March 2024.

Finally, I would like to thank my friends and family — my parents, my younger brother, and my beloved dog — for their unwavering encouragement and support throughout my extended student life. I am forever thankful to my mother for her kindness and assistance in every aspect of my daily life, always given without a single complaint. I also extend a heartfelt thank you to my father, who has consistently provided me with invaluable mental and financial support over the years.

Contents

List of Figures

List of Tables

List of Notations

The next list describes abbreviations and notations that will used in this dissertation. In mathematical notation, indices are represented using subscripts, e.g., $\mathbf{o}_t$ for an acoustic feature at frame t, and variable types are indicated using superscripts, e.g., N^{enc} for the number of encoder layers.

Abbreviations

AED	Attention-Based Encoder Decoder
ASR	Automatic Speech Recognition
BBD	Block Boundary Detection
CBS	Contextual Block Streaming
CER	Character Error Rate
CMLM	Conditional Masked Language Model
CSJ	Corpus of Spontaneous Japanese
CTC	Connectionist Temporal Classification
DNN	Deep Neural Network
FFN	Feed-Forward Network
GLU	Gated Linear Unit
GMM	Gaussian Mixture Model
HC-CTC	Hierarchical Conditional CTC
HMM	Hidden Markov Model
HR-CTC	Hierarchical Recursive CTC
LF-MMI	Lattice-Free Maximum Mutual Information
LLM	Large Language Model
LSTM	Long Short-Term Memory
MHA	Multi-Head Attention
MHSA	Multi-Head Self-Attention
MPL	Momentum Pseudo-Labeling
NLP	Natural Language Processing
OOV	Out-of-Vocabulary
ReLU	Rectified Linear Unit

RNN	Recurrent Neural Network
RNN-LM	Recurrent Neural Network-Based Language Model
RTF	Real-Time Factor
SC-CTC	Self-Conditioned CTC
SLU	Spoken Language Understanding
WER	Word Error Rate
WRR	Word Error Rate Recovery Rate
WSJ	Wall Street Journal

General Notations

λ	Loss weight
$\mathbb{E}[\cdot]$	Expectation
$\mathcal{D}$	Set of training samples
$\mathcal{F}(\cdot)$	General function
$\mathcal{L}$	Loss function
D	Number of dimensions
d	Dimension index
K	Number of iterations
k	Iteration count
N	Number of elements
n	Element index
P	Probability
$p(\cdot)$	Probabilistic distribution function

Neural Networks

β_1, β_2, ϵ	Beta coefficients and epsilon parameter for Adam optimizer
$\mathbf{b}$	Bias vector
$\mathbf{c}$	Context vector from attention network
$\mathbf{e}$	Output feature vector of Llama2
$\mathbf{g}$	Output feature vector of concatenation network
$\mathbf{h}$, H	Output feature vector, feature sequence of encoder network
$\mathbf{K}$	Convolution kernel matrix
$\mathbf{q}$, Q	Output feature vector, feature sequence of decoder network
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	Query, key, value matrices
$\mathbf{r}$	Output feature vector of joint network
$\mathbf{s}$	Output feature vector of prediction network

$\mathbf{v}$	Weight vector
$\mathbf{W}$	Weight matrix
$\mathbf{y}, Y$	Output feature vector, feature sequence of Transformer decoder
$\mathcal{I}$	Set of encoder layer indices
Θ, Φ	Model parameters
$a, \mathbf{a}$	Attention weight, attention weight vector
E	Transformed feature sequence of BERT output
e	Attention score
i, j	Layer indices for encoder, decoder networks
X	General feature sequence

Speech Recognition

α	Momentum coefficient
$\mathbf{o}, O$	Acoustic feature vector, feature sequence
η, τ	Coverage coefficient, threshold
γ	Insertion penalty
$\mathcal{B}(\cdot)$	Collapsing function of CTC or transducer
$\mathcal{H}(\cdot)$	Mapping function that returns a set of all possible hypotheses
$\mathcal{M}(\cdot)$	Mapping function that returns a set of all possible token masking patterns
$\mathcal{P}$	Set of phonemes
$\mathcal{T}$	Set of frame indices
$\mathcal{U}$	Set of token positions replaced with mask token
$\mathcal{V}$	Vocabulary
$\mathcal{W}$	Set of tokens
ω	Weight for MPL
ξ	Weight for joint CTC and AED decoding
a, A	Token, token sequence of CTC
B	Beam size
F	Acoustic feature dimension
J	Length of phoneme sequence
L	Length of token sequence
l	Token index
m, M	Phoneme, phoneme sequence
R	Number of recursive operations in HR-CTC
r	Count of recursive operations in HR-CTC
s, S	HMM state, HMM state sequence
T	Length of acoustic feature sequence

t	Frame index
T'	Length of down-sampled acoustic feature sequence
u	Token index of transducer
V	Sequence of token emission probabilities
w, W	Token, token sequence
z, Z	Token, token sequence of transducer

1
Introduction

1.1 Background

Speech is an efficient and cost-effective medium for expressing one's thoughts and feelings, enabling rapid and direct communication with minimal mental effort. Automatic speech recognition (ASR) is the fundamental technology that facilitates spoken communication, allowing for natural interactions between humans and machines, as well as aiding interactions among humans. By converting speech information to text information, ASR underpins a variety of applications, including transcribing lectures and meetings, assisting in telephony services, enhancing accessibility in television broadcasting, and building interactive spoken dialogue systems. Fueled by the growing demand for these practical applications, the performance of ASR systems has witnessed remarkable improvements, primarily thanks to emerging developments in deep learning-based techniques (Hinton et al., 2012; Graves et al., 2013). Much of the recent advancements is attributed to the end-to-end framework (Graves and Jaitly, 2014; Chorowski et al., 2015; Chan et al., 2016; Li, 2022; Prabhavalkar et al., 2024) that directly models the speech-to-text conversion using a single deep neural network (DNN) model. Building on foundational advances in sequence-to-sequence modeling techniques (Graves et al., 2006; Graves, 2012; Sutskever et al., 2014; Bahdanau et al., 2014), neural network architectures (Dong et al., 2018; Kriman et al., 2020; Gulati et al., 2020), and large-scale training methods (Baevski et al., 2020; Hsu et al., 2021; Radford et al., 2023; Zhang et al., 2023), end-to-end ASR has demonstrated promising results across a range of benchmarks and even exhibited reasonable performance in real-world environments.

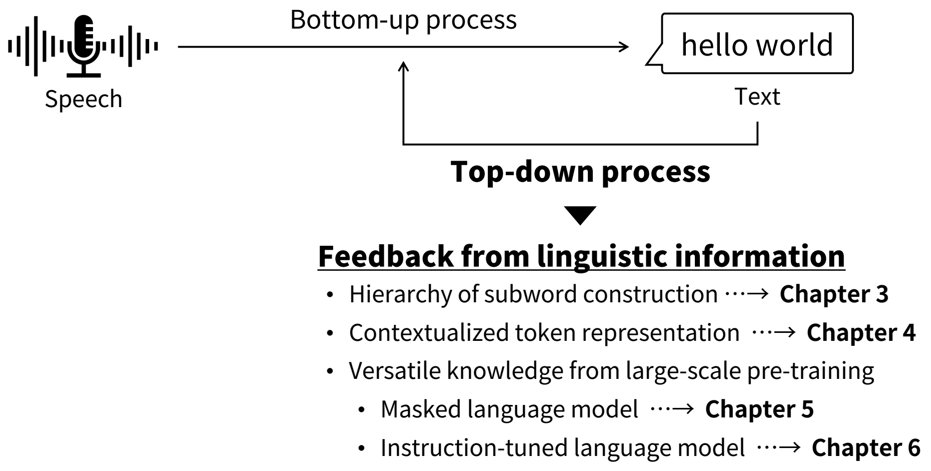

Bottom-up process
Speech
hello world
Text
Top-down process
Feedback from linguistic information
Hierarchy of subword construction ⋯→ Chapter 3
Contextualized token representation ⋯→ Chapter 4
Versatile knowledge from large-scale pre-training
Masked language model ⋯→ Chapter 5
Instruction-tuned language model ⋯→ Chapter 6

Figure 1-1: Dissertation overview. In addition to unidirectional, bottom-up modeling of ASR in conventional end-to-end approaches, bidirectional interactions with the top-down process, particularly through the use of linguistic information, are explored.

As illustrated in Figure 1-1, end-to-end ASR typically operates as a one-way process between two different modalities, directly mapping speech input into text output. This is often achieved through a sequence of stacked DNN layers, with each layer successively increasing the abstraction level of linguistic information being processed. As the speech input progresses through these layers, the model initially works with phonetic-level representations and gradually assembles them to form token (grapheme or word)-level representations (Belinkov and Glass, 2017; Shim et al., 2021). In contrast to this unidirectional, bottom-up approach of end-to-end ASR, human transcription involves a more dynamic and adaptive process, where both the speech input and the resulting text output play integral roles in informing one another. This bidirectional, interactive mechanism has been well-studied in traditional models of speech perception as part of human cognitive processes (McClelland and Elman, 1986; Norris, 1994). For example, consider a dictation assessment scenario in which individuals are tasked with transcribing a sentence spoken aloud, often repeated a couple of times by the speaker. In this situation, transcribers do more than merely write down the words they hear; they actively engage with the linguistic information derived from what they write, including semantics (the meaning of words and phrases), syntax (the arrangement of words and phrases to create complete sentences), and context (the relationship of words to the surrounding words and sentences). Such top-down clues from linguistic understanding can enhance their transcription accuracy, enabling a more focused and precise interpretation of speech, particularly in segments with challenging or ambiguous words.

In speech recognition, speech and text do not necessarily have a sequential or hierarchical relationship, and end-to-end ASR could benefit from incorporating top-down linguistic feedback or cues into its feature extraction and interpretation process. To explore this potential, this dissertation delves into approaches for effectively capturing and utilizing linguistic information within end-to-end ASR models, with the primary objective of enhancing the accuracy and reliability of transcription generation. For the rest of this chapter, I first provide a review of prior work relevant to the focus of this dissertation, along with the challenges and research questions associated with end-to-end ASR. I then outline the proposed approaches for integrating top-down linguistic cues in end-to-end ASR.

1.2 Related Work

This section reviews related work focusing on the processing of linguistic information in end-to-end ASR. It begins by exploring the roles of decoder networks within end-to-end ASR models, including a discussion regarding the significance of an explicit language modeling mechanism. Subsequently, different techniques for integrating separate language models are compared, emphasizing their use of linguistic knowledge independently of acoustic information.

1.2.1 Language Modeling Mechanism in End-to-End ASR

In end-to-end ASR, attention-based encoder-decoder (Chorowski et al., 2015; Chan et al., 2016; Bahdanau et al., 2016) and transducer (Graves, 2012; Graves et al., 2013) models are the current predominant choices for model structures, which primarily consist of encoder and decoder networks. In alignment with the principles of the classical noisy channel framework (Jelinek, 1998), the encoder network functions similarly to an acoustic model, extracting fine-grained phonetic patterns from the input acoustic signal. Concurrently, the decoder network is analogous to a language model, capturing causal dependencies among the output linguistic symbols (e.g., graphemes or subwords).

While the encoder and decoder networks each perform distinct roles in processing different modalities, recent findings suggest that the language modeling function in the decoder network may be less significant than the language model established in the traditional framework. The state-of-the-art encoder architecture, Conformer (Gulati et al., 2020; Tüske et al., 2021), only adopts a single long short-term memory (LSTM) layer for its decoder network. In certain scenarios, the capacity of the decoder network can be further reduced to an n-gram (Tripathi et al., 2020; Ghodsi et al., 2020; Prabhavalkar et al., 2021; Botros et al., 2021) or even a randomly initialized embedding layer (Shrivastava et al., 2021) without sacrificing recognition accuracy. Moreover, a model based on connectionist temporal classification (CTC) (Graves et al., 2006) has shown practical performance even without the decoder network, particularly with the encoder pre-training (Baevski et al., 2020; Zhang et al., 2022c) or intermediate regularization techniques (Sanabria and Metze, 2018; Lee and Watanabe, 2021; Nozaki and Komatsu, 2021). In this context, with the sophisticated network architectures and training methods, the encoder network is capable of taking responsibility for a substantial amount of learning text information (Variani et al., 2020; Ghodsi et al., 2020), reducing the reliance on the decoder network for explicit language modeling.

Consequently, the conventional end-to-end ASR models tend to lack a dedicated structure for language modeling. Yet, the ability to handle linguistic sequences is crucial for advancing ASR systems beyond merely generating accurate transcriptions, particularly when extending ASR to tasks that demand an understanding of spoken language. Such comprehension cannot be achieved by simply looking at the limited context of the output linguistic symbols. This dissertation introduces effective methods for incorporating linguistic knowledge into end-to-end ASR models, suggesting the importance of explicit language modeling in enhancing the overall performance and functionality of ASR systems.

1.2.2 Integration Methods with Separate Language Model

In classical ASR systems based on the noisy channel model (Jelinek, 1998), a separate language model is commonly employed during the inference stage, facilitating the generation of the most probable hypothesis in terms of language. Correspondingly, numerous studies have demonstrated that incorporating separate language models similarly improves the recognition accuracy in end-to-end ASR systems. Training the end-to-end models generally requires a substantial amount of paired speech and text data. Language models, in contrast, can be trained solely with text data, which is easier to collect and scale than paired data, such as through web crawling. Accordingly, a separate language model can be trained using additional external text data, thereby providing end-to-end ASR models with comprehensive linguistic knowledge for further enhancing their performance.

There is a line of previous studies that have explored the incorporation of separate language models into end-to-end ASR. In this context, these language models exhibit diverse architectures, including n-gram (Jelinek, 1998; Miao et al., 2015; Chorowski and Jaitly, 2017), recurrent neural network (RNN) (Sundermeyer et al., 2012; Mikolov et al., 2010), and Transformer models (Irie et al., 2019; Karita et al., 2019), which are selected depending on the situation, such as balancing accuracy with speed. Shallow fusion (Hannun et al., 2014a; Gulcehre et al., 2015; Chorowski and Jaitly, 2017; Kannan et al., 2018) has been the predominant integration approach for utilizing a separate language model during inference. This involves scoring hypotheses through a simple linear interpolation of output probabilities, which are computed by end-to-end ASR and language models. Deep fusion (Gulcehre et al., 2015) concatenates the hidden states from a separate language model with those from the decoder in a trained end-to-end ASR model. Both models are fine-tuned jointly to compute the output probability from the combined states, allowing for the selective use of information from the language model. Cold fusion (Sriram et al., 2018; Shan et al., 2019) is another integration approach, similar to deep fusion, where the end-to-end ASR model is trained from scratch with a frozen language model. This enables the use of the language model for capturing language-specific information. Consequently, the end-to-end ASR model can concentrate on learning relevant information conducive to speech-to-text mapping, while also alleviating the language bias inherent in the limited training text data.

Similar to the focus of this dissertation, such fusion techniques with separate language models can be viewed as a top-down process for end-to-end ASR models. However, these approaches primarily concentrate on refining the model outputs through the use of linguistic knowledge. In contrast, my research explores the more direct incorporation of linguistic information into the speech-to-text process of end-to-end ASR. This enables a flexible and adaptable interplay between acoustic and linguistic information, thereby enhancing the capacity of end-to-end ASR models to handle complex tasks that require both speech cues (such as prosody and emotion) and linguistic understanding. This dissertation presents methods for integrating linguistic information into end-to-end ASR, which shows potential not only in improving the recognition accuracy but also in expanding the applicability in more linguistically challenging tasks.

1.3 Goal

The primary goal of this dissertation is to advance end-to-end ASR through the incorporation of top-down linguistic cues into the direct speech-to-text conversion process. Recognizing the role of linguistic knowledge in human speech perception, my research explores end-to-end ASR approaches that emphasize a dynamic and adaptive interaction between speech and text information.

More concretely, the modeling of top-down linguistic cues is achieved through the introduction of a "latent" linguistic sequence, which represents text information that is recognizable from speech at any given moment. Given an input speech sequence O and an output linguistic sequence W, the posterior probability of ASR $p(W|O)$ is factorized into two models by marginalizing over all possible latent linguistic sequences as

$$p(W|O) = \sum_{\tilde{W} \in \mathcal{F}(W)} p(W, \tilde{W}|O), \tag{1.1}$$

$$= \sum_{\tilde{W} \in \mathcal{F}(W)} \underbrace{p(W|\tilde{W}, O)}_{\substack{\text{Top-down} \\ \text{modeling}}} \underbrace{p(\tilde{W}|O)}_{\substack{\text{Bottom-up} \\ \text{modeling}}}, \tag{1.2}$$

where a latent linguistic sequence $\tilde{W}$ is derived from the output sequence through a mapping function of $\mathcal{F}(\cdot)$. In Eq. (1.2), the term $p(\tilde{W}|O)$ is interpreted as a distribution of text sequences that are readily recognizable from the speech input alone, indicative of bottom-up processing, and the term $p(W|\tilde{W}, O)$ is interpreted as a distribution of text sequences that necessitate additional linguistic context beyond the speech input for accurate recognition, representing top-down processing. In this context, top-down linguistic cues are defined as linguistic information supplied based on the latent linguistic sequence, and the objective of this dissertation is to investigate methodologies for effectively designing these cues.

To this end, top-down linguistic cues, derived from various perspectives of language, are designed to enhance ASR performance, as listed in Figure 1-1. Firstly, the hierarchy in constructing subwords provides systematic information, involving the combination of smaller linguistic elements to form complete words. Given the huge modality gap between speech and text, such hierarchy in linguistic structures is expected to aid end-to-end ASR models in constructing word-level outputs more effectively. The second focus is on masked language modeling, originally developed for language model pre-training, which captures semantic information through the token-infilling task. This is expected to empower end-to-end ASR models to account for long-range, bi-directional dependencies among the output tokens, thereby yielding contextualized linguistic representations for improving the accuracy of token prediction. The final approach examines the effectiveness of integrating a pre-trained language model into end-to-end ASR models, aiming to utilize versatile linguistic knowledge for accurate text generation. More specifically, two distinct categories of pre-trained language models are explored. The first encompasses masked language models (e.g., BERT), which are adept at processing semantic and syntactic information. The second involves the more recent instruction-tuned large language models (e.g., ChatGPT), which can further consider contexts through prompting.

Developing explicit mechanisms for handling linguistic information in end-to-end ASR models, I expect that the proposed approaches improve transcription accuracy while alleviating the heavy reliance on paired data inherent in the fully data-driven approach. The incorporation of comprehensive linguistic knowledge can also expand the applicability of end-to-end ASR models in other speech-related tasks that necessitate an understanding of language.

1.4 Dissertation Organization

This chapter provided an introduction to the objectives and significance of my research, as well as a review of the relevant technical trends associated with the topic. The rest of this dissertation begins with an overview of background knowledge about end-to-end ASR, followed by my work on incorporating top-down linguistic cues for end-to-end ASR (Figure 1-1). An overview of each chapter is as follows, along with references to the corresponding papers that were published as a result of my doctoral research.

- Chapter 2 reviews the fundamentals of end-to-end ASR, beginning with a comparison to traditional hybrid systems. Subsequently, I compare the principal modeling approaches employed in end-to-end ASR, with a particular focus on the distinctions in their formulations. Lastly, I explore neural network architectures utilized in sequence processing.

- Chapter 3 presents methods to hierarchically increase the abstraction level in linguistic outputs, with the goal of efficiently learning representations for sparse word-level units. In end-to-end ASR, models are expected to implicitly learn representations conducive to word prediction. However, this requires an extensive amount of training data to overcome the substantial abstraction gap between input acoustic signals and output linguistic tokens. To facilitate word-level representation learning, I first develop a hierarchical conditional model. The proposed model is trained by auxiliary CTC losses applied to intermediate layers, where the vocabulary size of each target subword sequence is gradually increased as the layers become close to the word-level output. Each level of sequence prediction is explicitly conditioned on the sequences predicted at previous levels, enabling the model to progressively construct word-level representations by considering a hierarchy of linguistic structures. Then, I enhance the generation capability of the hierarchical model by employing a refinement mechanism at each stage of intermediate prediction. This mechanism repeatedly uses the shared model parameters to refine its intermediate representations, which leads to an improvement in the overall performance of the model. Finally, I design an efficient pseudo-labeling-based algorithm for the proposed hierarchical model, which utilizes audio-only data within a semi-supervised learning framework to further boost the model performance. The hierarchical modeling approach was published as a conference paper at ICASSP 2022 (Higuchi et al., 2022a). The efficient semi-supervised learning algorithm using pseudo-labeling was introduced in a conference paper at Interspeech 2021 (Higuchi et al., 2021c) and a journal paper at JSTSP 2022 (Higuchi et al., 2022c), which was expanded into the hierarchical model in a publication at ICASSP 2023 (Higuchi et al., 2023c).

- Chapter 4 proposes to incorporate the concept of masked language modeling into end-to-end ASR, aiming to augment the model's ability to capture long-range linguistic contexts. To this end, I first propose joint training and decoding strategies that synergize CTC with masked language modeling. To mitigate the limitation of CTC in explicitly modeling dependencies between output tokens, contextual information derived from masked language modeling is used to enhance the performance of CTC-based ASR. The non-autoregressive nature shared by both CTC and masked language modeling also enables fast inference without compromising recognition accuracy. In addition to end-to-end ASR, the proposed framework shows promise for application to end-to-end speech translation tasks, adeptly handling semantic information vital for producing translated sequences. Then, I establish that the adoption of the masked language model mechanism offers benefits to other end-to-end ASR models. The proposed model architecture augments the transducer-based model by injecting explicit contextual linguistic cues into the speech encoding process, which is shown to push the limits of prior state-of-the-art results. Lastly, I demonstrate that masked language modeling is advantageous in acquiring representations beneficial for streaming end-to-end ASR models, allowing for the extraction of anticipated linguistic contexts from constrained speech input. This chapter includes conference papers published at Interspeech 2020 (Higuchi et al., 2020) and ICASSP 2021 (Higuchi et al., 2021b), which proposed the joint CTC and masked language modeling framework. The application to the transducer-based model was presented at ASRU 2023 (Higuchi et al., 2023d). The representation learning method for streaming end-to-end ASR was introduced in conference papers at APSIPA 2021 (Zhao et al., 2021) and EUSIPCO 2023 (Zhao et al., 2023).

- Chapter 5 focuses on the application of pre-trained masked language models (e.g., BERT) in end-to-end ASR, utilizing their versatile linguistic knowledge to guide the generation process of linguistic sequences. In the field of natural language processing, large-scale pre-training using vast amounts of text data has greatly advanced language models' ability to learn diverse aspects of linguistic information. Such capabilities are expected to enhance end-to-end ASR systems by empowering models to effectively interpret complex linguistic elements, which is important for choosing words that are both grammatically correct and contextually appropriate. To harness this potential, I first present a novel end-to-end ASR formulation that explicitly conditions BERT's contextualized word embeddings on the ASR process, adapting BERT for the CTC-based training and inference framework. By seamlessly infusing BERT knowledge into audio information, the proposed model improves over conventional approaches. Additionally, I demonstrate its potential application in end-to-end spoken language understanding tasks, which typically require more abstract linguistic processing. Subsequently, I introduce an extension of the proposed BERT-based model by implementing an additional transducer-based decoder. The decoder is trained using a vocabulary suitable for ASR training, aiming to bridge the gap between the text processed in end-to-end ASR and BERT. This is shown crucial as these models utilize distinct vocabularies and exhibit different text formats and styles, including variations in punctuation usage. This chapter covers a conference paper published at EMNLP 2023 (Higuchi et al., 2022d), which explored the use of BERT in end-to-end ASR. The further improvement using the additional decoder was presented at ICASSP 2023 (Higuchi et al., 2023b).

- Chapter 6 further delves into the integration of pre-trained language models in end-to-end ASR, with a specific emphasis on larger-sized and controllable language models. Modern large language models (e.g., ChatGPT) are capable of performing a wide range of linguistic tasks within zero-shot learning, guided by precise instructions or prompts to direct the text generation process toward the desired task. I explore using this zero-shot capability inherent in large language models to enhance end-to-end ASR. The proposed approach involves guiding a large language model to perform zero-shot grammatical error correction, thereby extracting linguistic information that contributes to improving ASR performance. The linguistic knowledge drawn from the large language model is then used to trigger the ASR decoding process, along with acoustic information, for achieving accurate sequence generation. The findings are reported in a preprint (Higuchi et al., 2023a).

- Chapter 7 concludes this dissertation by summarizing the studies presented and highlighting their core contributions. Additionally, I discuss future directions that are informed by the findings gained from my research.

2
Overview of End-to-End Speech Recognition

ASR is the process of mapping sequences, converting acoustic speech signals into corresponding written text. Unlike natural language processing (NLP) tasks (e.g., machine translation) that primarily focus on mapping discrete linguistic symbols, ASR presents unique properties due to the complex nature of continuous speech signals. The high variability inherent in speech poses challenges in developing accurate ASR systems, arising from various factors such as the diverse characteristics of individual speakers, a wide range of accents and dialects, inconsistencies in recording conditions, and varying levels of background noise. Meanwhile, speech contains paralinguistic information, such as fillers and laughter, which offer valuable insights into the speaker's mental state. This can contribute to a richer understanding of spoken language that cannot be achieved solely through linguistic symbols. Therefore, it is crucial to develop mechanisms in ASR systems that can adeptly handle the complexities of both speech and text, thereby enriching the process of linguistic understanding with additional layers of meaning.

Let $O = (\mathbf{o}_t \in \mathbb{R}^F | t = 1, \cdots, T)$ be an input sequence of length T, and $W = (w_l \in \mathcal{V} | l = 1, \cdots, L)$ be the corresponding output sequence of length L, where $\mathbf{o}_t$ is an F-dimensional acoustic feature (e.g., log-mel filterbanks) at frame t, w_l is an output token (e.g., a character, subword[1], or word) at position l, and $\mathcal{V}$ is a vocabulary. In general, the output length is much shorter than the input length (i.e., $L \ll T$). The objective of ASR is to identify the most probable

[1]This refers to a language unit that forms part of a larger word, typically obtained through algorithms like byte pair encoding (Sennrich et al., 2016).

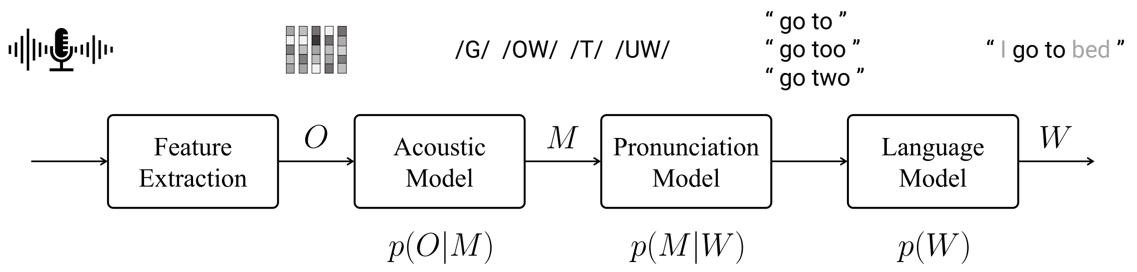


Figure 2-1: Overview of hybrid ASR system. The posterior probability distribution of ASR $p(W|O)$ is decomposed into three separate probabilistic models through the intermediary of a phoneme representation. These models include an acoustic model, a pronunciation model, and a language model.

output sequence $\hat{W}$ that matches the input sequence O, defined as

$$\hat{W} = \arg\max_{W \in \mathcal{V}^*} p(W|O), \tag{2.1}$$

where $\mathcal{V}^*$ represents a set of all possible token sequences. In modeling the posterior probability distribution of $p(W|O)$ in Eq. (2.1), there are two predominant approaches: a hybrid system (Figure 2-1) and an end-to-end system (Figure 2-2). This chapter primarily focuses on reviewing the end-to-end ASR system, beginning with a comparison to the hybrid system. It then discusses the principal frameworks of end-to-end ASR and explains the neural network architectures used in building end-to-end ASR models.

2.1 Hybrid ASR System

The traditional approach to ASR employs a hybrid system (Bourlard and Morgan, 1994), grounded in the probabilistic noisy channel model (Jelinek, 1998). As illustrated in Figure 2-1, this system breaks down the speech-to-text conversion process into several modules, including an acoustic model, a pronunciation model, and a language model. Each module is responsible for a specialized role, working in coordination to effectively achieve the complex task of converting speech into text.

The hybrid system factorizes the posterior distribution of $p(W|O)$ in Eq. (2.1) based on Bayes' theorem as

$$\hat{W} = \arg\max_{W} p(O|W)p(W). \tag{2.2}$$

Eq. (2.2) is further factorized by introducing a phoneme sequence $M = (m_i \in \mathcal{P} | i = 1, \cdots, J)$

and marginalizing $p(O|W)$ over all valid M for W as

$$
\begin{aligned}
\text{Eq. (2.2)} &= \arg\max_{W} \sum_{M} p(O, M|W)p(W), && (2.3)\\
&\approx \arg\max_{W} \sum_{M} p(O|M, \cancel{W})p(M|W)p(W), && (2.4)\\
&\approx \arg\max_{W} \left\{ \max_{M} p(O|M)p(M|W)p(W) \right\}, && (2.5)
\end{aligned}
$$

where the three posterior probability distributions $p(O|M)$, $p(M|W)$, and $p(W)$ are an acoustic model, a pronunciation model, and a language model, respectively. Eq. (2.4) makes a conditional independence assumption of W (indicated by a slash sign), which is a reasonable assumption for reducing the dependence in $p(O|M)$. Eq. (2.5) makes the Viterbi approximation for the summation over the phoneme sequences. The resulting formulation presented in Eq. (2.5) assumes that the input speech signal O is not directly dependent on the output token sequence W; instead, it is indirectly determined through the intermediary of the phoneme sequence M. As a result, an ASR system is implemented as a straightforward combination of the three simple sub-models.

Acoustic Model The acoustic model $p(O|M)$ represents the emission probability of the input speech signal O, given a phoneme sequence M. $p(O|M)$ can be further factorized by introducing a sequence of hidden Markov model (HMM) states $S = (s_t \in \{1, \cdots, J\}|t = 1, \cdots, T)$ (Gales and Young, 2008) as

$$
p(O|M) \approx \max_{S} p(O|S)p(S|M), \tag{2.6}
$$

which is derived similarly to Eq. (2.5) using Bayes' theorem and the conditional independence assumption. In Eq. (2.6), $p(S|M)$ is typically computed as the product of HMM state transition probabilities, which correspond to a phoneme sequence M. $p(O|S)$ is further factorized by using a probabilistic chain rule with a conditional independence assumption as

$$
\begin{aligned}
p(O|S) &= \prod_{t=1}^{T} p(\mathbf{o}_t|\mathbf{o}_1, \cdots, \mathbf{o}_{t-1}, S), && (2.7)\\
&\approx \prod_{t=1}^{T} p(\mathbf{o}_t|s_t). && (2.8)
\end{aligned}
$$

Before the deep learning era, the framewise likelihood function $p(\mathbf{o}_t|s_t)$ was commonly computed using Gaussian mixture models (GMMs) (Juang, 1985), an approach widely known as the GMM-HMM system. The current mainstream has shifted from the use of GMM to DNN, resulting in the widespread adoption of the hybrid DNN-HMM system (Bourlard and Morgan, 1994) due to its superior recognition performance (Mohamed et al., 2011; Dahl et al., 2011; Hinton et al., 2012).

In the DNN-HMM system, $p(\mathbf{o}_t|s_t)$ is computed using Bayes' theorem as

$$p(\mathbf{o}_t|s_t) = \frac{p(s_t|\mathbf{o}_t)p(\mathbf{o}_t)}{p(s_t)} \propto \frac{p(s_t|\mathbf{o}_t)}{p(s_t)}, \tag{2.9}$$

where the state posterior probability $p(s_t|\mathbf{o}_t)$ is modeled using a DNN, and $p(s_t)$ is the prior probability that can be obtained through the unigram count of target labels. To train the DNN model for state classification, a framewise alignment between O and S is required for assigning a specific target state label to each frame, which is often provided by a pre-trained GMM-HMM system. Alternatively, the acoustic model can be trained without this explicit alignment by utilizing sequence-level optimization techniques, such as lattice-free maximum mutual information (LF-MMI) training (Povey et al., 2016).

Pronunciation Model The pronunciation model $p(M|W)$ defines a mapping between a phoneme sequence M and a corresponding token sequence W, primarily words. Unlike the other sub-models in the hybrid ASR system, the development of the pronunciation model relies predominantly on human expertise. Specifically, the mapping between phonemes and words is typically based on manually created pronunciation dictionaries, such as the CMU Dictionary (CMUdict, 1993) for American English and the Julius dictionary for Japanese (Lee et al., 2001). These pre-defined dictionaries are often used to build a finite-state transducer for representing the conversion from a phoneme sequence to a word.

Language Model The language model $p(W)$ represents the prior probability of a token sequence W. $p(W)$ can be further factorized using a probabilistic chain rule as

$$p(W) = \prod_{l=1}^{L} p(w_l|w_1, \cdots, w_{l-1}). \tag{2.10}$$

With a conditional independence assumption, or an $(n-1)$th-order Markov assumption, Eq. (2.10) is approximated as an n-gram language model as

$$\text{Eq. (2.10)} \approx \prod_{l=1}^{L} p(w_l|w_{l-n+1}, \cdots, w_{l-1}), \tag{2.11}$$

where the likelihood of each token is estimated based on the occurrence of its preceding $n-1$ tokens in a training dataset. To prevent the model from assigning zero probabilities to unseen n-grams, smoothing techniques (Jelinek, 1980; Chen and Goodman, 1999) are applied to adjust the probability estimates. Eq. (2.10) can also be modeled directly without the conditional independence assumption by training an RNN to predict the next token given the previous token sequence (Mikolov et al., 2010; Sundermeyer et al., 2012). While this RNN-based language model (RNN-LM) allows for handling long-term dependencies, it makes the decoding process of an ASR system more complex and computationally intensive compared to the n-gram language

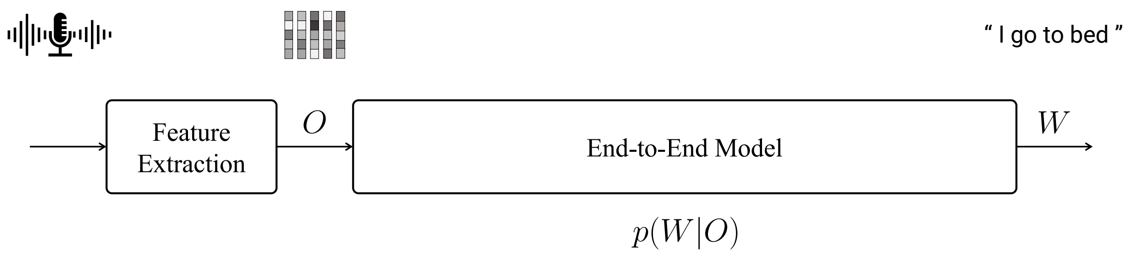


Figure 2-2: Overview of end-to-end ASR system. The posterior probability distribution of ASR $p(W|O)$ is directly modeled using a single deep neural network model.

model. Consequently, the RNN-LM is frequently used as a second-pass rescorer, evaluating the hypotheses generated by the n-gram language model in the first pass.

The modularity of the hybrid ASR system features the separate and independent training of its distinct sub-models. This design allows for the flexible customization of each individual model, making it especially advantageous for deploying ASR systems in production environments (Saon et al., 2017; Xiong et al., 2018). For example, the pronunciation model can be customized to accommodate new vocabularies by expanding its pronunciation dictionary with additional phoneme-word pairs. The language model can be trained using large amounts of text-only data, thereby enhancing the system's generalization capability or adapting it to a specific domain of interest. However, the independent optimization of the sub-models, each with different objectives, can result in incoherent optimization when considering the system as a whole. The heavy reliance on the conditional independence assumptions within each model (as described in the above formulations) may also be insufficient for accurately representing the complexities inherent in real-world data.

2.2 End-to-End ASR System

The end-to-end ASR system aims to achieve a direct mapping from speech to text using a single DNN model, as illustrated in Figure 2-2. In contrast to the hybrid system shown in Figure 2-1, the end-to-end approach does not require the independent training and integration of the sub-models. This greatly simplifies both the training and inference processes of ASR, reducing the complexity and costs associated with system development. Furthermore, the direct optimization of ASR can lead to superior performance compared to the hybrid systems (Chiu et al., 2018; Karita et al., 2019), particularly when an ample amount of training data is available.

This section reviews three principal approaches for directly modeling the posterior distribution of ASR $p(W|O)$ as defined in Eq. (2.1): CTC (Graves et al., 2006), transducer (Graves, 2012), and attention-based encoder-decoder (AED) (Chorowski et al., 2015; Chan et al., 2016). Figure 2-3 presents a comparison of end-to-end ASR models constructed by these approaches. The

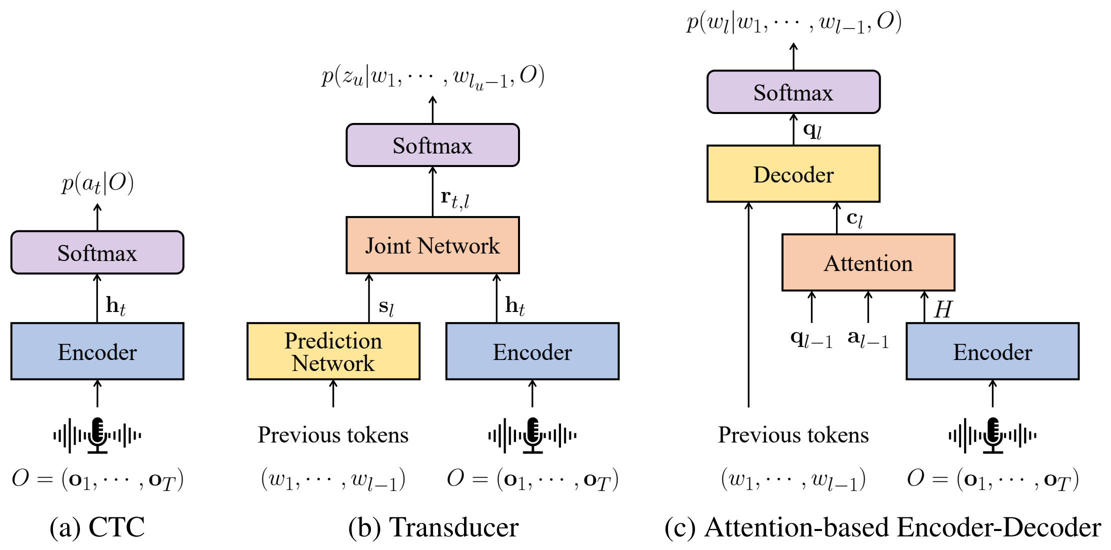


Figure 2-3: Comparison of major end-to-end ASR models. All the models share a similar encoder structure, but each has a unique decoder structure for estimating the output probability.

subsequent sections provide a detailed explanation of each model, with a specific focus on the differences in their probabilistic formulations.

2.2.1 Connectionist Temporal Classification (CTC)

CTC (Graves et al., 2006) formulates end-to-end ASR by evaluating all possible alignments between an input sequence O and the corresponding output sequence W. To align the sequences at the frame level, W is augmented by permitting repeated occurrences of the same token and inserting a special blank symbol <b> for representing "no output token" (e.g., silence). Let $A = (a_t \in \mathcal{V} \cup \{\texttt{<b>}\} | t = 1, \cdots, T)$ be an augmented sequence, which I refer to as an alignment sequence that is valid for aligning O and W. An example alignment sequence is shown in Figure 2-4, where the length of the input sequence T is 10 and the output token sequence W is $(\texttt{S}, \texttt{E}, \texttt{E})$. In this case, a feasible alignment sequence can be $A = (\texttt{<b>}, \texttt{S}, \texttt{<b>}, \texttt{<b>}, \texttt{E}, \texttt{E}, \texttt{E}, \texttt{<b>}, \texttt{E}, \texttt{E})$, which corresponds to W by removing repeated tokens and blank symbols.

With the introduction of the frame-level alignment sequence, CTC factorizes the posterior distribution of $p(W|O)$ from Eq. (2.1) as

$$p(W|O) \approx \sum_{A \in \mathcal{B}^{\text{ctc}^{-1}}(W)} p(W|A, \cancel{O}) p(A|O) \tag{2.12}$$

$$\approx \sum_{A \in \mathcal{B}^{\text{ctc}^{-1}}(W)} p(A|O), \tag{2.13}$$

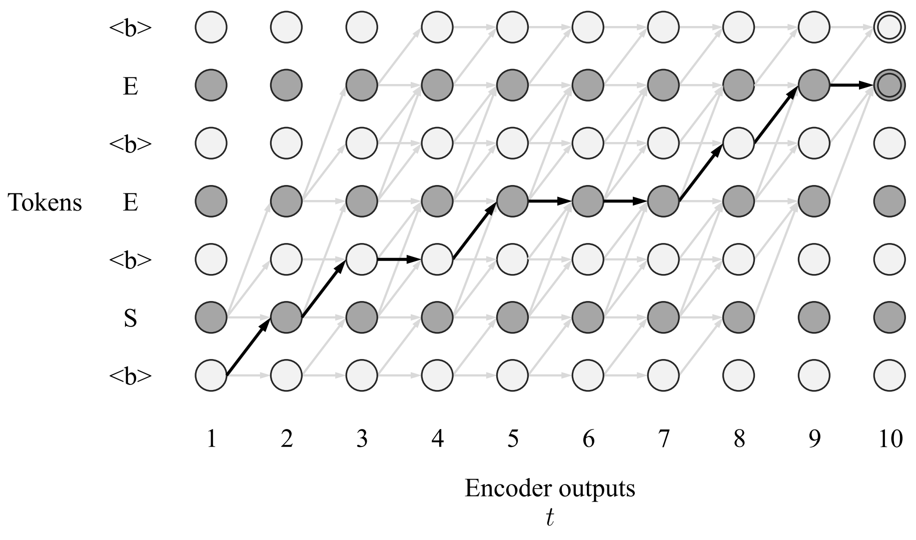


Figure 2-4: Example alignment sequence for CTC model. The arrows represent all possible alignment paths between the input encoder (speech) sequence and the target token sequence.

where $\mathcal{B}^{\mathsf{ctc}} : A \mapsto W$ is a collapsing function that removes repeated tokens and blank symbols in A, and ${\mathcal{B}^{\mathsf{ctc}}}^{-1}(W)$ represents a set of all possible alignments that are compatible with W. Similarly to Eq. (2.4), CTC assumes a conditional independence of O in Eq. (2.12), which is a reasonable assumption to simplify the dependency of $p(W|A)$. Furthermore, to obtain Eq. (2.13), the deterministic transformation from A to W is assumed (i.e., $p(W|A) = 1$), as W can be determined uniquely by the collapsing function. The posterior distribution $p(A|O)$ in Eq. (2.13) is further factorized by a probabilistic chain rule as

$$p(A|O) = \prod_{t=1}^{T} p(a_t|a_1, \cdots, a_{t-1}, O), \tag{2.14}$$

$$\approx \prod_{t=1}^{T} p(a_t|O). \tag{2.15}$$

In Eq. (2.15), CTC makes a conditional independence assumption between output tokens, where $p(A|O)$ is approximated as the product of token emission probabilities at each time frame. This resembles the HMM-based acoustic model in Eq. (2.8) in that both formulations assume conditional independence between observations at different time steps. To summarize, the posterior distribution modeled by CTC is defined by substituting Eq. (2.15) to Eq. (2.13) as

$$p^{\mathsf{ctc}}(W|O) \triangleq \sum_{A \in {\mathcal{B}^{\mathsf{ctc}}}^{-1}(W)} \prod_{t=1}^{T} p(a_t|O). \tag{2.16}$$

As illustrated in Figure 2-3(a), the token emission probability $p(a_t|O)$ in Eq. (2.16) is computed by building an encoder network that takes a speech sequence O as an input and produces a D^{enc}-dimensional hidden vector $\mathbf{h}_t$ at each time frame:

$$\mathbf{h}_t = \text{Encoder}_t(O) \in \mathbb{R}^{D^{\mathsf{enc}}}, \tag{2.17}$$

$$p(a_t|O) = \text{Softmax}\left(\text{Linear}_{D^{\mathsf{enc}} \to |\mathcal{V}|+1}(\mathbf{h}_t)\right) \in [0,1]^{|\mathcal{V}|+1}, \tag{2.18}$$

where $\text{Softmax}(\cdot)$ is a softmax activation function, $\text{Linear}_{D^{\mathsf{enc}} \to |\mathcal{V}|+1}(\cdot)$ is a linear layer for converting a D^{enc}-dimensional vector into a $(|\mathcal{V}|+1)$-dimensional vector[2], and $\text{Encoder}_t(\cdot)$ represents an output of the encoder network at frame t.

Inference

Substituting Eq. (2.16) into Eq. (2.1), CTC estimates the most probable token sequence $\hat{W}$ using the best path decoding algorithm (Graves et al., 2006). In this algorithm, the most probable alignment $\hat{A}$ is first obtained by concatenating the most active tokens at each time frame in H:

$$\hat{A} = \left(\hat{a}_t = \arg\max_{a_t} p(a_t|O) \,\middle|\, t = 1, \cdots, T \right). \tag{2.19}$$

$\hat{W}$ is then derived by applying the collapsing function to $\hat{A}$ as $\hat{W} = \mathcal{B}^{\mathsf{ctc}}(\hat{A})$. Although this greedy algorithm does not explicitly guarantee the identification of the most probable sequence, it has been empirically demonstrated to deliver satisfactory results, particularly when the model is trained using the recent advanced modeling techniques (Higuchi et al., 2021a). An alternative, more sophisticated approach is based on the prefix search decoding algorithm (Graves et al., 2006). This algorithm computes the probabilities of prefixes (i.e., a partial sequence in each hypothesis) during beam search decoding, where the summation over all possible alignments of each prefix is computed efficiently by a modified forward-backward algorithm. While prefix search decoding enables the exploration of multiple alignment paths, it generally yields results only slightly better than best path decoding. This is attributed to the "peaky" output distributions inherent in CTC-based models, meaning that these models tend to exhibit overconfidence in their alignment predictions. Therefore, prefix search decoding is commonly used in situations where the output probabilities of a CTC-based model are combined with those from a separate language model (Hannun et al., 2014b; Graves and Jaitly, 2014) or other end-to-end ASR models (Watanabe et al., 2017; Jeon and Kim, 2021).

[2] $|\mathcal{V}|+1$ indicates the addition of the blank symbol `<b>` to $\mathcal{V}$.

Training

The CTC-based model is optimized to minimize the negative log-likelihood of Eq. (2.16) as

$$\mathcal{L}^{\mathsf{ctc}} \triangleq -\log p^{\mathsf{ctc}}(W|O). \tag{2.20}$$

During the computation of $p^{\mathsf{ctc}}(W|O)$, the summation over all possible alignments is efficiently computed using dynamic programming (Viterbi or forward-backward algorithm).

2.2.2 Transducer

CTC estimates the distribution over alignment sequences solely based on the speech input (i.e., Eq. (2.15)), which can lead to an inaccurate capture of the conditional dependence of output tokens, often referred to as the multimodality problem (Gu et al., 2018). The transducer (Graves, 2012) overcomes this problem by making each token prediction explicitly conditioned on a previous sequence of output tokens, i.e., $(w_1, \cdots, w_{l-1})$. Similar to A defined by CTC, let $Z = (z_u \in \mathcal{V} \cup \{\texttt{<b>}\} | u = 1, \cdots, T+L)$ be an alignment sequence used in the transducer-based model. Here, the function of the blank symbol is slightly different from its role in CTC, which primarily serves as a placeholder for permitting the model to emit nothing at a particular time frame. In the transducer, the blank symbol is instead interpreted as a trigger for the model to proceed to the next frame, accumulating information over multiple frames until the model is ready to emit a non-blank symbol. Figure 2-4 shows an example alignment sequence of the transducer, representing $Z = (\texttt{<b>}, \texttt{<b>}, \texttt{S}, \texttt{<b>}, \texttt{<b>}, \texttt{<b>}, \texttt{<b>}, \texttt{E}, \texttt{E}, \texttt{<b>}, \texttt{<b>}, \texttt{<b>})$, where the length of the input sequence T is 9 and the output token sequence W is $(\texttt{S}, \texttt{E}, \texttt{E})$. Notice that the length of each transducer alignment is $T + L$, as the model does not consume acoustic frames when emitting non-blank tokens (or during vertical transitions). In the alignment sequence of CTC in Figure 2-4, the blank symbol is used to signify the separation of identical consecutive tokens, e.g., $\texttt{E}, \texttt{<b>}, \texttt{E}$. The transducer model, in contrast, does not necessitate such specific treatment.

Similarly to the derivation of Eq. (2.13), the transducer marginalizes the posterior distribution of $p(W|O)$ over all possible alignment sequences as

$$p(W|O) \approx \sum_{Z \in \mathcal{B}^{\mathsf{tra}^{-1}}(W)} p(W|Z, \cancel{O}) p(Z|O) \tag{2.21}$$

$$\approx \sum_{Z \in \mathcal{B}^{\mathsf{tra}^{-1}}(W)} p(Z|O), \tag{2.22}$$

where $\mathcal{B}^{\mathsf{tra}} : Z \mapsto W$ is a collapsing function for the transducer that removes all blank symbols from Z, and $\mathcal{B}^{\mathsf{tra}^{-1}}(W)$ is a set of all valid alignments for W. The posterior distribution $p(Z|O)$

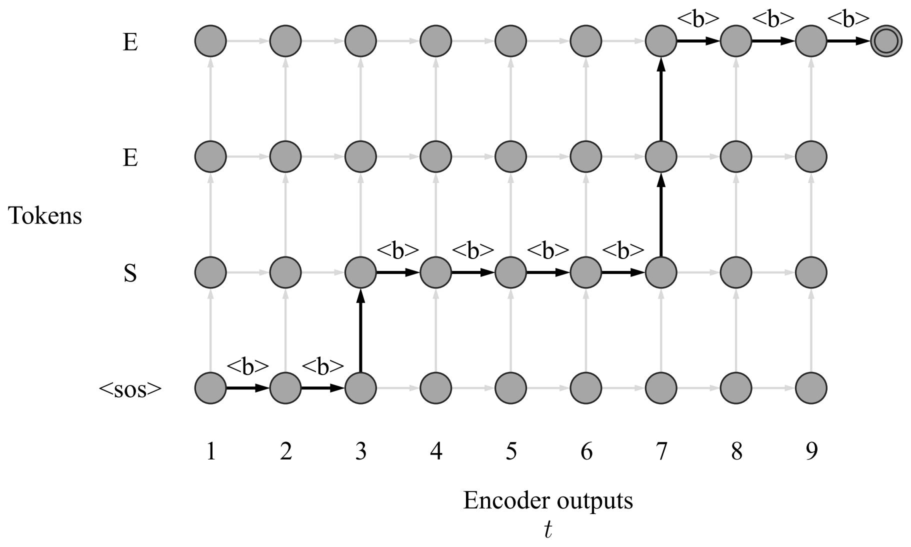


Figure 2-5: Example alignment sequence for transducer model. The arrows represent all possible alignment paths between the input encoder (speech) sequence and the target token sequence.

is further factorized without the conditional independence assumption (cf. Eq. (2.15)) as

$$p(Z|O) = \prod_{u=1}^{T+L} p(z_u|z_1, \cdots, z_{u-1}, O), \tag{2.23}$$

$$\approx \prod_{u=1}^{T+L} p(z_u|\underbrace{w_0, \cdots, w_{l_u-1}}_{=\mathcal{B}^{\mathsf{tra}}(z_1,\cdots,z_{u-1})}, O), \tag{2.24}$$

where $w_0 = \texttt{<sos>}$ is a special start-of-sentence symbol for indicating the beginning of a token sequence, and l_u is the number of non-blank tokens emitted up to an alignment index of u. To obtain Eq. (2.24), the transducer approximates $(z_1, \cdots, z_{u-1}) \approx (w_0, \cdots, w_{l_u-1})$, which is reasonable because W can be determined uniquely from Z using the collapsing function. As a result, the posterior distribution modeled by the transducer is defined by substituting Eq. (2.24) to Eq. (2.22) as

$$p^{\mathsf{tra}}(W|O) \triangleq \sum_{Z \in \mathcal{B}^{\mathsf{tra}-1}(W)} \prod_{u=1}^{T+L} p(z_u|w_0, \cdots, w_{l_u-1}, O). \tag{2.25}$$

The model structure for the transducer is illustrated in Figure 2-3(b), which consists of an encoder network, a prediction network, and a joint network. Using these networks, the token emission probability $p(z_u|w_0, \cdots, w_{l_u-1}, O)$ in Eq. (2.25) is computed as

$$\mathbf{h}_t = \text{Encoder}_t(O) \in \mathbb{R}^{D^{\text{enc}}}, \tag{2.26}$$

$$\mathbf{s}_{l_u} = \text{Prediction}(w_0, \cdots, w_{l_u-1}) \in \mathbb{R}^{D^{\text{pred}}}, \tag{2.27}$$

$$\mathbf{r}_{t,l_u} = \text{Joint}(\mathbf{h}_t, \mathbf{s}_{l_u}) \in \mathbb{R}^{D^{\text{joint}}}, \tag{2.28}$$

$$p(z_u|w_0, \cdots, w_{l_u-1}, O) = \text{Softmax}\left(\text{Linear}_{D^{\text{joint}} \to |\mathcal{V}|+1}(\mathbf{r}_{t,l_u})\right) \in [0,1]^{|\mathcal{V}|+1}. \tag{2.29}$$

In Eq. (2.26), the encoder network $\text{Encoder}_t(\cdot)$ takes the speech sequence O as an input and outputs a D^{enc}-dimensional vector $\mathbf{h}_t$ at each frame. In Eq. (2.27), the prediction network $\text{Prediction}(\cdot)$ embeds the previous output tokens $(w_0, \cdots, w_{l_u-1})$ to a D^{pred}-dimensional hidden vector $\mathbf{s}_{l_u}$. In Eq. (2.28), the joint network $\text{Joint}(\cdot)$ transforms each hidden vector from the encoder and prediction networks into a D^{joint}-dimensional hidden vector, which are then added together and passed through a hyperbolic tangent activation function. In Eq. (2.29), $\text{Linear}_{D^{\text{joint}} \to |\mathcal{V}|+1}(\cdot)$ is a linear layer for converting a D^{joint}-dimensional vector into a $(|\mathcal{V}|+1)$-dimensional vector. The introduction of the prediction network is the key difference from CTC (Figure 2-3(a) vs. 2-3(b)), which enables the explicit capture of causal dependencies in the output tokens.

Inference

Substituting Eq. (2.25) into Eq. (2.1), the transducer employs the beam search algorithm to identify the most probable token sequence $\hat{W}$ (Graves, 2012). The algorithm operates by exploring and evaluating multiple hypotheses, keeping the top B hypotheses based on probability scores at each prediction step. For the transducer model, the score of each hypothesis is calculated using Eq. (2.25), taking into account the sum of all possible alignments that are compatible with the current hypothesis. While prefix search decoding of CTC is also a viable choice for the transducer, beam search decoding has been shown to yield both faster and more effective results (Graves et al., 2013). All the experiments conducted in this dissertation follow the algorithm implementation as detailed in Boyer et al. (2021).

Training

The transducer model is optimized by minimizing the negative log-likelihood of Eq. (2.25) as

$$\mathcal{L}^{\text{tra}} \triangleq -\log p^{\text{tra}}(W|O). \tag{2.30}$$

Similarly to the calculation of the CTC objective in Eq. (2.20), the summation over alignments is efficiently computed using dynamic programming. For the previous tokens $(w_0, \cdots, w_{l_u-1})$ inputted into the prediction network in Eq. (2.27), the ground truth tokens are used in a teacher-forcing manner (Williams and Zipser, 1989). Compared with the other end-to-end ASR models,

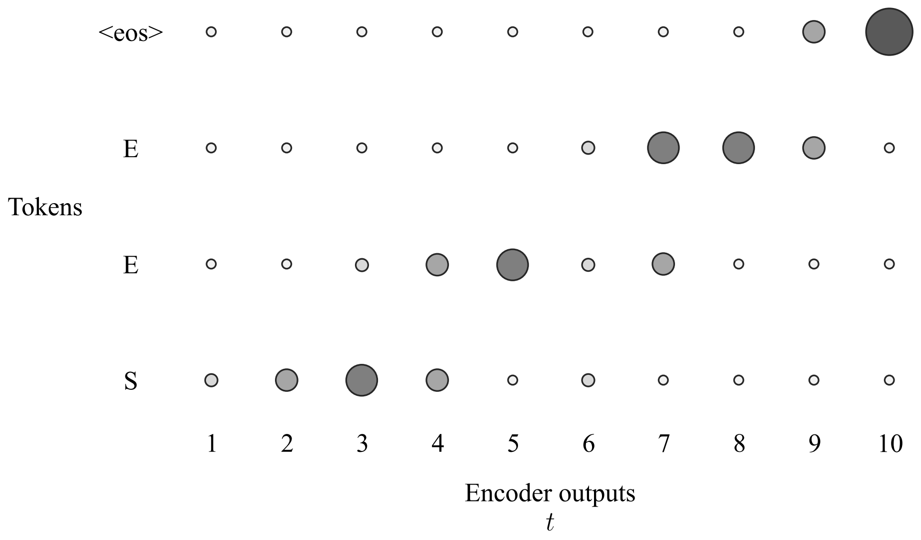


Figure 2-6: Example soft alignment learned by AED model. The size and color of each circle represent the magnitude of the attention weights computed between the input acoustic frames and each output token.

the training process of the transducer model tends to consume significantly large memory, requiring the total memory complexity of $\mathcal{O}(T \cdot L \cdot |\mathcal{V}|)$ for computing the loss. This can be mitigated to some extent through the optimization of the training algorithm (Bagby et al., 2018; Li et al., 2019b) or the adoption of advanced training techniques (Saon et al., 2021; Panchapagesan et al., 2021; Lee et al., 2022).

2.2.3 Attention-Based Encoder-Decoder (AED)

AED is an alternative approach to addressing end-to-end ASR (Chorowski et al., 2015; Chan et al., 2016; Bahdanau et al., 2016), functioning without the need for the explicit alignment modeling required by the CTC and transducer models. This is achieved by employing the attention mechanism (Bahdanau et al., 2014), which implicitly learns soft alignments between the input speech sequence and the output token sequence. An example alignment obtained by AED is visualized in Figure 2-6, where each circle represents the strength of the relationship, or the *attention weight*, between the input acoustic frames and each output token. Contrasting with the deterministic assignment of each acoustic frame to a specific token in the CTC and transducer alignments (as illustrated in Figures 2-4 and 2-5), AED allows for the flexibility of assigning a single acoustic frame to multiple tokens.

Unlike the formulations of CTC and the transducer, AED formulates end-to-end ASR without relying on any conditional independence assumptions, which enables more precise sequence modeling and has been shown to achieve higher recognition accuracy compared to the other end-to-end models (Prabhavalkar et al., 2017; Chiu et al., 2018). The posterior distribution $p(W|O)$ for Eq. (2.1) is estimated directly, using a probabilistic chain rule as

$$p^{\mathsf{aed}}(W|O) \triangleq \prod_{l=1}^{L+1} p(w_l|w_0, \cdots, w_{l-1}, O), \tag{2.31}$$

where $w_0 = \texttt{<sos>}$ is a start-of-sentence symbol. AED also predicts an additional end-of-sentence symbol $w_{L+1} = \texttt{<eos>}$ for identifying the completion of the output token sequence.

Figure 2-3(c) depicts the model structure for AED, which consists of an encoder network, a decoder network, and an attention network. The token emission probability $p(w_l|w_0, \cdots, w_{l-1}, O)$ in Eq. (2.31) is computed using these networks as

$$H = (\mathbf{h}_1, \cdots, \mathbf{h}_T) = \text{Encoder}(O) \in \mathbb{R}^{T \times D^{\mathsf{enc}}}, \tag{2.32}$$

$$\mathbf{q}_l = \text{AttentionDecoder}(w_0, \cdots, w_{l-1}, H) \in \mathbb{R}^{D^{\mathsf{dec}}}, \tag{2.33}$$

$$p(w_l|w_0, \cdots, w_{l-1}, O) = \text{Softmax}\left(\text{Linear}_{D^{\mathsf{dec}} \to |\mathcal{V}|+1}(\mathbf{q}_l)\right) \in [0, 1]^{|\mathcal{V}|+1}. \tag{2.34}$$

In Eq. (2.32), the encoder network Encoder$(\cdot)$ takes the speech sequence O as an input and outputs a sequence of D^{enc}-dimensional hidden vectors H. In Eq. (2.33), AttentionDecoder$(\cdot)$ indicates the combined attention and decoder networks, producing a D^{dec}-dimensional hidden vector $\mathbf{q}_l$ given the previous output tokens $(w_0, \cdots, w_{l-1})$ and the encoder output H. In Eq. (2.34), Linear$_{D^{\mathsf{dec}} \to |\mathcal{V}|+1}(\cdot)$ represents a linear layer for converting a D^{dec}-dimensional vector into a $(|\mathcal{V}| + 1)$-dimensional vector[3].

AttentionDecoder$(\cdot)$ in Eq. (2.33) computes the hidden vector $\mathbf{q}_l$ as

$$\mathbf{c}_l = \text{Attention}(\mathbf{q}_{l-1}, \mathbf{a}_{l-1}, H) \in \mathbb{R}^{D^{\mathsf{enc}}}, \tag{2.35}$$

$$\mathbf{q}_l = \text{Decoder}(\mathbf{c}_l, \mathbf{q}_{l-1}, w_{l-1}) \in \mathbb{R}^{D^{\mathsf{dec}}}, \tag{2.36}$$

In Eq. (2.35), Attention$(\cdot)$ is the attention network that produces a D^{enc}-dimensional context vector $\mathbf{c}_l$, using an attention mechanism to extract contextual acoustic representations useful for predicting the l-th token. In Eq. (2.36), Decoder$(\cdot)$ outputs a D^{dec}-dimensional hidden vector $\mathbf{q}_l$, using a recurrent network (e.g., LSTM) conditioned on a context vector $\mathbf{c}_l$, a previous hidden state $\mathbf{q}_{l-1}$, and a previously predicted token w_{l-1}. Using the location-aware attention mechanism (Chorowski et al., 2015) as an example, the attention network in Eq. (2.35) computes the

[3]$|\mathcal{V}| + 1$ indicates the addition of the end-of-sentence symbol <eos> to $\mathcal{V}$.

context vector $\mathbf{c}_l$ as

$$[\mathbf{f}_{l,1}, \cdots, \mathbf{f}_{l,T}] = \mathbf{K} * \mathbf{a}_{l-1}, \tag{2.37}$$

$$e_{l,t} = \mathbf{v}^\top \tanh(\mathbf{W}^{\mathsf{q}}\mathbf{q}_{l-1} + \mathbf{W}^{\mathsf{h}}\mathbf{h}_t + \mathbf{W}^{\mathsf{f}}\mathbf{f}_{l,t} + \mathbf{b}), \tag{2.38}$$

$$\mathbf{a}_l = \mathrm{Softmax}\left([e_{l,1}, \cdots, e_{l,T}]^\top\right), \tag{2.39}$$

$$\mathbf{c}_l = \sum_{t=1}^{T} a_{l,t}\mathbf{h}_t. \tag{2.40}$$

In Eq. (2.37), $*$ represents a one-dimensional convolution operation with a learnable kernel matrix $\mathbf{K}$, which convolves the previous attention weights $\mathbf{a}_{l-1}$ along the frame axis to produce the output feature $\mathbf{f}_{l,t}$ at each frame t. This convolution mechanism enables the attention network to focus more on local frames, considering that the alignment between input and output sequences is monotonic in ASR. Eq. (2.38) calculates the attention score $e_{l,t}$ for quantifying the relevance between the previous decoder state $\mathbf{q}_{l-1}$ and the encoder output $\mathbf{h}_t$ at frame t, where $\mathbf{v}$, $\mathbf{W}^{\mathsf{q}}$, $\mathbf{W}^{\mathsf{h}}$, $\mathbf{W}^{\mathsf{f}}$, and $\mathbf{b}$ are learnable parameters. Eq. (2.39) computes the attention weights $\mathbf{a}_l$ at each output step l, normalizing the scores over the entire frames using a softmax activation function. Finally, in Eq. (2.40), the context vector $\mathbf{c}_l$ is obtained at each output step l by calculating the weighted sum of the encoder outputs based on the attention weights.

AttentionDecoder$(\cdot)$ in Eq. (2.33) can alternatively be implemented using the Transformer decoder (Vaswani et al., 2017), which incorporates the self-attention mechanism within the decoder architecture (see Section 2.3.2).

Inference

Substituting Eq. (2.31) into Eq. (2.1), AED generally utilizes the beam search algorithm to find the most probable token sequence $\hat{W}$, keeping a subset of partial hypotheses with the top B probability scores (computed by Eq. (2.31)) at each prediction step. However, with the standard beam search, the AED model often experiences deletion and insertion errors due to the lack of explicit alignment information (Chorowski and Jaitly, 2017), allowing the attention mechanism to flexibly refer to any segment of the encoder output for token predictions (i.e., Eq. (2.35)). This issue can be mitigated by incorporating heuristics into the decoding process. One straightforward solution is to limit the length of a hypothesis by setting minimum and maximum length parameters, which are manually determined according to the length of an input sequence (i.e., $|O|$). The other solution is to add extra terms for adjusting the calculation of the probability score. The score of a hypothesis $\hat{W}$ is computed using additional heuristic terms as

$$\log p(\hat{W}|O) = \log p^{\mathsf{aed}}(\hat{W}|O) + \gamma|\hat{W}| + \eta \sum_{t=1}^{T}\left[\left(\sum_{l=1}^{L} a_{l,t}\right) > \tau\right], \tag{2.41}$$

where γ and η are tunable parameters for the corresponding terms. In Eq. (2.41), the second term serves as the length penalty to prevent the generation of overly long hypotheses (Chorowski et al., 2015). The third term is the coverage term for penalizing the excessive repetition of references to the same frames (Tu et al., 2016; Chorowski and Jaitly, 2017).

Training

The AED model is optimized by minimizing the negative log-likelihood of Eq. (2.31) as

$$\mathcal{L}^{\mathsf{aed}} \triangleq -\log p^{\mathsf{aed}}(W|O). \tag{2.42}$$

For the previous tokens $(w_0, \cdots, w_{l_u-1})$ fed into the decoder network in Eq. (2.33), the ground truth tokens are used in a teacher-forcing manner (Williams and Zipser, 1989). To mitigate the mismatch between training and inference conditions caused by teacher-forcing training, techniques such as scheduled sampling (Ranzato et al., 2016; Bengio et al., 2015) or minimum word error rate training (Prabhavalkar et al., 2018) are commonly used.

Joint CTC/Attention-based Model

The AED model can be enhanced by integrating the CTC framework (Kim et al., 2017; Watanabe et al., 2017) in a multi-task learning manner, applying an auxiliary CTC loss to the output of a shared encoder network. The computation of the CTC loss, with its left-to-right constraint in the forward-backward algorithm, aids the attention mechanism in effectively extracting a monotonic alignment between the input and output sequences. The objective function of the joint CTC and AED model is defined as a linear interpolation of $\mathcal{L}^{\mathsf{ctc}}$ from Eq. (2.20) and $\mathcal{L}^{\mathsf{aed}}$ from Eq. (2.42) as

$$\mathcal{L}^{\mathsf{ctc\text{-}aed}} = \lambda^{\mathsf{ctc\text{-}aed}}\mathcal{L}^{\mathsf{ctc}} + (1 - \lambda^{\mathsf{ctc\text{-}aed}})\mathcal{L}^{\mathsf{aed}}, \tag{2.43}$$

where $\lambda^{\mathsf{ctc\text{-}aed}}$ $(0 \leq \lambda^{\mathsf{ctc\text{-}aed}} \leq 1)$ is a tunable weight to control the balance between the two losses.

Similarly to the pure AED model, the joint model uses the beam search algorithm during inference, but additionally considers the score derived from CTC (Hori et al., 2017a). The score for a hypothesis $\hat{W}$ is defined as a log-linear interpolation of the posterior probabilities derived from both CTC and AED:

$$\log p(\hat{W}|O) = \xi \log p^{\mathsf{ctc}}(\hat{W}|O) + (1 - \xi) \log p^{\mathsf{aed}}(\hat{W}|O), \tag{2.44}$$

where ξ $(0 \leq \xi \leq 1)$ is a tunable weight to define the importance of each score. In order to synchronize the scores at the token level, the CTC score is calculated based on the prefix probability (Graves et al., 2006), which involves calculating the total probability of all token sequences that share the same prefix. Eq. (2.44) can also incorporate the penalty terms used in the pure

AED decoding process (as described in Eq. (2.41)). However, as the CTC probability serves as an effective regularization for achieving a robust alignment, these penalty terms are not commonly considered for the joint CTC and AED model.

2.3 Network Architectures

This section reviews network architectures utilized for implementing key network components (e.g., encoder network) within the end-to-end ASR models explained in the previous section.

2.3.1 Long Short-Term Memory (LSTM)

LSTM (Hochreiter and Schmidhuber, 1997) networks are an advanced form of RNNs that is designed to effectively process sequential data of variable lengths. The traditional RNNs often struggle with handling long sequences due to the vanishing or exploding gradient problem, where training becomes ineffective as gradients diminish or grow excessively. LSTMs address this issue by introducing a gating mechanism, which maintains the consistency of gradient flow from previous steps. This enhancement has improved the network's ability to capture longer-term dependencies, which is particularly crucial in language and speech-processing tasks. In the context of end-to-end ASR, the majority of the initial models have primarily utilized the LSTM-based networks to construct their network components (Graves and Jaitly, 2014; Chorowski et al., 2015; Chan et al., 2016). An RNN-LM (for modeling Eq. (2.10)) is typically trained using LSTMs, which is also used in end-to-end ASR to incorporate linguistic information into the decoding process (Hwang and Sung, 2017; Hori et al., 2017b; Kannan et al., 2018).

In an N^{lstm}-layer LSTM model, the j-th layer generates a D^{lstm}-dimensional hidden vector $\mathbf{s}_l^{(j)} \in \mathbb{R}^{D^{\mathsf{lstm}}}$ at state index l as

$$\mathbf{s}_l^{(j)} = \mathrm{LSTM}^{(j)}(\mathbf{s}_l^{(j-1)}, \mathbf{s}_{l-1}^{(j)}), \tag{2.45}$$

where $\mathbf{s}_l^{(j-1)}$ is the output from the preceding layer, and $\mathbf{s}_{l-1}^{(j)}$ represents the previous hidden state. In the case of language modeling, such as in the construction of RNN-LM or the prediction network of the transducer, the input of the initial layer $\mathbf{s}_l^{(0)}$ is obtained by embedding each token w_l in the output sequence W into a D^{lstm}-dimensional vector.

2.3.2 Transformer

Transformer (Vaswani et al., 2017) follows the AED structure, wherein both the encoder and decoder networks are built based on the multi-head self-attention mechanism. Unlike RNNs, which rely on recurrent connections to model contextual dependencies among inputs, Transformer computes attention matrices to capture these dependencies. This exclusive reliance on the attention mechanism has allowed Transformer-based models to outperform RNN-based models across a range of tasks, including end-to-end ASR (Dong et al., 2018; Karita et al., 2019; Zeyer et al., 2019). Additionally, Transformer permits a more parallel and efficient sequence processing, which is particularly advantageous for training on large-scale datasets (Devlin et al., 2019; Radford et al., 2018; Baevski et al., 2020; Hsu et al., 2021).

Scaled dot-product attention is the fundamental component of the Transformer architecture, which is defined as

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{D^{\mathsf{att}}}}\right)\mathbf{V}, \tag{2.46}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ represent query, key, and value matrices, respectively, and D^{att} denotes the dimension of the query and key matrices. In Eq. (2.46), the softmax function operates on each row of the attention score matrix $\mathbf{Q}\mathbf{K}^{\top}$, transforming the scores into a set of weights for the corresponding values. These weights are then used to compute a weighted sum of the values, yielding an output context vector for each query. Scaled dot-product attention modifies the standard dot-product computation by scaling down the attention score matrix with a factor of $\sqrt{D^{\mathsf{att}}}$. This scaling factor plays a crucial role in preventing the occurrence of small gradients, which can arise when large values are fed into the softmax operation.

To efficiently capture features from various representation subspaces at different positions, Transformer employs the multi-head attention mechanism. Equipped with N^{head} parallel heads, multi-head attention simultaneously applies the scaled dot-product attention mechanism to inputs that are projected into different subspaces. Given sequences of queries $X^{\mathsf{q}} \in \mathbb{R}^{T^{\mathsf{q}} \times D^{\mathsf{model}}}$, keys $X^{\mathsf{k}} \in \mathbb{R}^{T^{\mathsf{k}} \times D^{\mathsf{model}}}$, and values $X^{\mathsf{v}} \in \mathbb{R}^{T^{\mathsf{k}} \times D^{\mathsf{model}}}$, the output of the n-th attention head is computed using Eq. (2.46) as

$$\mathrm{Head}_n = \mathrm{Attention}(X^{\mathsf{q}}\mathbf{W}^{\mathsf{q}}_n, X^{\mathsf{k}}\mathbf{W}^{\mathsf{k}}_n, X^{\mathsf{v}}\mathbf{W}^{\mathsf{v}}_n), \tag{2.47}$$

where $\mathbf{W}^{\mathsf{q}}_n \in \mathbb{R}^{D^{\mathsf{model}} \times D^{\mathsf{att}}}$, $\mathbf{W}^{\mathsf{k}}_n \in \mathbb{R}^{D^{\mathsf{model}} \times D^{\mathsf{att}}}$, and $\mathbf{W}^{\mathsf{v}}_n \in \mathbb{R}^{D^{\mathsf{model}} \times D^{\mathsf{att}}}$ are learnable weight matrices for the projections. The multi-head attention (MHA) output is then obtained by concatenating and transforming the outputs from each attention head as

$$\mathrm{MHA}(X^{\mathsf{q}}, X^{\mathsf{k}}, X^{\mathsf{v}}) = \mathrm{Concat}(\mathrm{Head}_1, \cdots, \mathrm{Head}_{N^{\mathsf{head}}})\mathbf{W}^{\mathsf{o}}, \tag{2.48}$$

where $\mathbf{W}^{\mathsf{o}} \in \mathbb{R}^{N^{\mathsf{head}} D^{\mathsf{att}} \times D^{\mathsf{model}}}$ is a learnable weight matrix. Throughout the experiments conducted in this dissertation, I consistently configured D^{att} to $D^{\mathsf{model}}/N^{\mathsf{head}}$.

Figure 2-7 illustrates the overall architecture of Transformer, which is built upon the multi-head attention mechanism as defined in Eq. (2.48). The following provides detailed structures of the encoder and decoder networks within the Transformer model.

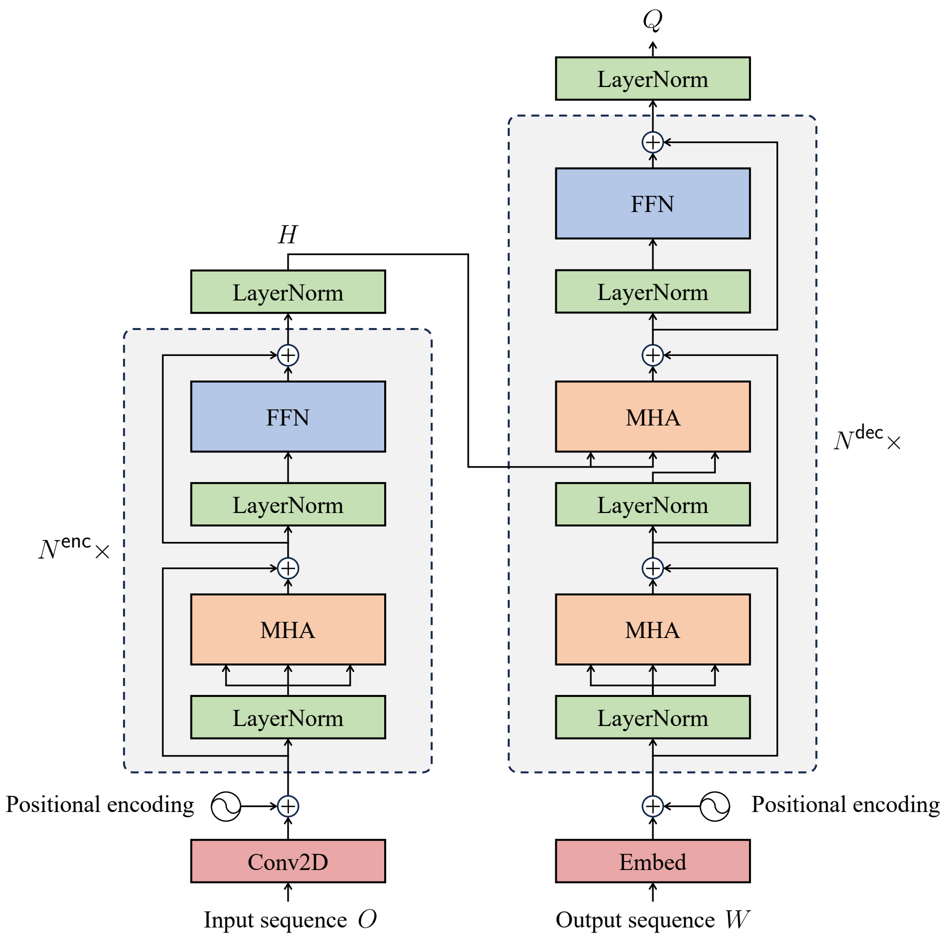


Figure 2-7: Block diagram of Transformer architecture, mainly consisting of multi-head attention (MHA) and feed-forward network (FFN) modules. Different from the original architecture (Vaswani et al., 2017), layer normalization (LayerNorm) is applied before each module. The multi-head attention module receives three inputs: key, value, and query from left to right.

Transformer Encoder

The Transformer encoder maps an input audio sequence $O \in \mathbb{R}^{T \times F}$ into a discriminative latent space as

$$H = \text{TransformerEncoder}(O) \in \mathbb{R}^{T' \times D^{\text{model}}}. \tag{2.49}$$

To obtain the hidden sequence H in Eq. (2.49), the encoder first applies two-dimensional convolution (Conv2D) layers to the input sequence (Hori et al., 2017b), which down-samples the sequence length to T' and transforms the feature dimensions to D^{model}. Positional encoding is then added to each frame of the down-sampled sequence, resulting in an initial hidden sequence,

$$H^{(0)} = \text{Conv2D}(O) + \text{PosEmb}(T') \in \mathbb{R}^{T' \times D^{\text{model}}}, \tag{2.50}$$

where the positional embeddings $\text{PosEmb}(T') \in \mathbb{R}^{T' \times D^{\mathsf{model}}}$ assign a unique representation to each frame within the down-sampled input sequence, thereby enabling the model to recognize and differentiate the specific location of each position. Transformer adopts the absolute positional encoding method, which computes each embedding based on sinusoidal functions as

$$\text{PosEmb}_{t,d}(T') = \begin{cases} \sin\left(\frac{t}{10000^{d/D^{\mathsf{model}}}}\right) & \text{if } d \text{ is even,} \\ \cos\left(\frac{t}{10000^{d/D^{\mathsf{model}}}}\right) & \text{if } d \text{ is odd,} \end{cases} \tag{2.51}$$

where t is the position in the T'-length sequence and d is the dimension of the D^{model}-dimensional vector. Subsequently, the initial sequence $H^{(0)}$ is fed into a stack of N^{enc} identical encoder blocks. The i-th encoder block takes input as a previous sequence $H^{(i-1)} \in \mathbb{R}^{T' \times D^{\mathsf{model}}}$ and outputs a current sequence $H^{(i)} \in \mathbb{R}^{T' \times D^{\mathsf{model}}}$ as

$$\bar{H}^{(i)} = H^{(i-1)} + \text{MHSA}^{(i)}(\text{LayerNorm}(H^{(i-1)})), \tag{2.52}$$

$$H^{(i)} = \bar{H}^{(i)} + \text{FFN}^{(i)}(\text{LayerNorm}(\bar{H}^{(i)})). \tag{2.53}$$

In Eq. (2.52), $\text{MHSA}(\cdot)$ indicates a multi-head self-attention module, which computes Eq. (2.48) using the same input source, i.e., $X^{\mathsf{q}} = X^{\mathsf{k}} = X^{\mathsf{v}}$. In Eq. (2.53), $\text{FFN}(\cdot)$ indicates a point-wise feed-forward module, which consists of two linear layers with an inner dimensionality of D^{ff} and a rectified linear unit (ReLU) activation applied between the layers. Slightly different from the original architecture presented in Vaswani et al. (2017), layer normalization (LayerNorm) (Ba et al., 2016) is applied before each module to stabilize the model training process (Wang et al., 2019; Baevski and Auli, 2019; Brown et al., 2020). Finally, the output of the Transformer encoder H is obtained by applying layer normalization to the output of the final block as

$$H = \text{LayerNorm}(H^{(N^{\mathsf{enc}})}). \tag{2.54}$$

Transformer Decoder

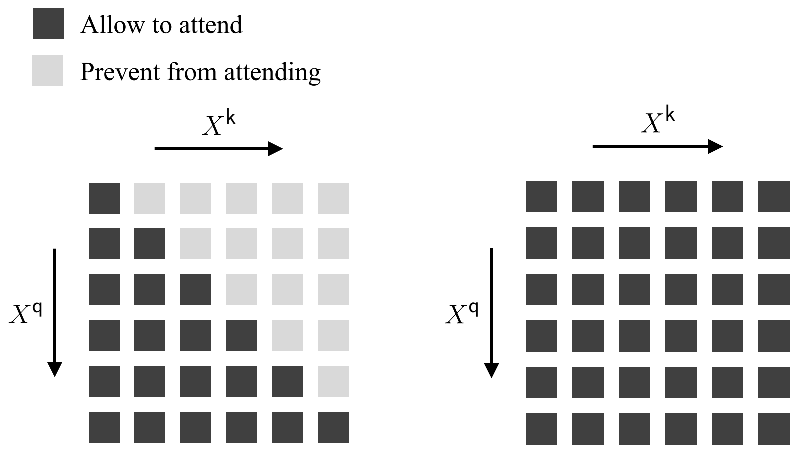
The Transformer decoder embeds an output token sequence $W \in \mathcal{V}^L$, utilizing the acoustic conditioning provided by the encoder output H as

$$Q = \text{TransformerDecoder}(W, H) \in \mathbb{R}^{L \times D^{\mathsf{model}}}. \tag{2.55}$$

In the embedding process of Eq. (2.55), the output sequence is initially transformed into vectors with a dimensionality of D^{model} as

$$Q^{(0)} = \text{Embed}(W) + \text{PosEmb}(L) \in \mathbb{R}^{L \times D^{\mathsf{model}}}, \tag{2.56}$$

where $\text{Embed}(\cdot)$ maps one-hot representations of output tokens into D^{model}-dimensional vectors,



(a) Causal masking (b) Full-context masking

Figure 2-8: Masking schemes for self-attention.

and $\text{PosEmb}(L) \in \mathbb{R}^{L \times D^{\mathsf{model}}}$ represents the absolute positional embeddings computed in the same manner as in Eq. (2.51). Then, the initial sequence $Q^{(0)}$ is inputted into a stack of N^{dec} identical decoder blocks. The j-th decoder block takes input as a previous sequence $Q^{(j-1)} \in \mathbb{R}^{L \times D^{\mathsf{model}}}$ and outputs a current sequence $Q^{(j)} \in \mathbb{R}^{L \times D^{\mathsf{model}}}$ as

$$\bar{Q}^{(j)} = Q^{(j-1)} + \text{MHSA}^{(j)}(\text{LayerNorm}(Q^{(j-1)})), \tag{2.57}$$

$$\bar{\bar{Q}}^{(j)} = \bar{Q}^{(j)} + \text{MHA}^{(j)}(\text{LayerNorm}(\bar{Q}^{(j)}), H, H), \tag{2.58}$$

$$Q^{(j)} = \bar{\bar{Q}}^{(j)} + \text{FFN}^{(j)}(\text{LayerNorm}(\bar{\bar{Q}}^{(j)})). \tag{2.59}$$

The multi-head attention mechanism in Eq. (2.58) takes queries from the decoder's hidden states, along with keys and values from the output of the encoder. This functions as the source-target attention (or cross-attention) mechanism, which facilitates the integration of encoder and decoder information. Instead of employing a separate neural network to calculate attention weights (i.e., Eq. (2.35)), the Transformer decoder derives these weights by performing a scaled dot product on its intermediate representations. Finally, the output of the Transformer decoder Q is obtained by applying layer normalization to the output of the final block as

$$Q = \text{LayerNorm}(Q^{(N^{\mathsf{dec}})}). \tag{2.60}$$

The Transformer decoder often adopts different masking strategies for its self-attention process to control how each token in a sequence attends to other tokens. Figure 2-8 depicts mask matrices that are used to determine whether to consider or ignore attention scores computed in Eq. (2.46). In causal attention masking (Figure 2-8(a)), each token is restricted to attending only to itself and preceding tokens. This approach is intended for learning the generative process of the output sequence (i.e., autoregressive language modeling), particularly applicable for the prediction network of the transducer (in Eq. (2.27)) or the decoder network of AED (in Eq. (2.33)). Full-context attention masking (Figure 2-8(b)) enables the attention mechanism to attend to every other token in a sequence, irrespective of their positions. This is especially helpful for capturing bidirectional information from both past and future contexts, facilitating the decoder's ability to process and interpret the complex interplay of tokens throughout the entire sentence (Devlin et al., 2019; Ghazvininejad et al., 2019).

2.3.3 Conformer

Conformer (Gulati et al., 2020) is a variant of the Transformer encoder architecture, which modifies each encoder block by incorporating a convolution layer. The self-attention mechanism is adept at modeling long-range global contexts. Meanwhile, the additional convolution layer enhances the capability of capturing local patterns within a sequence, which is particularly vital for feature extraction from speech signals. This synergistic effect has demonstrated promising results in various speech-related tasks (Guo et al., 2021).

Figure 2-9 presents a detailed view of the Conformer encoder architecture. In contrast to the Transformer encoder depicted in Figure 2-7, each encoder block of Conformer is enhanced with additional modules. A convolution module is introduced immediately after the multi-head self-attention module. Moreover, the self-attention and convolution modules are sandwiched between two feed-forward modules with half-step residual connections, a design inspired by the structure of Macaron-Net (Lu et al., 2020). As shown in the right side of Figure 2-9, the convolution module initiates with a gating mechanism (Dauphin et al., 2017), which is composed of a pointwise convolution layer with a gated linear unit (GLU). This is then followed by a one-dimensional (1D) depthwise convolution layer, batch normalization (BatchNorm), a swish activation function (Ramachandran et al., 2017), and another pointwise convolution layer. Under certain conditions, batch normalization within the convolution module can negatively impact the generalization ability of a Conformer-based model (Li et al., 2021; Liu et al., 2021; Kim and Lee, 2022; Higuchi et al., 2022b). This is a common issue in batch normalization (Ioffe, 2017), particularly when faced with constraints such as limited training data, high variability in data distributions, or significantly imbalanced datasets. In such cases, layer normalization or group normalization (Wu and He, 2018) can be utilized as stable alternatives in the convolution module.

The Conformer encoder transforms an input audio sequence $O \in \mathbb{R}^{T \times F}$ into a sequence of higher-level representations as

$$H = \text{ConformerEncoder}(O) \in \mathbb{R}^{T' \times D^{\text{model}}}. \tag{2.61}$$

Similar to the computational processes of the Transformer encoder block in Eqs. (2.52) and (2.53),

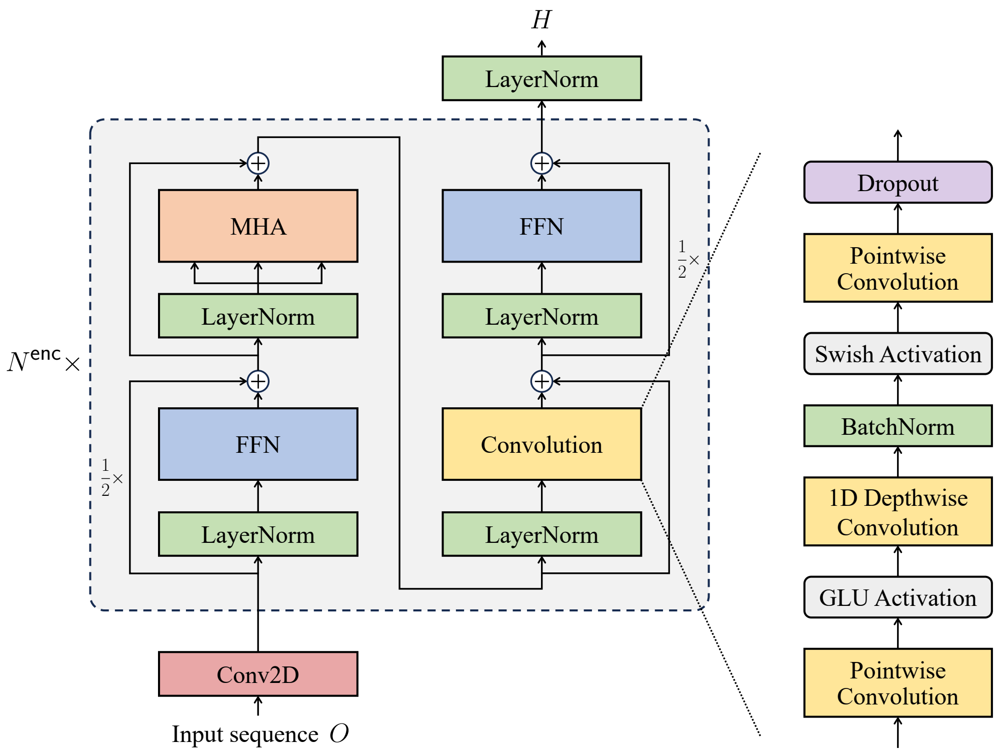


Figure 2-9: Block diagram of Conformer architecture. Conformer enhances the Transformer encoder by integrating the convolution module applied subsequent to the multi-head self-attention module.

the i-th Conformer encoder block takes input as a previous sequence $H^{(i-1)} \in \mathbb{R}^{T' \times D^{\text{model}}}$ and outputs a current sequence $H^{(i)} \in \mathbb{R}^{T' \times D^{\text{model}}}$ as

$$\bar{H}^{(i)} = H^{(i-1)} + \frac{1}{2}\text{FFN}^{(i)}(\text{LayerNorm}(H^{(i-1)})), \tag{2.62}$$

$$\bar{\bar{H}}^{(i)} = \bar{H}^{(i)} + \text{MHSA}^{(i)}(\text{LayerNorm}(\bar{H}^{(i)})), \tag{2.63}$$

$$\tilde{H}^{(i)} = \bar{\bar{H}}^{(i)} + \text{Convlution}^{(i)}(\text{LayerNorm}(\bar{\bar{H}}^{(i)})), \tag{2.64}$$

$$H^{(i)} = \tilde{H}^{(i)} + \frac{1}{2}\text{FFN}^{(i)}(\text{LayerNorm}(\tilde{H}^{(i)})). \tag{2.65}$$

Unlike the Transformer encoder, the Conformer encoder solely utilizes the convolution down-sampling layer to obtain the initial sequence, i.e., $H^{(0)} = \text{Conv2D}(O)$. Instead of adding absolute positional embeddings to the input, Conformer employs relative positional encoding (Dai et al., 2019) within each multi-head self-attention module, which helps the model to generalize better to varying input lengths.

2.4 Evaluation Metrics

This section outlines the key metrics utilized for evaluating the performance of ASR systems, including the word error rate (WER) and the real-time factor (RTF).

2.4.1 Word Error Rate (WER)

WER is a common metric used for assessing the recognition accuracy of ASR systems. WER measures the frequency of errors, including substitutions, deletions, and insertions, calculating the Levenshtein distance between a generated hypothesis and a reference transcription as

$$\text{WER}[\%] = \frac{N^{\mathsf{sub}} + N^{\mathsf{del}} + N^{\mathsf{ins}}}{N^{\mathsf{word}}} \times 100, \tag{2.66}$$

where N^{sub}, N^{del}, and N^{ins} are the number of substitution, deletion, and insertion errors, respectively, and N^{word} is the total number of words in the reference. In languages with the absence of explicit word boundaries in their written forms, such as Mandarin and Japanese, Eq. (2.66) is computed at the character level, which is known as the character error rate (CER).

2.4.2 Real-Time Factor (RTF)

RTF is a widely used metric for evaluating the decoding speed of ASR systems. RTF calculates the ratio of the time taken by a system to recognize speech to the actual duration of the speech itself as

$$\text{RTF} = \frac{\text{Time taken to recognize speech}}{\text{Duration of the speech segment}}. \tag{2.67}$$

An RTF value lower than 1 indicates that the system is processing speech faster than real time, thus delivering ASR results more promptly.

2.5 Summary

This chapter provided a comprehensive overview of end-to-end ASR, starting with a contrast against traditional hybrid systems. Then, the major modeling approaches were compared, emphasizing the unique aspects of their formulations. Lastly, various network architectures were introduced for developing end-to-end ASR models. Many parts of this chapter will be frequently referred to throughout the following chapters.

3 End-to-End Speech Recognition Guided by Hierarchy of Subword Construction

3.1 Introduction

In the field of deep learning, there has been a notable trend towards building end-to-end models, which rely entirely on data to learn the process of mapping inputs to outputs. These models are adept at implicitly acquiring representations that are intricately optimized for solving specific tasks. For example, in image classification, such models have successfully captured different types of shape features (Zeiler and Fergus, 2014). Similarly in language modeling, they have shown high proficiency in understanding complex syntactic structures (Peters et al., 2018). However, in the context of ASR, it can be more challenging for end-to-end models to learn effective representations purely from data. Having no access to segmentation or alignment information, end-to-end ASR models are required to predict word-level linguistic tokens directly from frame-level acoustic signals. This input-output gap in the level of abstraction makes it difficult to optimize end-to-end ASR, unless a large amount of data or a strong language model is accessible during training or inference (Zhang et al., 2020b; Irie et al., 2019).

With the aim of facilitating the representation learning process in end-to-end ASR, this chapter introduces a mechanism designed to gradually increase the abstraction level in linguistic information. Such a progressive approach is inspired by the traditional hybrid ASR system (outlined in Section 2.1), where the processing of information transitions stepwise from speech to text, i.e., speech $\rightarrow$ phonemes $\rightarrow$ words $\rightarrow$ text. The proposed concept involves training end-to-end ASR models to explicitly use lower-level linguistic elements to capture more abstract, higher-level linguistic information. This training approach is expected to enhance the models' ability to learn more precise representations for performing ASR, particularly aiding them in managing word-level outputs, which are notably sparse and challenging for end-to-end ASR models to process.

To this end, Section 3.2 proposes *hierarchical conditional* modeling of end-to-end ASR. The proposed model consists of multiple CTC (Graves et al., 2006) losses hierarchically applied to the intermediate and last layers, inspired by previous studies (Fernández et al., 2007; Rao and Sak, 2017; Toshniwal et al., 2017; Sanabria and Metze, 2018; Krishna et al., 2018; Tjandra et al., 2020; Lee and Watanabe, 2021). Each loss calculation targets sequences with a different granularity of linguistic information: sequences with lower abstraction levels are predicted from the intermediate layers, and a word-level sequence is predicted from the last layer. Specifically, I focus on subwords (n-gram characters) (Sennrich et al., 2016) and increase the vocabulary size to word-level as the model layer becomes close to the output (e.g., 256 $\rightarrow$ 2k $\rightarrow$ 16k). In addition to this hierarchical structure, the model is trained to predict each sequence at a specific level of abstraction, while receiving explicit conditioning from sequences predicted at lower levels. This enables the model to maintain subword information attributed to composing the higher-level sequence.

Section 3.3 further enhances the proposed hierarchical conditional model by designing a refinement mechanism within each intermediate prediction. Specifically, I build a Transformer-based model (Vaswani et al., 2017) that incorporates recursive operations, which has been well-studied in various fields including NLP (Dehghani et al., 2019; Bai et al., 2019; Lan et al., 2020), computer vision (Shen et al., 2022), and ASR (Li et al., 2019c; Chi et al., 2021b; Komatsu, 2022). This model involves the repeated use of Transformer layers with shared parameters, iteratively refining its representations without the need for extra parameters. By employing the recursive operation for each sequence prediction, I expect that the model accurately estimates the intermediate sequences, which in turn should improve overall performance. Additionally, I implement a conditioning feedback technique into the recursive process, enabling the model to explicitly refine its output based on its previous predictions.

Section 3.4 introduces a semi-supervised learning approach designed to enhance the performance of the hierarchical model. To efficiently train the model using unlabeled speech-only data, I propose to extend the hierarchical training process to a pseudo-labeling-based framework (Lee, 2013), a simple yet effective semi-supervised learning method commonly adopted for ASR (Li et al., 2019a; Kahn et al., 2020a; Masumura et al., 2020; Weninger et al., 2020; Hsu et al., 2020; Xu et al., 2020; Chen et al., 2020; Park et al., 2020b; Likhomanenko et al., 2021; Moritz et al., 2021). In typical pseudo-labeling, a teacher (seed) model is first trained on labeled data and used to generate pseudo-labels by transcribing unlabeled data. A student model is then trained using the labeled and pseudo-labeled data, to perform better than the teacher. The proposed approach generates multiple pseudo-labels from the intermediate layers of the hierarchical model. Here, linguis-

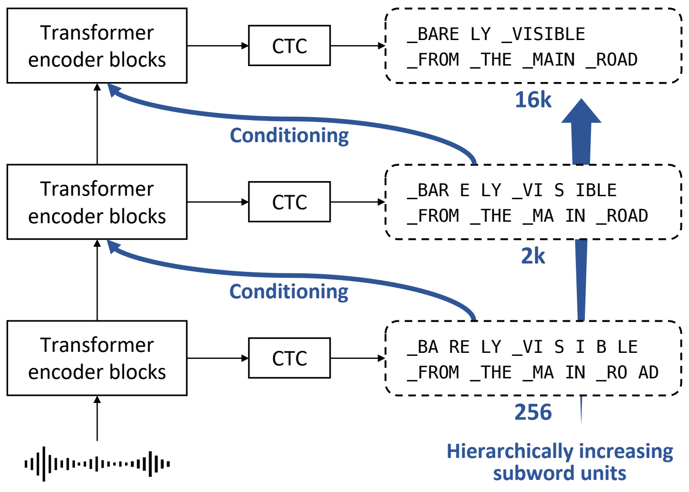

Transformer encoder blocks
CTC
_BARE LY _VISIBLE _FROM _THE _MAIN _ROAD
16k
Conditioning
Transformer encoder blocks
CTC
_BAR E LY _VI S IBLE _FROM _THE _MA IN _ROAD
2k
Conditioning
Transformer encoder blocks
CTC
_BA RE LY _VI S I B LE _FROM _THE _MA IN _RO AD
256
Hierarchically increasing subword units

Figure 3-1: Proposed hierarchical conditional CTC for end-to-end ASR. The model employs multiple CTC losses applied to intermediate layers, with the output linguistic unit gradually increasing towards the last layer. The granularity of the output units is adjusted by modifying the subword vocabulary size. This figure is reproduced from my conference paper (Higuchi et al., 2022a)

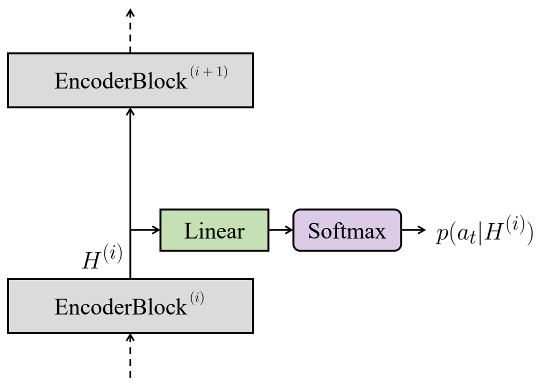
tic information captured at varying levels of granularity is expected to facilitate semi-supervised learning by providing additional training signals.

3.2 Hierarchical Conditional ASR with CTC and Multi-Granular Subword Units

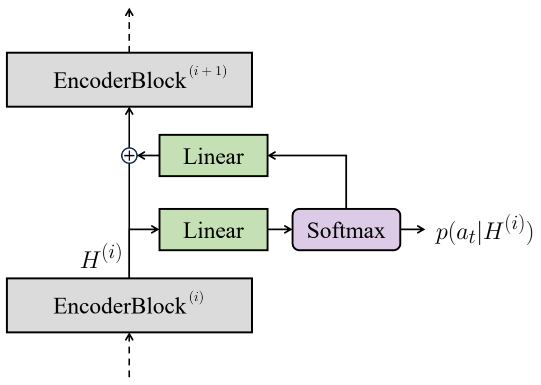
This section proposes hierarchical conditional CTC (HC-CTC), designed to facilitate the representation learning process for end-to-end ASR by increasing the granularity of output linguistic units in a progressive manner. Figure 3-1 illustrates the concept of the proposed HC-CTC model. The model applies multiple CTC losses to intermediate layers, calculated using different sizes of subword vocabularies. Each loss calculation is explicitly conditioned on the sequences predicted at a lower level, which is expected to help the sequence predictions at higher levels.

3.2.1 Intermediate CTC Regularization

The proposed approach is based on the intermediate CTC method (Tjandra et al., 2020; Lee and Watanabe, 2021). Intermediate CTC serves as a regularization technique for the training of a CTC-based end-to-end ASR model, applying auxiliary CTC losses to the intermediate model layers. Considering a model based on the Transformer encoder (from Eq. (2.49)) or the Conformer encoder (from Eq. (2.61)), an auxiliary CTC loss is computed similarly to the CTC loss $\mathcal{L}^{\text{ctc}}$



(a) Intermediate CTC



(b) Self-conditioned CTC

Figure 3-2: Schematic diagrams of intermediate CTC techniques.

(defined in Eq. (2.20)) as

$$p(a_t|H^{(i)}) = \text{Softmax}\left(\text{Linear}_{D^{\text{enc}}\to|\mathcal{V}|+1}(\mathbf{h}_t^{(i)})\right) \in [0,1]^{|\mathcal{V}|+1}, \tag{3.1}$$

$$\mathcal{L}^{\text{ctc}}(W|H^{(i)}) = -\log \sum_{A\in\mathcal{B}^{\text{ctc}-1}(W)} \prod_{t=1}^{T'} p(a_t|H^{(i)}), \tag{3.2}$$

where $H^{(i)} = (\mathbf{h}_t^{(i)}|t=1,\cdots,T')$ is the output of the i-th encoder layer (e.g., Eq. (2.53)). When using the final output of the encoder layer H (e.g., Eq. (2.54)), Eq. (3.2) becomes equivalent to the standard CTC loss, i.e., $\mathcal{L}^{\text{ctc}} = \mathcal{L}^{\text{ctc}}(W|H)$. The overall loss for an intermediate CTC-based model is defined by combining $\mathcal{L}^{\text{ctc}}$ and the loss computed in Eq. (3.2) as

$$\mathcal{L}^{\text{i-ctc}} = \frac{1}{|\mathcal{I}|+1}\left\{\mathcal{L}^{\text{ctc}} + \sum_{i\in\mathcal{I}} \mathcal{L}^{\text{ctc}}(W|H^{(i)})\right\}, \tag{3.3}$$

where $\mathcal{I} = \{i \mid 1 \le i < N^{\text{enc}}\}$ is a set of layer indices where the CTC losses are computed, and I equally distribute the weight across the losses.

Self-Conditioned CTC

Intermediate CTC can be further extended by introducing the self-conditioning technique (Nozaki and Komatsu, 2021), which conditions the encoder network using a sequence predicted at each intermediate layer. Specifically, after calculating the framewise probability distribution in Eq. (3.1), the output of the i-the encoder layer is updated as

$$\mathbf{h}_t^{(i)} \leftarrow \mathbf{h}_t^{(i)} + \text{Linear}_{|\mathcal{V}|+1\to D^{\text{model}}}(p(a_t|H^{(i)})), \tag{3.4}$$

where $\mathrm{Linear}_{|\mathcal{V}|+1\rightarrow D^{\mathsf{model}}}(\cdot)$ is a linear layer for mapping the probability to a D^{model}-dimensional vector. With this modification applied to the calculation processes in Eqs. (3.1) and (3.2), the overall loss for a self-conditioned CTC-based model is defined similarly to Eq. (3.3) as

$$\mathcal{L}^{\mathsf{sc\text{-}ctc}} = \frac{1}{|\mathcal{I}|+1}\left\{\mathcal{L}^{\mathsf{ctc}} + \sum_{i\in\mathcal{I}} \mathcal{L}^{\mathsf{ctc}}(W|H^{(i)})\right\}. \tag{3.5}$$

Figure 3-2 compares this self-conditioned CTC method to intermediate CTC. In addition to incorporating the intermediate loss, self-conditioned CTC feeds the predicted sequence back into the forward calculation of the encoder network, explicitly adding the framewise probabilities to the current encoder states. This provides the subsequent loss calculation with output contexts, which has been demonstrated to be effective in relaxing the conditional independence assumption in CTC.

3.2.2 Subword Segmentation

For tokenizing text sequences, subword segmentation is a widely used approach for alleviating the out-of-vocabulary (OOV) problem (Sennrich et al., 2016), where words in a sentence are split into subword units. In the general algorithm for building a subword vocabulary, pairs of subword units are repeatedly merged on the basis of the frequency appearing in a text corpus. The iteration stops when the vocabulary reaches an arbitrary size.

The proposed approach employs subwords for tokenizing ASR transcriptions. As opposed to characters, subwords can provide the model with shorter output sequences, thus reducing the difficulty of modeling the dependency between outputs. This can be especially important for CTC-based modeling with the conditional independence assumption. However, it should be noted that increasing the subword vocabulary size makes a sequence close to word-level and potentially lead to the data-sparsity problem (Soltau et al., 2017).

3.2.3 Hierarchical Conditional CTC (HC-CTC)

Figure 3-1 represents an overview of the proposed HC-CTC for training end-to-end ASR. It is similar to the intermediate CTC method, but the granularity of subword units is gradually increased to the word level as the sequence transduction proceeds in the encoder layers. Let $W^{(i)} = (w_l^{(i)} \in \mathcal{V}^{(i)}|l = 1, \dots, L^{(i)})$ be an $L^{(i)}$-length target subword sequence for calculating the i-th intermediate loss, which is generated by the corresponding subword tokenizer with a vocabulary of $\mathcal{V}^{(i)}$. HC-CTC hierarchically increases the vocabulary size, as the position of the CTC loss becomes close to the output layer (i.e., $|\mathcal{V}^{(<i)}| < |\mathcal{V}^{(i)}|$). Given the target sequences with different units, the intermediate loss of the proposed model is defined by modifying Eq. (3.2) as

$$\mathcal{L}^{\mathsf{ctc}}(W^{(i)}|H^{(i)}) = -\log \sum_{A^{(i)} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{(i)})} \prod_{t=1}^{T'} p(a_t^{(i)}|H^{(i)}), \tag{3.6}$$

where $A^{(i)} = (a_t^{(i)} \in \mathcal{V}^{(i)} \cup \{\text{<b>}\} | t = 1, \cdots, T')$ is an alignment sequence compatiable with $W^{(i)}$. With the self-conditioning mechanism realized by Eq. (3.4), the overall loss for an HC-CTC-based model is defined as

$$\mathcal{L}^{\mathsf{hc\text{-}ctc}} = \frac{1}{|\mathcal{I}|+1} \left\{ \mathcal{L}^{\mathsf{ctc}} + \sum_{i \in \mathcal{I}} \mathcal{L}^{\mathsf{ctc}}(W^{(i)}|H^{(i)}) \right\}. \tag{3.7}$$

The training based on Eq. (3.7) conditions each loss calculation on previously predicted sequences with lower levels of subword units.

In the proposed HC-CTC model, the word-level recognition is achieved by progressively integrating subwords in a fine-to-coarse manner. By having the shallower layers predict frequent subwords with small units and the deeper layers predict sparse subwords with large units, the model is expected to use a hierarchy of linguistic structures and effectively learn word-level representations.

3.2.4 Conventional Model with Parallel CTC Losses

To verify the effectiveness of the proposed model with the hierarchical structure, I also consider training with CTC losses applied in parallel to the final layer, as it has been shown to be effective in several studies (Li et al., 2017; Sanabria and Metze, 2018; Kremer et al., 2018; Heba et al., 2019). The objective for the parallel CTC losses is defined by modifying Eq. (3.7) as

$$\mathcal{L}^{\mathsf{para\text{-}ctc}} = \frac{1}{|\mathcal{I}|+1} \left\{ \mathcal{L}^{\mathsf{ctc}} + \sum_{i \in \mathcal{I}} \mathcal{L}^{\mathsf{ctc}}(W^{(i)}|H) \right\}. \tag{3.8}$$

Such training with parallel CTC losses treats the predictions of multi-granular sequences equally, where finer subword predictions provide an inductive bias to promote coarse word-level modeling (Kremer et al., 2018).

3.2.5 Relationship to Prior Work on Hierarchical End-to-End ASR

Several studies have explored introducing auxiliary CTC losses to intermediate model layers and demonstrated its effectiveness in improving various end-to-end ASR systems, based on the AED (Kim et al., 2017; Moriya et al., 2018), transducer (Jeon and Kim, 2021), and CTC (Zweig et al., 2017; Tjandra et al., 2020; Chi et al., 2021a; Lee and Watanabe, 2021) models. For the

CTC-based system, hierarchically applying low-level supervision (e.g., phonemes) to the intermediate CTC losses has been shown to improve a primary CTC loss with higher-level recognition (Fernández et al., 2007; Toshniwal et al., 2017; Rao and Sak, 2017; Krishna et al., 2018; Sanabria and Metze, 2018). The proposed model can be considered an extension of these hierarchical CTC-based models. However, it differs from prior work in the following perspectives. 1) Each CTC loss is explicitly conditioned on the sequences predicted previously at lower abstraction levels. In this way, the model is expected to maintain subwords that contribute to composing a word-level sequence and promote the CTC training with conditional dependencies (Nozaki and Komatsu, 2021). 2) Given that, in recent studies (Tjandra et al., 2020; Lee and Watanabe, 2021), the intermediate CTC losses are effective even without the hierarchical supervision, I carefully conduct a comparative experiment and further analyze the effectiveness of hierarchical modeling. 3) Target sequences are tokenized only using subwords, which does not require additional labeling effort and is easy to control the granularity of target sequences. 4) The models are evaluated using the recent state-of-the-art architectures (i.e., Transformer (Vaswani et al., 2017) and Conformer (Gulati et al., 2020)).

3.2.6 Experimental Setting

Data

The experiments were carried out using the LibriSpeech (LS) and TED-LIUM2 (TED2) datasets. For LS, models were trained using the 100-hour subset (LS-100) or the 960-hour full set (LS-960). See Appendix A for corpus details. As input speech features, I extracted 80 mel-scale filterbank coefficients with three-dimensional pitch features using Kaldi (Povey et al., 2011), which were augmented by speed perturbation (Ko et al., 2015) and SpecAugment (Park et al., 2019). I used SentencePiece (Kudo, 2018) to construct subword vocabularies for each dataset.

Evaluated Models

CTC denotes a standard CTC-based model trained with $\mathcal{L}^{\text{ctc}}$ from Eq. (2.20) (Graves and Jaitly, 2014). **SC-CTC** is a conventional model trained with the intermediate CTC losses (Tjandra et al., 2020; Lee and Watanabe, 2021) and the self-conditioning mechanism (Nozaki and Komatsu, 2021) defined by $\mathcal{L}^{\text{sc-ctc}}$ in Eq. (3.5). **HC-CTC** is the proposed hierarchical conditional model trained with $\mathcal{L}^{\text{hc-ctc}}$ from Eq. (3.7). **ParaCTC** is a conventional model trained with the parallel CTC losses (Li et al., 2017; Sanabria and Metze, 2018; Kremer et al., 2018) defined by $\mathcal{L}^{\text{para-ctc}}$ in Eq. (3.8).

Training and Decoding Configurations

The ESPnet toolkit (Watanabe et al., 2018) was used for conducting the experiments. I used the Transformer (Vaswani et al., 2017) architecture to train the above models, which consisted of two Conv2D layers followed by a stack of $N^{\mathsf{enc}} = 18$ encoder layers. The number of heads N^{head}, dimension of a self-attention layer D^{model}, and dimension of a feed-forward network D^{ff} were set to 4, 256, and 2048, respectively. I also trained the models using the Conformer architecture (Gulati et al., 2020), which had a kernel size of 15 and the same configurations as the Transformer-based models, except D^{ff} was set to 1024. The models were trained up to 100 epochs, using the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, and Noam learning rate scheduling (Vaswani et al., 2017). Warm-up steps and a learning rate factor were set to 25k and 5.0, respectively. SC-CTC and HC-CTC had a total of 3 CTC losses applied to the final layer and $\mathcal{I} = \{6, 12\}$. The output vocabulary sizes $|\mathcal{V}|$ for LS-100, LS-960, and TED2 were set to 16384, 32768, and 16384, respectively. Each vocabulary size was determined on the basis of the maximum number that could be set using SentencePiece, which can be large enough to be considered as word-level. SC-CTC had intermediate losses with the same vocabulary size as the output. For HC-CTC and ParaCTC, ($|\mathcal{V}^{(6)}|$, $|\mathcal{V}^{(12)}|$) was set to (256, 2048) for LS-100 and TED2, and (512, 4096) for LS-960. After training, a final model was obtained by averaging model parameters over 10 to 20 checkpoints with the best validation performance. During decoding, I did not use any language model and carried out the best path decoding of CTC (see Section 2.2.1). All the codes and recipes are publicly available to ensure reproducibility.[1]

3.2.7 Main Results

Table 3.1 lists the results on LS-100, LS-960, and TED2 in terms of the WER. Looking at the Transformer results, all the models trained with multiple CTC losses led to an improvement over the standard CTC-based model. Especially, SC-CTC and HC-CTC significantly reduced the WER on all of the tasks. On LS-100, HC-CTC showed a clear improvement over SC-CTC, indicating the effectiveness of hierarchically increasing subword units. In contrast, on LS-960 and TED2 with more data, the performance gap was reduced, and HC-CTC performed slightly better than SC-CTC. Therefore, it can be concluded that the proposed model is particularly effective for smaller-scale data, in which the word-level units are likely to become sparser. SC-CTC was capable of handling word-level units when there was a sufficient amount of data. However, the large vocabulary-sized softmax calculation (in Eq. (3.1)) led to a severe slow-down of the SC-CTC training and inference processes. HC-CTC, on the other hand, was able to perform faster training and inference, using finer units for the losses from intermediate layers. Due to the same reason regarding the softmax calculation, the model size of HC-CTC was much smaller than that of SC-CTC (e.g., 36.4M vs. 67.6M on LS-960). By comparing HC-CTC with ParaCTC, HC-CTC

[1] `https://github.com/YosukeHiguchi/espnet/tree/hierctc`

Table 3.1: WERs on LibriSpeech-{100h, 960h} and TED-LIUM2 tasks. Output subword vocabulary size was set to 16k for LibriSpeech-100h and TED-LIUM2, and 32k for LibriSpeech-960h. Decoding was performed without using a language model and beam search.

		WER [%] (↓)									
		LibriSpeech-100h				LibriSpeech-960h				TED-LIUM2	
		Dev		Test		Dev		Test		Dev	Test
Model		clean	other	clean	other	clean	other	clean	other		
Transformer	CTC	11.5	24.8	11.8	25.5	4.2	10.0	4.5	9.9	11.8	10.7
	SC-CTC	8.9	21.0	9.1	21.7	3.2	8.2	3.5	8.2	9.4	**8.6**
	HC-CTC	**8.2**	**19.9**	**8.4**	**20.6**	**3.1**	**8.0**	**3.4**	**8.0**	**9.1**	**8.6**
	ParaCTC	10.4	24.0	10.9	24.3	4.6	10.3	4.8	10.3	10.9	10.2
Conformer	SC-CTC	7.1	17.7	7.7	18.3	**2.8**	**6.7**	**3.0**	6.9	8.5	7.8
	HC-CTC	**6.9**	**17.1**	**7.1**	**17.8**	**2.8**	6.9	**3.0**	**6.8**	**8.0**	**7.6**

Table 3.2: WER on LibriSpeech-100h task for comparing Transformer-based models trained with different combinations of subword vocabulary sizes.

		WER [%] (↓)	
Model	$\vert\mathcal{V}^{(1)}\vert$-$\vert\mathcal{V}^{(2)}\vert$-$\vert\mathcal{V}^{(3)}\vert$	dev-clean	dev-other
SC-CTC	256 - 256 - 256	8.4	22.8
SC-CTC	2k - 2k - 2k	8.5	22.0
SC-CTC	16k - 16k - 16k	8.9	21.0
HC-CTC	256 - 256 - 16k	**8.2**	20.2
HC-CTC	2k - 2k - 16k	8.4	20.2
HC-CTC	256 - 2k - 16k	**8.2**	**19.9**

achieved much lower WERs on all tasks, demonstrating the effectiveness of applying CTC losses to intermediate layers as well as gradually increasing the subword units in a hierarchical manner.

Using Conformer further improved the performance of SC-CTC and HC-CTC, and HC-CTC again achieved more favorable performance than SC-CTC with faster training and inference. The presented Conformer results are comparable with other strong CTC-based models of the same size (Ng et al., 2021; Majumdar et al., 2021; Higuchi et al., 2021a), even without requiring exhaustive tuning.

3.2.8 Analysis

Analysis on Subword Vocabulary Size

While using sparse word-level units can make training of an ASR model challenging (Soltau et al., 2017), it is observed that the standard CTC-based model, with the Transformer-based architecture, benefits from training with a large subword vocabulary size. By increasing the output vocabulary size from 256 to 16k, the WERs for development sets changed from 11.1/28.1% to 11.5/24.8% on LS-360, and 12.3% to 11.8% on TED2. Similarly, the performance on LS-960 changed from 4.6/12.1% to 4.4/10.5% by changing the vocabulary size from 2k to 32k. These decent improvements from increasing the subword vocabulary size can be attributed to compensating for the CTC's incapability of modeling output dependencies (cf. Eq.(2.15)).

Considering the above observation, I evaluated SC-CTC and HC-CTC with different combinations of vocabulary sizes, focusing on Transformer-based models trained on LS-100. From the results for SC-CTC in Table 3.2, the performance on the dev-other set improved by increasing the vocabulary size, benefiting from the CTC training with large subword units. HC-CTC performed better than the 16k result of SC-CTC, indicating HC-CTC was more effective at modeling word-level recognition besides the advantage of CTC training with a large vocabulary size. While the SC-CTC performance on the dev-clean set degraded by increasing the vocabulary size, HC-CTC succeeded in learning robust word-level representations and achieved the lowest WER with the 16k-vocabulary size. Comparing the HC-CTC results, hierarchically increasing the subword units resulted in better performance than using the same vocabulary size for intermediate losses, suggesting the importance of gradually increasing the abstraction level for learning word-level representations effectively.

Importance of Conditioning

I studied the effectiveness of the self-conditioning mechanism, which is one of the important factors of the proposed model (see Section 3.2.1). The Transformer-based HC-CTC was trained on LS-100 without conditioning each CTC loss. Note that this model becomes similar to those from previous studies in Fernández et al. (2007); Toshniwal et al. (2017); Rao and Sak (2017); Krishna et al. (2018); Sanabria and Metze (2018). Without the conditioning mechanism, HC-CTC achieved WERs of 8.7/20.7% and 9.0/21.3% on the development sets and test sets, respectively. While these results are better than those obtained from CTC, SC-CTC, and ParaCTC in Table 3.1, HC-CTC with the conditioning mechanism achieved much lower WERs. Overall, it can be concluded that 1) hierarchical modeling based on multi-granular subword units as well as 2) the conditioning mechanism for explicitly maintaining lower levels of predictions are effective for learning word-level representations.

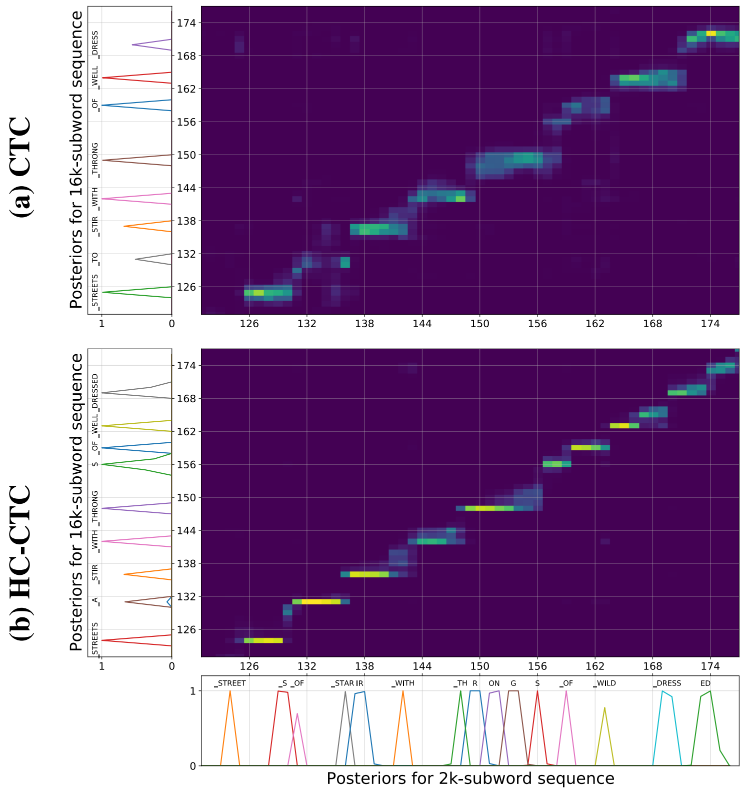


Figure 3-3: Visualizations of attention weights and CTC spikes derived from (a) CTC and (b) HC-CTC models trained on LibriSpeech-100h. A partial utterance was manually chosen from dev-other set (116-288045-0000), transcription of which was *"STREETS ASTIR WITH THRONGS OF WELL DRESSED."* This figure is reproduced from my conference paper (Higuchi et al., 2022a).

Attention visualization

Figure 3-3 visualizes attention weights between a source (x-axis) and target (y-axis) sequences, comparing Transformer-based (a) CTC and (b) HC-CTC models trained on LS-100 from Table 3.1. I focused on weights that seemed to contribute to predicting a 16k-subword sequence in the final CTC (from the 18-th layer). For both models, the CTC posteriors from the final 16k prediction are visualized. For HC-CTC, I also show the CTC posteriors (from the 12-th layer) for predicting a 2k-subword sequence in advance to see the relationship to the 16k prediction. Comparing the overall attention weights, it appeared that HC-CTC learned more solid and confident weights than CTC. HC-CTC seemed to exploit the lower-level 2k predictions to detect important frames for predicting each token, effectively composing complex word-level tokens using the lower-level tokens. For example, HC-CTC successfully recognized the words "THRONGS" and "DRESSED" with proper conjunctions, while CTC failed to handle these infrequent words.

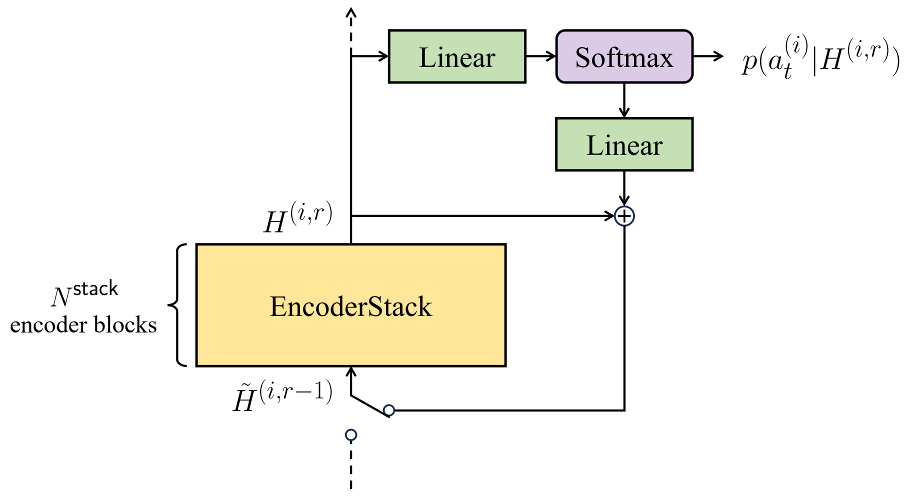


Figure 3-4: Schematic diagram of recursive operation in hierarchical recursive CTC. In each iteration, the output of the encoder is continuously looped back as input into the encoder stack. The self-conditioning technique is also employed to condition the encoding process on the previous predictions.

3.3 Hierarchical Multi-Task Learning with CTC and Recursive Feedback

This section introduces a simple extension to HC-CTC, aimed at enhancing the model's hierarchical generation capability. While the self-conditioning mechanism in HC-CTC is an important component of HC-CTC (see Section 3.2.8), there remains a potential issue associated with it; the explicit dependence on intermediate predictions could propagate errors through the model layers. Specifically, since the intermediate losses are applied to the shallower layers, they only affect the updates of only a limited portion of the model parameters. As a result, the model may struggle to learn to estimate lower-level targets accurately, consequently hurting the overall prediction quality at the very last layer. To address this, hierarchical recursive CTC (HR-CTC) is proposed as an enhancement to HC-CTC, which involves incorporating a refinement mechanism into each intermediate prediction by repeatedly utilizing shared model layers for estimating targets.

3.3.1 Hierarchical Recursive CTC (HR-CTC)

HR-CTC is realized by incorporating recursive operations into HC-CTC. Specifically, using a partial stack of encoder blocks, HR-CTC repeatedly performs forward computations for R times. As illustrated in Figure 3-4, at the r-th iteration, a stack of encoder blocks outputs a hidden sequence

$H^{(i,r)} \in \mathbb{R}^{T' \times D^{\text{model}}}$ as

$$\bar{H}^{(i,r-1)} = \begin{cases} H^{(i-N^{\text{stack}},R)} & (r=1), \\ H^{(i,r-1)} + \text{Linear}_{|V^{(i)}|+1 \to D^{\text{model}}}(V^{(i,r-1)}) & (r>1), \end{cases} \tag{3.9}$$

$$H^{(i,r)} = \text{EncoderStack}(\bar{H}^{(i,r-1)}). \tag{3.10}$$

where $i \in \mathcal{I} \cup \{N^{\text{enc}}\}$ is the layer index where the CTC loss is computed, and N^{stack} ($\leq N^{\text{enc}}$) represents the number of encoder blocks within each partial stack. In Eq. (3.9), when $r = 1$, the input sequence is obtained from the output of the preceding encoder stack. Otherwise, when $r > 1$, it is based on the output from the previous recursion, where self-conditioning is employed using the previous predictions,

$$V^{(i,r-1)} = \left(p(a_t^{(i)} | H^{(i,r-1)}) \,\middle|\, t = 1, \cdots, T' \right) \in [0,1]^{T' \times (|\mathcal{V}^{(i)}|+1)}, \tag{3.11}$$

with each framewise probability calculated similarly to Eq. (3.1). In Eq. (3.10), EncoderStack$(\cdot)$ represents a series of forward computations performed by N^{stack} encoder blocks, ranging from layer index of $i - N^{\text{stack}} + 1$ to i. The above recursion process within the same encoder stack enables the model to refine its representations via the repeated use of its encoder layers. This thereby virtually increases the model's depth and facilitates more precise intermediate predictions compared to HC-CTC. Furthermore, the conditioning mechanism similar to HC-CTC (with Eq. (3.9)) is expected to convey refined lower-level information that contributes to the improvement of the final prediction.

The i-th encoder layer of the HR-CTC model accumulates the CTC losses computed at each recursive operation as

$$\mathcal{L}^{(i)}(W^{(i)}|O) = \frac{1}{R} \sum_{r=1}^{R} \mathcal{L}^{\text{ctc}}(W^{(i)}|H^{(i,r)}), \tag{3.12}$$

where $\mathcal{L}^{\text{ctc}}(W^{(i)}|H^{(i,r)})$ is computed following the same procedure as described in Eq. (3.2). The overall HR-CTC loss is defined as the summation of the losses calculated across all encoder blocks:

$$\mathcal{L}^{\text{hr-ctc}} = \frac{1}{|\mathcal{I}|+1} \sum_{i \in \mathcal{I} \cup \{N^{\text{enc}}\}} \mathcal{L}^{(i)}(W^{(i)}|O), \tag{3.13}$$

where $W^{(N^{\text{enc}})} = W$ is the target sequence for the final loss.

3.3.2 Experimental Setting

I used the ESPnet toolkit (Watanabe et al., 2018) for conducting ASR experiments.

Data

I used various datasets with different amounts of data, speaking styles, and languages, including LibriSpeech (LS), TED-LIUM2 (TED2), and Corpus of Spontaneous Japanese (CSJ). See Appendix A for corpus details. For LS, in addition to the full 960-hour training set (LS-960), I used the 100-hour *train-clean-100* subset (LS-100) for performing additional investigations and analyses. For LS-100 and TED2, I constructed a subword vocabulary for tokenizing output texts, which were extracted from each training set using SentencePiece (Kudo, 2018). For CSJ, I used Japanese syllable characters (Kana) or Chinese characters (Kanji). As input speech features, I extracted 80 mel-scale filterbank coefficients with three-dimensional pitch features using Kaldi (Povey et al., 2011). To avoid overfitting, I applied speed perturbation (Ko et al., 2015) and SpecAugment (Park et al., 2019) to the input speech from LS and TED2, and only SpecuAugment to the input speech from CSJ.

Evaluated Models

I conducted training and evaluation on models trained by HC-CTC and HR-CTC. HC-CTC is the baseline model trained by $\mathcal{L}^{\text{hc-ctc}}$ from Eq. (3.7). HR-CTC is the proposed model trained by $\mathcal{L}^{\text{hr-ctc}}$ from Eq. (3.13).

Network Architecture

All the models were composed of two Conv2D down-sampling layers followed by $N^{\text{enc}} = 12$ encoder layers. The encoder layer was constructed using the Conformer (Gulati et al., 2020) architecture, with each layer having the number of heads N^{head}, dimension of a self-attention layer D^{model}, dimension of a feed-forward network D^{ff}, and kernel size of 4, 256, 1024, and 31, respectively. For LS and TED2, the intermediate CTC losses were applied at $\mathcal{I} = \{6, 9\}$, and N^{stack} was set to 3. For CSJ, the intermediate CTC loss was applied at $\mathcal{I} = \{8\}$, and N^{stack} was set to 4.

Training and Decoding

The models were trained up to 100 epochs for LS-100 and TED2, and 50 epochs for LS-960 and CSJ. The Adam (Kingma and Ba, 2015) optimizer with the Noam learning rate scheduling (Vaswani et al., 2017) was used for updating model parameters, with warmup steps and a learning rare factor set to 25k and 5.0, respectively. I followed the same setup as in Guo et al. (2021) for regularization hyper-parameters (e.g., dropout rate and label-smoothing weight). For the English tasks, each encoder block had targets with varying subword vocabulary sizes, where $(|\mathcal{V}^{(6)}|, |\mathcal{V}^{(9)}|, |\mathcal{V}^{(12)}|)$ was set to $(256, 2048, 16384)$ for LS-100 and TED2, and $(512, 4096, 32768)$ for LS-960. For the Japanese task, $\mathcal{V}^{(8)}$ consisted of Japanese syllable characters (181 in total) and

Table 3.3: WERs on LibriSpeech-{100h, 960h} and TED-LIUM2 tasks, and CER on CSJ task.

	WER [%] (↓)										**CER** [%] (↓)		
	LibriSpeech-100h				LibriSpeech-960h				TED-LIUM2		CSJ		
	Dev		Test		Dev		Test						
Model	clean	other	clean	other	clean	other	clean	other	Dev	Test	eval1	eval2	eval3
HC-CTC	7.0	17.8	7.4	18.2	3.3	8.3	3.5	8.2	8.2	7.8	6.1	4.3	4.5
HR-CTC	**6.6**	**16.7**	**6.8**	**17.0**	**2.9**	**7.2**	**3.0**	**7.3**	**7.7**	**7.2**	**5.6**	**4.0**	**4.3**

Table 3.4: Ablation studies on LibriSpeech-100h task evaluated by WER. The hierarchical structure or self-conditioning mechanism was removed from HR-CTC.

	WER [%] (↓)			
	Dev		Test	
Model	clean	other	clean	other
HR-CTC	6.6	16.7	6.8	17.0
$(\lvert\mathcal{V}^{(6)}\rvert, \lvert\mathcal{V}^{(9)}\rvert, \lvert\mathcal{V}^{(12)}\rvert) = (256, 256, 256)$	6.2	18.5	6.7	18.7
$(\lvert\mathcal{V}^{(6)}\rvert, \lvert\mathcal{V}^{(9)}\rvert, \lvert\mathcal{V}^{(12)}\rvert) = (16\text{k}, 16\text{k}, 16\text{k})$	7.9	18.7	8.1	19.3
w/o conditioning	6.7	17.2	6.9	17.5

$\mathcal{V}^{(12)}$ consisted of both Japanese syllable and Chinese characters (3260 in total). The number of the feedback operations in HR-CTC was fixed to $R = 3$, unless otherwise specified. After training, a final model was obtained by averaging model parameters over 10 checkpoints with the best validation performance. For CTC decoding, the best path search algorithm was performed (see Section 2.2.1).

3.3.3 Results

Table 3.3 shows the WER for LS-100, LS-960, and TED2, along with the CER for CSJ, evaluated for HC-CTC and HR-CTC models. HR-CTC consistently outperformed HC-CTC across all tasks, demonstrating its effectiveness regardless of data quantities, speaking styles, and languages. The number of parameters remains consistent across all the models, and the improvement in HR-CTC was accomplished without the need for additional parameters. This improvement is attributed to the recursive feedback mechanism in HR-CTC, which is further analyzed in the following sections.

Table 3.5: WERs of HC-CTC and HR-CTC on LibriSpeech-100h task, comparing predictions generated by each encoder block. The results are divided into "dev-{clean / other}" sets.

		Dev WER [%] (↓)	
Output Block Index		HC-CTC	HR-CTC
$i = 6$	$(\lvert\mathcal{V}^{(6)}\rvert = 256)$	10.3 / 27.4	**8.0 / 21.9**
$i = 9$	$(\lvert\mathcal{V}^{(9)}\rvert = 2\text{k})$	7.4 / 20.3	**6.4 / 17.7**
$i = 12$	$(\lvert\mathcal{V}^{(12)}\rvert = 16\text{k})$	7.0 / 17.8	**6.6 / 16.7**

3.3.4 Analysis

Ablation Study

To validate the effectiveness of our HR-CTC model design, I conducted ablation studies to assess the impact of both the hierarchical loss and conditioning mechanism on its performance.

Hierarchical Loss Table 3.4 presents the HR-CTC results, which employed a consistent vocabulary across the encoder blocks. As a result, HR-CTC with the hierarchical structure demonstrated superior performance, achieving the averaged WERs of 11.7% and 11.9% on development and test sets, respectively. This gain can be attributed to the model's ability to learn helpful representations for accurate word-level predictions, which is consistent with the findings presented in Section 3.2.8.

Conditioning Mechanism In Table 3.4, I also report the HR-CTC results obtained without incorporating conditioning feedback, specifically, ablating the inclusion of posterior probability distributions in Eq. (3.9). The performance of HR-CTC was degraded in the absence of this conditioning mechanism, indicating its significance. However, even without the conditioning mechanism, HR-CTC still outperformed the HC-CTC results presented in Table 3.3. This observation underscores the critical role played by the recursive operations in improving the model performance.

WER on Intermediate Predictions

Table 3.5 provides a comparison of WERs for predictions produced by each encoder block within HC-CTC and HR-CTC-based models. HR-CTC consistently outperformed HC-CTC across all encoder blocks, indicating the effectiveness of performing recursive operations at each block. Remarkably, the first encoder block (at $i = 6$) demonstrated the most substantial performance improvement when compared to the other models. Given that the second and third blocks of HR-CTC showed similar performance levels, the enhanced accuracy of the first encoder block in particular attributed to the notable improvement in the quality of subsequent predictions.

Table 3.6: WER and RTF on LibriSpeech-100h task, comparing HC-CTC to HR-CTC decoded with different numbers of recursive operations.

			WER [%] (↓)			
			Dev		Test	
Model	**#Repeat** R	**RTF** (↓)	clean	other	clean	other
HC-CTC	–	0.06	7.0	17.8	7.4	18.2
	1	0.06	10.3	23.8	10.5	24.7
HR-CTC	2	0.09	6.8	17.2	7.1	17.8
	3	0.12	6.6	16.7	6.8	17.0

Controllability of Balance Between Accuracy and Speed

Thanks to its ability to use the same encoder block repeatedly, HR-CTC can control the number of recursive operations executed during inference. Table 3.6 compares HC-CTC to HR-CTC decoded with different settings of recursive operations, where recognition accuracy is measured by the WER and inference speed is measured by the RTF. The RTF was measured using the LS-100 development sets using Intel(R) Core(TM) i9-10920X CPU, 3.50GHz. The WER of HR-CTC was largely improved by increasing the number of repetitions, with the optimal performance observed at $R = 3$. Nevertheless, this improvement comes at the expense of slower inference speed due to the increased forward computation cost, resulting in a speed twice as slow as HC-CTC. This slowdown, however, can be mitigated by decreasing the number of repetitions. For instance, by setting $R = 2$, HR-CTC achieves an RTF of under 0.1 while still outperforming HC-CTC. Different from HC-CTC, HR-CTC has the ability to control both recognition accuracy and inference speed within a single model, adjusting its performance based on the specific demands of the ASR application.

3.4 Momentum Pseudo-Labeling with Intermediate CTC Loss

This section presents InterMPL, an advanced semi-supervised learning approach for improving end-to-end ASR models trained with the intermediate CTC losses (i.e., SC-CTC and HC-CTC from Section 3.2). By integrating intermediate losses into the training framework of momentum pseudo-labeling (MPL) (Higuchi et al., 2021c, 2022c), InterMPL allows the model to benefit from intermediate supervision through the use of multiple pseudo-labels. This is expected to facilitate the model's ability to effectively learn from unlabeled data.

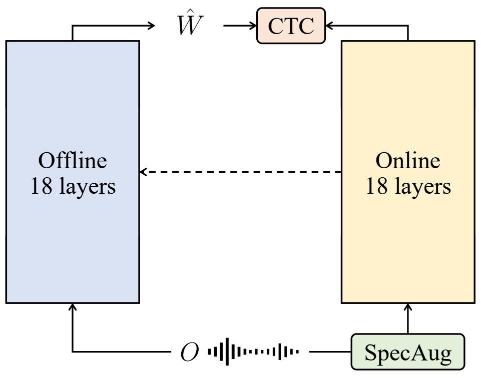


Figure 3-5: Overview of momentum pseudo-labeling for semi-supervised ASR. A dashed line ($\leftarrow$--) indicates the momentum update of the offline model using the online model parameters. This figure is reproduced from my conference paper (Higuchi et al., 2023c).

3.4.1 Semi-Supervised ASR with Momentum Pseudo-Labeling

In semi-supervised ASR, a seed model is first trained on labeled data $\mathcal{D}^{\mathsf{lab}} = \{\langle O_n, W_n\rangle | n = 1, \ldots, N^{\mathsf{lab}}\}$ using the CTC loss $\mathcal{L}^{\mathsf{ctc}}$ from Eq. (2.20). MPL (Higuchi et al., 2022c) is then applied to the seed model to improve the performance using unlabeled speech-only data $\mathcal{D}^{\mathsf{unlab}} = \{O_{n'} | n' = N^{\mathsf{lab}}+1, \ldots, N^{\mathsf{lab}}+N^{\mathsf{unlab}}\}$. Figure 3-5 and Algorithm 1 describe the training process of MPL based on a pair of *online* and *offline* models. Let Θ and Φ denote the parameters of the online and offline models, which are initialized with the pre-trained seed model parameters.

Online Model Training

Given the n'-th unlabeled sample $O_{n'} \in \mathcal{D}^{\mathsf{unlab}}$ and its encoded sequence $H_{n'}$ from an encoder network (e.g., Eq. (2.54)), the online model is trained on pseudo-labels $\hat{W}_{n'}$ generated on the fly by the offline model with Φ:

$$\hat{W}_{n'} = \mathcal{B}^{\mathsf{ctc}}\left(\operatorname*{arg\,max}_{a_t} p(a_t | H_{n'}, \Phi) \,\middle|\, t = 0, \cdots, T'\right), \tag{3.14}$$

which is obtained via the best path decoding algorithm of CTC (see Section 2.2.1). With the pseudo-labeled sample $\langle O_{n'}, \hat{W}_{n'}\rangle$, the online model with Θ is trained via a gradient descent optimization based on the CTC loss $\mathcal{L}^{\mathsf{ctc}}(\hat{W}_{n'} | H_{n'}, \Theta)$, as calculated in Eq. (3.2). Here, the unlabeled speech input $O_{n'}$ is augmented by SpecAugment (Park et al., 2019) (as shown in Figure 3-5) to facilitate the model training on pseudo-labels (Masumura et al., 2020; Chen et al., 2020). Note that MPL also uses the n-th labeled sample $\langle O_n, W_n\rangle \in \mathcal{D}^{\mathsf{lab}}$ and trains the online model with a supervised loss $\mathcal{L}^{\mathsf{ctc}}(W_n | H_n, \Theta)$, which helps the online model stabilize and promote learning from unlabeled data.

Algorithm 1 Momentum pseudo-labeling

```
    Input:
      $\mathcal{D}^{\mathsf{lab}}, \mathcal{D}^{\mathsf{unlab}}$      ▷ Labeled and unlabeled data
      $\Theta, \Phi$            ▷ Parameters of online and offline models
      $\alpha$                ▷ A momentum coefficient
 1: Train a seed model on $\mathcal{D}^{\mathsf{lab}}$ using the CTC loss from Eq. (2.20)
 2: Initialize online and offline models with the parameters of the seed model
 3: for the number of training epochs do
 4:     for all $\mathcal{D} \in \mathcal{D}^{\mathsf{lab}} \cup \mathcal{D}^{\mathsf{unlab}}$ do
 5:         Obtain $O \sim \mathcal{D}$
 6:         Obtain $W = \begin{cases} W \sim \mathcal{D} & (\mathcal{D} \in \mathcal{D}^{\mathsf{lab}}) \\ \hat{W} = \arg\max_W p(W|O,\Phi) & (\mathcal{D} \in \mathcal{D}^{\mathsf{unlab}}) \end{cases}$
 7:         Compute encoder states $H$
 8:         Compute the CTC loss $\mathcal{L}^{\mathsf{ctc}}(W|H)$ and update the parameters $\Theta$ of the online model
 9:         Update the offline model as $\Phi \leftarrow \alpha\Phi + (1-\alpha)\Theta$
10:     end for
11: end for
12: return $\Theta$          ▷ Online model is returned for final evaluation
```

Offline Model Training

After every update of the online model, the offline model accumulates the parameters of the online model as

$$\Phi \leftarrow \alpha\Phi + (1-\alpha)\Theta, \tag{3.15}$$

an exponential moving average with a momentum coefficient of α $(0 \leq \alpha \leq 1)$. This momentum update makes the offline model serves as an ensemble of the online models at different training steps (Tarvainen and Valpola, 2017), preventing the pseudo-labels from deviating too quickly from the labels initially generated by the seed model. Through the above interaction between the two models, MPL realizes stable and continuous ASR training on unlabeled data, concurrently improving the quality of pseudo-labels.

Tuning Momentum Coefficient

Instead of directly tuning α in Eq. (3.15), a more intuitive method is designed to derive an appropriate value of α. Based on Eq. (3.15), the parameters of the offline model after K updates can be written as

$$\Phi^{(K)} = \alpha^K \Phi^{(0)} + (1-\alpha) \sum_{k=1}^{K} \alpha^{K-k} \Theta^{(k)}, \tag{3.16}$$

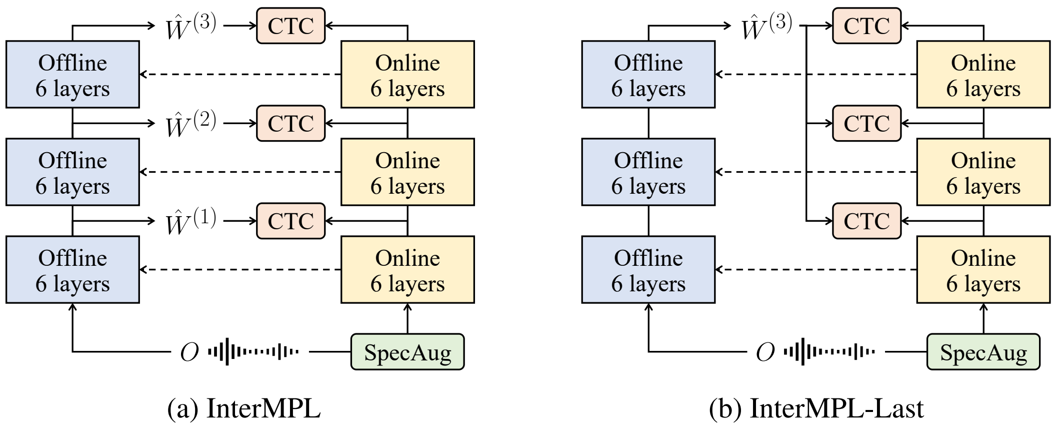


Figure 3-6: Proposed InterMPL for semi-supervised ASR. A dashed line (←--) indicates the momentum update of the offline model using the online model parameters. The number of CTC losses is set to three. These figures are reproduced from my conference paper (Higuchi et al., 2023c).

where $\Phi^{(k)}$ and $\Theta^{(k)}$ denote the parameters of each model at the k-th update, and $\Phi^{(0)} = \Theta^{(0)}$. Here, I assume that it is important to retain some influence of the seed model to stabilize the pseudo-label generation. As a measure of this influence, I focus on the term $\alpha^K \Phi^{(0)}$ in Eq. (3.16) and define a weight ω of the seed model in $\Phi^{(K)}$ as

$$\omega = \alpha^K, \tag{3.17}$$

where K is the number of training iterations (i.e., mini-batches) in a training epoch. As K can often be in the thousands, small changes in α lead to huge differences in ω (e.g., $0.999^{3000} \ll 0.9997^{3000}$), requiring small adjustments on α for different amounts of training data. Instead of directly tuning α, the weight ω is used to tune the momentum update, which can be regarded as the proportion of the seed model parameters retained after a training epoch. Given ω and K, α is calculated as

$$\alpha = e^{(1/K)\log \omega}. \tag{3.18}$$

By controlling the update through ω, MPL becomes less affected by the amount of training data, as demonstrated in Higuchi et al. (2021c).

3.4.2 InterMPL

I propose a semi-supervised ASR method that introduces the intermediate CTC technique to MPL. The conventional MPL is founded on CTC-based modeling, whose performance can be limited due to the conditional independence assumption. To further enhance MPL, I adopt SC-CTC or HC-CTC for constructing a seed model, which is expected to facilitate better CTC training/decoding and thus promote the succeeding semi-supervised process with higher-quality pseudo-labels.

Given labeled data $\mathcal{D}^{\mathsf{lab}}$, a seed model is trained by a supervised loss based on SC-CTC (with Eq. (3.5)) or HC-CTC (with Eq. (3.7)).

Initialized with the seed model trained by SC-CTC or HC-CTC, the online model can accept intermediate supervision using pseudo-labels, and the offline model can generate multiple pseudo-labels from its intermediate layers. This leads to the exploration of two different advancements in MPL, namely **InterMPL** (Figure 3-6(a)) and **InterMPL-Last** (Figure 3-6(b)), for fully utilizing the intermediate mechanism in SC-CTC and HC-CTC to enhance the learning process with unlabeled data.

InterMPL

In InterMPL, the offline model generates pseudo-labels from each prediction layer, as shown in Figure 3-6(a), with three different outputs (one from the final layer and two from the intermediate layers). These pseudo-labels are used to calculate a loss for the corresponding layer of the online model. Given the n'-th unlabeled sample $O_{n'} \in \mathcal{D}^{\mathsf{unlab}}$, the i-th offline encoder layer emits hidden vectors $H_{n'}^{(i)}$ and generates an i-th prediction $\hat{W}_{n'}^{(i)}$ as in Eq. (3.14) as

$$\hat{W}_{n'}^{(i)} = \mathcal{B}^{\mathsf{ctc}} \left(\underset{a_t^{(i)}}{\arg\max}\, p(a_t^{(i)} | H_{n'}^{(i)}, \Phi) \middle| t = 0, \cdots, T' \right), \tag{3.19}$$

where $i \in \mathcal{I} \cup \{N^{\mathsf{enc}}\}$. With the pair of unlabeled speech sample and multiple pseudo-labels $\langle O_{n'}, \{\hat{W}_{n'}^{(i)}\}_{i \in \mathcal{I} \cup \{N^{\mathsf{enc}}\}} \rangle$, the objective function of the online model is defined similarly to Eq. (3.5) and Eq. (3.7) as

$$\frac{1}{|\mathcal{I}| + 1} \sum_{i \in \mathcal{I} \cup \{N^{\mathsf{enc}}\}} \mathcal{L}^{\mathsf{ctc}}(\hat{W}_{n'}^{(i)} | H^{(i)}, \Theta). \tag{3.20}$$

This training strategy is compatible with both SC-CTC and HC-CTC-based InterMPL, which I assume is particularly effective for HC-CTC with varying output units. HC-CTC trains an ASR model to learn a progressive generation of a target sequence, using the intermediate loss with increasing subword vocabulary size. I expect pseudo-labels generated at different granularities to facilitate semi-supervised learning by providing ancillary training signals.

InterMPL-Last

For SC-CTC, InterMPL may not be an optimal choice, as SC-CTC calculates intermediate losses using the same sequence targeted in the last layer. Hence, I design another variant called InterMPL-Last. Different from InterMPL (Figure 3-6(a) vs. Figure 3-6(b)), InterMPL-Last utilizes only the final hypothesis of the offline model as pseudo-labels for calculating all the losses in the online model. Given the n'-th unlabeled sample $O_{n'} \in \mathcal{D}^{\mathsf{unlab}}$ and the last pseudo-labels generated by the offline model $\hat{W}_{n'}^{(N^{\mathsf{enc}})}$, the objective function of the online model is defined similarly to Eq. (3.5) as

$$\frac{1}{|\mathcal{I}|+1} \sum_{i \in \mathcal{I} \cup \{N^{\mathsf{enc}}\}} \mathcal{L}^{\mathsf{ctc}}(\hat{W}_{n'}^{(N^{\mathsf{enc}})} | H^{(i)}, \Theta). \tag{3.21}$$

InterMPL-Last enables the online model to be trained on the most accurate pseudo-labels predicted by the offline model, which permits more effective use of SC-CTC for semi-supervised training.

3.4.3 Experimental Setting

I used the ESPnet toolkit (Watanabe et al., 2018) for conducting the experiments, and all the codes and recipes are made publicly available.[2]

Data

The experiments were carried out using LibriSpeech (LS) and TED-LIUM3 (TED3). See Appendix A for corpus details. As input speech features, I extracted 80 mel-scale filterbank coefficients with three-dimensional pitch features using Kaldi (Povey et al., 2011). I used SentencePiece (Kudo, 2018) to construct subword vocabularies from the *train-clean-100* transcriptions.

Semi-Supervised Settings

I regarded *train-clean-100* (LS-100) as the labeled data $\mathcal{D}^{\mathsf{lab}}$. Based on a seed model trained on LS-100, I simulated three semi-supervised settings using different unlabeled data $\mathcal{D}^{\mathsf{unlab}}$: LS-100/LS-360, an in-domain setting using unlabeled *train-clean-360* (LS-360); LS-100/LS-860, an in-domain setting using unlabeled *train-clean-360* and *train-other-500* (LS-860); and LS-100/TED3, an out-of-domain setting using unlabeled TED3 train set.

Model Architecture

I used the Conformer architecture (Gulati et al., 2020; Guo et al., 2021) consisting of two Conv2D layers followed by a stack of $N^{\mathsf{enc}} = 18$ encoder blocks. The number of heads N^{head}, the dimension of a self-attention layer D^{model}, the dimension of a feed-forward network D^{ff}, and the kernel size were set to were set to 4, 256, 1024, and 7, respectively. Following Higuchi et al. (2022b), batch normalization in the convolution module is replaced with group normalization (Wu and He, 2018) with the group size of 4.

Training and Decoding Configurations

The seed model was trained for 150 epochs using the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, and Noam learning rate scheduling (Vaswani et al., 2017).

[2] `https://github.com/YosukeHiguchi/espnet/tree/intermpl`

Warmup steps and a learning rate factor were set to 25k and 5.0. The MPL training was iterated up to 200 epochs, and the online model was trained using the Adam optimizer with an initial learning rate of 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The weight ω for deriving the momentum coefficient, defined in Eq. (3.17), was set to 0.5. The subword vocabulary size of a CTC-based model was set to 1024. SC-CTC and HC-CTC applied the intermediate CTC losses to the 6th and 12th encoder layers (i.e., $\mathcal{I} = \{6, 12\}$). The output vocabulary size for each loss ($|\mathcal{V}^{(6)}|$, $|\mathcal{V}^{(12)}|$, $|\mathcal{V}^{(18)}|$) was set to $(1024, 1024, 1024)$ for SC-CTC and $(256, 1024, 4096)$ for HC-CTC. A final model was obtained for evaluation by averaging model parameters over 10 checkpoints that gave the best validation performance. For the MPL-based methods, I followed Higuchi et al. (2021c) and used the online model for evaluation. During decoding, the best path decoding algorithm of CTC was performed (see Section 2.2.1).

Evaluation Metrics

The WER was used to measure the ASR performance. For evaluating the performance of semi-supervised training, I measured the WER recovery rate (WRR) (Ma and Schwartz, 2008; Kahn et al., 2020a). WRR compares WERs of the oracle model (trained using ground-truth transcriptions for the unlabeled data as well) and the semi-supervised model by calculating the ratio between their absolute reductions from the seed model's WER:

$$\text{WRR}[\%] = \frac{\text{WER}^{\textsf{seed}} - \text{WER}^{\textsf{semi-supervised}}}{\text{WER}^{\textsf{seed}} - \text{WER}^{\textsf{oracle}}}, \tag{3.22}$$

where WER^* denotes WER for each model.

3.4.4 Results

Supervised Baseline and Oracle Results

Table 3.7 shows the WER of seed models trained on LS-100 (`S⋆`) and oracle models trained on fully labeled data in each semi-supervised setting (`A⋆`, `B⋆`, and `C⋆`). Overall, consisted with the findings presented in Section 3.2.7, SC-CTC and HC-CTC outperformed CTC by a large margin. The quality of pseudo-labels is crucial for effective semi-supervised training, and I can expect MPL to benefit from the seed models trained with the intermediate loss.

Main Results

In-domain setting Table 3.8 shows WER on LS, comparing the conventional MPL (Higuchi et al., 2022b) against the proposed InterMPL and InterMPL-Last. In Figure 3-7, I also compare the performance of each semi-supervised training in the WRR, where the results were averaged on the clean and other sets for LS. Note that the seed models (`S⋆`) from Table 3.7 were used for the

Table 3.7: WERs for models trained on fully labeled data. `A⋆`, `B⋆`, and `C⋆` indicate the oracle results for each semi-supervised setting, indicated by "supervised data / unsupervised data."

		WER [%] (↓)		
		LibriSpeech		TED-LIUM3
Setting	**Model**	test-clean	test-other	Test
LS-100	`S1` CTC	8.4	23.1	26.7
	`S2` SC-CTC	7.5	21.3	24.2
	`S3` HC-CTC	**7.4**	**20.4**	**23.8**
LS-100 / LS-360	`A1` CTC	4.6	13.5	–
	`A2` SC-CTC	**3.9**	12.0	–
	`A3` HC-CTC	4.0	**11.6**	–
LS-100 / LS-860	`B1` CTC	3.5	8.8	–
	`B2` SC-CTC	**3.1**	7.8	–
	`B3` HC-CTC	3.2	**7.7**	–
LS-100 / TED3	`C1` CTC	–	–	7.5
	`C2` SC-CTC	–	–	**6.8**
	`C3` HC-CTC	–	–	7.1

initialization in each method. Looking at the results on the LS-360 setting (`X⋆`) in Table 3.8, both InterMPL and InterMPL-Last led to distinct improvements over MPL (`X1` vs. `X2`, `X3`, `X4`), indicating the effectiveness of using the well-trained seed models and applying intermediate CTC loss during semi-supervised training. Comparing SC-CTC and HC-CTC-based InterMPL, HC-CTC resulted in better performance by benefiting from using the pseudo-labels at different granularity (`X2` vs. `X3`). InterMPL-Last was better suited for SC-CTC-based training than InterMPL (`X2` vs. `X4`), as it was hypothesized that higher-quality labels are more appropriate for intermediate supervision. Overall, HC-CTC-based InterMPL and InterMPL-Last similarly achieved the best performance, while InterMPL-Last gave higher WRRs in Figure 3-7(a). In the LS-860 setting with more unlabeled data (`Y⋆`) in Table 3.8, the general trend was consistent with what was observed in the LS-360 setting. In terms of the WRR in Figure 3-7, InterMPL-Last had the most significant gain, which was even higher than those of MPL. Both InterMPL and InterMPL-Last were scalable to larger amounts of unlabeled data.

Out-of-domain setting Table 3.9 lists results on the out-of-domain TED3 setting. Both InterMPL and InterMPL-Last outperformed MPL (`Z1` vs. `Z2`, `Z3`, `Z4`), demonstrating stable training on unlabeled data under the domain-mismatched condition. In contrast to the in-domain results, SC-CTC and HC-CTC-based InterMPL resulted in a similar performance (`Z2` vs. `Z3`), and InterMPL-Last achieved lower WERs than InterMPL (`Z4` vs. `Z2`, `Z3`). HC-CTC was less significant in the out-of-domain semi-supervised scenario, including the oracle results in Table 3.7,

Table 3.8: WERs on in-domain LS-100/LS-360 and LS-100/LS-860 settings.

			WER [%] (↓)	
Setting	**Method**	**Init.**	test-clean	test-other
LS-100 / LS-360	`X1` MPL	`S1` (CTC)	6.3	15.4
	`X2` InterMPL	`S2` (SC-CTC)	5.7	14.5
	`X3` InterMPL	`S3` (HC-CTC)	5.5	**14.1**
	`X4` InterMPL-Last	`S2` (SC-CTC)	**5.4**	**14.1**
LS-100 / LS-860	`Y1` MPL	`S1` (CTC)	6.0	11.9
	`Y2` InterMPL	`S2` (SC-CTC)	5.4	11.0
	`Y3` InterMPL	`S3` (HC-CTC)	5.3	**10.7**
	`Y4` InterMPL-Last	`S2` (SC-CTC)	**5.1**	**10.7**

Table 3.9: WER on out-of-domain LS-100/TED3 setting.

Method	**Init.**	**Test WER** [%] (↓)
`Z1` MPL	`S1` (CTC)	13.4
`Z2` InterMPL	`S2` (SC-CTC)	12.8
`Z3` InterMPL	`S3` (HC-CTC)	12.6
`Z4` InterMPL-Last	`S2` (SC-CTC)	**12.1**

which indicates its inferior generalization capability. Subword vocabularies were constructed from the small LS-100 text set, and the large vocabulary size used in HC-CTC (i.e., 4096) was not generalized well to the TED3 domain.

In all of the settings, InterMPL consistently delivered substantial improvements compared to the seed models (`S*` in Table 3.7), demonstrating that the models incorporating intermediate losses, i.e., SC-CTC and HC-CTC, experienced significant gains by learning from unsupervised data through the proposed semi-supervised training strategies.

Ablation Study on Intermediate Loss

Table 3.10 shows an ablation study validating the effectiveness of InterMPL. I initialized a model using parameters of SC-CTC (`S2`) or HC-CTC (`S3`) and performed standard MPL without using intermediate loss. Compared to the InterMPL results (`X2`, `X3`), it was observed that removing intermediate loss led to worsening WERs with degraded WRRs. Interestingly, MPL based on SC-CTC initialization resulted in a similar performance as that of standard MPL (`X1`). This indicates the importance of applying intermediate loss during semi-supervised training. I also performed InterMPL-Last initialized from CTC (`S1`), which gave better results than those of MPL (`X1`).

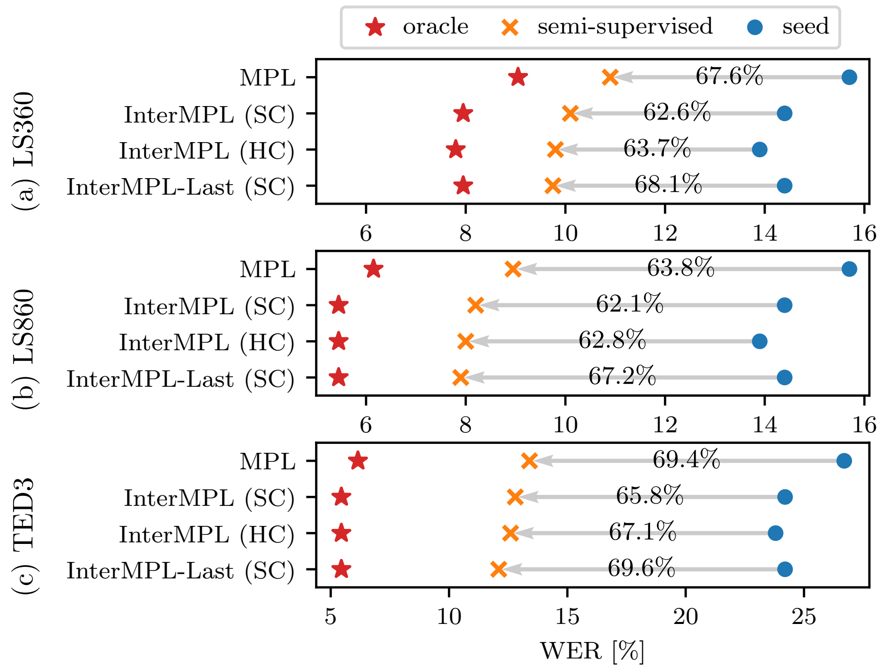


Figure 3-7: Visualization of WRR in each semi-supervised setting. This figure is reproduced from my conference paper (Higuchi et al., 2023c).

These results suggest the importance of applying intermediate loss to both the seed model and semi-supervised training.

3.5 Summary

This chapter presented hierarchical modeling methods, aiming at improving the representation learning capabilities of end-to-end ASR models.

First, I proposed HC-CTC, which hierarchically increased the abstraction level in linguistic outputs to effectively learn representations for predicting sparse word-level units. The proposed model was trained using auxiliary CTC losses applied to intermediate layers, where the vocabulary size of each target subword sequence gradually increased as the layers progressed closer to the word-level output. In this hierarchical process, each level of sequence prediction was explicitly conditioned on the sequences predicted at the lower levels, thereby encouraging the model to exploit a hierarchy of linguistic structures to extract word-level representations. The experimental results demonstrated that HC-CTC outperforms the standard CTC-based model and other conventional models trained with the intermediate CTC techniques.

Subsequently, I made an architectural improvement to the HC-CTC-based model by integrating a refinement mechanism into each stage of intermediate prediction. Specifically, the

Table 3.10: Ablation study on LS-100/LS-360 setting.

	WER [%] ($\downarrow$)		**WRR** [%] ($\uparrow$)	
Method	test-clean	test-other	test-clean	test-other
InterMPL (`X2`)	**5.7**	**14.5**	**51.9**	**73.3**
w/o inter. loss	6.4	15.5	30.9	62.6
InterMPL (`X3`)	**5.5**	**14.1**	**55.3**	**72.0**
w/o inter. loss	5.8	14.8	45.8	63.8
InterMPL-Last (`X4`)	**5.4**	**14.1**	**59.2**	**77.0**
w/ init. from `S1`	5.9	14.4	46.4	74.1

Transformer-based encoder layers were augmented with recursive operations, which involved repeatedly using shared model layers to refine intermediate representations. This enabled the model to make more accurate sequence predictions at the lower levels, thereby improving its overall performance, especially in predicting sparser outputs at the final layer. The experimental results showed that the proposed enhancement further improves HC-CTC, yielding effective results across various conditions, including varying data quantities, speaking styles, and languages.

Lastly, I developed an efficient semi-supervised learning method for improving the end-to-end ASR models trained with intermediate CTC losses. The proposed approach, InterMPL, extended the functionality of MPL by enabling it to utilize multiple pseudo-labels obtained from intermediate predictions. In the context of HC-CTC, the pseudo-labels were generated at varying levels of granularity, which helped the model learn linguistic information from unlabeled audio-only data. The experimental results in different semi-supervised settings consistently showed that InterMPL outperforms standard MPL. InterMPL proved to be effective for both SC-CTC and HC-CTC-based models, underscoring the significance of incorporating intermediate losses with pseudo-labels in the semi-supervised training process.

4 End-to-End Speech Recognition Guided by Masked Language Modeling

4.1 Introduction

Masked language modeling has proven to be highly effective in extracting and comprehending linguistic information from text sequences, training a language model on the token in-filling task (Williams and Zipser, 1989; Fedus et al., 2018; Devlin et al., 2019). This method involves first masking some of the tokens in a text sequence, and then predicting the masked tokens based on the surrounding context. For example, given a masked sequence

“Dogs `[MASK]` and `[MASK]` meow,”

a model is trained to predict “barks” and “cats” at the masked positions indicated by the special mask token `[MASK]`. This task is different from the standard approach of language modeling, which focuses on capturing unidirectional left-to-right dependencies in text through an autoregressive structure (Radford et al., 2018) (cf. Eq. (2.10)). Unlike left-to-right language modeling, masked language modeling facilitates the acquisition of contextualized representations that integrate both left and right contexts, enabling a model to capture bidirectional token dependencies at the sentence level.

This chapter presents methods for incorporating the masked language modeling mechanism into end-to-end ASR models, with the primary goal of augmenting the models’ capability to effectively utilize long-range linguistic contexts. End-to-end ASR models are designed to directly transform acoustic signals, which naturally exhibit local dependencies, into linguistic tokens that often depend on broader contexts. By employing masked language modeling, it is expected to provide end-to-end ASR models with guides to understanding token dependencies, thereby facilitating the extraction of linguistic information for producing accurate textual outputs.

Section 4.2 proposes Mask-CTC, an end-to-end ASR framework that trains an AED-based model with joint CTC and conditional masked language model (CMLM) (Ghazvininejad et al., 2019) objectives. The inference process in Mask-CTC is designed by combining the strengths of both CTC and CMLM-based modeling. Initially, CTC decoding is performed to rapidly produce an ASR hypothesis, during which tokens of low confidence are selectively masked based on their posterior probabilities. Then, the CMLM-based decoder further refines the hypothesis by repredicting the masked tokens, utilizing both the audio information and the linguistic context derived from the other unmasked tokens. Such a decoding algorithm enables the model to execute non-autoregressive sequence generation, resulting in faster inference speed compared to conventional autoregressive models. Additionally, the CMLM decoder allows for correcting errors caused by the conditional independence assumption in CTC, thus producing more accurate predictions by accounting for token dependencies.

Section 4.3 delves into advanced strategies for leveraging the contextualized representations derived from the CMLM. The proposed Mask-Conformer model enhances the Conformer architecture by applying the CMLM decoder to multiple encoder layers. The token-level representations, embedded by the decoder, are fed back to subsequent encoder layers through the cross-attention mechanism. This conditioning by the decoder enables the encoder to explicitly incorporate linguistic information during its speech encoding process. Mask-CTC inference refines a hypothesis using the CMLM decoder, while preserving the initial interpretation of speech information (from the encoder output). In contrast, Mask-Conformer allows for the reinterpretation of speech information based on explicit linguistic information acquired by the CMLM. This leads to the effective utilization of language model information for improving ASR performance.

Section 4.4 explores the application of Mask-CTC as a pre-training method for streaming end-to-end ASR models. Achieving high accuracy with low latency has always been a challenge in streaming end-to-end ASR systems. By attending to more future contexts, a streaming ASR model achieves higher accuracy but this comes at the cost of increased latency, adversely affecting the streaming performance. With the CMLM decoder that captures token dependencies from both past and future contexts, Mask-CTC is capable of learning feature representations that anticipate long-term contexts, which can be particularly beneficial for streaming ASR models. To this end, I propose a pre-training method that first trains an end-to-end ASR model based on Mask-CTC and then utilizes its parameters to initialize a streaming model. This implicitly transfers the advantageous capabilities of Mask-CTC into the training process for streaming ASR, thereby achieving high recognition accuracy while maintaining low latency.

Chapter 3 proposed the hierarchical model to implicitly assimilate linguistic information through a hierarchy of subword units. This chapter, in contrast, shifts focus to integrating explicit language modeling, aiming for a more direct approach to learning text sequences.

4.2 Mask-CTC: Non-Autoregressive End-to-End ASR with CTC and Mask-Predict

This section proposes Mask-CTC, a novel end-to-end ASR model that permits non-autoregressive sequence generation. Many of the previous studies on end-to-end ASR have concentrated on developing an autoregressive model (i.e., AED described in Section 2.2.3), which estimates the likelihood of sequence generation based on a left-to-right probabilistic chain rule. The autoregressive model often achieves the best performance among other end-to-end ASR approaches (Chiu et al., 2018). However, its application is constrained by slower inference speed, requiring incremental calculations within the model to produce each token of the output sequence. This limitation is a significant drawback for the practical deployment of ASR systems in real-world scenarios, particularly for on-device applications where efficiency and speed are crucial. Unlike the autoregressive model, a non-autoregressive model generates an output sequence within a constant number of the inference steps (Graves et al., 2006; Gu et al., 2018). Non-autoregressive models are actively studied in the field of neural machine translation, which have demonstrated performance on par with autoregressive models while achieving faster inference speed. Innovative approaches being explored include iterative refinement decoding (Lee et al., 2018), insertion-based sequence generation (Stern et al., 2019; Gu et al., 2019), masked language modeling (Ghazvininejad et al., 2019, 2020; Saharia et al., 2020), and generative flow techniques (Ma et al., 2019). Mask-CTC achieves non-autoregressive ASR, drawing inspiration from the principles of masked language modeling (Ghazvininejad et al., 2019) for enabling parallel prediction of tokens.

4.2.1 Related Work on Non-Autoregressive End-to-End ASR

CTC (Graves et al., 2006) forms a fundamental basis for realizing non-autoregressive end-to-end ASR. CTC makes a strong conditional independence assumption between token frame predictions, which facilitates fast inference speed but can constrain recognition accuracy. Inspired by CMLM (Ghazvininejad et al., 2019), Audio-CMLM (Chen et al., 2021b) effectively introduces masked language modeling to learn the conditional distribution of output tokens over a partially observed sequence. Imputer (Chan et al., 2020) combines CTC with masked language modeling to improve the frame-level token predictions, eliminating the need for the length prediction mechanism employed in previous approaches (Gu et al., 2018). Align-Refine (Chi et al., 2021a) and Align-Denoise (Chen et al., 2021c) incorporate iterative refinement (Lee et al., 2018) to directly optimize and refine CTC-based predictions.

Recent efforts have focused on improving the performance of the standard CTC-based model.

Notably, intermediate CTC (Lee and Watanabe, 2021) and self-conditioned CTC (Nozaki and Komatsu, 2021) introduce auxiliary CTC losses to the model's intermediate layers (Tjandra et al., 2020), facilitating the learning of intermediate representations for enhancing CTC-based training and inference. Convolution-based neural network architectures have been shown to improve the CTC-based and the other end-to-end ASR models in general (Ng et al., 2021; Majumdar et al., 2021). When a large amount of speech data is available for pre-training, powerful speech representations learned by wav2vec 2.0 (Baevski et al., 2020) can significantly boost the performance of the CTC-based model (Ng et al., 2021).

An alternative approach to non-autoregressive ASR utilizes insertion-based modeling, which allows for generating tokens in a flexible arbitrary order without the left-to-right constraint in the autoregressive formulation. Demonstrating notable success in neural machine translation, Insertion Transformer (Stern et al., 2019) and Kontextuell Encoder Representations Made by Insertion Transformations (KERMIT) (Chan et al., 2019) have been effectively adapted for end-to-end ASR (Fujita et al., 2020).

The proposed Mask-CTC shares similarities to the models that integrate CTC with masked language modeling, especially the approach based on Imputer (Chan et al., 2020). However, Imputer operates by processing sequences at the input frame level, frequently dealing with hundreds of units, using self-attention layers (Vaswani et al., 2017). Given that these self-attention layers require computational costs proportional to the square of the sequence length, the sequence processing in Imputer tends to be computationally intensive. On the other hand, Mask-CTC deals with the output sequence at the token level, which is significantly shorter than the input length, thereby resulting in a more computationally efficient approach.

4.2.2 Mask-CTC

Joint CTC and CMLM Training

Mask-CTC adopts non-autoregressive modeling based on CMLM (Ghazvininejad et al., 2019; Chen et al., 2021b), where the model is trained to predict masked tokens in a target output sequence (Devlin et al., 2019). Taking advantage of Transformer's parallel computation (see Section 2.3.2 for details), CMLM can predict any arbitrary subset of tokens in an output sequence, attending to the entire sequence including tokens in the past and the future positions.

Given a target token sequence W, let $\tilde{W} = (\tilde{w}_l \in \mathcal{V} \cup \{\texttt{[MASK]}\} | l = 1, \cdots, L)$ be a partially masked sequence, with each token $\tilde{w}_l$ defined as

$$\tilde{w}_l = \begin{cases} \texttt{[MASK]} & (l \in \mathcal{U}), \\ w_l \in W & (\text{otherwise}), \end{cases} \tag{4.1}$$

where $\mathcal{U}$ represents a set of token positions that are replaced with the mask token. During training, to obtain $\tilde{W}$, the ground truth tokens are randomly replaced by the mask token, where the number of tokens to be masked is sampled from a uniform distribution between 1 to L (Ghazvininejad et al., 2019). CMLM predicts a set of target tokens $\mathcal{W} = \{w_l \in W | l \in \mathcal{U}\}$, conditioning on the input sequence O and the masked sequence $\tilde{W}$ as

$$p(\mathcal{W}|\tilde{W}, O) = \prod_{w_l \in \mathcal{W}} p(w_l|\tilde{W}, O), \tag{4.2}$$

which represents a posterior probability distribution of masked tokens.

The token emission probability in Eq. (4.2) is computed using the Transformer-based AED architecture (see Section 2.3.2) as

$$H = \text{TransformerEncoder}(O) \in \mathbb{R}^{T' \times D^{\mathsf{model}}}, \tag{4.3}$$

$$Q = (\mathbf{q}_1, \cdots, \mathbf{q}_L) = \text{TransformerDecoder}(\tilde{W}, H) \in \mathbb{R}^{L \times D^{\mathsf{model}}}, \tag{4.4}$$

$$p(w_l|\tilde{W}, O) = \text{Softmax}\left(\text{Linear}_{D^{\mathsf{model}} \to |\mathcal{V}|}(\mathbf{q}_l)\right) \in [0, 1]^{|\mathcal{V}|}, \tag{4.5}$$

where $\text{Linear}_{D^{\mathsf{model}} \to |\mathcal{V}|}(\cdot)$ represents a linear layer for converting a D^{model}-dimensional vector into a $|\mathcal{V}|$-dimensional vector. Using the full-context attention mask (see Figure 2-8), the Transformer decoder in Eq. (4.4) takes query as the masked sequence $\tilde{W}$ and takes the key and value from the encoder output H. This enables the model to consider the bidirectional context within the output sequence for accurately predicting masked tokens. The objective function of the CMLM is defined by the negative log-likelihood of Eq. (4.2) as

$$\mathcal{L}^{\mathsf{cmlm}} \triangleq -\log p(\mathcal{W}|\tilde{W}, O). \tag{4.6}$$

I observed that training end-to-end ASR solely with the CMLM loss in Eq. (4.6) leads to poor performance, having issues such as the skipping and repetition of output tokens. To address this, a joint training approach combining CMLM with CTC (similar to the joint CTC and AED training described in Section 2.2.3) is proposed to facilitate the extraction of a robust alignment between the encoder output and the target sequence. With the shared encoder network, the objective function of the proposed joint training is defined as a linear interpolation of $\mathcal{L}^{\mathsf{ctc}}$ (from Eq. (2.20)) and $\mathcal{L}^{\mathsf{cmlm}}$ (from Eq. (4.6)) as

$$\mathcal{L}^{\mathsf{mask\text{-}ctc}} = \lambda^{\mathsf{mask\text{-}ctc}} \mathcal{L}^{\mathsf{ctc}} + (1 - \lambda^{\mathsf{mask\text{-}ctc}}) \mathcal{L}^{\mathsf{cmlm}}, \tag{4.7}$$

where $\lambda^{\mathsf{mask\text{-}ctc}}$ ($0 \leq \lambda^{\mathsf{mask\text{-}ctc}} \leq 1$) is a tunable weight to control the balance between the two losses.

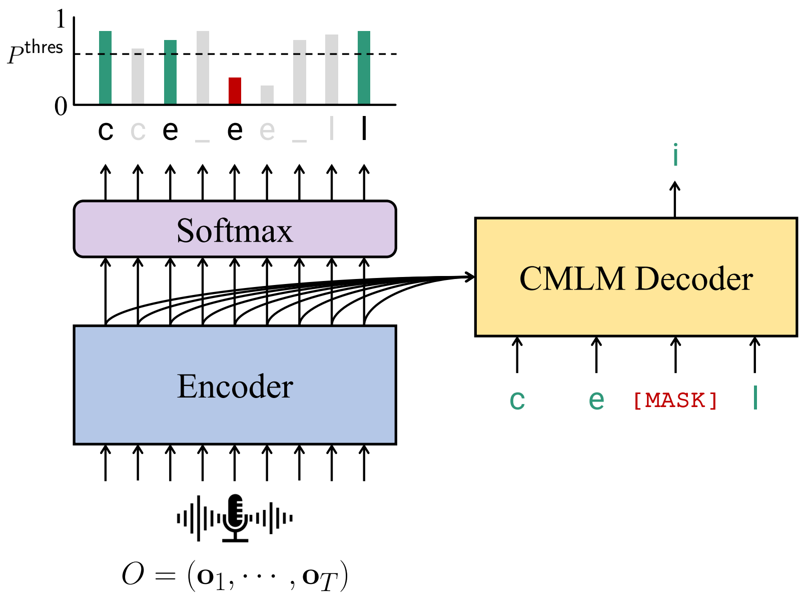


Figure 4-1: Overview of Mask-CTC decoding, correcting inaccurate hypothesis "ceel" to accurately predict "ceil." The output sequence begins with the results of greedy decoding from CTC, and tokens with low confidence are substituted with `[MASK]`, based on a threshold applied to their posterior probabilities. Then, the conditional masked language model (CMLM) decoder predicts the masked tokens by explicitly taking into account the context provided by the observed tokens.

Mask-Predict Decoding Based on CTC Predictions

Generally, in non-autoregressive modeling, the length of the output sequence must be predicted in advance to initiate the decoding process. In neural machine translation, the output length has been predicted explicitly from the fertility model (Gu et al., 2018) or a special length token `[LENGTH]` in the encoder (Ghazvininejad et al., 2019). In ASR, however, the task of predicting the output length is particularly challenging due to the distinct characteristics of the input acoustic signals and the output linguistic symbols. For example, the lengths of input utterances can vary significantly, influenced by factors such as speaking rate and the duration of silences. Audio-CMLM (Chen et al., 2021b) has predicted the position of the end-of-sentence token to determine the output length, but it has been argued that this strategy is only effective for short output sequences.

To overcome the challenge of initializing the output sequence, the proposed inference algorithm for Mask-CTC considers using predictions made by CTC, as illustrated in Figure 4-1. First, using the encoder output, the frame-level output sequence $\hat{A}$ is obtained by performing best path decoding of CTC (from Eq. (2.19)), and the token-level sequence $\hat{W}$ is derived by applying the collapsing function to $\hat{A}$, i.e., $\hat{W} = \mathcal{B}^{\text{ctc}}(\hat{A})$. Subsequently, the emission probability of each token

$\hat{w}_l \in \hat{W}$ is approximated based on the frame-level probabilities calculated in Eq. (2.18) as

$$p(w_l = \hat{w}_l | O) \approx \max_{t \in \mathcal{T}_l} p(a_t = \hat{w}_l | O), \tag{4.8}$$

where $\mathcal{T}_l$ represents a set of frame indices that correspond to the l-th token $\hat{w}_l$ after applying the collapsing function. Then, the masked sequence $\tilde{W}$ is obtained by replacing low-confidence tokens in $\hat{W}$ with the mask token as

$$\tilde{w}_l = \begin{cases} \texttt{[MASK]} & (p(w_l = \hat{w}_l | O) < P^{\mathsf{thres}}), \\ \hat{w}_l & (\text{otherwise}), \end{cases} \tag{4.9}$$

where P^{thres} is a threshold probability for determining whether to mask each output token. Finally, the masked tokens are predicted by the CMLM decoder for refinement, which utilizes the context from the unmasked high-confidence tokens and the conditioning from the encoder output.

The above mask prediction process in the CMLM decoder can be iteratively performed, allowing for the gradual prediction of masked tokens. Given the masked sequence $\tilde{W}$ from Eq. (4.9), the CMLM decoder updates a token at a masked position l as

$$\tilde{w}_l \leftarrow \arg\max_{w \in \mathcal{V}} p(w_l = w | \tilde{W}, O). \tag{4.10}$$

The update process is repeated for a fixed number of K iterations. In each iteration, tokens with the highest probabilities, specifically the top $\lfloor L/K \rfloor$, are selected for prediction.

With the proposed non-autoregressive decoding in Mask-CTC, the model is designed to operate without the need to predict the output length. Furthermore, the refinement of a CTC hypothesis through mask prediction is expected to correct errors caused by the conditional independence assumption.

4.2.3 Experimental Setting

To evaluate the effectiveness of Mask-CTC, I conducted speech recognition experiments using ESPnet (Watanabe et al., 2018) for comparing different end-to-end ASR models. The recognition performance was evaluated based on the CER or WER without using an external language model for decoding. Additionally, the inference speed was measured using the RTF.

Data

The experiments were carried out using three different tasks: the Wall Street Journal (WSJ) and TED-LIUM2 datasets were used for English ASR, and the VoxForge dataset was used for Italian ASR. For the network inputs, I extracted 80 mel-scale filterbank coefficients with three-dimensional pitch features using Kaldi (Povey et al., 2011). To avoid overfitting, the speech inputs were augmented using speed perturbation (Ko et al., 2015) or SpecAugment (Park et al., 2019), depending on the tasks and models. For the tokenization of target transcriptions, characters (Latin alphabets) were used for WSJ and VoxForge. For TED-LIUM2, I used SentencePiece (Kudo, 2018) to construct a subword vocabulary, where the vocabulary size was set to 500.

Evaluated Models

I evaluated and compared different end-to-end ASR models, each of which could either be autoregressive or non-autoregressive. **AR** indicates an autoregressive model trained by the joint CTC and AED objective, as defined in Eq. (2.43). **CTC** indicates a non-autoregressive model trained by the CTC objective, as defined in Eq. (2.20). **Mask-CTC** is the proposed non-autoregressive model trained by the joint CTC and CMLM objective, as defined in Eq. (4.7).

Model Architecture

The above models were constructed using either the Transformer or Conformer-based architecture. For Transformer, the number of heads N^{head}, the dimension of a self-attention layer D^{model}, and the dimension of a feed-forward layer D^{ff} were set to 4, 256, and 2048, respectively. For Conformer, I used the same configuration for the self-attention layer as in Transformer. The only difference was that D^{ff} was adjusted to 1024 to prevent an increase in the number of parameters. The kernel size of the convolution module was tuned from values of 7, 15, and 31, depending on the task. The encoder network consisted of two Conv2D down-sampling layers followed by $N^{\mathsf{enc}} = 12$ Transformer or Conformer encoder layers. For AR and Mask-CTC, the decoder network consisted of $N^{\mathsf{dec}} = 6$ Transformer decoder layers.

Training and Decoding Configurations

The hyper-parameters for model training, including the number of training epochs and mini-batch size, were tuned based on the recipes provided by ESPnet (as specified in Appendix A). All the codes and recipes used in the experiments have been made publicly available to ensure reproducibility.[1] For training the AR model, the loss weight $\lambda^{\mathsf{ctc\text{-}aed}}$ in Eq. (2.43) was set to 0.3. For training the Mask-CTC model, the loss weight $\lambda^{\mathsf{mask\text{-}ctc}}$ in Eq. (4.7) was set to 0.3. After training, a final model was obtained for evaluation by averaging model parameters over 10 to 50 checkpoints with the best validation performance. For decoding the AR model, the joint CTC and AED decoding algorithm was performed (see Section 2.2.3 for details), using the weight ξ set to 0.3 and a beam size of 10. For decoding the CTC model, the best path decoding algorithm was performed (see Section 2.2.1 for details). For decoding the Mask-CTC model, the threshold probability P^{thres}

[1] https://github.com/espnet/espnet

Table 4.1: WER and RTF on WSJ task. K denotes the number of inference steps required to generate each output token. RTF was measured on dev93 using CPU. For each Mask-CTC model, RTF was calculated for $K = 10$. †Transformer-based architecture trained without SpecAugment.

			WER [%] (↓)									
ID	**Model**	**Params** [M]	dev93				eval92				**RTF** (↓)	**Speedup** (↑)
	Autoregressive		$K = L$ (avg. 99.9)				$K = L$ (avg. 100.3)					
A1	Transformer-AR	27.2	13.5				10.8				$0.456_{\pm 0.005}$	1.00×
A2	+ beam search	27.2	12.8				10.6				$5.067_{\pm 0.012}$	0.09×
A3	Conformer-AR	30.4	11.4				8.8				$0.474_{\pm 0.009}$	0.96×
A4	+ beam search	30.4	**11.1**				**8.5**				$5.094_{\pm 0.031}$	0.09×
			K				K					
	Non-autoregressive		0	1	5	10	0	1	5	10		
B1	Transformer-CTC	17.7	19.4	–	–	–	15.5	–	–	–	$0.021_{\pm 0.000}$	21.71×
B2	Transformer-Mask-CTC	27.2	15.5	15.2	**14.9**	**14.9**	12.5	12.2	**12.0**	**12.0**	$0.063_{\pm 0.001}$	7.24×
C1	Conformer-CTC	20.9	13.0	–	–	–	10.8	–	–	–	$0.033_{\pm 0.000}$	13.81×
C2	Conformer-Mask-CTC	30.4	11.9	11.8	**11.7**	**11.7**	9.4	9.2	9.2	**9.1**	$0.063_{\pm 0.000}$	7.24×
--	Imputer† (Chan et al., 2020)	–	–				12.7 ($K = 8$)				–	–

was tuned from values of 0.9, 0.99, and 0.999. RTF was measured using utterances in the WSJ dev93 set using Intel(R) Xeon(R) Gold 6148 CPU, 2.40GHz.

4.2.4 Results

Main Results

Table 4.1 lists the WSJ results evaluated on the WER and RTF. Here, K represents the number of inference steps required to produce an output sequence, and when $K = 0$ for Mask-CTC, the evaluation was based on the CTC predictions derived from best path decoding. By comparing the results for non-autoregressive models, the greedy CTC predictions of Mask-CTC outperformed the standard CTC-based model (e.g., B1 vs. B2), owing to the synergistic effect of joint training with the CMLM objective. By applying the proposed iterative refinement algorithm to CTC predictions, Mask-CTC consistently improved its performance. The performance of CTC significantly improved by using Conformer (B1 vs. C1), and similarly, Mask-CTC greatly benefited from using the better architecture (C2). Compared to the autoregressive models, Mask-CTC yielded results more closely aligned with those of AR than with CTC (A3 vs. C1 vs. C2). Regarding inference speed measured in the RTF, Mask-CTC achieved speeds up to 7.24 times faster compared to AR (A3 vs. C2), while maintaining a satisfactory level of recognition performance. Mask-CTC exhibited superior performance compared to previous results from a non-autoregressive end-to-end ASR model (Chan et al., 2020) that also combines CTC with masked language modeling.

Table 4.2: Decoding example for utterance 443c040i in WSJ eval92. The output sequence was first initialized by the CTC predictions, and low-confidence tokens were replaced with mask tokens ("–") based on their output probabilities. The masked tokens were then iteratively predicted conditioning on the other unmasked tokens. Red indicates characters with errors and blue indicates ones recovered by Mask-CTC decoding.

CTC	`they favor un anounced checks by roving rather than in house anspectors ··· in sefood processing`
$k = 0$	`they favor un--noun--d ch--ks by roving rather than -n-house -nspectors ··· in --food processing`
$k = 1$	`they favor un--noun--d ch--ks by roving rather than -n-house -nspectors ··· in --food processing`
$k = 2$	`they favor unannoun--d ch--ks by roving rather than -n-house inspectors ··· in --food processing`
$k = 3$	`they favor unannounc-d ch-cks by roving rather than in house inspectors ··· in --food processing`
Mask-CTC	`they favor unannounced checks by roving rather than in house inspectors ··· in sifood processing`
Refeference	`they favor unannounced checks by roving rather than in house inspectors ··· in seafood processing`

Table 4.3: WERs on VoxForge Italian and TED-LIUM2 tasks. K denotes the number of inference steps required to generate each output token. Results with beam search are reported in parentheses.

		Test WER [%] (↓)	
Model	K	VoxForge	TED-LIUM2
Transformer-AR	L	35.5	9.5
+ beam search		35.7	8.9
Conformer-AR	L	**29.8**	8.4
+ beam search		**29.8**	**7.9**
Transformer-CTC	0	56.1	16.6
Transformer-Mask-CTC	10	**38.3**	**10.9**
Conformer-CTC	0	31.8	9.5
Conformer-Mask-CTC	10	**29.2**	**8.6**

Example Decoding Process of Mask-CTC

Table 4.2 shows an example decoding process of Mask-CTC recognizing a sample in the WSJ evaluation set. Here, it can be observed that the CTC predictions included errors mainly coming from substitution errors due to incomplete word spelling. By applying Mask-CTC decoding, the spelling errors were successfully recovered by considering contextual dependencies between characters at the word level. However, as can be seen in the error for "sifood," Mask-CTC was not capable of recovering errors derived from character-level insertion or deletion errors because the length allocated to each word was fixed by the CTC predictions. Such errors can be resolved by introducing an additional mechanism that allows the model to delete and insert tokens during the mask prediction process (Higuchi et al., 2021b).

Table 4.4: BLEU scores of speech translation models on Fisher-CallHome Spanish.

	BLEU (↑)				
	Fisher			CallHome	
Model	dev	dev2	test	devtest	evltest
AR	47.01	47.89	47.19	18.11	17.95
CTC	45.57	46.97	45.97	15.99	15.91
Mask-CTC					
+ CTC greedy	45.93	46.82	46.17	15.73	15.60
+ original decoding	44.80	45.40	44.39	14.14	14.14
+ mask-predict	47.43	48.14	46.96	16.52	16.42
+ restricted mask-predict	**49.94**	**49.42**	**48.66**	**16.96**	**16.79**

Results on Other Corpora

Table 4.3 shows the results on VoxForge and TED-LIUM2. Consistent with the findings in the WSJ task, Mask-CTC outperformed the standard CTC-based model, indicating its effectiveness across different languages and varying amounts of data. Remarkably, with Conformer, Mask-CTC achieved results on par with AR in the VoxForge task, which suggests that masked language modeling is especially effective for capturing linguistic information in limited resource settings.

4.2.5 Application to End-to-End Speech Translation

To see a potential application to other speech tasks, I applied the Mask-CTC framework to the end-to-end speech translation task, following the ESPnet-ST toolkit (Inaguma et al., 2020). For non-autoregressive models, including CTC and Mask-CTC, sequence-level knowledge distillation (Kim and Rush, 2016) was used during training. Table 4.4 shows the results on the Fisher-CallHome Spanish corpus (see Appendix A.10 for corpus details). Unlike ASR, input-output alignments are not monotonic in this task, and the confidence filtering based on the output probabilities did not work well. Next, I performed the original mask-predict decoding algorithm proposed in neural machine translation (Ghazvininejad et al., 2019) by starting from all mask tokens, which resulted in some gains over CTC. Finally, I initialized an output sequence with the filtered CTC output as in the ASR task and then performed the mask-predict decoding. Here, the number of masked tokens at each iteration is truncated by the number of initial masked tokens (*restricted mask-predict*) to keep information from the CTC predictions for the later iterations. This way, the results were further improved from the CTC greedy results by a large margin. Moreover, interestingly, Mask-CTC outperformed the autoregressive model on this corpus.

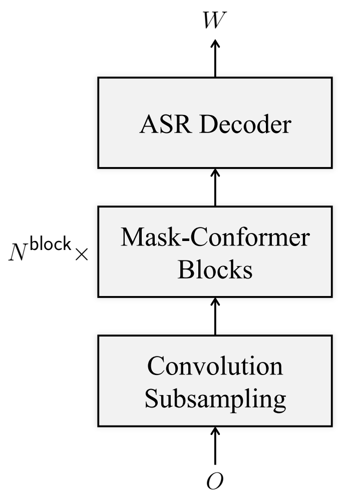


Figure 4-2: Overview of proposed Mask-Conformer model and its network components.

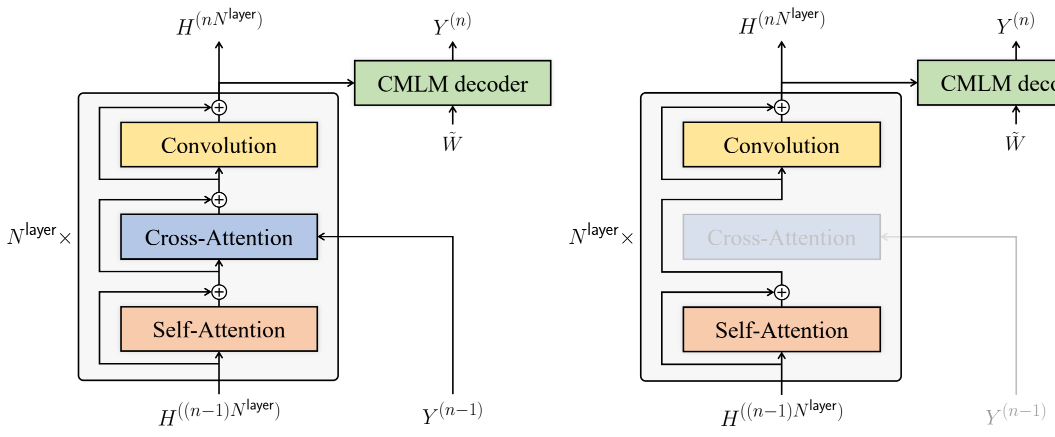
4.3 Mask-Conformer: Augmenting Conformer with CMLM Decoder

This section introduces Mask-Conformer, a novel end-to-end ASR model designed to effectively utilize linguistic information derived from the CMLM. In contrast to Mask-CTC, which focuses on refining output sequences at the token level, Mask-Conformer aims to enhance the frame-level speech encoding process within an end-to-end ASR model. This is achieved by explicitly conditioning the encoder on the outputs of the CMLM decoder, which enables the model to reinterpret speech inputs based on contextualized linguistic representations.

4.3.1 Mask-Conformer

Figure 4-2 illustrates the proposed **Mask-Conformer**, which introduces a CMLM decoder to the Conformer encoder architecture (see Section 2.3.3 for details). Mask-Conformer is composed of N^{block} sub-blocks sandwiched between the subsampling and ASR decoder (i.e., the transducer decoder) layers. Figure 4-3 shows the detailed structure of the sub-blocks. Each sub-block is constructed of N^{layer} Conformer encoder layers, with an additional CMLM decoder and cross-attention module. The CMLM decoder solves the token in-filling task (Devlin et al., 2019) while conditioned on the encoder outputs. Token-level linguistic information captured by the CMLM decoder is explicitly fed to the next block via the cross-attention module (Figure 4-3(a)). The cross-attention module can be enabled or disabled (Figure 4-3(b)) based on what information is available during training and inference. The following delves into the CMLM decoder and cross-attention modules implemented in the Mask-Conformer model.

CMLM Decoder Given a partially masked output sequence $\tilde{W}$ (as defined by Eq. (4.1)) and the output of the i-th Conformer encoder layer $H^{(i)}$, the CMLM decoder computes a posterior



(a) Block with cross-attention. (b) Block without cross-attention.

Figure 4-3: Mask-Conformer block. Mask-Conformer builds its sub-block by augmenting a Conformer encoder layer with an additional CMLM decoder and cross-attention module. The CMLM decoder is trained to capture token-level linguistic information explicitly, which is used to condition the encoder layers via cross-attention. The cross-attention module can be enabled (a) or disabled (b) based on what information is available during training and inference.

probability distribution of mask tokens similarly to Eq. (4.2) as

$$p(\mathcal{W}|\tilde{W}, H^{(i)}) = \prod_{w_l \in \mathcal{W}} p(w_l|\tilde{W}, H^{(i)}), \tag{4.11}$$

where $i \in \{nN^{\text{layer}}|n = 1, \cdots, N^{\text{block}}\}$ is the layer index at which the CMLM decoder is applied, i.e., every N^{layer} layers of Conformer encoder (see Figure 4-3). The token emission probability in Eq. (4.11) is computed similarly to Eq. (4.5) as

$$Y^{(n)} = (\mathbf{y}_1^{(n)}, \cdots, \mathbf{y}_L^{(n)}) = \text{TransformerDecoder}(\tilde{W}, H^{(i)}) \in \mathbb{R}^{L \times D^{\text{model}}}, \tag{4.12}$$

$$p(w_l|\tilde{W}, H^{(i)}) = \text{Softmax}\left(\text{Linear}_{D^{\text{model}} \to |\mathcal{V}|}(\mathbf{y}_l^{(n)})\right) \in [0,1]^{|\mathcal{V}|}, \tag{4.13}$$

where $n \in \{1, \cdots, N^{\text{block}}\}$. Note that the parameters of the Transformer decoder are shared among the Mask-Conformer blocks. However, an alternate structure could use different decoder modules per block.

Conformer with Cross-Attention With the cross-attention (CA) mechanism, the Conformer's embedding process of O in Eq. (2.61) is redefined as

$$H' = \text{ConformerEncoderWithCA}(O, \tilde{W}) \in \mathbb{R}^{T' \times D^{\text{model}}}, \tag{4.14}$$

where $\tilde{W}$ is a masked output sequence used as an input to the CMLM decoder (i.e., Eq. (4.12)).

Algorithm 2 Inference algorithm of Mask-Conformer

```
Input: Input sequence $O$; Beam size $B$; Threshold probability $P^{\text{thres}}$
1: $H = \text{ConformerEncoder}(O)$                         ▷ Forward w/o cross-attention
2: $\hat{W} = \text{DecodeTransducer}(H, B)$                ▷ First-pass decoding
3: for all $\hat{w}_l \in \hat{W}$ do                       ▷ Mask low-confidence tokens
4:     $\hat{w}_l = \begin{cases} \texttt{[MASK]}, & \text{if } p(w_l = \hat{w}_l | \hat{W}, H) \leq P^{\text{thres}} \\ \hat{w}_l, & \text{otherwise} \end{cases}$
5: end for
6: $H' = \text{ConformerEncoderWithCA}(O, \hat{W})$         ▷ Forward w/ cross-attention
7: $\hat{\hat{W}} = \text{DecodeTransducer}(H', B)$         ▷ Second-pass decoding
8: return $\hat{\hat{W}}$
```

The modified encoder layer operates similarly to Eqs. (2.62) to (2.65), with an additional cross-attention module inserted between Eqs. (2.63) and (2.64) as

$$\bar{\bar{H}}^{(i)} \leftarrow \begin{cases} \bar{\bar{H}}^{(i)} & (n \leq N^{\text{layer}}), \\ \bar{\bar{H}}^{(i)} + \text{MHA}^{(i)}(\text{LayerNorm}(\bar{\bar{H}}^{(i)}), Y^{(n)}, Y^{(n)}) & (n > N^{\text{layer}}), \end{cases} \tag{4.15}$$

where $i \in \{1, \cdots, N^{\text{block}} \times N^{\text{layer}}\}$, $n = \lfloor (i-1)/N^{\text{layer}} \rfloor$, and the left arrow symbol ($\leftarrow$) represents the reassignment of $\bar{\bar{H}}^{(i)}$. Similarly to the cross-attention mechanism in Eq. (2.58), $\text{MHA}(\cdot)$ in Eq. (4.15) takes queries from the encoder's hidden states, along with keys and values from the output of the CMLM decoder. This enables the model to infuse linguistic information from the CMLM decoder into speech information processed in the encoder. Note that when $n \leq N^{\text{layer}}$, the block behaves the same as the standard Conformer encoder, since $Y^{(0)}$ is not available.

Inference

Algorithm 2 explains the inference algorithm of Mask-Conformer, designed to explicitly utilize token-level contextualized information learned in the CMLM decoder. As the decoder input (i.e., a masked token sequence $\hat{W}$) is not initially available, a two-pass inference strategy is employed based on confidence-based masking, which is similar to Mask-CTC (see Section 4.2.2) and prior studies in Ghazvininejad et al. (2019); Baskar et al. (2022). The algorithm begins by computing encoder outputs H *without the cross-attention module* (i.e., Figure 4-3(b)), where the forward process is the same as the standard Conformer in Eq. (2.61) (line 1). With the encoder outputs H and a beam size of B, a hypothesis $\hat{W}$ is obtained via transducer decoding (see Section 2.2.2 for details) (line 2). Each token in $\hat{W}$ is then scored using the CMLM decoder with Eq. (4.13), where a token is substituted with `[MASK]` if its output probability is less than or equal to a predetermined threshold P^{thres} (line 4). Using the masked sequence, the algorithm computes encoder outputs H'

with cross-attention as in Eq. (4.14) (i.e., Figure 4-3(a)), infusing decoder information explicitly into the encoder (line 5). Finally, with the updated encoder outputs H', a second-pass result $\hat{\hat{W}}$ is obtained from transducer decoding with a beam size of B (line 6). With this two-pass decoding strategy, it is expected that $\hat{\hat{W}}$ will be refined from $\hat{W}$ with the aid of the linguistic information extracted by the CMLM decoder.

Training

Given ground-truth and masked sequences W and $\tilde{W}$, the loss for the CMLM decoder is defined by the negative log-likelihood of Eq. (4.11) as

$$\mathcal{L}^{\text{cmlm}}(\mathcal{W}|\tilde{W}, H^{(i)}) = -\log p(\mathcal{W}|\tilde{W}, H^{(i)}), \tag{4.16}$$

which is computed based on the i-th encoder output $H^{(i)}$. To obtain $\tilde{W}$ during training, $P^{\text{mask}}\%$ of the tokens in the ground truth sequence W are replaced with `[MASK]`, where the masked positions are randomly sampled from a Bernoulli distribution as in Devlin et al. (2019). Additionally, $P^{\text{rep}}\%$ of the tokens are replaced with random tokens in $\mathcal{V}$ to simulate the possibility of unmasked incorrect tokens during the two-pass decoding process. Note that the positions selected for the randomly masked and replaced tokens do not overlap.

Multi-tasking the transducer objective from Eq. (2.30) and the CMLM decoder objective from Eq. (4.16), Mask-Conformer computes its loss by combining losses with and without cross-attention (Figures 4-3(a) and 4-3(b)). The loss with cross-attention is defined using the encoder outputs derived from Eq. (4.14) as

$$\mathcal{L}^{\text{ca}} = \mathcal{L}^{\text{tra}}(W|H') + \sum_{n=1}^{N^{\text{block}}} \mathcal{L}^{\text{cmlm}}(\mathcal{W}|\tilde{W}, H'^{(nN^{\text{layer}})}), \tag{4.17}$$

where $H'^{(nN^{\text{layer}})}$ represents the outputs from the (nN^{layer})-th encoder layer. Similarly, the loss without cross-attention is defined as

$$\mathcal{L}^{\text{noca}} = \mathcal{L}^{\text{tra}}(W|H) + \sum_{n=1}^{N^{\text{block}}} \mathcal{L}^{\text{cmlm}}(\mathcal{W}|\tilde{W}, H^{(nN^{\text{layer}})}), \tag{4.18}$$

where the encoder outputs are obtained from the standard forward computation of Conformer (with Eq. (2.61)). Combining $\mathcal{L}^{\text{ca}}$ from Eq. (4.17) and $\mathcal{L}^{\text{noca}}$ from Eq.(4.18), the overall Mask-Conformer loss is defined as

$$\mathcal{L}^{\text{mask-cfm}} = \mathcal{L}^{\text{ca}} + \mathcal{L}^{\text{noca}}. \tag{4.19}$$

Note that the losses are weighted equally among the CMLM losses in Eqs. (4.17) and (4.18) as well as between $\mathcal{L}^{\text{ca}}$ and $\mathcal{L}^{\text{noca}}$ in Eq. (4.19).

4.3.2 Experimental Setting

Data

The experiments were carried out using English ASR tasks, including single-domain LibriSpeech and multi-domain SpeechStew. See Appendix A for corpus details. For all data preparation, I adhered to the prior work on LibriSpeech (Gulati et al., 2020) and SpeechStew (Chan et al., 2021).

Models and Architectures

The baseline end-to-end ASR model is a well-tuned Conformer model from Gulati et al. (2020); Chan et al. (2021), which achieves state-of-the-art performance with standard transducer training as described in Section 2.2.2. The models were constructed using the medium (`M`) or large (`L`) Conformer-transducer architectures, as specified in Gulati et al. (2020). All the models employed a single LSTM layer with $D^{\mathsf{lstm}} = 640$ units for the transducer decoder. The proposed Mask-Conformer uses a CMLM decoder with a stack of $N^{\mathsf{dec}} = 6$ Transformer decoder layers (Vaswani et al., 2017), with hyper-parameters (e.g., number of heads) matching those of the self-attention modules in the Conformer encoder. Each Mask-Conformer block consisted of $N^{\mathsf{layer}} = 4$ encoder layers. All the models were implemented based on the Lingvo (Shen et al., 2019) and TensorFlow (Abadi et al., 2016) toolkits.

Training Configuration

I followed the previous work in Gulati et al. (2020); Chan et al. (2021) for the training configurations. The models were trained up to 150k steps using the Adam optimizer (Kingma and Ba, 2015) with $(\beta_1, \beta_2, \epsilon) = (0.9, 0.98, 10^{-9})$. The Noam learning rate scheduling (Vaswani et al., 2017) was used with warmup steps of 10k, where the learning rate constant was tuned from $\{2.5, 5.0\}$. The batch size was set to 2048 for LibriSpeech and 8192 for SpeechStew. For regularization, I applied residual dropout (Srivastava et al., 2014) with a probability of 0.1, variational noise (Jim et al., 1996) with a standard deviation of 0.075, and L2 regularization with a weight of 10^{-6}. I augmented speech data using SpecAugment (Park et al., 2020a), where the number of frequency and time masks were 2 and 10, and the size of frequency and time masks were 27 and $0.05T$. For Mask-Conformer training, both the token masking and replacing ratios, P^{mask} and P^{rep}, were set to 15%.

Decoding Configuration

Exponential-moving-averaged model weights aggregated with decay rate 0.9999 were used for final evaluation (Zhang et al., 2020b). With the transducer decoder, beam search decoding was performed with a beam size of $B = 8$. The threshold probability for Mask-Conformer's second-pass decoding was set to $P^{\mathsf{thres}} = 0.9$.

Table 4.5: WER on single-domain LibriSpeech task.

		Params [M]		**WER** [%] (↓)			
				Dev		Test	
`ID`	**Model**	Training	Inference	clean	other	clean	other
`M0`	Conformer-`M`	30.7	30.7	2.1	5.0	2.2	5.1
	+ larger decoder (1 → 4)	40.5	40.5	2.0	5.0	2.2	5.1
`M1`	Mask-Conformer-`M`	43.8	30.7	**1.9**	4.8	**2.1**	4.8
`M2`	+ two-pass decoding	43.8	43.8	**1.9**	**4.7**	**2.1**	**4.7**
`L0`	Conformer-`L`	118.8	118.8	1.8	4.2	2.0	4.3
	+ larger encoder (17 → 23)	156.6	156.6	1.9	4.2	2.0	4.4
`L1`	Mask-Conformer-`L`	158.6	118.8	1.9	4.1	2.0	4.2
`L2`	+ two-pass decoding	158.6	158.6	**1.8**	**4.0**	**1.9**	**4.1**

4.3.3 Results

Main Results on Single-Domain LibriSpeech

Table 4.5 compares the WER of the baseline Conformer with the proposed Mask-Conformer on LibriSpeech. The suffixes `M` and `L` refer to model sizes *Medium* and *Large*. With the introduction of the CMLM decoder, Mask-Conformer performed better (`M0` vs. `M1,M2`). Note that during first-pass decoding, the number of parameters used in Mask-Conformer was the same as that of the baseline. The second-pass mask-predict algorithm further improved the performance (`M1` vs. `M2`), improving an already strong baseline even further by explicitly exploiting the token-level linguistic information captured by the CMLM decoder. The above findings remain consistent across the larger models (`L⋆`), with Mask-Conformer achieving distinct improvements over Conformer.

Results on Multi-Domain SpeechStew

Table 4.6 lists results on the multi-domain SpeechStew task. Mask-Conformer resulted in superior performance compared to Conformer on Common Voice, LibriSpeech, Switchboard and WSJ. I attribute this to the fact that multiple domains exhibit differing error patterns or that length differences in their utterances require different masking strategies. Moreover, I find that two-pass rescoring demonstrated a small but consistent improvement across AMI, Common Voice and TED-LIUM3. This suggests that conditioning with the mask-predict decoder is particularly advantageous for domain-specific tasks, where the decoder can be more useful within an adapter or domain-conditioned setting. I also compare Mask-Conformer to large foundation models (Zhang et al., 2022c; Chen et al., 2022b) pre-trained on a vast amount of unlabeled data and in the case of Chen et al. (2022b) additional text. Notice that these models perform exceptionally well on

Table 4.6: WER on multi-domain SpeechStew task, evaluated across various test sets, including AMI, Common Voice (CV), LibriSpeech (LS), Switchboard/Fisher (SWBD), TED-LIUM3 (TED3), and Wall Street Journal (WSJ). In addition to the baseline Conformer-L model, results of pre-existing foundation models pre-trained with unlabeled data are listed. The numbers of the previous models were obtained from the published papers. †Evaluated with punctuation removal following Likhomanenko et al. (2020).

		WER [%] (↓)								
		AMI		CV†	LS		SWBD		TED3	WSJ
Model	**Params** [B]	IHM	SDM1	Test	clean	other	SWBD	CH	Test	eval92
Conformer-L (Chan et al., 2021)	0.1	9.0	21.7	9.7	2.0	4.0	4.7	8.3	5.3	1.3
MAESTRO (Chen et al., 2022b)	0.6	8.5	21.9	8.1	1.5	2.8	4.3	8.0	4.9	–
ConformerXXL (Zhang et al., 2022c)	1.0	8.6	17.7	7.8	1.9	3.5	4.8	10.6	5.9	1.3
Mask-Conformer-L	0.1	9.1	21.8	9.6	1.9	3.8	4.5	8.7	5.6	1.2
+ two-pass decoding		9.0	21.6	9.4	1.9	3.8	4.5	8.7	5.5	1.2

specific tasks (e.g., LibriSpeech and TED-LIUM3) whose domains match the ones of unlabeled data used for pre-training. Interestingly, the performance of Mask-Conformer outperformed the much larger Conformer-XXL model on CallHome, Fisher, WSJ, and TED-LIUM3 indicating the proposed architecture led to improving the model's generalization capability without an increase in parameters and training data.

Comparison to Prior Work

Table 4.7 presents WERs on LibriSpeech, comparing the baseline and Mask-Conformer models with the previously proposed competitive models. It should be noted that I refer to the published papers for the prior results, and the comparison is not necessarily in an equivalent setting (e.g., deep learning library and computational environment). Overall, Mask-Conformer exhibited highly competitive performance, pushing the limits of the previous state-of-the-art results.

4.3.4 Analysis

Does the gain come from more parameters?

For a fairer comparison in terms of the model capacity, in Table 4.5, I list WERs of the baseline models that were trained with an increased depth in either the decoder or encoder network.

Decoder I first augmented the number of LSTM layers ($1 \rightarrow 4$) in the transducer decoder of Conformer-M (M0), thereby attaining an equivalent model size to that of Mask-Conformer-M (M2). The deeper decoder yielded minimal improvements in ASR performance, which aligns with the observations from previous works (Tripathi et al., 2020; Ghodsi et al., 2020; Botros et al., 2021; Shrivastava et al., 2021). In contrast, Mask-Conformer has successfully enhanced ASR perfor-

Table 4.7: WER on LibriSpeech task, comparing Mask-Conformer with previous models. The numbers of the previous models are obtained from the published papers.

		WER [%] (↓)			
		Dev		Test	
Model	**Params** [M]	clean	other	clean	other
Previous Models					
ContextNet (M) (Han et al., 2020)	31.4	–	–	2.4	5.4
Conformer (M) (Gulati et al., 2020)	30.7	–	–	**2.3**	**5.0**
Citrinet (Majumdar et al., 2021)	36.5	–	–	3.1	7.8
E-Branchformer (M) (Kim et al., 2023)	41.1	2.3	5.7	2.5	5.6
Ours					
Conformer-`M` (`M0`)	30.7	2.1	5.0	2.2	5.1
Mask-Conformer-`M` (`M2`)	43.8	1.9	4.7	**2.1**	**4.7**
Previous Models					
Transformer (Zhang et al., 2020a)	139.0	–	–	2.4	5.6
ContextNet (L) (Han et al., 2020)	112.7	2.0	4.6	**2.1**	4.6
Conformer (L) (Gulati et al., 2020)	118.8	1.9	4.4	**2.1**	**4.3**
Conformer CTC/Att. (Peng et al., 2022)	116.2	2.2	5.6	2.5	5.5
Citrinet (Majumdar et al., 2021)	142.0	–	–	2.5	6.2
Branchformer (Peng et al., 2022)	116.2	2.2	5.5	2.4	5.5
E-Branchformer (L) (Kim et al., 2023)	148.9	–	–	**2.1**	4.6
Ours					
Conformer-`L` (`L0`)	118.8	1.8	4.2	2.0	4.3
Mask-Conformer-`L` (`L2`)	158.6	1.8	4.0	**1.9**	**4.1**

mance through the effective design and integration of the decoder network.

Encoder I then increased the number of Conformer encoder layers ($17 \rightarrow 23$) in Conformer-`L` (`L0`), which resulted in the same model size as that of Mask-Conformer-`L` (`L2`). It appeared that there was little performance gain from increasing the encoder size. This can be attributed to the over-parameterization of the encoder in relation to the amount of data available in LibriSpeech (Zhang et al., 2020b). On the other hand, Mask-Conformer strategically allocates parameters to the decoder network, enabling it to effectively capture token-level linguistic information for facilitating the training process of the encoder network.

How effective is second-pass mask-predict decoding?

Figure 4-4 shows WERs of Mask-Conformer (`M2`, `L2`), averaged on the dev-clean and -other sets, depending on the threshold probability P^{thres} used to filter low-confidence tokens (Algorithm 2

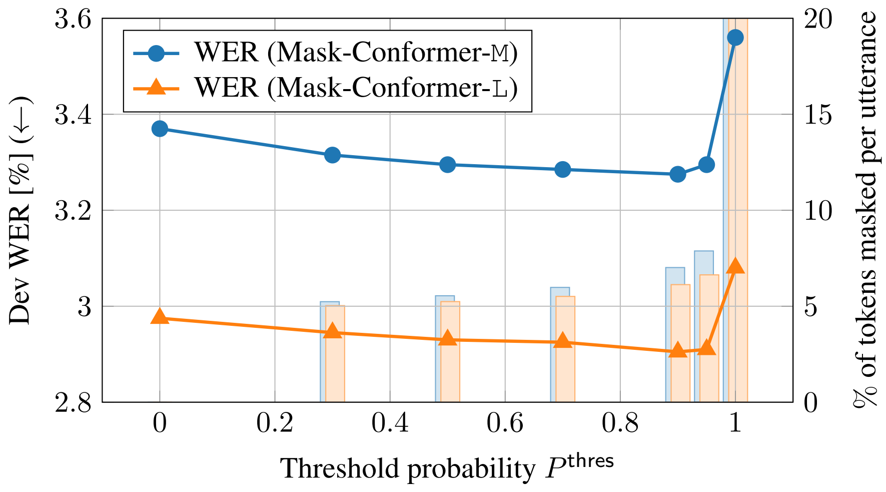


Figure 4-4: Influence of threshold probability P^{thres} on WERs during two-pass decoding. The bar plots show the percentage of tokens that were replaced with `[MASK]` in each output sequence. This figure is reproduced from my conference paper (Higuchi et al., 2023d).

line 4). Filtering too many tokens can leave too much uncertainty for the model to recover from, filtering too few can prevent any error correction. In this section, I explore the optimal threshold, all else being held constant. When $P^{\text{thres}} = 0.0$, no masks were applied to the first-pass result, and, during the second pass, the model had full access to the entire output sequence. While with $P^{\text{thres}} = 1.0$, all tokens were masked. As the threshold P^{thres} is increased from 0.0, there was a gradual increase in the number of masked tokens, which correspondingly led to improving WERs. Mask-Conformer achieved its best performance with $P^{\text{thres}} = 0.9$ by effectively applying partial masks and repredicting them based on explicit contextual information. Though the reduction in WER may appear modest, the observed gain from two-pass decoding is promising, particularly considering the competitiveness and saturation of the LibriSpeech task.

Random masking/replacing ratios during training.

Table 4.8 investigates WERs of Mask-Conformer (`M2`) trained with different combinations of random masking and replacing ratios, P^{mask} and P^{rep}. Merely applying random masks still performed better than the baseline Conformer-`M` (`M0`). Furthermore, the inclusion of random replacement further improved the performance, anticipating the possibility of unmasked incorrect tokens during two-pass decoding. Increasing the ratios led to a slight degradation, suggesting that a similar masking ratio should be used during training and inference.

Table 4.8: WER of Mask-Conformer-`M` (`M2` from Table 4.5) trained with different combinations of random masking and replacing ratios, P^{mask} and P^{rep}.

		WER [%] (↓)	
P^{mask}	P^{rep}	dev-clean	dev-other
15%	0%	2.0	4.8
15%	15%	**1.9**	**4.7**
30%	0%	**1.9**	4.8
30%	30%	2.0	4.8

Table 4.9: WER on LibriSpeech task for comparing models trained with different types of intermediate decoders.

	WER [%] (↓)			
	Dev		Test	
Intermediate Decoder Type	clean	other	clean	other
None (`M0`)	2.1	5.0	2.2	5.1
CTC	2.1	5.1	2.2	5.2
Transducer	2.0	5.0	**2.1**	4.9
Attention	2.0	**4.7**	2.2	**4.8**
CMLM	**1.9**	4.8	**2.1**	**4.8**

Comparison Across Different Decoder Types

Mask-Conformer was effective *without* two-pass decoding (e.g., `M1`) (Section 4.3.3). Thus, I trained the model exclusively using Eq. (4.18), with the option to substitute the CMLM decoder loss $\mathcal{L}^{\text{cmlm}}$ with other loss functions. Such training resembles the intermediate regularization technique that applies auxiliary losses to intermediate encoder layers, which has been explored using auxiliary CTC (Sanabria and Metze, 2018; Tjandra et al., 2020; Lee and Watanabe, 2021; Lee et al., 2022) or attention losses (Zhang et al., 2022b; Komatsu and Fujita, 2023), as also introduced in Chapter 3. Table 4.9 lists WERs on LibriSpeech, comparing Conformer-`M` (`M0`) trained with different types of intermediate decoder losses. In my experimental setup, CTC had a limited impact on enhancing the model performance, likely due to the absence of the decoder network. The other losses effectively improved the baseline, indicating the importance of decoder capacity. While the attention decoder is also a viable option, Mask-Conformer benefits from the efficient non-autoregressive sequence processing, particularly during inference, thanks to the CMLM decoder.

4.3.5 Limitations

Mask-Conformer's novelty is in its structure of providing both forward and backward propagation training signals at multiple points in the encoder. This is accomplished through incremental decoding signal through the CMLM decoder, and an ability to inject hypotheses at multiple layers of the encoder network. The latter provides the capability to perform recognition and hypothesis correction in a single model.

However, I acknowledge three limitations to the current formulation of the Mask-Conformer. 1) The two-pass recognition requires a second pass that is as computationally intensive as the first pass. 2) While the Mask-Conformer requires very few additional parameters to accomplish this capability, its structure introduces more hyper-parameters (see Section 4.3.4). These include the position, number, and size of CMLM decoders as well as the masking threshold. While I have analyzed many of these, they likely interact in non-obvious ways. 3) The presentation of the model's current hypotheses as the correction signal may be exacerbating exposure bias for the model. This is similar to the risk of previous self-training techniques. While this is addressed to some degree by masking the hypotheses prior to injection, further mitigation may be necessary to improve the second-pass inference performance.

4.4 Mask-CTC-Based Encoder Pre-Training for Streaming End-to-End ASR

This section aims to employ Mask-CTC for pre-training end-to-end streaming ASR models. By jointly optimizing CTC and CMLM objectives, Mask-CTC trains a model to extract acoustic feature representations that adeptly capture long-term output dependencies by anticipating both past and future contexts. This has been proven effective in ASR tasks, as demonstrated in earlier sections. Additionally, the context-rich representations harnessed by Mask-CTC have been validated in several follow-up studies focusing on spoken language understanding tasks (Li and Doddipatla, 2023; Meeus et al., 2023), which require the comprehension of more abstract linguistic context. This section focuses on streaming tasks, and I propose to pre-train end-to-end streaming ASR models using the Mask-CTC framework, expecting that the context-aware representations are advantageous in reducing latency requirements, particularly by effectively anticipating future contexts.

4.4.1 Streaming End-to-End ASR Models

In this section, I introduce two types of streaming ASR approaches that are central to this study: Streaming Transformer-transducer (Transformer-T) (Chen et al., 2021d) and contextual block streaming AED (CBS-AED) (Tsunoo et al., 2021).

Streaming Transformer-Transducer (Transformer-T)

The transducer-based end-to-end ASR model (as described in Section 2.2.2) operates in a frame-synchronous manner and can easily be adapted for an online streaming model. To this end, various architectures have been developed to enable effective streaming processing in the encoder network (Zhang et al., 2016; Rao et al., 2017; Dong et al., 2019). In this study, I focus on the Transformer-based encoder (Chen et al., 2021d), which has been successfully adopted for streaming ASR models. To enable streaming feature extraction in the Transformer encoder, a chunk-wise attention mask is applied during the computation of self-attention (in Eq. (2.46)). The chunk-wise attention mask segments the input sequence into fixed-width chunks (sub-sequences) with a specific chunk size. Here, frames located within the same chunk can attend to one another. However, if two frames are located in separate chunks, the frame on the left cannot attend to the frame on the right, regardless of the offset. Therefore, the extent of the look-ahead range is controlled by the pre-defined chunk size. Refer to the original work of Chen et al. (2021d) for detailed information on the design of the chunk-wise attention mask. Hereafter, this architecture is referred to as Transformer-transducer (Transformer-T).

Contextual Block Streaming AED (CBS-AED)

Contextual block streaming AED (CBS-AED) (Tsunoo et al., 2021) introduces streaming properties to the AED-based end-to-end ASR model (as described in Section 2.2.3). The AED model, operating in a label-synchronous fashion, is not inherently suited for streaming applications. To address this, CBS-AED employs a block processing technique, which involves dividing the input sequence into small blocks of frames. In this way, streaming processing can be realized by shifting these blocks sequentially across the entire input sequence.

During the processing of each block, the speech input is first divided into segments containing past, central, and future frames with the numbers of N^{l}, N^{c}, and N^{r}. These segments are then fed together into the encoder network, where the central frames are utilized for output generation, while the past and future frames provide local contexts. To accommodate the consideration of longer-term contexts from past blocks, the context inheritance mechanism (Tsunoo et al., 2019) is employed by introducing an additional contextual embedding vector to the encoder input. The encoder computes hidden states for this vector, which are subsequently inherited to the computation of the next block. Consequently, this facilitates the inheritance of a global context that contributes to enhancing the feature extraction process for streaming inputs. To enable streaming decoding using the decoder network of AED, the block boundary detection (BBD) algorithm (Tsunoo et al., 2021) is implemented. The role of the BBD algorithm is to determine the length of a hypothesis that the current block can support. If a hypothesis exceeds the length that the encoder output can accommodate, it is likely to become unreliable. In such situations, the decoder network often encounters two predominant errors, including prematurely predicting the end-of-sentence token or

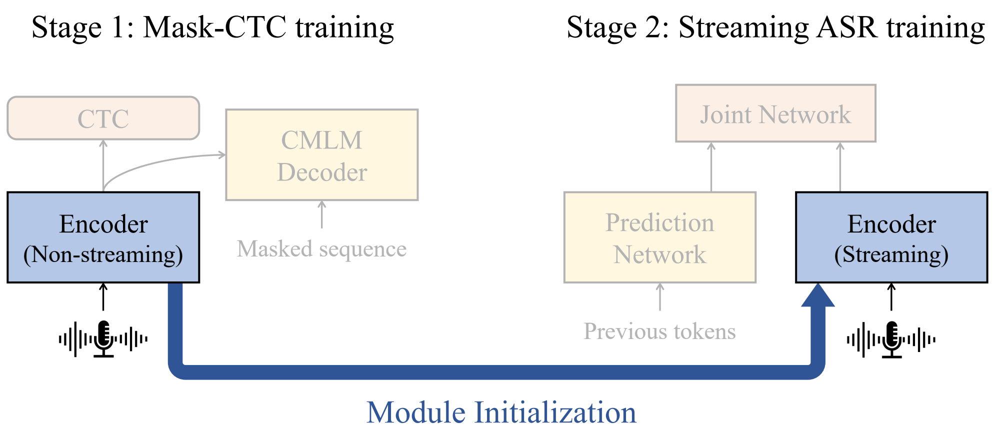


Figure 4-5: Illustration of Mask-CTC-based pre-training for streaming end-to-end ASR model (e.g., Transformer-transducer). In Stage 1, the encoder network is trained within the Mask-CTC framework. In Stage 2, the Transformer-transducer model is initialized using the pre-trained encoder and subsequently fine-tuned to align with a streaming objective.

generating repetitive tokens. The BBD algorithm utilizes such observations as stopping criteria for detecting the block boundary on the fly and performs beam search decoding synchronously with the encoded blocks. For more detailed information on CBS-AED, refer to the original work of Tsunoo et al. (2021).

4.4.2 Mask-CTC-Based Pre-Training Method

A simple and general pre-training method based on Mask-CTC is proposed to achieve high-accuracy and low-latency streaming ASR. Specifically, this study aims to demonstrate the effectiveness of Mask-CTC-based pre-training regardless of model architectures and discusses whether such pre-training can extract features suitable for anticipation as intended, focusing on the alignment of the output tokens.

As different end-to-end streaming ASR models, I focus on Transformer-T and CBS-AED, as introduced in the previous section, covering both the transducer and AED-based model architectures. For both models, the adoption of Transformer has realized high recognition accuracy in their non-streaming baselines. However, when applied to streaming scenarios, the look-ahead ranges of self-attention layers are limited from global to local. This leads to an inevitable performance drop by degrading the Transformer's capability to capture long-range contexts, which limits applications where low latency is a top priority for recognition.

To remedy such an effect, it is necessary to develop a feature representation for the input sequence that not only accounts for long-term contextual dependencies but also anticipates future information effectively, which corresponds to the properties of Mask-CTC as described in Section 4.2. To introduce the desirable properties of the Mask-CTC model into the streaming ASR, I propose a simple two-step training method as follows, which is also described in Figure 4-5:

- **Stage 1 (Mask-CTC training):** The Mask-CTC model is trained to obtain an encoder network that can extract 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 Transformer-T, the encoder network with the chunk-wise attention mask is initialized with the Mask-CTC encoder. For CBS-AED, both the Mask-CTC encoder and CTC networks are used to initialize the corresponding components.

With this two-step training method, I expect to inherit the characteristics of Mask-CTC to a streaming ASR model to capture long-term contextual information and reduce the latency dependency.

4.4.3 Experimental Setting

Speech recognition experiments were conducted to examine the effectiveness of the Mask-CTC-based pre-training method using ESPnet2 (Watanabe et al., 2018; Boyer et al., 2021). I also investigated the essential effect of the proposed pre-training method by studying the output token alignments of the streaming ASR models.

Data

The models were trained and evaluated using the WSJ and TED-LIUM2 (TED2) datasets (see Appendix A for corpus details). For the output tokens, I used SentencePiece (Kudo, 2018) to construct a 80 subword vocabulary for WSJ and a 500 subword vocabulary for TED2, respectively. For robust model training, I applied SpecAugment (Park et al., 2019) to the input data.

Model Settings

For the Transformer-T model, the encoder network was implemented with $N^{\mathsf{enc}} = 12$ Transformer encoder layers and a single LSTM layer for the prediction network. For streaming feature extraction, a chunk-wise attention mask was implemented and applied to the encoder layers. 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.

For WSJ experiments, the CBS-AED model consisted of $N^{\mathsf{enc}} = 6$ Conformer encoder layers and $N^{\mathsf{dec}} = 6$ Transformer decoder layers. The input block settings followed $N^{\mathsf{l}} = 8$, $N^{\mathsf{c}} = 4$, and N^{r} varying from 0 to 6. The latency for CBS-AED was calculated as the product of the maximum look-ahead range in the block (i.e., $N^{\mathsf{c}} + N^{\mathsf{r}} - 1$) and a frame rate of 40ms. For TED2 experiments, the CBS-AED model consisted of $N^{\mathsf{enc}} = 12$ Conformer encoder layers and $N^{\mathsf{dec}} = 6$ Transformer decoder layers. The configuration of the input blocks remained identical

Table 4.10: WER on WSJ task.

Model	**Latency** [ms]	**Initialization**	**WER** [%] (↓) eval92	dev93
Baseline				
Transformer-T	120	Random	19.5	23.3
	160		16.8	20.9
	200		**15.1**	**18.9**
	∞	Random	14.7	17.3
CBS-AED	200	Random	14.4	18.1
	280		13.2	16.2
	360		**12.9**	**16.1**
	1240	Random	11.2	14.2
Enhanced				
Transformer-T	120	Mask-CTC	16.6	20.8
	160		15.0	19.0
	200		**14.8**	**18.5**
CBS-AED	200	Mask-CTC	13.5	17.2
	280		12.9	**16.0**
	360		**12.2**	16.1

to those used in the WSJ task except that N^{r} was fixed at 6. For the pre-trained Mask-CTC model, the encoder network was constructed with the identical setting as the target streaming model. The CMLM decoder was built with $N^{\mathsf{dec}} = 6$ 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 minimal validation loss for Transformer-T and the best validation accuracy for CBS-AED. Beam search decoding was conducted with a beam size of 10 for all the models.

4.4.4 Results

Main Results

For both the Transformer-T and CBS-AED systems, the performance of the following models was compared.

- **Baseline** (Chen et al., 2021d; Tsunoo et al., 2021): Existing streaming ASR models, including Transformer-T and CBS-AED. The parameters for all the network components were randomly initialized.

Table 4.11: WER on TED-LIUM2 task.

Model	**Latency** [ms]	**Initialization**	**WER** [%] ($\downarrow$)
Baseline			
CBS-AED	280	Random	11.3
	1240	Random	9.8
Enhanced			
CBS-AED	280	Mask-CTC	11.1

- **Enhanced**: Streaming ASR models with the proposed Mask-CTC-based pre-training. Network components of the streaming ASR were initialized with pre-trained Mask-CTC modules. For Transformer-T, the encoder module was initialized with the Mask-CTC encoder. For CBS-AED, both encoder and CTC modules were initialized with the corresponding Mask-CTC modules.

The experimental results of Transformer-T and CBS-AED are summarized in Table 4.10 and Table 4.11. Non-streaming Transformer-T and CBS-AED with 1240ms latency were used as lower bounds in these experiments.

The results on WSJ show that for both Transformer-T and CBS-AED, 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. For the WSJ task, 40ms and 80ms latency reductions were reached for Transformer-T and CBS-AED, respectively, while achieving better or equal recognition accuracy than the baseline models. For instance, the enhanced Transformer-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). Such results demonstrated that the proposed method contributed to the construction of streaming ASR models with low latency and high accuracy. For the TED2 task, the enhanced CBS-AED 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. The results for systems with different architectures, such as the transducer and AED, also demonstrated that the Mask-CTC-based pre-training was effective regardless of the model architecture.

Analysis of Output Token Alignments

It has been argued that streaming models tend to shift the token boundaries to the future side to obtain more contextual information (Inaguma and Kawahara, 2021), which results in a delay of the posterior probability spikes for the output tokens compared to non-streaming models. In contrast,

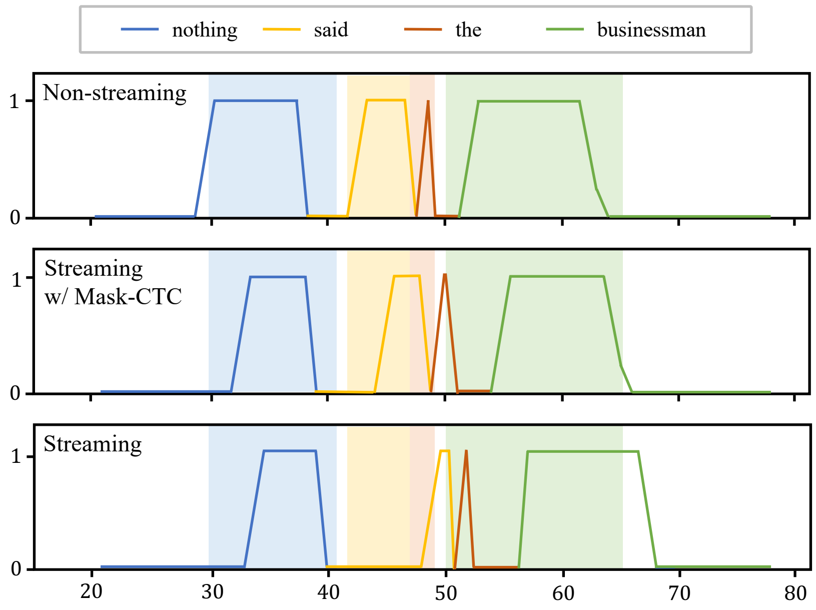


Figure 4-6: Output token alignments of non-streaming and streaming Transformer-T models. The background color in each figure represents the ground-truth alignment. The solid lines in the top, middle, and bottom figures illustrate alignments derived from different models: a non-streaming model, a streaming model pre-trained by Mask-CTC, and a baseline streaming model, respectively. This figure is reproduced from my conference paper (Zhao et al., 2023).

if the encoder network learns the feature representations that anticipate future information, the output tokens can be confirmed earlier and the token boundary shifting issue should be remedied in some instances. To investigate this further, I measured the delay of the spike occurrences in streaming models by comparing them to the alignments obtained from a non-streaming model. The delay is expected to be reduced with the Mask-CTC-based pre-training method.

I conducted measurements on the dev93 validation set of WSJ. Specifically, I evaluated the baseline and enhanced models with 200ms latency settings for Transformer-T and compared their alignments with a non-streaming Transformer-T model. The alignments were obtained from the output of the joint network. For CBS-AED, the latency was also set to 200ms, and I compared the output token boundaries between the baseline and enhanced models. The ASR alignments were obtained from the CTC predictions of CBS-AED in the same manner as Inaguma and Kawahara (2021) and the reference alignments were obtained using the Montreal Forced Aligner (McAuliffe et al., 2017). Figure 4-6 illustrates one example of output token alignments given by Transformer-T. Here, the color in the background represents the reference alignment to the speech input. The non-streaming ASR model (top) managed to predict accurate token alignments. However, the baseline streaming ASR model (bottom) showed a significant delay in the alignments, indicating token boundary shifting due to the lack of contexts. Meanwhile, the enhanced streaming

ASR model (middle), with a Mask-CTC-based pre-trained encoder network, largely improved the alignments of streaming ASR. I calculated the average output delay reduction across the dev93 validation set for both Transformer-T and CBS-AED. For Transformer-T, the spike output delay was reduced by 44ms, and for CBS-AED, 46ms. Such results help the understanding of the knowledge learned from the Mask-CTC-based pre-training method and the reason for the latency reduction capability.

4.5 Summary

This chapter proposed methods for integrating the masked language modeling mechanism into end-to-end ASR models, to enhance their ability to utilize long-range linguistic contexts effectively.

First, I introduced Mask-CTC, a non-autoregressive end-to-end ASR model that was trained using the joint CTC and CMLM objective. During inference, the initial sequence was generated through greedy CTC decoding, and low-confidence tokens were subsequently replaced with the mask token by thresholding their posterior probabilities. This masked output sequence was then fed into the CMLM decoder, where the masked tokens were iteratively predicted by taking into account both the input speech features and the context provided by other unmasked tokens. The experimental results across different ASR tasks demonstrated that Mask-CTC significantly surpasses the performance of the standard CTC-based model, highlighting its effective incorporation of linguistic context into the ASR process. Furthermore, Mask-CTC yielded results competitive with the AED model, while requiring much less inference time. Mask-CTC also showed potential when applied to speech translation tasks.

Subsequently, I further explored the effective use of contextualized linguistic representations in end-to-end ASR. The proposed Mask-Conformer employed the CMLM decoder to enrich speech encoding with linguistic information. Specifically, this involved enhancing the Conformer encoder with the cross-attention mechanism, allowing it to incorporate the outputs of the CMLM decoder into the encoder states. The experimental results on single- and multi-domain ASR tasks demonstrated that Mask-Conformer outperforms strong Conformer-transducer baselines by fully utilizing the decoder information. Moreover, comprehensive analyses validated the effectiveness of the proposed architectural design.

Lastly, I proposed to adopt Mask-CTC for the pre-training of streaming end-to-end ASR models. In Mask-CTC, the CMLM decoder served to enhance the encoder network by supplying contextual linguistic information, thereby improving the performance of CTC. Such capability to integrate long-term linguistic context, including information from future contexts, was expected to aid the training of streaming ASR, enabling it to effectively anticipate and incorporate future information. By initializing an encoder network of a streaming end-to-end ASR model with a pre-trained Mask-CTC encoder, the inherent capabilities of Mask-CTC were implicitly leveraged to optimize the streaming objective. The experimental results showed the effectiveness of the proposed pre-training method on various streaming models, including Transformer-T and CBS-AED. Additionally, the analysis of output spike timings in the streaming models revealed that the pre-training led to more precise estimations of input-output alignments, which contributed to reduced latency.

5 End-to-End Speech Recognition Guided by Pre-Trained Masked Language Model

5.1 Introduction

In the field of NLP, pre-training of language models (Devlin et al., 2019; Radford et al., 2018) have ascended to a predominant paradigm. This approach involves training language models on a large quantity of text-only data using well-designed self-supervised objectives (Devlin et al., 2019; Brown et al., 2020), which leads to the acquisition of versatile linguistic knowledge. Such pre-trained models provide sophisticated representations that enhance the performance of a diverse range of downstream NLP tasks (Wang et al., 2018; Gao et al., 2021), while also mitigating the need for extensive supervised training data. In light of their remarkable success in NLP, pre-trained language models have been actively adopted for various end-to-end speech processing tasks (Shin et al., 2019; Huang et al., 2021; Chuang et al., 2020; Chung et al., 2021b; Hayashi et al., 2019; Kenter et al., 2020; Bang et al., 2022). Particularly in end-to-end ASR, the integration of linguistic knowledge, including semantic and morphosyntax information (Tenney et al., 2019), is crucial for generating accurate textual output, holding great promise in enhancing the model's performance.

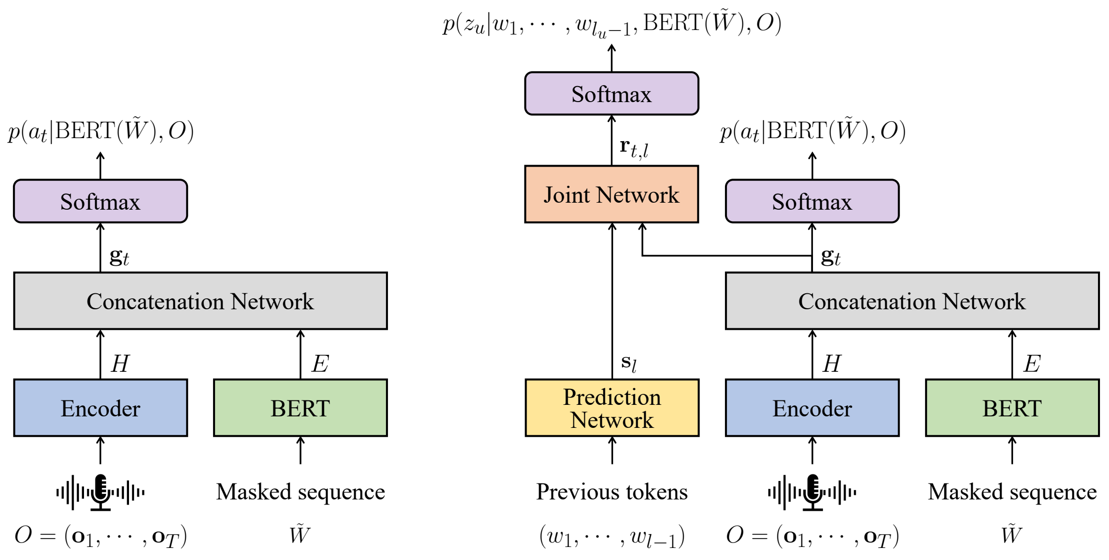
Several attempts have been made to indirectly employ pre-trained language models to improve end-to-end ASR models, such as through knowledge distillation (Futami et al., 2020; Bai et al., 2021; Kubo et al., 2022; Lu and Chen, 2022) and N-best hypothesis rescoring (Shin et al., 2019; Salazar et al., 2020; Chiu and Chen, 2021; Futami et al., 2021; Udagawa et al., 2022). Although these approaches are straightforward and do not interfere with the original end-to-end ASR structures, they can only benefit from the powerful linguistic knowledge either during training or inference. More recently, there have been efforts to integrate pre-trained language models directly into end-to-end ASR models, accomplished by fine-tuning the language models in conjunction with a speech processing network (Huang et al., 2021; Yi et al., 2021; Zheng et al., 2021; Deng et al., 2021; Yu et al., 2022). This enables explicit adaptation of language models to ASR, while allowing models to exploit the linguistic knowledge during both training and inference. However, these approaches require summarizing the speech input into a sequence of appropriate output length before it can be fed into the language models. Moreover, to effectively optimize the unified model, the fine-tuning process entails precise calibration and scheduling of hyper-parameters.

In this chapter, I provide an alternative view of incorporating pre-trained language models into end-to-end ASR, with a particular emphasis on masked language models such as BERT (Devlin et al., 2019). To achieve this, I propose two models, **BERT-CTC** and **BECTRA**, specifically designed to overcome the challenges associated with the integration. BERT-CTC facilitates the combination of audio and linguistic features, based on the formulation of CTC (Graves et al., 2006). More precisely, BERT embeddings are used to explicitly condition CTC on context-aware linguistic information, thereby mitigating the conditional independence in outputs (refer to Eq. (2.15)). BERT-CTC exploits the capabilities of BERT without requiring fine-tuning, while enabling end-to-end training and inference using BERT knowledge and retaining the advantages of the efficient CTC framework. BERT-CTC-Transducer (BECTRA) is an extension of BERT-CTC developed to address the discrepancy between text formats and styles employed in end-to-end ASR and BERT. BECTRA expands BERT-CTC to the transducer-based model (Graves, 2012) and trains the decoder (i.e., prediction and joint networks) using a vocabulary tailored to the target ASR task. This distinct decoder allows for more accurate text generation by alleviating a crucial limitation in BERT-CTC, wherein the model training is constrained on a word-level and domain-mismatched vocabulary used in BERT.

While Chapter 4 also focused on masked language modeling to obtain contextual linguistic information, its effectiveness may be limited due to the small training text data, which was derived solely from the transcriptions of an ASR corpus. This chapter delves into the utilization of contextualized representations extracted by pre-trained masked language models, which are trained on a vast amount of text data, and examines the potential of using versatile linguistic knowledge for enhancing end-to-end ASR models.

5.2 Proposed Integration of Pre-Trained Masked Language Models into End-to-End ASR

I propose a novel approach to end-to-end ASR that utilizes a pre-trained masked language model in its formulation, focusing on BERT (Devlin et al., 2019) as a case in point. The proposed approach



(a) BERT-CTC (b) BERT-CTC-Transducer (BECTRA)

Figure 5-1: Proposed end-to-end ASR models integrating pre-trained masked language model (i.e., BERT). The models can be described in relation to the CTC and transducer-based models illustrated in Figure 2-3. BERT-CTC conditions CTC on contextualized linguistic representations that are obtained from BERT. BECTRA is an extension of BERT-CTC that incorporates prediction and joint networks, benefiting from both the transducer framework and the use of BERT.

is designed to address the following key points:

- How to make end-to-end ASR conditioned on BERT information?
- How to bridge the gap between text processed in end-to-end ASR and BERT?

The former is solved via **BERT-CTC**, which adapts BERT representations to explicitly condition CTC on linguistic contexts (Section 5.2.1). The latter is tackled by **BE**RT-**C**TC-**Tra**nsducer (**BECTRA**) by extending BERT-CTC to the transducer framework. While the BERT-CTC formulation is restricted to the vocabulary used in BERT, BECTRA overcomes this limitation by enabling the handling of different text formats and styles (Section 5.2.2).

To provide a precise explanation of the proposed formulation, I define output sequences that are tokenized using two varying vocabularies: $W^{\mathsf{a}} = (w_l^{\mathsf{a}} \in \mathcal{V}^{\mathsf{a}} | l = 1, \cdots, L^{\mathsf{a}})$ and $W^{\mathsf{b}} = (w_l^{\mathsf{b}} \in \mathcal{V}^{\mathsf{b}} | l = 1, \cdots, L^{\mathsf{b}})$, where $\mathcal{V}^{\mathsf{a}}$ is a vocabulary constructed from ASR training text, and $\mathcal{V}^{\mathsf{b}}$ is a vocabulary of BERT, with the superscripts a and b indicating ASR and BERT, respectively. Typically, $\mathcal{V}^{\mathsf{b}}$ consists of almost word-level tokens with a large subword vocabulary size, while $\mathcal{V}^{\mathsf{a}}$ comprises smaller subword units (i.e., $|\mathcal{V}^{\mathsf{b}}| \gg |\mathcal{V}^{\mathsf{a}}|$). Moreover, $\mathcal{V}^{\mathsf{b}}$ may contain written symbols, including punctuation and casing, whereas they are often disregarded in $\mathcal{V}^{\mathsf{a}}$.

5.2.1 BERT-CTC

Overview: Figure 5-1(a) illustrates the overall model architecture of BERT-CTC, which can be compared with the CTC and transducer-based models shown in Figure 2-3. BERT-CTC utilizes powerful contextualized representations from BERT to make CTC's training and inference explicitly conditioned on linguistic information (Figure 2-3(a) vs. Figure 5-1(a)). BERT-CTC can be similar to the transducer in that it fuses audio and token representations to estimate the distribution over alignments (Figure 2-3(b) vs. Figure 5-1(a)). However, by employing a concatenation network that attends to the full contexts of the input and output sequences, BERT-CTC permits learning inner and inter-dependencies within and between the sequences, facilitating the integration of information from the different modalities (Fujita et al., 2020).

BERT-CTC formulates end-to-end ASR by introducing a partially masked ($\Leftrightarrow$ partially observed) sequence $\tilde{W}^{\mathsf{b}} = (\tilde{w}^{\mathsf{b}}_l \in \mathcal{V}^{\mathsf{b}} | l = 1, \cdots, L^{\mathsf{b}})$, which is obtained by replacing some tokens in an output sequence W^{b} with a special mask token `[MASK]`. This masked sequence is similar to the one introduced in Chapter 4 (i.e., Eq. (4.1)). Note that `[MASK]` is included in the BERT vocabulary $\mathcal{V}^{\mathsf{b}}$. This masked sequence is obtained by applying masks to a ground-truth sequence during training or a hypothesized sequence during inference.

Taking account of all possible masked sequences, the posterior probability distribution of ASR $p(W^{\mathsf{b}}|O)$ (from Eq. (2.1)) is factorized as

$$p(W^{\mathsf{b}}|O) = \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} p(W^{\mathsf{b}}, \tilde{W}^{\mathsf{b}}|O) \tag{5.1}$$

$$= \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, O) p(\tilde{W}^{\mathsf{b}}|O), \tag{5.2}$$

where $\mathcal{M}(W^{\mathsf{b}})$ covers W^{b} with all possible masking patterns. In Eq. (5.2), $p(\tilde{W}^{\mathsf{b}}|O)$ is interpreted as a distribution of sequences that consist of unmasked (observed) tokens, which are readily recognizable from the speech input alone. The other masked tokens, in contrast, are difficult to determine (e.g., homophones) and require context from observed tokens, which is modeled by $p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, O)$. I provide a more intuitive explanation of this interpretation in the inference section (Section 5.2.1).

Similarly to Eq. (2.13), $p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, O)$ in Eq. (5.2) is further factorized by introducing alignment sequences of CTC as

$$p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, O) = \sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} p(W^{\mathsf{b}}, A^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, O) \tag{5.3}$$

$$\approx \sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} p(A^{\mathsf{b}}|W^{\mathsf{b}}, \cancel{\tilde{W}^{\mathsf{b}}}, O) p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, \cancel{O}), \tag{5.4}$$

where $A^{\mathsf{b}} = (a_t^{\mathsf{b}} \in \mathcal{V}^{\mathsf{b}} \cup \{\texttt{<b>}\} | t = 1, \cdots, T)$ is an alignment sequence corresponding to W^{b} with the BERT vocabulary. Eq. (5.4) makes two conditional independence assumptions. The first is that given W^{b} and O, $\tilde{W}^{\mathsf{b}}$ is not required to determine A^{b}. This is reasonable because W^{b} already contains observed tokens in $\tilde{W}^{\mathsf{b}}$ and is helpful in avoiding the combination of all possible masked sequences and alignments (i.e., the Cartesian product of $\mathcal{M} \times \mathcal{B}^{\mathsf{ctc}-1}$). The second is that given $\tilde{W}^{\mathsf{b}}$, O is not required to determine W^{b} because $p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}})$ can be considered as a strong prior distribution modeled by a pre-trained masked language model (e.g., BERT), which can be achieved without the observation from O. I empirically show that this assumption holds in Section 5.6.4.

The posterior probability distribution of $p(A^{\mathsf{b}}|W^{\mathsf{b}}, O)$ in Eq. (5.4) is factorized using a probabilistic chain rule as

$$p(A^{\mathsf{b}}|W^{\mathsf{b}}, O) \approx \prod_{t=1}^{T} p(a_t^{\mathsf{b}}|\cancel{a_1^{\mathsf{b}}, \cdots, a_{t-1}^{\mathsf{b}}}, W^{\mathsf{b}}, O), \tag{5.5}$$

which makes the conditional independence assumption similar to CTC in Eq. (2.15). However, Eq. (5.5) is conditioned on an output sequence W^{b}, which enables the explicit use of linguistic information to estimate the distribution over alignments. This is somewhat similar to the transducer formulation in Eq. (2.24), but is different in that BERT-CTC allows for attending to the entire output sequence $(w_1^{\mathsf{b}}, \cdots, w_{L^{\mathsf{b}}}^{\mathsf{b}})$.

Substituting Eq. (5.5) into Eq. (5.4), BERT-CTC models the product of $p(a_t^{\mathsf{b}}|W^{\mathsf{b}}, O)$ and $p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}})$ as

$$\text{Eq. (5.4)} \triangleq \sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} \prod_{t=1}^{T} p(a_t^{\mathsf{b}}|\text{BERT}(\tilde{W}^{\mathsf{b}}), O), \tag{5.6}$$

where $\text{BERT}(\cdot)$ returns the final hidden states of BERT or any pre-trained masked language model, representing the distribution of target sequences.[1] By substituting Eq. (5.6) into Eq. (5.4), the posterior distribution modeled by BERT-CTC is defined as

$$p^{\mathsf{bert\text{-}ctc}}(W^{\mathsf{b}}|O) \triangleq \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} \sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} \prod_{t=1}^{T} p(a_t^{\mathsf{b}}|\text{BERT}(\tilde{W}^{\mathsf{b}}), O) p(\tilde{W}^{\mathsf{b}}|O). \tag{5.7}$$

The BERT-CTC model can be realized with a single differentiable model, enabling the whole network to be trained end-to-end while being conditioned on BERT knowledge. As illustrated in Figure 5-1(a), the model consists of an encoder network, pre-trained BERT, and a concatenation network. These networks are used to compute the token emission probability at each time frame

[1]As the proposed formulation assumes $p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}})$ to be a strong prior distribution of a masked language model, BERT is used as a feature extractor for an output sequence *without fine-tuning*, which has been reported to still be effective for several NLP tasks (Peters et al., 2019; Zhu et al., 2020; Stappen et al., 2020). In Section 5.6.4, I provide empirical evidence that fine-tuning is not necessary for the proposed approach.

Algorithm 3 Decoding algorithm of BERT-CTC

```
function DecodeBertctc($H$, $K$)
    1. Initialize a masked sequence $\tilde{W}'^{\mathsf{b}}$ by filling mask tokens at all positions
    for $k = 1$ to $K$ do
        2. Forward BERT($\tilde{W}'^{\mathsf{b}}$) and update $E$ with Eq. (5.9)
        3. Forward ConcatNetwork($(H, E)$) and compute token emission probs. with Eq. (5.11)
        4. Generate a hypothesis $\hat{W}^{\mathsf{b}}$ via best path decoding
        5. Compute $N^{\mathsf{mask}} = \lfloor |\hat{W}^{\mathsf{b}}| \cdot \frac{K-k}{K} \rfloor$
        6. Update $\tilde{W}'^{\mathsf{b}}$ by masking $N^{\mathsf{mask}}$ tokens in $\hat{W}^{\mathsf{b}}$ with the lowest prob. scores from Step 3
    return $\hat{W}^{\mathsf{b}}$, $E$
```

in Eq. (5.6) as follows:

$$H = \text{ConformerEncoder}(O) \in \mathbb{R}^{T' \times D^{\mathsf{model}}}, \tag{5.8}$$

$$E = \text{Linear}_{D^{\mathsf{bert}} \to D^{\mathsf{model}}}(\text{BERT}(\tilde{W}^{\mathsf{b}})) \in \mathbb{R}^{L^{\mathsf{b}} \times D^{\mathsf{model}}}, \tag{5.9}$$

$$(\mathbf{g}_1, \cdots, \mathbf{g}_{T'}, \mathbf{g}_{T'+1}, \cdots, \mathbf{g}_{T'+L^{\mathsf{b}}}) = \text{ConcatNetwork}((H, E)) \in \mathbb{R}^{(T'+L^{\mathsf{b}}) \times D^{\mathsf{model}}}, \tag{5.10}$$

$$p(a_t^{\mathsf{b}} | \text{BERT}(\tilde{W}^{\mathsf{b}}), O) = \text{Softmax}(\text{Linear}_{D^{\mathsf{model}} \to |\mathcal{V}^{\mathsf{b}}|+1}(\mathbf{g}_t)) \in [0, 1]^{|\mathcal{V}^{\mathsf{b}}|+1}. \tag{5.11}$$

Eq. (5.8) represents the Conformer encoder from Eq. (2.61). In Eq. (5.9), $\text{Linear}_{D^{\mathsf{bert}} \to D^{\mathsf{model}}}(\cdot)$ is a linear layer for converting the D^{bert}-dimensional BERT outputs to a sequence of D^{model}-dimensional vectors E. In Eq. (5.10), ConcatNetwork$(\cdot)$ is the concatenation network that processes the concatenated sequence (H, E) using a stack of Transformer encoder blocks (see Section 2.3.2 for details). The self-attention mechanism enables the model to capture dependencies within and between the audio and token sequences, H and E, which I analyze in Section 5.6.3. In Eq. (5.11), $\text{Linear}_{D^{\mathsf{model}} \to |\mathcal{V}^{\mathsf{b}}|+1}(\cdot)$ transforms the contextualized representation $\mathbf{g}_t$ to a logit.

Inference

The most probable token sequence is estimated by solving Eq. (2.1) with Eq. (5.2) as

$$\hat{W}^{\mathsf{b}} = \underset{W^{\mathsf{b}}}{\arg\max} \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} p(W^{\mathsf{b}} | \tilde{W}^{\mathsf{b}}, O) p(\tilde{W}^{\mathsf{b}} | O), \tag{5.12}$$

$$\approx \underset{W^{\mathsf{b}}}{\arg\max}\, p(W^{\mathsf{b}} | \tilde{W}'^{\mathsf{b}}, O), \tag{5.13}$$

$$\text{where } \tilde{W}'^{\mathsf{b}} = \underset{\tilde{W}^{\mathsf{b}}}{\arg\max}\, p(\tilde{W}^{\mathsf{b}} | O). \tag{5.14}$$

Eq. (5.13) is derived by applying the Viterbi approximation to Eq. (5.12) in order to handle the intractable summation over all possible masked sequences.

The inference formulation with Eqs. (5.13) and (5.14) can be viewed as the process of human speech recognition, which involves "top-down" and "bottom-up" processing (McClelland and Elman, 1986; Norris, 1994), similar to the concept introduced in Chapter 1. Determining $\tilde{W}'^{\mathsf{b}}$ in Eq. (5.14) is analogous to bottom-up processing, where the model analyzes the individual low-level sounds that make up words. However, certain words, particularly those with homophones, are difficult to identify solely from their sounds, requiring higher-level linguistic information for accurate recognition. This is solved via top-down processing in Eq. (5.13), where the conditioning from $\tilde{W}'^{\mathsf{b}}$ enables the model to leverage linguistic knowledge, context, and anticipation for identifying words from low-level sounds.

To solve Eqs. (5.13) and (5.14), I design a fill-mask-style decoding algorithm based on mask-predict (Ghazvininejad et al., 2019) and CTC inference, which is also inspired by the inference process of Mask-CTC presented in Chapter 4. Algorithm 3, consisting of Steps `1` to `6`, describes the proposed algorithm. At the beginning of decoding, a masked sequence $\tilde{W}'^{\mathsf{b}}$ is initialized by replacing all token positions with the mask token `[MASK]` (Step `1`).[2] The algorithm then proceeds to generate a hypothesis by gradually filling in the masked tokens over K iterations. At each iteration $k \in \{1, \cdots, K\}$, the current masked sequence $\tilde{W}'^{\mathsf{b}}$ is fed into BERT to obtain contextual embeddings E, as defined by Eq. (5.9) (Step `2`). The encoder output H from Eq. (5.8) and E are concatenated and input into the concatenation network, which computes the framewise probability distribution $p(a_t^{\mathsf{b}}|\text{BERT}(\tilde{W}'^{\mathsf{b}}), O)$ as in Eq. (5.11) (Step `3`). Using the probabilities computed at each frame, a hypothesis $\hat{W}^{\mathsf{b}}$ is generated through best path decoding, in the same manner as described in Section 2.2.1 (Step `4`). The number of tokens that will be masked N^{mask} is determined by a linear decay function as $N^{\mathsf{mask}} = \lfloor |\hat{W}^{\mathsf{b}}| \cdot \frac{K-k}{K} \rfloor$ (Step `5`), e.g., if K is set to five, N^{mask} decreases by 20% at each iteration. The masked sequence $\tilde{W}'^{\mathsf{b}}$ is updated by replacing N^{mask} tokens in the hypothesis $\hat{W}^{\mathsf{b}}$ with the mask token `[MASK]` (Step `6`). Here, tokens are selected for masking according to their confidence scores, which are measured by calculating the output probability of each token. Using the framewise probabilities from Step `3`, the output probability for a token $\hat{w}_l^{\mathsf{b}} \in \hat{W}^{\mathsf{b}}$ is derived similarly to Eq. (4.8) as

$$p(w_l^{\mathsf{b}} = \hat{w}_l^{\mathsf{b}}|\text{BERT}(\tilde{W}'^{\mathsf{b}}), O) \approx \max_{t \in \mathcal{T}_l} p(a_t^{\mathsf{b}} = \hat{w}_l^{\mathsf{b}}|\text{BERT}(\tilde{W}'^{\mathsf{b}}), O), \tag{5.15}$$

where $\mathcal{T}_l$ is a set of frame indices that correspond to the l-th token $\hat{w}_l^{\mathsf{b}}$ after applying the collapsing function. With Eq. (5.15), N^{mask} tokens with the lowest probability scores are masked.

In the first iteration (i.e., $k = 1$), the model generates a hypothesis solely based on the speech input, without any linguistic cues from the output tokens, which are all masked. This can be aligned with the concept of bottom-up processing as formulated by Eq. (5.14). As the iterations proceed (i.e., $1 < k \leq K$), the output tokens become gradually observable, providing additional

[2] The algorithm is a non-autoregressive process, and it requires predicting the target length beforehand (Gu et al., 2018), which is obtained from intermediate predictions from the encoder network (see Section 5.4.4 for details).

linguistic information for generating a more precise hypothesis. This can be interpreted as top-down processing as formulated by Eq. (5.13).

Training

The BERT-CTC model is trained by minimizing the negative log-likelihood of Eq. (5.2), defined as

$$-\log \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, O) p(\tilde{W}^{\mathsf{b}}|O), \tag{5.16}$$

which is further expanded using Eqs. (5.4) and (5.5) as

$$\begin{aligned} \text{Eq. (5.16)} &= -\log \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} \sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} p(A^{\mathsf{b}}|W^{\mathsf{b}}, O) p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}) p(\tilde{W}^{\mathsf{b}}|O) & (5.17) \\ &\approx -\log \mathbb{E}_{\tilde{W}^{\mathsf{b}} \sim \mathcal{M}'(W^{\mathsf{b}})} \left[\sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} p(A^{\mathsf{b}}|W^{\mathsf{b}}, O) p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}) \right]. & (5.18) \end{aligned}$$

To obtain Eq. (5.18), the intractable marginalization over $\tilde{W}^{\mathsf{b}}$ is approximated as expectation with respect to the sampling distribution $\mathcal{M}'(W^{\mathsf{b}})$, which is calculated on the probability distribution of $p(\tilde{W}^{\mathsf{b}}|O)$. The upper bound of Eq. (5.18) can be derived by applying Jensen's inequality as

$$\text{Eq. (5.18)} \leq -\mathbb{E}_{\tilde{W}^{\mathsf{b}} \sim \mathcal{M}'(W^{\mathsf{b}})} \left[\log \sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} p(A^{\mathsf{b}}|W^{\mathsf{b}}, O) p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}) \right]. \tag{5.19}$$

Substituting Eqs. (5.5) and (5.6) into Eq. (5.19), the loss for BERT-CTC training is defined as

$$\mathcal{L}^{\mathsf{bert\text{-}ctc}} \triangleq -\mathbb{E}_{\tilde{W}^{\mathsf{b}} \sim \mathcal{M}'(W^{\mathsf{b}})} \left[\log \sum_{A^{\mathsf{b}} \in \mathcal{B}^{\mathsf{ctc}-1}(W^{\mathsf{b}})} \prod_{t=1}^{T} p(a_t^{\mathsf{b}}|\mathrm{BERT}(\tilde{W}^{\mathsf{b}}), O) \right]. \tag{5.20}$$

In comparison to the CTC objective defined in Eq. (2.20), each token prediction in Eq. (5.20) is explicitly conditioned on the contextualized embeddings from BERT. This allows an explicit consideration of the contextual dependencies among token predictions while retaining the efficient optimization strategy as in CTC.

For the sampling process of $\tilde{W}^{\mathsf{b}}$ in Eq. (5.20), random sampling from a uniform distribution is used to approximate the probability distribution of $\mathcal{M}'(W^{\mathsf{b}})$, for the sake of simplicity. The sampling strategy is similar to the one employed in Ghazvininejad et al. (2019), where a random number N^{mask} is sampled from a uniform distribution ranging between one and the target sequence length L^{b}, i.e., $N^{\mathsf{mask}} \sim \mathrm{Uniform}(1, L^{\mathsf{b}})$. Subsequently, N^{mask} tokens are randomly selected from a ground-truth sequence, which are then replaced with `[MASK]`.

5.2.2 BERT-CTC-Transducer (BECTRA)

BERT-CTC progressively conditions CTC on contextual linguistic information by gradually predicting tokens and updating BERT embeddings correspondingly. This BERT-based refinement requires the model to work with the BERT's text format, which has the vocabulary $\mathcal{V}^{\mathsf{b}}$ that can be too large for ASR training, and could lead to a mismatch against the target ASR domain. BERT-CTC-Transducer (BECTRA) is designed to extend BERT-CTC for handling such mismatches while still utilizing BERT knowledge to enhance ASR performance.

Overview: Figure 5-1(b) illustrates the overall model architecture of BECTRA, which can be compared with the conventional transducer-based models shown in Figure 2-3(b). BECTRA formulates end-to-end ASR based on BERT-CTC, using the output of the concatenation network for calculating the transducer loss (Figure 5-1(a) vs. Figure 5-1(b)). Here, the joint and prediction networks are trained with an ASR-specific vocabulary $\mathcal{V}^{\mathsf{a}}$, which allows for more suitable end-to-end ASR training without being limited by the BERT vocabulary $\mathcal{V}^{\mathsf{b}}$. Hence, unlike targeting W^{b} in Eq. (5.1), BECTRA utilizes W^{a} tokenized by $\mathcal{V}^{\mathsf{a}}$ as its target sequence.

Similarly to Eq. (5.2), BECTRA formulates end-to-end ASR by marginalizing the posterior distribution of $p(W^{\mathsf{a}}|O)$ over all possible masked sequences as

$$p(W^{\mathsf{a}}|O) = \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} p(W^{\mathsf{a}}|\tilde{W}^{\mathsf{b}}, O)p(\tilde{W}^{\mathsf{b}}|O). \tag{5.21}$$

In Eq. (5.21), $\tilde{W}^{\mathsf{b}}$ is obtained by masking an output sequence with the BERT unit W^{b} ($\neq W^{\mathsf{a}}$). Similarly to Eq. (2.22), the posterior distribution of $p(W^{\mathsf{a}}|\tilde{W}^{\mathsf{b}}, O)$ in Eq. (5.21) is further factorized by considering all possible alignment sequences of the transducer as

$$p(W^{\mathsf{a}}|\tilde{W}^{\mathsf{b}}, O) = \sum_{Z^{\mathsf{a}} \in \mathcal{B}^{\mathsf{tra}\,-1}(W^{\mathsf{a}})} p(W^{\mathsf{a}}, Z^{\mathsf{a}}|\tilde{W}^{\mathsf{b}}, O) \tag{5.22}$$

$$\approx \sum_{Z^{\mathsf{a}} \in \mathcal{B}^{\mathsf{tra}\,-1}(W^{\mathsf{a}})} p(Z^{\mathsf{a}}|W^{\mathsf{a}}, \cancel{\tilde{W}^{\mathsf{b}}}, O)p(W^{\mathsf{a}}|\tilde{W}^{\mathsf{b}}, \cancel{O}), \tag{5.23}$$

where $Z^{\mathsf{a}} = (z^{\mathsf{a}}_u \in \mathcal{V}^{\mathsf{a}} \cup \{\text{<b>}\}|u = 1, \cdots, T + L^{\mathsf{a}})$ is an alignment sequence corresponding to W^{a} with the ASR vocabulary $\mathcal{V}^{\mathsf{a}}$, as defined by the transducer (refer to Eq. (2.22)). Eq. (5.23) makes the same approximations employed in Eq. (5.4). The posterior probability $p(Z^{\mathsf{a}}|W^{\mathsf{a}}, O)$ in Eq. (5.23) is further factorized by a probabilistic chain rule without a conditional independence assumption (cf. Eq. (5.5)) as

$$p(Z^{\mathsf{a}}|W^{\mathsf{a}}, O) = \sum_{Z^{\mathsf{a}} \in \mathcal{B}^{\mathsf{tra}-1}(W^{\mathsf{a}})} \prod_{u=1}^{T+L^{\mathsf{a}}} p(z^{\mathsf{a}}_u | z^{\mathsf{a}}_1, \cdots, z^{\mathsf{a}}_{u-1}, W^{\mathsf{a}}, O), \tag{5.24}$$

$$\approx \sum_{Z^{\mathsf{a}} \in \mathcal{B}^{\mathsf{tra}-1}(W^{\mathsf{a}})} \prod_{u=1}^{T+L^{\mathsf{a}}} p(z^{\mathsf{a}}_u | \underbrace{w^{\mathsf{a}}_0, \cdots, w^{\mathsf{a}}_{l_u-1}}_{=\mathcal{B}^{\mathsf{tra}}(z^{\mathsf{a}}_1, \cdots, z^{\mathsf{a}}_{u-1})}, W^{\mathsf{a}}, O), \tag{5.25}$$

where $w^{\mathsf{a}}_0 = \texttt{<sos>}$, and l_u is the number of non-blank tokens predicted up to an alignment index of u. Eq. (5.25) assumes $(z^{\mathsf{a}}_1, \cdots, z^{\mathsf{a}}_{u-1}) \approx (w^{\mathsf{a}}_0, \cdots, w^{\mathsf{a}}_{l_u-1})$, using the same approximation as the transducer in Eq. (2.24). Similarly to the BERT-CTC formulation in Eq. (5.6), BECTRA models Eq. (5.23) using Eq. (5.25) as

$$\text{Eq. (5.23)} \triangleq \sum_{Z^{\mathsf{a}} \in \mathcal{B}^{\mathsf{tra}-1}(W^{\mathsf{a}})} \prod_{u=1}^{T+L^{\mathsf{a}}} p(z^{\mathsf{a}}_u | w^{\mathsf{a}}_0, \cdots, w^{\mathsf{a}}_{l_u-1}, \text{BERT}(\tilde{W}^{\mathsf{b}}), O), \tag{5.26}$$

where BERT is employed to model $p(W^{\mathsf{a}}|\tilde{W}^{\mathsf{b}})$. This is a reasonable approximation because both W^{b} and W^{a} represent the same target sentence. Thus, W^{a} can be derived easily by first converting W^{b} into a word sequence and then tokenizing it using $\mathcal{V}^{\mathsf{a}}$. By substituting Eq. (5.26) into Eq. (5.21), the posterior distribution modeled by BECTRA is defined as

$$p^{\mathsf{bectra}}(W^{\mathsf{a}}|O) \triangleq \sum_{\tilde{W}^{\mathsf{b}} \in \mathcal{M}(W^{\mathsf{b}})} \sum_{Z^{\mathsf{a}} \in \mathcal{B}^{\mathsf{tra}-1}(W^{\mathsf{a}})} \prod_{u=1}^{T+L^{\mathsf{a}}} p(z^{\mathsf{a}}_u | W^{\mathsf{a}}_{<l_u}, \text{BERT}(\tilde{W}^{\mathsf{b}}), O) p(\tilde{W}^{\mathsf{b}}|O), \tag{5.27}$$

where $W^{\mathsf{a}}_{<l_u} = (w^{\mathsf{a}}_0, \cdots, w^{\mathsf{a}}_{l_u-1})$.

As shown in Figure 5-1(a), the BECTRA model is based on the architecture of BERT-CTC, with the additional prediction and joint networks from the transducer model (see Figure 2-3(b)). The token emission probability in Eq. (5.26) is computed as

$$\mathbf{s}_{l_u} = \text{Prediction}(w^{\mathsf{a}}_0, \cdots, w^{\mathsf{a}}_{l_u-1}) \in \mathbb{R}^{D^{\mathsf{pred}}}, \tag{5.28}$$

$$\mathbf{r}_{t,l_u} = \text{Joint}(\mathbf{g}_t, \mathbf{s}_{l_u}) \in \mathbb{R}^{D^{\mathsf{joint}}}, \tag{5.29}$$

$$p(z^{\mathsf{a}}_u | W^{\mathsf{a}}_{<l_u}, \text{BERT}(\tilde{W}^{\mathsf{b}}), O) = \text{Softmax}\left(\text{Linear}_{D^{\mathsf{joint}} \to |\mathcal{V}^{\mathsf{a}}|+1}(\mathbf{r}_{t,l_u})\right) \in [0,1]^{|\mathcal{V}^{\mathsf{a}}|+1}. \tag{5.30}$$

In Eq. (5.29), $\mathbf{g}_t$ is obtained from the output of the concatenation network in Eq. (5.10). In Eq. (5.30), $\text{Linear}_{D^{\mathsf{joint}} \to |\mathcal{V}^{\mathsf{a}}|+1}(\cdot)$ transforms the output of the joint network $\mathbf{r}_{t,l_u}$ to a logit. This network architecture is almost identical to the transducer presented in Eqs. (2.26) to (2.29), but it differs in that the joint network takes the output of the concatenation network as its input. This enables the integration of BERT knowledge into the transducer-based model. By adopting the

Algorithm 4 Decoding algorithm of BECTRA

```
function DecodeBectra($H$, $K$, $B$)
    1. Perform DecodeBertctc($H$, $K$) and obtain the final BERT output $E$
    2. Generate a hypothesis $\hat{W}^{\mathsf{a}}$ via beam search decoding with a beam size of $B$,
           using output probabilities computed by Eq. (5.30)
    return $\hat{W}^{\mathsf{a}}$
```

prediction network, BECTRA can explicitly capture the causal dependencies between output tokens, leading to better sequence modeling. This is another key advantage of BECTRA compared to BERT-CTC, beyond the use of ASR-specific vocabulary, which is discussed in Section 5.6.5.

Inference

Algorithm 4 shows the inference algorithm of BECTRA, which includes Steps 1 and 2. The algorithm is implemented with BERT-CTC decoding (see Section 5.2.1) followed by beam search decoding of the transducer (see Section 2.2.2). BERT-CTC decoding provides the model with a fully contextualized BERT output E, which is obtained from the final hypothesis estimated by the iterative refinement (Step 1). To find the optimal sequence with the highest sequence-level generation probability, beam search decoding is performed using the token emission probabilities computed from Eq. (5.30) (Step 2). With this combined inference algorithm, BECTRA can leverage the BERT's ability to capture the bi-directional context in an output sequence, providing the benefit of non-autoregressive decoding. Furthermore, transducer-based decoding enables the model to refine a sequence in an autoregressive manner, utilizing a more appropriate output unit for performing ASR.

Training

The transducer loss of BECTRA is defined by substituting Eq. (5.26) into Eq. (5.21) and following the same derivation process as Eq. (5.20), resulting in

$$\mathcal{L}'^{\mathsf{bectra}} \triangleq -\mathbb{E}_{\tilde{W}^{\mathsf{b}} \sim \mathcal{M}(W^{\mathsf{b}})}\left[\log \sum_{Z^{\mathsf{a}} \in \mathcal{B}^{\mathsf{tra}\,-1}(W^{\mathsf{a}})} \prod_{u=1}^{T+L^{\mathsf{a}}} p(z^{\mathsf{a}}_u | w^{\mathsf{a}}_0, \cdots, w^{\mathsf{a}}_{l_u-1}, \mathrm{BERT}(\tilde{W}^{\mathsf{b}}), O)\right], \tag{5.31}$$

where the summation over Z^{a} can be efficiently computed using the same approach as in transducer training (see Section 2.2.2). The sampling strategy for $\tilde{W}^{\mathsf{b}}$ is described in BERT-CTC training (see Section 5.2.1). The objective function of BECTRA is defined by combining $\mathcal{L}^{\mathsf{bert\text{-}ctc}}$ from Eq. (5.20) and $\mathcal{L}'^{\mathsf{bectra}}$ from Eq. (5.31) as

$$\mathcal{L}^{\mathsf{bectra}} = (1 - \lambda^{\mathsf{bectra}})\mathcal{L}^{\mathsf{bert\text{-}ctc}} + \lambda^{\mathsf{bectra}}\mathcal{L}'^{\mathsf{bectra}}, \tag{5.32}$$

Table 5.1: Comparisons between end-to-end ASR formulations for modeling distribution over alignments.

Model	**Formulation**
CTC	$\sum_{A \in \mathcal{B}^{\mathsf{ctc}^{-1}}(W)} \prod_{t=1}^{T} p(a_t \| O)$
Transducer	$\sum_{Z \in \mathcal{B}^{\mathsf{tra}^{-1}}(W)} \prod_{u=1}^{T+L} p(z_u \| w_0, \cdots, w_{l_u - 1}, O)$
BERT-CTC	$\sum_{A \in \mathcal{B}^{\mathsf{ctc}^{-1}}(W)} \prod_{t=1}^{T} p(a_t \| \text{BERT}(\tilde{W}), O)$
BECTRA	$\sum_{Z \in \mathcal{B}^{\mathsf{tra}^{-1}}(W)} \prod_{u=1}^{T+L} p(z_u \| w_0, \cdots, w_{l_u - 1}, \text{BERT}(\tilde{W}), O)$

where $\lambda^{\mathsf{bectra}}$ $(0 \leq \lambda^{\mathsf{bectra}} \leq 1)$ is a tunable parameter.

5.2.3 Overall Comparison of End-to-End ASR Formulations

As summarized in Table 5.1, the key difference between the end-to-end ASR formulations discussed thus far lies in how the distribution over alignments is computed. CTC estimates the distribution based solely on the speech input O, assuming that the output tokens are independent of one another. The transducer conditions each token prediction explicitly on the preceding non-blank tokens $(w_0, \cdots, w_{l_u - 1})$, introducing the prediction and joint networks. BERT-CTC achieves similar conditioning using BERT's contextualized embeddings BERT$(\tilde{W})$ through the concatenation network. BECTRA is conditioned on both $(w_0, \cdots, w_{l_u - 1})$ and BERT$(\tilde{W})$, enabling a model to benefit from the information provided by both sources.

5.3 Relationship to Previous Work

In this section, I further clarify the position of this work in relation to other relevant topics in end-to-end ASR, focusing on masked language model modeling and external language model integration.

5.3.1 End-to-End ASR and Masked Language Modeling

CMLM (Ghazvininejad et al., 2019), one of the successful approaches in non-autoregressive neural machine translation, has been introduced to solve end-to-end ASR, including the Mask-CTC approach presented in Chapter 4. CMLM utilizes an encoder-decoder structure, wherein its decoder is trained with the masked language model objective (Devlin et al., 2019) while being conditioned on the encoder outputs through the cross-attention mechanism. Audio-CMLM (Chen et al., 2021b) employs CMLM to enable non-autoregressive end-to-end ASR by conditioning the decoder on audio information to learn the fill-mask process. By combining CMLM-based modeling with CTC, Imputer (Chan et al., 2020) and Mask-CTC extend the mask-predict algorithm to refine either a frame-level or token-level sequence predicted in the CTC framework. Several studies have trained CMLM as an error-correction model for predictions generated by an end-to-end ASR system (Futami et al., 2022; Fan et al., 2022).

The proposed approach of incorporating masked language modeling with the CTC and transducer frameworks is relevant to the above studies. However, it differs in that this work aims to leverage the pre-existing knowledge acquired by pre-trained masked language models to enhance end-to-end ASR performance.

5.3.2 Language Model Integration for End-to-End ASR

The proposed approach also shares similarities with the language model integration approaches, which are outlined in Section 1.2.2. Particularly, it is similar to the cold fusion method by combining an end-to-end ASR model and a pre-trained masked language model using the self-attention mechanism to selectively merge audio and linguistic representations. However, this work focuses on exploring how the versatile linguistic knowledge acquired from pre-trained language models (i.e., BERT) can be utilized to improve end-to-end ASR. Additionally, I demonstrate that the conventional fusion technique is complementary to the proposed approach, allowing for the incorporation of a domain-specific RNN-LM to further enhance performance.

5.4 Experimental Setting

I used the ESPnet toolkit (Watanabe et al., 2018, 2021) for conducting the experiments. All the codes and recipes used in the experiments are made publicly available.[3]

5.4.1 Data

The experiments were carried out using various corpora, as summarized in Table 5.2, which comprised different quantities of data, languages, and speech and text styles. See Appendix A for more detailed information on each corpus. For LS, the 100-hour subset (LS-100) was also used in addition to the full set (LS-960), which was mainly employed for conducting additional investigations and analyses. For LL, I used the 10-hour training set (LL-10) for evaluating the proposed models

[3] https://github.com/YosukeHiguchi/espnet/tree/bectra

Table 5.2: Dataset descriptions

Dataset	Hours	Language	Speech Style	Text Style
LibriSpeech (LS)	100	English	Read	Normalized
	960	English	Read	Normalized
Libri-Light (LL-10)	10	English	Read	Normalized
TED-LIUM2 (TED2)	207	English	Spontaneous	Normalized
AISHELL-1 (AS1)	150	Mandarin	Read	Normalized
CoVoST2 (CV2)	407	English	Read	Punct./Casing

in a low-resource setting. Notice that all corpora except CV2 only provide normalized transcriptions, in which punctuation is removed and casing is standardized to upper or lower case. This potentially limits the capabilities of BERT, which is often trained on written-form text with punctuation and casing preserved. In contrast, CV2 provides unnormalized transcriptions with a decent amount of ASR training data, which makes it an ideal resource for evaluating the effectiveness of the proposed approach.

I used SentencePiece (Kudo, 2018) to construct subword vocabularies from ASR transcriptions in order to obtain the ASR vocabulary $\mathcal{V}^{\mathsf{a}}$. The vocabulary sizes were set to 300, 5k, 100, 500, and 500 for LS-100, LS-960, LL-10, TED2, and CV2, respectively. For AS1, I used character-level tokenization with 4231 Chinese characters. It should be noted that before extracting subwords or characters, the ASR transcriptions were normalized regardless of the corpus.

5.4.2 Model and Network Architecture

I evaluated end-to-end ASR models illustrated in Figures 2-3 and 5-1. **CTC** and **Transducer** are the baseline models trained based on $\mathcal{L}^{\mathsf{ctc}}$ and $\mathcal{L}^{\mathsf{tra}}$, as defined by Eqs. (2.20) and (2.30), respectively. **BERT-CTC** and **BECTRA** are the proposed models trained based on $\mathcal{L}^{\mathsf{bert\text{-}ctc}}$ and $\mathcal{L}^{\mathsf{bectra}}$, as defined by Eqs. (5.20) and (5.32), respectively.

The Conformer encoder in Eq. (5.8) consisted of two Conv2D layers followed by a stack of $N^{\mathsf{enc}} = 12$ encoder blocks. The Conv2D layers had 256 channels, a kernel size of 3×3, and a stride size of 2, which resulted in down-sampling the input length by a factor of 4 (i.e., $T' = T/4$). For the self-attention module in Eq. (2.63), the number of heads N^{head}, dimension of a self-attention layer D^{model}, and dimension of a feed-forward network D^{ff}, were set to 4, 256, and 1024, respectively. For the convolution module in Eq. (2.64), I used a kernel size of 31.

For the transducer-based models, including Transducer and BECTRA, the prediction network was an LSTM layer with $D^{\mathsf{pred}} = 256$ hidden units. The joint network consisted of a single linear layer with $D^{\mathsf{joint}} = 256$ hidden units, followed by a hyperbolic tangent activation function.

Regarding the BERT-based models, including BERT-CTC and BECTRA, the concatenation network was the Transformer encoder (Vaswani et al., 2017) with $N^{\mathsf{enc}} = 6$ blocks, where N^{head}, D^{model}, and D^{ff} were configured to 4, 256, and 2048, respectively. I applied two Conv2D layers to the encoder output before passing it through the concatenation network, using the same configuration as that of the down-sampling layer in the Conformer encoder. Unless stated otherwise, I used a BERT$_{\text{BASE}}$ (uncased) model trained for each language, which were downloaded from the HuggingFace Transformers library[4] (Wolf et al., 2020): English[5] (with $|\mathcal{V}^{\mathsf{b}}| = 30522$) and Mandarin[6] (with $|\mathcal{V}^{\mathsf{b}}| = 21128$).

5.4.3 Training Configuration

For each dataset, I mostly adhered to the configurations provided by the ESPnet2 recipe (as specified in Appendix A). The models were trained for 100 epochs on LS-100, LL-10, and AS1; 70 epochs on TED2 and LS-960; and 50 epochs on CV2. I used the Adam optimizer (Kingma and Ba, 2015) for weight updates with the beta coefficients (β_1, β_2), epsilon parameter ϵ, and weight decay rate of $(0.9, 0.999)$, 10^{-8}, and 10^{-6}, respectively. I used Noam learning rate scheduling (Vaswani et al., 2017), where the number of warmup steps was set to 15k, and a peak learning rate was tuned from $\{1.0, 2.0\} \times 10^{-3}$. The batch size was set to 256, except for LL-10, which was set to 32. Speech data was augmented using speed perturbation (Ko et al., 2015) with a factor of 3 and SpecAugment (Park et al., 2019, 2020a). For the hyper-parameters in SpecAugment, the number of frequency and time masks were set to 2 and 5, and the size of frequency and time masks were set to 27 and $0.05T$. For BECTRA, $\lambda^{\mathsf{bectra}}$ in Eq. (5.32) was set to 0.5.

For all of the models, the intermediate CTC regularization technique (Tjandra et al., 2020; Lee and Watanabe, 2021) was applied to the Conformer encoder, which has been demonstrated to enhance ASR performance, as also introduced in Section 3.2.1. An additional CTC loss was applied to the output of the 6-th encoder block $H^{(i=6)}$, which was calculated using a target sequence W^{a} tokenized by the ASR vocabulary $\mathcal{V}^{\mathsf{a}}$. This is especially effective for training end-to-end ASR models with the large BERT vocabulary $\mathcal{V}^{\mathsf{b}}$, facilitating the prediction of sparse word-level tokens in a hierarchical multi-tasking manner (Fernández et al., 2007; Sanabria and Metze, 2018; Krishna et al., 2018), as also demonstrated in Chapter 3.

5.4.4 Decoding Configuration

A final model was obtained for evaluation by averaging model parameters over 10 checkpoints with the best validation performance. For the number of iterations in BERT-CTC decoding (in Algorithm 3), K was to 20 for BERT-CTC and 10 for BECTRA. The beam search decoding was

[4]https://github.com/huggingface/transformers
[5]https://huggingface.co/bert-base-uncased
[6]https://huggingface.co/bert-base-chinese

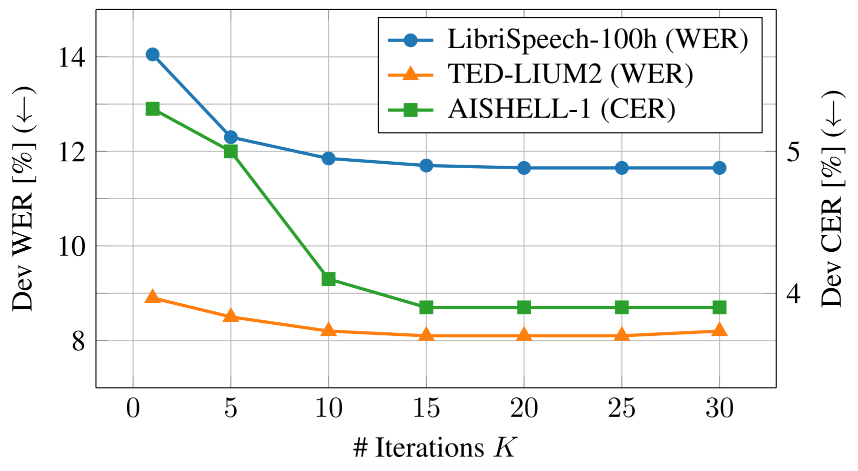


Figure 5-2: WER or CER of BERT-CTC evaluated on development sets, using varying numbers of decoding iterations K. When $K = 1$, the model relies solely on audio information to predict output tokens. When $K > 1$, the model incorporates linguistic information from BERT to refine its outputs.

performed with a beam size of 10 for Transducer and 5 for BECTRA.

During the initialization process of a masked sequence in BERT-CTC decoding (Algorithm 3 Step 1), the output length was determined based on intermediate predictions. More specifically, the best path decoding was performed using the intermediate encoder states, where the auxiliary CTC loss was applied (i.e., $H^{(i=6)}$). This allowed the models to generate a sequence tokenized based on the ASR vocabulary $\mathcal{V}^{\mathsf{a}}$, which was then retokenized using the BERT vocabulary $\mathcal{V}^{\mathsf{b}}$ to estimate the initial length.

5.5 Results

5.5.1 Effectiveness of BERT-CTC

I first investigated the effectiveness of BERT-CTC, which is designed to enhance ASR performance by integrating audio information with linguistic knowledge from BERT. Figure 5-2 depicts the relationship between the number of decoding iterations (K in Algorithm 3) and the BERT-CTC results, as evaluated by the WER for LS-100 and TED2, and the CER for AS-1. Note that WERs for LS-100 were obtained by averaging scores on the *dev-clean* and *dev-other* sets. During decoding with $K = 1$, the model relied on the speech input only, and the model did not have access to any linguistic cues, as all output tokens were masked (i.e., bottom-up processing modeled by Eq. (5.14)). By increasing the number of iterations with $K > 1$, the model successfully utilized the knowledge from BERT to refine the output tokens (i.e., top-down processing modeled by Eq. (5.13)), leading to more effective performance across all tasks.

Table 5.3: WER of CTC and Transducer baselines compared to proposed BERT-CTC, evaluated on LibriSpeech-100h task. CTC and Transducer were trained using either the ASR vocabulary $\mathcal{V}^{\mathsf{a}}$ or the BERT vocabulary $\mathcal{V}^{\mathsf{b}}$.

	Output Vocab.		**WER** [%] (↓)			
			Dev		Test	
Model	$\mathcal{V}^{\mathsf{a}}$	$\mathcal{V}^{\mathsf{b}}$	clean	other	clean	other
CTC	✓		6.9	20.1	7.0	20.2
		✓	11.2	21.4	11.4	22.0
Transducer	✓		5.9	17.7	6.0	17.6
		✓	9.7	21.5	9.8	22.3
BERT-CTC		✓	7.0	16.4	7.1	16.5

5.5.2 Difficulty of Training ASR with BERT Vocabulary

Table 5.3 compares WERs obtained by training CTC and Transducer models on LS-100, using either the ASR or BERT vocabulary ($\mathcal{V}^{\mathsf{a}}$ vs. $\mathcal{V}^{\mathsf{b}}$). It is apparent that CTC and Transducer produced notably higher WERs when trained with the BERT vocabulary, indicating that employing word-level BERT units was not an optimal choice for ASR training (Soltau et al., 2017). Moreover, I also mention that the BERT vocabulary is not precisely aligned with the intended domain of the LibriSpeech task (e.g., Wikipedia vs. audiobook). As a result, there was a potential domain mismatch between the ASR training text and BERT vocabulary.

In contrast, despite using the same BERT vocabulary, the proposed BERT-CTC model significantly outperformed both the conventional CTC and Transducer models. The gain from CTC demonstrates the effectiveness of leveraging BERT's contextualized linguistic embeddings to explicitly consider output dependencies (refer to Table 5.1). BERT-CTC improved over Transducer by modeling output dependencies using BERT, allowing the model to consider the bi-directional context in the target sequence (refer to Table 5.1). BERT-CTC addressed the domain mismatch issue through the effective use of powerful representations obtained from BERT. However, the effectiveness of BERT-CTC diminishes when compared to CTC and Transducer models trained on the ASR vocabulary, thereby reducing the advantage gained from using BERT.

5.5.3 Main Results

Table 5.4 lists results on the major ASR tasks, including LS-100, LS-960, TED2, and AS1, evaluated in terms of the WER or CER. Here, the proposed models, BERT-CTC and BECTRA, are compared with Transducer trained on the ASR vocabulary, which was the best-performing baseline from Table 5.3. As discussed in Section 5.5.2, BERT-CTC underperformed compared to

Table 5.4: WER or CER of proposed BERT-CTC and BECTRA compared to Transducer baseline, evaluated on major ASR tasks.

			WER [%] (↓)										**CER** [%] (↓)	
			LibriSpeech-100h				LibriSpeech-960h				TED-LIUM2		AISHELL-1	
	Output Vocab.		Dev		Test		Dev		Test					
Model	$\mathcal{V}^\mathsf{a}$	$\mathcal{V}^\mathsf{b}$	clean	other	clean	other	clean	other	clean	other	Dev	Test	Dev	Test
Transducer	✓		5.9	17.7	6.0	17.6	**2.5**	6.8	**2.8**	6.8	7.8	7.4	4.9	5.3
BERT-CTC		✓	7.0	16.4	7.1	16.5	3.1	7.1	3.2	7.1	8.3	7.6	3.9	4.0
BECTRA	✓		**5.1**	**15.4**	**5.4**	**15.5**	2.6	**6.7**	2.9	**6.7**	**7.3**	**6.9**	**3.7**	**3.9**

Table 5.5: WER comparison on low-resource Libri-Light-10h task.

			WER [%] (↓)			
	Output Vocab.		Dev		Test	
Model	$\mathcal{V}^\mathsf{a}$	$\mathcal{V}^\mathsf{b}$	clean	other	clean	other
CTC		✓	36.2	46.9	36.8	47.7
Transducer		✓	34.9	45.9	35.6	46.8
BERT-CTC		✓	**27.2**	**39.2**	**28.3**	**40.4**
CTC	✓		24.8	43.2	25.8	44.4
Transducer	✓		21.7	38.8	22.3	39.7
BECTRA	✓		**19.9**	**36.1**	**20.3**	**37.2**

Transducer in several tasks, which is attributed to the vocabulary discrepancy. Overall, BECTRA achieved the highest performance compared to all other models, taking the advantages of both BERT-CTC and Transducer. BECTRA effectively utilized BERT knowledge by adopting BERT-CTC-based feature extraction and incorporated the transducer framework to enable more suitable and flexible token generation using the ASR vocabulary. In Section 5.6.2, I present the specific errors that BECTRA succeeded in recovering, as compared to BERT-CTC.

Another notable observation was that with more training data in LS-960, the performance gap between Transducer and BECTRA narrowed, and the impact of BERT became less pronounced. This finding led me to explore two further directions for investigation, which I discuss in the following two subsections.

5.5.4 Results on Low-Resource Setting

The proposed models, BERT-CTC and BECTRA, were less effective in the LS-960 task, likely because the dataset already contained a sufficient amount of text data. Consequently, the ASR models were capable of acquiring rich linguistic information specific to the LibriSpeech domain, without relying on BERT knowledge. This can be consistent with the recent findings from other

Table 5.6: WER on CoVoST2 task comparing with and without considering punctuation or casing. CTC and Transducer were trained on the ASR vocabulary. †Trained using a "cased" model.

		Text Style		**WER** [%] (↓)	
Model	**Masked LM**	Punct.	Casing	Dev	Test
CTC	–	–	–	18.7	23.2
Transducer	–	–	–	15.2	18.8
BECTRA	$\text{BERT}_{\text{BASE}}$	✗	✗	14.4	17.6
	$\text{BERT}_{\text{BASE}}$	✓	✗	**14.0**	17.2
	$\text{BERT}^{\dagger}_{\text{BASE}}$	✓	✓	**14.0**	**17.1**
	$\text{RoBERTa}^{\dagger}_{\text{BASE}}$	✓	✓	14.1	**17.1**
	$\text{BERT}_{\text{LARGE}}$	✓	✗	**13.4**	**16.4**
	$\text{BERT}^{\dagger}_{\text{LARGE}}$	✓	✓	13.8	16.7
	$\text{DistilBERT}_{\text{BASE}}$	✓	✗	**14.3**	**17.6**
	$\text{DistilBERT}^{\dagger}_{\text{BASE}}$	✓	✓	14.9	17.9
	$\text{ALBERT}_{\text{BASE}}$	✓	✗	14.6	17.7

studies on LibriSpeech (Zhang et al., 2020b), which has indicated that a language model (used for shallow fusion) provides limited gains in conjunction with a well-trained ASR model.

I, thus, examined the other end of the spectrum, evaluating the proposed models on LL-10, an extremely low-resource condition with only 10 hours of training data. Table 5.5 lists WERs obtained by training the models using either the ASR or BERT vocabulary ($\mathcal{V}^{\mathsf{a}}$ vs. $\mathcal{V}^{\mathsf{b}}$). The overall trend was in line with what was observed in the previous results in Tables 5.3 and 5.4, highlighting the ability of the BERT-based approaches to compensate for the limited availability of training text data.

5.5.5 Results on Preserving Punctuation and Casing

The experimental setups used to obtain results in Table 5.4 could potentially limit the full capabilities of BERT since the training text data was normalized for ASR training (see Section 5.4.1). Pre-trained language models are typically trained on written-style text, preserving punctuation and casing as a standard practice. Therefore, the prior experimental condition may have caused a discrepancy regarding the text format used as input into BERT.

To verify the above consideration, I conducted experiments on CV2 while explicitly controlling the preservation of punctuation and casing. Table 5.6 presents the results of BECTRA in comparison to CTC and Transducer trained on the ASR vocabulary. Note that punctuation and casing only affect the BERT-CTC processing (i.e., Algorithm 4 Step 1) with the BERT vocabulary $\mathcal{V}^{\mathsf{b}}$, and the WER is calculated using the normalized text, which is obtained from a hypothesis tokenized by the ASR vocabulary $\mathcal{V}^{\mathsf{a}}$. For training with casing preserved, I used the "cased" model[7]. Looking at the BECTRA results based on BERT$_{\text{BASE}}$, BECTRA outperformed the baseline models even without considering punctuation and casing, which is consistent with the outcome in Table 5.4. The addition of punctuation provided further improvement, whereas the impact of adding casing was less significant. This suggests the importance of matching the input text format used for BERT with that used for input during pre-training.

5.5.6 Application of BERT Variants

In Table 5.6, I compare the BECTRA results obtained from using various pre-trained masked language models other than BERT$_{\text{BASE}}$. RoBERTa$_{\text{BASE}}$[8] is an extension of BERT that is constructed with an improved pre-training procedure (Liu et al., 2019). However, there was little improvement over the results using vanilla BERT$_{\text{BASE}}$. BECTRA greatly benefited from increasing the capacity of the pre-trained language model, with BERT$_{\text{LARGE}}$[9] achieving the best overall performance.

BECTRA incurs a high computational cost, especially during inference, primarily due to the multiple forward passes in BERT (i.e., $K = 10$ times) with the $\mathcal{O}(N^2)$ computational and memory complexities in self-attention layers. To mitigate this drawback, I explored lightweight variants, including DistilBERT$_{\text{BASE}}$[10] and ALBERT$_{\text{BASE}}$[11]. DistilBERT distills BERT's knowledge into a more compact model (Sanh et al., 2019), while ALBERT reduces the model size by sharing common parameters across layers (Lan et al., 2020). Both lightweight models achieved superior results compared to the baseline models, with only minor performance degradation compared to BERT$_{\text{BASE}}$.

In alignment with the observation in Section 5.5.5, the BERT variants gave more importance to considering punctuation than casing.

5.5.7 Combination with Shallow Fusion

I examined the feasibility of utilizing an in-domain language model during BECTRA inference. BECTRA can adopt the commonly used shallow fusion technique, as its inference process relies on the standard transducer framework (Algorithm 4 Step 2). I used the external text data provided by LibriSpeech to train an RNN-LM, which consisted of 4 LSTM layers with 2048 units. The language model weight and beam size for shallow fusion were configured to 0.5 and 20, respectively.

Table 5.7 compares the WER between the Transducer and BECTRA models, which were trained on LS-100 and decoded with or without the RNN-LM. Through the incorporation of linguistic knowledge from RNN-LM via shallow fusion, Transducer significantly improved the per-

[7]`https://huggingface.co/bert-base-cased`
[8]`https://huggingface.co/roberta-base`
[9]`https://huggingface.co/bert-large-{cased,uncased}`
[10]`https://huggingface.co/distilbert-base-{cased,uncased}`
[11]`https://huggingface.co/albert-base-v2`

Table 5.7: WER on LibriSpeech-100h task comparing with and without performing shallow fusion during inference.

	WER [%] (↓)							
	w/o Shallow Fusion				w/ Shallow Fusion			
	Dev		Test		Dev		Test	
Model	clean	other	clean	other	clean	other	clean	other
Transducer	5.9	17.7	6.0	17.6	5.1	15.0	5.1	15.1
BECTRA	**5.1**	**15.4**	**5.4**	**15.5**	**4.5**	**14.2**	**4.9**	**14.2**

Table 5.8: WER and classification accuracy on SLURP intent classification task.

Model	**WER** [%] (↓)	**Acc.** [%] (↑)
ESPnet-SLU (Arora et al., 2022)	–	86.3
ASR + BERT (Arora et al., 2022)	–	85.7
BERT-CTC ($K = 1$)	19.1	87.0
BERT-CTC ($K = 20$)	**18.2**	**87.8**

formance, resulting in lower WERs compared to BECTRA, which by default employs BERT in its formulation. Similarly, the performance of BECTRA was further improved by utilizing shallow fusion. This indicates that BECTRA effectively integrated general knowledge from BERT and domain-specific knowledge from the RNN-LM, thereby enhancing its ability to consider linguistic information.

5.5.8 Application to Spoken Language Understanding

Besides end-to-end ASR, I explored the potential application of BERT-CTC to end-to-end spoken language understanding (SLU), focusing on an intent classification task as a case in point. To this end, an additional cross-entropy loss for intent classification was computed from $\mathbf{g}_{T'+1}$, the $(T' + 1)$-th output of the concatenation network that corresponds to the classification symbol `[CLS]` of BERT (Devlin et al., 2019). This modified BERT-CTC was trained by simply adding the SLU loss to the ASR loss $\mathcal{L}^{\textsf{bert-ctc}}$ without applying any weighting factors. During inference, BERT-CTC decoding was first performed to obtain a refined ASR hypothesis, and an intent label was then predicted in the final iteration.

Table 5.8 lists the results of the SLURP intent classification task (see Appendix A.11 for details), evaluated in accuracy. I refer to the ESPnet-SLU (Arora et al., 2022) result as a baseline, which has performed SLU along with ASR by prepending an intent label to the corresponding output sequence. I also refer to the ESPnet-SLU result obtained by stacking BERT on top of an

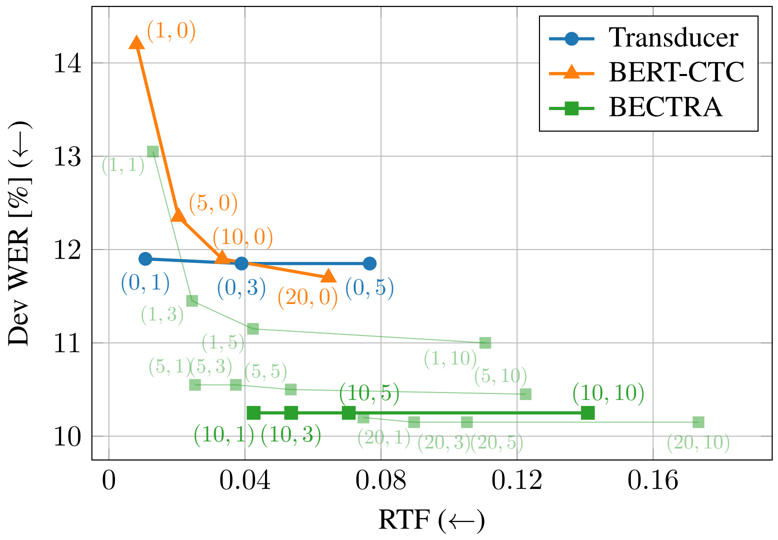


Figure 5-3: Comparison of WER versus RTF trade-offs for models evaluated on the averaged LibriSpeech development sets. The values in parentheses indicate the number of iterations and a beam size (K, B). This figure is reproduced from my conference paper (Higuchi et al., 2023b).

ASR model, which has been found to be less effective. BERT-CTC outperformed the baselines by effectively incorporating acoustic and linguistic information. By decoding in a single iteration ($K = 1$), BERT-CTC predicted an intent label only from speech, and the accuracy was already higher than those of baselines. I observed a slight but clear gain by increasing K, which improved both ASR and SLU performance thanks to the incorporation of BERT's knowledge.

5.6 Analysis

5.6.1 Trade-off Between WER and Inference Speed

Figure 5-3 depicts the trade-offs between the WER and RTF, comparing the proposed BERT-CTC and BECTRA to Transducer on LS-100. RTF was measured using a single V100 GPU with a batch size of 1. Upon analyzing the results obtained through greedy decoding with $K \leq 1$ and $B = 1$, Transducer exhibited the most favorable performance with the lowest values for both WER and RTF. Increasing the number of iterations ($K > 1$) in BERT-CTC resulted in a larger performance gain as compared to increasing the beam size ($B > 1$) in Transducer. BECTRA achieved substantial improvement through beam search decoding when $K = 1$; however, the extent of improvement diminished with an increase in the number of iterations at $K > 1$. The results suggest that BERT-CTC decoding effectively refined the output sequence, which reduced the search space during beam search decoding. As a consequence, a smaller beam size ($B = 3$ or 5) was sufficient without compromising inference speed. Overall, BECTRA with $K = 10$ and $B \in$

Table 5.9: Example inference process of BECTRA (Algorithm 4), decoding an utterance from CoVoST2 test set. During BERT-CTC inference (Algorithm 4 Step 1), the highlighted tokens were replaced with the mask token and subsequently re-predicted in the following iteration. The transducer decoding with beam search further refined the BERT-CTC result (Algorithm 4 Step 2). The corrected tokens are colored in blue, while the incorrect ones are in red. Punctuation is only considered during BERT-CTC decoding.

Refine ↓	BERT-CTC ($k = 1$)	car ’ s business in mc cly was in sawmill mills , tutine , lumber , and land .
	BERT-CTC ($k = 5$)	carr ’ s business in mclean was in sawmill mills , tu , lumber and land .
	BERT-CTC ($k = 10$)	carr ’ s business in mclean was in sawmills , carpenter , lumber and land .
	BECTRA ($B = 5$)	carr &apos s business in mcclenny was in sawmills turpentine lumber and land
	Reference	carr &apos s business in mcclenny was in sawmills turpentine lumber and land

$\{1, 3, 5\}$ achieved a desirable balance between WER and RTF, taking advantage of both BERT-CTC’s fast non-autoregressive decoding and Transducer’s accurate autoregressive decoding.

5.6.2 Example Decoding Process of BECTRA

Table 5.9 provides an example of the BECTRA inference process, which was obtained by decoding an utterance in the CoVoST2 test set. I used the model trained with BERT$_{\text{BASE}}$ from Table 5.6, which only preserved punctuation during BERT-CTC decoding. During the first iteration of BERT-CTC inference ($k = 1$), the model produced erroneous predictions that are phonetically similar to the actual tokens (e.g., “car” vs. “carr” and “mc cly” vs. “mcclenny”). The model was solely conditioned on acoustic information during the first iteration, which led to difficulties in accurately determining the target tokens. As the iteration progressed ($k \in \{5, 10\}$), the model was able to correct some of the errors by considering the output dependencies extracted from BERT. However, the model still struggled with recognizing rare words such as the place name “mcclenny.” In addition, the model mistakenly identified the technical term “turpentine” as “carpenter,” despite the two words sounding dissimilar. This error is likely due to the contextual information being influenced by the BERT knowledge. The transducer decoding in BECTRA effectively recovered these errors by accurately predicting the rare words. The autoregressive token generation facilitated a more flexible estimation of tokens using a vocabulary suited for ASR in the CoVoST2 domain.

5.6.3 Attention Visualization

Figure 5-4 presents example attention weight matrices that were obtained from the second self-attention layer of the concatenation network in BERT-CTC. Two major attention patterns were identified: weights aligning audio and token sequences by capturing their inter-dependencies (Figure 5-4 left) and weights attending to the inner-dependencies within each sequence (Figure 5-4 right). These attention weights support the effectiveness of the proposed architectural design for

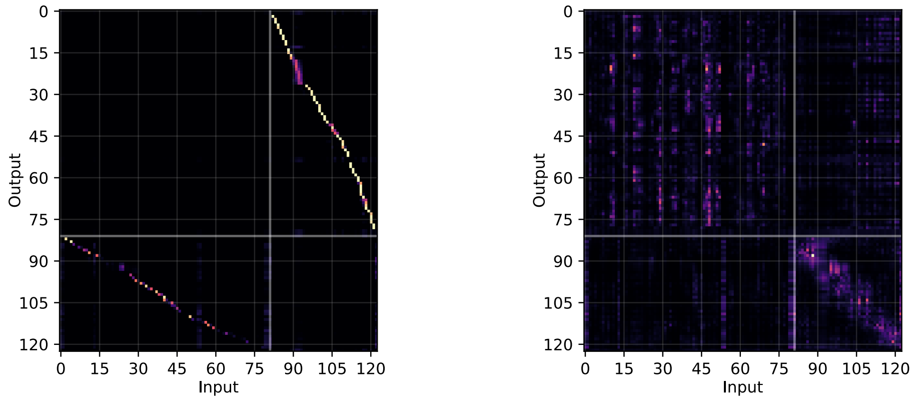


Figure 5-4: Visualization of self-attention weights learned in concatenation network. White lines indicate the boundaries of audio and token sequences, H and E, which are concatenated and processed by self-attention in Eq. (5.10). These figures are reproduced from my conference paper (Higuchi et al., 2022d).

BERT-CTC, indicating audio and linguistic information are dynamically merged by considering their inter and inner-dependencies.

5.6.4 Conditional Independence of Acoustic Observation

I empirically validated the conditional independence assumption made in Eq. (5.4), where the output sequence W^{b} depends solely on its masked sequence $\tilde{W}^{\mathsf{b}}$ without acoustic information O. To this end, I incorporated trainable cross-attention layers into the BERT module, which is similar to the technique proposed in Adapter-BERT Networks (Guo et al., 2020). These additional layers enable each BERT layer to attend to the audio encoder output H, thereby allowing BERT-CTC to achieve $p(W^{\mathsf{b}}|\tilde{W}^{\mathsf{b}}, O)$. After training the modified BERT-CTC on LS-100, I observed inferior WERs compared to the original BERT-CTC presented in Table 5.4, with 7.2%/17.9% and 7.3%/18.0% on the development and test sets, respectively.

The finding above suggests that BERT is capable of capturing sophisticated linguistic information without relying on audio input conditioning. Furthermore, this implies that the proposed formulation does not require any adaptation or fine-tuning of BERT.

5.6.5 BECTRA with BERT Vocabulary

To further examine the advantages of BECTRA compared to BERT-CTC, I utilized the BERT vocabulary to train the transducer decoder of BECTRA. Under the same experimental conditions using LS-100, BECTRA with the BERT vocabulary achieved 7.1%/16.6% on the LibriSpeech test sets, while BERT-CTC resulted in slightly worse scores of 7.3%/16.9%. This improvement over BERT-CTC can be attributed to the transducer-based formulation in BECTRA, which does not rely on the conditional independence assumption between outputs (Eq. (5.5) vs. Eq. (5.25)). Nevertheless, using the ASR vocabulary appeared to be the superior choice.

Note that, different from the results reported in Table 5.4, the above results were obtained by reducing the number of training epochs ($100 \rightarrow 50$). Training a transducer-based model with a large vocabulary size leads to a substantial increase in memory consumption (Lee et al., 2022), resulting in a significant extension of the training time. Therefore, employing the ASR vocabulary is the optimal approach for constructing the BECTRA model, as it also facilitates faster training and inference.

5.7 Summary

In this chapter, I introduced a novel approach to formulating end-to-end ASR, utilizing pre-trained masked language models to facilitate the extraction of linguistic information. The proposed models, BERT-CTC and BECTRA, were specifically designed to effectively integrate pre-trained language models (e.g., BERT) into end-to-end ASR models. BERT-CTC adapted BERT for CTC by addressing the constraint of the conditional independence assumption between output tokens. This enabled explicit conditioning of BERT's contextualized embeddings in the ASR process, seamlessly merging audio and linguistic information through an iterative refinement algorithm. BECTRA extended BERT-CTC to the transducer framework and trained the prediction and joint networks using a vocabulary suitable for ASR training. This aimed to bridge the gap between the text processed in end-to-end ASR and BERT, as these models have distinct vocabularies with varying text formats and styles, such as the presence of punctuation. Experimental results on various ASR tasks demonstrated that the proposed models improve over both the CTC and transducer-based baselines, owing to the incorporation of BERT knowledge. I also showed that BERT-CTC provides semantic representations beneficial for solving SLU tasks. The comprehensive analysis and investigation of various aspects verified the effectiveness of the proposed formulations and architectural designs.

6 End-to-End Speech Recognition Guided by Instruction-Tuned Large Language Model

6.1 Introduction

Driven by the remarkable achievements in self-supervised pre-training (Devlin et al., 2019; Radford et al., 2018), the scale of recent language models has witnessed an exponential surge. This growth is primarily fueled by the vast amount of data sourced from the internet, coupled with a continual increase in the size of DNN-based models, which now have billions or even trillions of parameters. These large language models (LLMs) have shown remarkable adaptability in acquiring various capabilities with minimal or even no task-specific training data (Radford et al., 2019; Brown et al., 2020; Scao et al., 2022; Chowdhery et al., 2022; Wei et al., 2022b). This has highlighted the potential of LLMs to solve different downstream tasks in a few-shot or even zero-shot manner, enabling highly flexible and efficient information processing that has been beneficial to end users (OpenAI, 2023).

Such inherent few-/zero-shot learning capabilities of LLMs can be improved by strategically using the prompting mechanism to control their behavior. In-context learning (Brown et al., 2020) allows LLMs to learn tasks without requiring parameter updates, which is achieved by adding a few input-output examples prior to a target input. This technique directs LLMs toward desired outputs by conditioning the model's text generation process on a specific domain or context. To further enhance the controllability of LLMs, instruction fine-tuning (Wei et al., 2022a; Chung et al., 2022; Ouyang et al., 2022) is adopted to make the model follow instructions more closely and generate responses that align with the desired outcome. This involves fine-tuning the model on a mixture of highly diverse tasks that are phrased as natural language instructions, and it has been shown to substantially boost zero-shot performance on unseen tasks.

This chapter explores the application of the LLM's zero-shot learning capability to address speech-processing tasks, specifically focusing on end-to-end ASR. I aim to align the target speech processing task with a related NLP task and leverage the linguistic information embedded within the LLM to improve the target speech task; I relate the end-to-end ASR task to (zero-shot) grammatical error correction (Nie et al., 2022; Wu et al., 2023; Fang et al., 2023). Recent efforts have demonstrated the effectiveness of using instruction-tuned LLMs for ASR error correction as post-processing (Ma et al., 2023a,b), wherein the model is instructed to select the most probable hypothesis from an ASR N-best list, similar to a rescoring approach. My approach, in contrast, attempts to directly integrate the LLM's ability to correct grammatical errors into an end-to-end ASR formulation.

Chapter 5 similarly proposed to enhance end-to-end ASR by utilizing BERT, a large-scale language model pre-trained through the masked language model objective. This chapter shifts the focus to a more recent language model with an order of magnitude larger in size (7B $\gg$ 100M), which is pre-trained using the standard left-to-right language model objective, as referencing in Eq. (2.10).

6.2 Methodology

Figure 6-1 illustrates the overview of the proposed model, which is built upon the hybrid CTC and AED architecture (as described in Section 2.2.3), integrating an autoregressive decoder-only LLM (e.g., GPT-3 (Brown et al., 2020) and LLaMA (Touvron et al., 2023a)) at the input of the decoder network. The LLM is utilized to extract linguistic information that contributes to solving an ASR task. To achieve this, I capitalize on the LLM's potential as a zero-shot grammatical error correction model (Wu et al., 2023; Fang et al., 2023; Ma et al., 2023b). The proposed model mainly involves the following processes. First, a hypothesis is obtained from the encoder output via CTC decoding. Subsequently, I feed this hypothesis to the LLM for a subject to correction, accompanied by an explicit instruction (or *prompt*) to precisely guide the LLM's interpretation of the hypothesis. The decoder network then takes as input the LLM output and performs text generation, which is trained using an ASR corpus with a specific domain of interest.

In the following sections, I present a brief explanation of an instruction-tuned LLM with an emphasis on its instruction-based controllability. I then delve into the proposed integration of the LLM and end-to-end ASR, providing a detailed formulation that substantiates the effectiveness of the proposed model design.

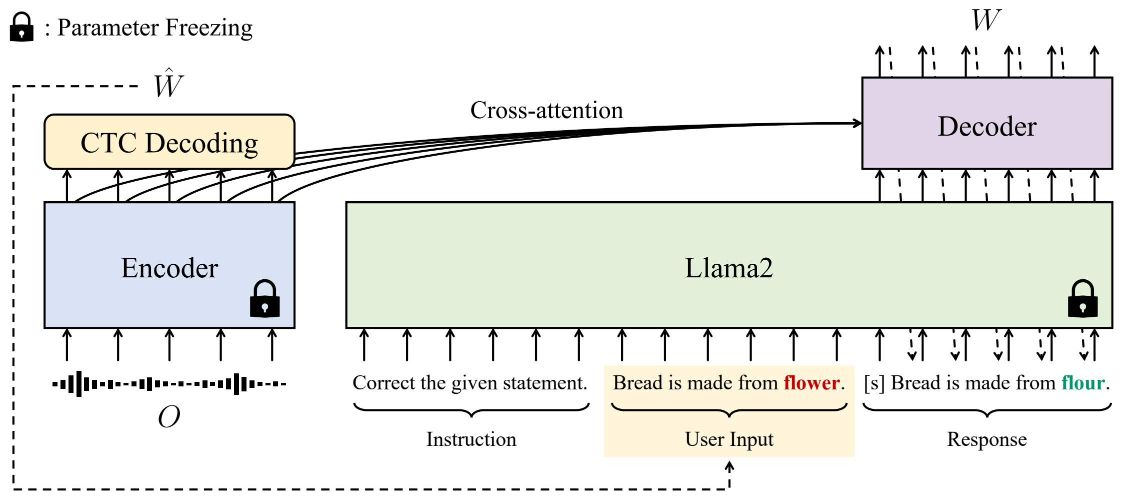


Figure 6-1: Overview of proposed integration of instruction-tuned LLM, i.e., Llama2(-Chat), and end-to-end ASR. A hybrid CTC and AED-based model is constructed, with Llama2 serving as the front end for the decoder. Given a hypothesis obtained from the encoder output, Llama2 is prompted to perform grammatical error correction. The decoder then generates a sequence using linguistic information derived from Llama2. This figure is reproduced from my preprint (Higuchi et al., 2023a).

6.2.1 Instruction-Tuned Large Language Model

Recent advances in LLMs have centered around the development of decoder-only models, which train deep stacks of self-attention layers (Vaswani et al., 2017) for autoregressive text generation. With access to extensive amounts of internet-derived text data and exponentially increasing parameter sizes, LLMs have shown their capacity to learn a range of NLP tasks without the need for explicit supervision. This has highlighted the LLM's potential for performing zero-shot task transfer without relying on gradient updates or fine-tuning (Radford et al., 2019). The latest LLMs have the capability to be "prompted" for executing specific tasks. This involves providing instructions or contexts that influence the subsequent output generated by the model, thereby enabling users to obtain desired information. Additionally, advancements in instruction fine-tuning have further enhanced the model's ability to produce responses that align with human expectations (Wei et al., 2022a; Ouyang et al., 2022).

I focus on Llama2-Chat, an instruction-fine-tuned version of Llama2 (Touvron et al., 2023b), as the LLM used in the proposed model. I hereafter refer to this chat model as "Llama2" for brevity. Given a series of input toke sequences, Llama2 outputs a D^{llama}-dimensional hidden vector $\mathbf{e}_l$ at a token position l as

$$\mathbf{e}_l = \text{Llama2}(\underbrace{W^{\mathsf{inst}}}_{\text{Instruction}}, \underbrace{W^{\mathsf{user}}}_{\text{User Input}}, \underbrace{W_{<l}}_{\text{Response}}) \in \mathbb{R}^{D^{\mathsf{llama}}}, \tag{6.1}$$

where $W^{\mathsf{inst}} \in \mathcal{V}^{L^{\mathsf{inst}}}$ is an L^{inst}-length instruction sequence, $W^{\mathsf{user}} \in \mathcal{V}^{L^{\mathsf{user}}}$ is an L^{user}-length user input sequence, $W \in \mathcal{V}^{L}$ is an L-length output token sequence with $W_{<l} = (w_0, \cdots, w_{l-1})$ and $w_0 = \texttt{<sos>}$, and $\mathcal{V}$ is the vocabulary of Llama2. Using the Llama2 output $\mathbf{e}_l$ from Eq. (6.1), the likelihood of a target sequence is computed as

$$p(W|W^{\mathsf{inst}}, W^{\mathsf{user}}) = \prod_{l=1}^{L} p(w_l|W_{<l}, W^{\mathsf{inst}}, W^{\mathsf{user}}), \tag{6.2}$$

$$p(w_l|W_{<l}, W^{\mathsf{inst}}, W^{\mathsf{user}}) = \text{Softmax}\left(\text{Linear}_{D^{\mathsf{llama}} \rightarrow |\mathcal{V}|}(\mathbf{e}_l)\right) \in [0, 1], \tag{6.3}$$

where $\text{Linear}_{D^{\mathsf{llama}} \rightarrow |\mathcal{V}|}(\cdot)$ transforms the feature vector to a logit.

In the context of ASR, the formulation presented in Eq. (6.2) is similar to the Transformer-based language models commonly used for shallow fusion or rescoring (Irie et al., 2019; Karita et al., 2019). However, Llama2 distinguishes itself through its vast scale (over billions of parameters), having superior versatility in extracting linguistic information, which can be explicitly controlled through the auxiliary inputs (i.e, W^{inst} and W^{user}) tailored to specific tasks. My primary focus is to explore the capability of such LLMs to effectively bridge the gap between speech and NLP tasks.

6.2.2 Proposed Integration of Large Language Models and End-to-End ASR

The proposed approach factorizes the posterior distribution of ASR $p(W|O)$ (from Eq. (2.1)) as

$$p(W|O) = \sum_{\tilde{W} \in \mathcal{H}(W)} p(W|\tilde{W}, O) p(\tilde{W}|O), \tag{6.4}$$

where $\tilde{W} \in \mathcal{V}^{L'}$ represents an L'-length ASR hypothesis, and $\mathcal{H}(W)$ is a set of all possible hypotheses compatible with W. In other words, $\mathcal{H}(W)$ comprises token sequences that are prone to being misrecognized from input speech O. On the right side of Eq. (6.4), $p(W|\tilde{W}, O)$ is further factorized by applying a probabilistic chain rule as

$$p(W|\tilde{W}, O) = \prod_{l=1}^{L+1} p(w_l|\tilde{W}, W_{<l}, O), \tag{6.5}$$

where $w_{L+1} = \texttt{<eos>}$. The posterior distribution $p(W|\tilde{W}, O)$ in Eq. (6.5) is modeled using the AED architecture, as described in Section 2.2.3. The emission probability at each token position in Eq. (6.5) is computed as follows:

$$H = \text{ConformerEncoder}(O) \in \mathbb{R}^{T' \times D^{\text{model}}}, \tag{6.6}$$

$$\mathbf{e}_l = \text{Llama2}(W^{\text{inst}}, \tilde{W}, W_{<l}) \in \mathbb{R}^{D^{\text{llama}}}, \tag{6.7}$$

$$\mathbf{q}_l = \text{TransformerDecoder}_l(\mathbf{e}_1, \cdots, \mathbf{e}_{L+1}, H) \in \mathbb{R}^{D^{\text{model}}}, \tag{6.8}$$

$$p(w_l|\tilde{W}, W_{<l}, O) = \text{Softmax}\left(\text{Linear}_{D^{\text{model}} \to |\mathcal{V}|+1}(\mathbf{q}_l)\right) \in [0, 1]. \tag{6.9}$$

Eq. (6.6) is the Conformer encoder from Eq. (2.61). In Eq. (6.7), the Llama2 output $\mathbf{e}_l$ is generated using the same process as described in Eq. (6.1), where an ASR hypothesis $\tilde{W}$ is fed into Llama2 as the user input, accompanied by an instruction W^{inst} for guiding the model toward the grammatical error correction task (see the Llama2 input depicted in Figure 6-1 for example). In Eq. (6.8), $\text{TransformerDecoder}_l(\cdot)$ represents the l-the output of the Transformer decoder from Eq. (2.55), where $Q^{(0)}$ is derived from the Llama2 outputs $(\mathbf{e}_1, \cdots, \mathbf{e}_{L+1})$ by applying a linear layer that projects D^{llama} to D^{model}, instead of using Eq. (2.56). $\mathbf{e}_{L+1}$ represents the Llama2 output corresponding to the final token w_L, as the input is right-shifted due to the insertion of `<sos>`. In Eq. (6.9), $\text{Linear}_{D^{\text{model}} \to |\mathcal{V}|+1}(\cdot)$ transforms the output vector $\mathbf{q}_l$ to a logit.

Training

The proposed model is constructed through the following two-stage process:

1. Train a hybrid CTC and AED-based end-to-end ASR model (see Section 2.2.3 for details);
2. Train the proposed model (Figure 6-1) by training a new decoder network from scratch using Llama2, along with the pre-trained encoder and CTC networks from Step 1.

In Step 2, the parameters of the decoder network are only updated.

The objective function of the proposed model is defined by the negative log-likelihood of Eq. (6.4) expanded with Eq. (6.5),

$$\mathcal{L} = -\log \sum_{\tilde{W} \in \mathcal{H}(W)} \prod_{l=1}^{L+1} p(w_l|\tilde{W}, W_{<l}, O) p(\tilde{W}|O), \tag{6.10}$$

$$\leq -\mathbb{E}_{\tilde{W} \sim \mathcal{H}(W)} \left[\log \prod_{l=1}^{L+1} p(w_l|\tilde{W}, W_{<l}, O) \right]. \tag{6.11}$$

In Eq. (6.11), the intractable marginalization over hypotheses is approximated under expectation with respect to the sampling distribution $\mathcal{H}(W)$. The sampling process is implemented by running the encoder network in "training mode" (with dropout enabled) and performing the best path decoding algorithm of CTC (Graves et al., 2006), which is a similar strategy utilized in uncertainty estimation (Gal and Ghahramani, 2016; Vyas et al., 2019).

Inference

Substituting Eq. (6.4) into Eq. (2.1), the most probable sequence $\hat{W}$ is estimated as

$$\hat{W} = \arg\max_{W} \sum_{\tilde{W} \in \mathcal{H}(W)} p(W|\tilde{W}, O) p(\tilde{W}|O), \tag{6.12}$$

$$\approx \arg\max_{W} p(W|\tilde{W}', O), \tag{6.13}$$

$$\text{where } \tilde{W}' = \arg\max_{\tilde{W}} p(\tilde{W}|O). \tag{6.14}$$

To handle the intractable marginalization in Eq. (6.12), the Viterbi approximation is applied with respect to the posterior distribution $p(\tilde{W}|O)$, as shown in Eqs. (6.13) and (6.14). The search process in Eq. (6.14) is implemented by calculating the encoder output (Eq. (6.6)) and performing the best path decoding algorithm of CTC (refer to Section 2.2.1). Subsequently, in Eq. (6.13), the joint CTC and AED decoding algorithm (Watanabe et al., 2017) is performed to obtain the final output (see Section 2.2.3 for details). Notice that this inference formulation incorporates top-down and bottom-up processing, similar to the BERT-CTC decoding algorithm presented in 5.2.1, also reflecting the concept introduced in Chapter 1.

6.3 Additional Related Work on Two-Pass End-to-End ASR

In this section, I further clarify the position of my research to another relevant topic in end-to-end ASR, focusing on the two-pass approaches.

In the context of two-pass decoding in ASR, it is common practice to employ a second-pass model to refine outputs produced by a first-pass model. For example, language models are often used to rescore and rerank multiple hypotheses generated during the first-pass ASR decoding process (Sundermeyer et al., 2015; Chan et al., 2016; Kannan et al., 2018; Salazar et al., 2020). Recent advances in deep learning have enabled an ASR model to train both the first-pass and second-pass models in an end-to-end fashion, introducing an additional decoder that refines a first-pass sequence with deliberation (Xia et al., 2017). The two-pass end-to-end ASR framework (Sainath et al., 2019) involves training a transducer-based model in conjunction with an attention decoder, which is specifically optimized to rescore hypotheses generated during transducer decoding. Additionally, acoustic embeddings from the encoder can be helpful in facilitating the training of the rescoring decoder (Hu et al., 2020; Wang et al., 2022).

The proposed formulation in Eq. (6.5) shares similarities with the conventional two-pass-based strategy employed in end-to-end ASR. However, it differs in that the decoder network does not specifically deliberate on hypotheses to generate an output sequence. Instead, it leverages linguistic information derived from the LLM, which is prompted to improve the hypotheses.

6.4 Experiments

6.4.1 Experimental Setting

I used the ESPnet toolkit (Watanabe et al., 2018) for conducting the experiments.

Data

The experiments were carried out using various corpora spanning different domains, including LibriSpeech (LS), TED-LIUM2 (TED2), and CoVoST2 (CV2). See Appendix A for the detailed statistics of each corpora. In addition to the 960-hour training set (LS-960) in LS, I used the *train-clean-100* subset (LS-100) for a lower-resource scenario. Note that transcriptions provided by the above corpora, except for CV2, were normalized by default for ASR training, where punctuation was removed, and casing was standardized to lowercase. I specifically used CV2 for training models with punctuation and casing preserved, as this can be crucial for the LLM to accurately capture linguistic information, a point emphasized in Chapter 5. During evaluation using CV2, I removed punctuations to exclusively assess ASR performance. All the text tokenization was performed using the vocabulary of Llama2, where $|\mathcal{V}| = 32$k.

Model and Network Architecture

The baseline model was the joint CTC and AED-based model (described in Section 2.2.3), which corresponds to the model trained in Step 1 of Section 6.2.2. The proposed model was trained as described in Step 2 of Section 6.2.2, using the loss defined in Eq. (6.11). The encoder network consisted of two Conv2D layers followed by a stack of $N^{\mathsf{enc}} = 12$ Conformer encoder blocks. The number of heads N^{head}, the dimension of a self-attention layer D^{model}, the dimension of a feed-forward network D^{ff}, and the kernel size were set to (4, 256, 1024, 31) for LS-100, TED2, and CV2; and (8, 512, 2048, 31) for LS-960. The decoder network was a stack of $N^{\mathsf{dec}} = 6$ Transformer decoder blocks, where (N^{head}, D^{model}, D^{ff}) were set to (4, 256, 2048) for LS-100, TED2, and CV2; and (8, 512, 2048) for LS-960. For the proposed model, I used the Llama2-Chat model with 7 billion parameters, which was accessed through the HuggingFace library (Wolf et al., 2020)[1].

Training and Decoding Configurations

I mostly followed configurations provided by the ESPnet2 recipe for each dataset (as specified in Appendix A). The baseline model was trained up to 50 epochs, and subsequently, the proposed model was trained up to 50 epochs for LS-100, and 25 epochs for the other datasets. The Adam optimizer (Kingma and Ba, 2015) with Noam learning rate scheduling (Vaswani et al., 2017) was

[1]`https://huggingface.co/meta-llama/Llama-2-7b-chat-hf`

used for weight updates, where the number of warmup steps was set to 15k, and a peak learning rate was tuned from $\{0.0015, 0.002\}$. I followed the default recipes for regularization hyper-parameters (e.g., dropout rate and label-smoothing weight). I augmented speech data using speed perturbation (Ko et al., 2015) and adaptive SpecAugment (Park et al., 2020a). I set the weight of $\lambda^{\text{ctc-aed}} = 0.3$ to the CTC loss during baseline model training (refer to Eq. (2.43)). After training, a final model was obtained for evaluation by averaging model parameters over 10 checkpoints with the best validation performance. For the joint CTC and AED decoding performed in Eq. (6.13), I used a beam size of $B = 20$ unless otherwise specified and the weight for the CTC probability of $\xi = 0.3$ (refer to Eq. (2.44)).

Prompt

I empirically designed a prompting instruction to guide Llama2 in performing grammatical error correction, where I set W^{inst} to "*You will be provided with a statement in quotes. Correct the wrong words and provide your revised version.*" As specified in the instruction, a user input (i.e., a hypothesis) was enclosed within double quotation marks. I followed the prompt format described in Touvron et al. (2023b)[2] for inputting sequences into Llama2, which resulted in a prompt sequence as follows

```
<s>[INST] <<SYS>>
You will be provided with a statement in quotes.  Correct the wrong words
and provide your revised version.
<</SYS>>

"${ASR_HYPOTHESIS}" [/INST]
```

6.4.2 Main Results

Table 6.1 lists results on all the tasks, which are evaluated using the WER. For tasks other than LS-960, the proposed model with Llama2 applied to the front end of the decoder consistently outperformed the standard CTC-attention-based model, requiring only the retraining of the decoder network with minimal parameters (e.g., 19 million on LS-100). These improvements over the baseline indicate the successful utilization of linguistic knowledge obtained from the LLM to enhance ASR performance, which I further analyze in Section 6.4.3. In CV2, the proposed model demonstrated a notably higher level of gain compared to those observed in other tasks. I attribute this to the use of unnormalized written-style text, which enabled Llama2 to extract precise linguistic information.

The proposed model showed limited improvements on LS-960. I observed the model's tendency to degrade in recognizing "uncommon" words (e.g., names of characters in a book), which

[2]https://huggingface.co/blog/llama2#how-to-prompt-llama-2

Table 6.1: WERs on various ASR tasks, comparing proposed Llama2-based model to hybrid CTC/attention baseline model.

		WER [%] (↓)											
		LibriSpeech-100h				LibriSpeech-960h				TED-LIUM2		CoVoST2	
		Dev		Test		Dev		Test					
Model	**#Beam**	clean	other	clean	other	clean	other	clean	other	Dev	Test	Dev	Test
CTC/Attention	1	9.9	20.6	10.7	21.1	3.2	**6.5**	3.4	**6.7**	11.6	8.8	18.9	21.8
+ Llama2 front-end	1	**6.7**	**17.5**	**7.3**	**17.9**	**2.6**	6.9	**2.8**	7.0	**9.5**	**7.8**	**15.6**	**18.1**
CTC/Attention	20	7.2	17.5	7.5	18.0	**2.3**	**5.7**	**2.6**	**5.7**	9.4	7.8	16.2	18.4
+ Llama2 front-end	20	**6.2**	**16.5**	**6.7**	**16.9**	2.6	6.8	2.8	7.0	**7.6**	**7.2**	**15.0**	**16.9**

Table 6.2: Ablation studies on LibriSpeech-100h task analyzing influence of LLM and prompt designs.

	Dev WER [%] (↓)			
	#Beam $= 1$		#Beam $= 20$	
Method	clean	other	clean	other
CTC/Attention	9.9	20.6	7.2	17.5
+ Llama2 front-end	**6.7**	**17.5**	**6.2**	**16.5**
`A1`: Without Llama2	9.3	20.7	7.1	17.8
`A2`: No prompt	9.3	19.7	6.5	16.7
`A3`: Mismatched prompt	6.8	17.8	6.5	16.8

can be considered as long-tail words rarely seen during the pre-training process of Llama2. This could potentially explain why beam search decoding failed to deliver any performance gains, as Llama2 might be excessively confident in its predictions for common and generic words within its vocabulary. This observation somewhat aligns with the findings presented in BERT-CTC (see Section 5.6.2), highlighting a potential limitation inherent in the use of pre-trained language models. However, with the instruction-tuned LLM, such an issue could be addressed by biasing the model through prompting.

6.4.3 Ablation Study

To validate the effectiveness of the proposed model, I conducted several ablation studies to assess the significance of both the integration of Llama2 and prompt design. Table 6.2 presents the results of the ablation studies, evaluated by WER using the LS-100 development sets.

Table 6.3: WER on LibriSpeech-100h task, comparing proposed approach with other major LLM integration methods.

	WER [%] (↓)			
	Dev		Test	
Integration Method	clean	other	clean	other
CTC/Attention	7.2	17.5	7.5	18.0
+ Shallow fusion	6.4	17.1	7.0	17.5
+ Rescoring	**6.1**	**15.6**	**6.4**	**16.1**
+ Error correction	13.8	22.8	13.4	23.3
+ Front-end	6.2	16.5	6.7	16.9

Importance of Llama2

I ablated Llama2 from the proposed model during the training process (A1); in Step 2 of Section 6.2.2, I trained the decoder from scratch, in the same way as in Step 1, without using Llama2 as its front end. This modified training resulted in improvements on the “clean” set compared to the baseline model. However, there was a slight decline in performance on the “other” set, indicating a decrease in generalizability. With the incorporation of Llama2, the proposed model achieved significantly better results with superior generalization ability.

Influence of prompt

First, I removed the prompt (i.e., the instruction W^{inst} and hypothesis $\hat{W}$) from the Llama2 input (A2). While this modification resulted in a slight improvement compared to the baseline performance, it had a negative impact on the proposed model. I then modify the prompt to focus on a speech translation task instead of grammatical error correction (A3), by setting W^{inst} to “*You will be provided with a statement in quotes, and your task is to translate it into Japanese.*” This greatly improved over the baseline, with a marginal performance degradation relative to the proposed model using the proper prompt. The findings from A2 and A3 suggest that, in the proposed model, a prompt is an essential factor in maximizing the zero-shot learning capability of the LLM to extract helpful linguistic information. Designing a prompt that aligns with the target task can further enhance the model performance.

6.4.4 Comparison and Combination with Previous Methods

Table 6.3 shows WERs on LS-100, comparing the proposed model and other approaches for using an LLM in end-to-end ASR. I performed shallow fusion (Hu et al., 2023) by incorporating the Llama2 probability (from Eq. (6.2)) into the joint CTC and AED decoding process, with the

Table 6.4: WER of proposed model combined with LLM rescoring.

	Test WER [%] (↓)					
	LS-100		LS-960		TED2	CV2
Model	clean	other	clean	other		
CTC/Attention	7.5	18.0	2.6	5.7	7.8	18.4
+ Llama2 rescoring	6.4	16.1	**2.3**	**5.1**	7.3	15.8
+ Llama2 front-end	6.7	16.9	2.8	7.0	7.2	16.9
++ Llama2 rescoring	**5.8**	**15.1**	2.4	6.0	**6.8**	**14.5**

language model weight set at 0.5. I also conducted rescoring (Udagawa et al., 2022) by using the Llama2 probability to rerank top-10 hypotheses obtained from the joint decoding process, where the scores derived from both the joint decoding and rescoring procedures were combined with a weight of 0.5 applied to the Llama2 score. Moreover, I assessed the inherent ability of Llama2 for grammatical error correction (Wu et al., 2023; Fang et al., 2023), directly evaluating the response generated by the prompt with the instruction to improve an ASR hypothesis (see Section 6.4.1). These methods were applied to the inference process of the baseline model. Looking at the results, both shallow fusion and rescoring resulted in notable performance improvements, with rescoring yielding larger gains than the proposed approach. Grammatical error correction appeared to be challenging due to the absence of explicit access to ASR probabilities. This led to the LLM producing hallucinations, resulting in the output of words not present in the input speech.

As the above-mentioned approaches are specifically designed for use during inference, they can complement the proposed model, which integrates the LLM directly into the decoder network. To validate this, I focus on combining the proposed model with the most promising rescoring method. Table 6.4 lists the results on all the tasks, demonstrating that the Llama2-based rescoring further enhanced the performance of the proposed model. The issue in LS-960 was mitigated to some extent, indicating the potential of the proposed model to retain uncommon words during its beam search process.

6.5 Summary

This chapter presented a novel integration of an instruction-tuned LLM and end-to-end ASR. I explored using the zero-shot capability of LLMs to extract linguistic information that can contribute to improving ASR performance. Specifically, an LLM was directed to correct grammatical errors in an ASR hypothesis, and the embedded linguistic knowledge was used to guide the text generation process in an end-to-end ASR model. The proposed model was built on the joint CTC and AED model, where an instruction-tuned LLM (i.e., Llama2) was employed as a front-end of the decoder network. An ASR hypothesis, subject to correction, was obtained from the encoder via CTC decoding, which was then fed into the LLM along with an instruction. The decoder network subsequently took as input the LLM embeddings to perform sequence generation, incorporating acoustic information from the encoder output. Experimental results and analyses demonstrated that the proposed integration yields promising performance improvements. Additionally, the performance could be further enhanced through the combination with LLM-based rescoring.

7 Conclusions

In this dissertation, I proposed approaches for advancing end-to-end ASR by incorporating top-down linguistic cues into the direct speech-to-text conversion process. This chapter summarizes this dissertation by reviewing the key content presented in each chapter and highlighting the major contribution of my doctoral research. I also mention the limitations and possible future directions for further studies.

7.1 Summary of the Dissertation

In Chapter 2, I provided a comprehensive overview of the core principles of end-to-end ASR. Initially, traditional hybrid ASR systems were explained to better understand the nuances of end-to-end systems. I then compared the major modeling approaches for end-to-end ASR, including the CTC, transducer, and AED models, focusing on the distinctions in their probabilistic formulations. Lastly, I reviewed neural network architectures that are commonly used for constructing end-to-end ASR models.

In Chapter 3, I presented hierarchical modeling methods to enhance the representation learning capabilities of end-to-end ASR models. Firstly, HC-CTC was proposed as a foundational approach for progressively increasing the granularity of output linguistic units, aiming to learn representations adept at predicting sparse word-level units. The HC-CTC model was trained by multiple CTC losses applied to intermediate encoder layers, where the subword vocabulary size of each target sequence gradually increased as the layers progressed closer to the word-level output. During this training process, each level of sequence prediction was explicitly conditioned on the sequences predicted at the lower levels. This encouraged the model to exploit a hierarchy of linguistic units to extract effective linguistic representations. Subsequently, I made an architectural improvement to HC-CTC by implementing a refinement mechanism for each stage of intermediate prediction. Specifically, the Transformer-based encoder layers were augmented with recursive operations, which involved repeatedly using shared model layers to refine intermediate representations. This enhancement allowed the model to make more precise predictions, especially at the lower levels, thereby improving its overall performance. Lastly, I developed an efficient semi-supervised learning method for improving the end-to-end ASR models trained with intermediate CTC losses. The proposed approach, InterMPL, extended the functionality of MPL by enabling it to utilize multiple pseudo-labels obtained from intermediate predictions. With HC-CTC, the pseudo-labels were generated at varying levels of granularity, which helped the model learn linguistic information from unlabeled audio-only data. Through a series of experiments, the effectiveness of adopting the hierarchical structure for end-to-end ASR was validated, which proved particularly beneficial for learning linguistic representations that are conducive to making sparse word-level predictions.

In Chapter 4, I developed approaches for integrating the masked language modeling mechanism into end-to-end ASR models, aiming to effectively incorporate long-range linguistic contexts. To this end, I first proposed Mask-CTC, a non-autoregressive end-to-end ASR model trained by the joint CTC and CMLM objective. Mask-CTC realized fast inference by generating the initial sequence rapidly via greedy CTC decoding and replacing low-confidence tokens with the mask token. This masked sequence was then fed into the CMLM decoder, where the masked tokens were predicted in parallel by considering both the input speech and the linguistic context provided by other unmasked tokens. Subsequently, I further explored the effective use of contextualized linguistic representations for improving end-to-end ASR performance, proposing Mask-Conformer. Mask-Conformer employed the CMLM decoder to explicitly condition the speech encoding process on linguistic information. This was achieved by augmenting the Conformer encoder architecture with the cross-attention mechanism, which enabled the integration of CMLM decoder outputs into the encoder's hidden states. Lastly, I proposed to use Mask-CTC for pre-training streaming end-to-end ASR models. In Mask-CTC, the CMLM decoder served to enhance the encoder network by providing contextual linguistic information, leading to improved performance of CTC. Such ability to capture long-term linguistic context, including information from future contexts, was expected to aid the training of streaming ASR, enabling it to effectively anticipate future information. By initializing an encoder network of a streaming end-to-end ASR model with a pre-trained Mask-CTC encoder, the inherent capabilities of Mask-CTC were implicitly leveraged to optimize the streaming objective. The experiments conducted for each approach demonstrated their effectiveness convincingly. They consistently showed that masked language modeling is beneficial in supplying contextual linguistic information to end-to-end ASR models, providing effective guidance for generating accurate text sequences.

In Chapter 5, I introduced a novel approach to formulating end-to-end ASR, utilizing pre-trained masked language models to facilitate the extraction of linguistic information. The proposed models, BERT-CTC and BECTRA, were specifically designed to effectively integrate pre-trained language models (e.g., BERT) into end-to-end ASR models. BERT-CTC adapted BERT for CTC by addressing the constraint of the conditional independence assumption between output tokens. This enabled explicit conditioning of BERT's contextualized embeddings in the speech-to-text conversion process, seamlessly merging audio and linguistic information through an iterative refinement algorithm. BECTRA extended BERT-CTC to the transducer framework and trained the prediction and joint networks using a vocabulary suitable for ASR training. This aimed to bridge the gap between the text processed in end-to-end ASR and BERT, as these models have distinct vocabularies with varying text formats and styles, such as the presence of punctuation and the distinction of casing. The experimental results on various ASR tasks demonstrated that the proposed models improve over both the CTC and transducer-based baselines, thanks to the incorporation of versatile linguistic knowledge obtained from BERT. I also showed that BERT-CTC provides semantic representations advantageous for addressing SLU tasks, enabling the integration and utilization of both audio and linguistic information. The comprehensive analysis and investigation of various aspects verified the effectiveness of the proposed formulations and architectural designs.

In Chapter 6, I presented an advanced method for integrating an instruction-tuned LLM with end-to-end ASR. I focused on harnessing the zero-shot capability of LLMs to extract linguistic information that can contribute to improving ASR performance. Specifically, an LLM was directed to correct grammatical errors in an ASR hypothesis through prompting, and the embedded linguistic knowledge was used to guide the text generation process in an end-to-end ASR model. The proposed model was built on the joint CTC and AED model, where an instruction-tuned LLM (i.e., Llama2) was employed as a front-end of the decoder network. An ASR hypothesis, subject to correction, was obtained from the encoder via CTC decoding, which was then fed into the LLM along with an instruction. The decoder network subsequently took as input the LLM embeddings to perform sequence generation, incorporating acoustic information from the encoder output. The experimental results and analyses revealed that the proposed integration leads to promising performance improvements by adeptly orienting the LLM towards the grammatical error correction task. Additionally, the performance could be further enhanced through the combination with LLM-based rescoring.

7.2 Summary of Contribution

This dissertation presented a series of studies that offer novel perspectives on the extraction and management of linguistic information during the process of direct speech-to-text conversion. Just as in the role of linguistic knowledge in human speech perception, speech and text do not neces-

Table 7.1: Comparison of proposed models in relation to design of latent linguistic sequence.

	Model	**Latent linguistic sequence $\tilde{W}$**
Chap. 3	HC-CTC (§3.2)	A sequence predicted with a finer granularity of linguistic units.
Chap. 4	Mask-CTC (§4.2) Mask-Conformer (§4.3)	A sequence in which low-confidence tokens are masked.
Chap. 5	BERT-CTC (§5.2.1) BECTRA (§5.2.2)	A sequence in which low-confidence tokens are masked, with contextual embeddings derived from a pre-trained masked language model.
Chap. 6	CTC/AED+LLM (§6.2)	A hypothesized sequence accompanied by an LLM's linguistic information aimed at correcting grammatical errors.

sarily maintain a sequential or hierarchical relationship in ASR, and I have demonstrated that the incorporation of top-down linguistic cues contributes to enhancing the overall performance and functionality of end-to-end ASR models. These linguistic cues were effectively designed through: i) a structured approach to constructing subword units, reflecting the fine-to-coarse processing inherent in ASR; ii) the employment of masked language modeling to capture semantic and contextual linguistic information; and iii) the utilization of pre-trained language models, which provide rich and versatile linguistic knowledge, with a focus on masked language models and instruction-tuned LLMs. Each of the proposed approaches was fundamentally grounded in the concept of bottom-up and top-down processing as outlined in Chapter 1 (specifically referenced in Eq. (1.2)). All of the approaches modeled bottom-up processing primarily based on the CTC framework, generating a hypothesized sequence that relied exclusively on the speech input. The key distinction lies in the design of top-down processing, as summarized in Table 7.1, where varying types of latent linguistic sequences ($\tilde{W}$ in Eq. (1.2)) were explored to condition end-to-end ASR models with effective linguistic information. In Chapter 3, HC-CTC predicted a latent sequence using lower-granularity linguistic units, which were used to effectively guide the subsequent predictions toward a higher and sparser level. In Chapter 4, Mask-CTC and Mask-Conformer introduced a latent sequence comprising partially masked tokens, utilizing tokens with high confidence to repredict those with low confidence, thereby facilitating the use of contextual information. Similarly, in Chapter 5, BERT-CTC and BECTRA employed the masked sequence to derive contextualized embeddings from pre-trained masked language models, which aided the model in generating accurate sequences. In Chapter 6, an ASR hypothesis is fed into an LLM for grammatical error correction, and the output embeddings from the LLM were then used to assist the text generation process within the decoder network of the joint CTC and AED model.

I believe that this dissertation delivers significant insights into the handling of linguistic information within end-to-end ASR, uncovering opportunities to enhance recognition accuracy and extend the applicability of end-to-end ASR models in tackling more linguistically challenging tasks.

7.3 Limitations and Future Directions

7.3.1 Assessment of Scalability with Increased Data Sizes

The experiments conducted in this dissertation were in a relatively low-resource setting (less than 1000 hours), compared to the continuously growing resources accompanying the increasing scale of recent ASR datasets (Chen et al., 2021a; Ando and Fujihara, 2021; Zhang et al., 2022a), which involve over 10k hours of paired speech and text data. Future research should focus on assessing whether the proposed models demonstrate similar improvements when trained on these extensive datasets. These datasets at scale contain ample text data that can facilitate the acquisition of essential linguistic information for achieving end-to-end ASR, without the need for additional training techniques or complex model architectures (Radford et al., 2023). Nonetheless, ASR datasets are likely to remain lower in resources compared to text-only datasets constructed for training modern LLMs. Consequently, I am convinced that there will always be room for improvement through the effective integration of linguistic cues into both end-to-end ASR and, further, into end-to-end SLU models.

My research primarily focused on the textual aspect of end-to-end ASR models, emphasizing the significance of using pre-trained language models, as detailed in Chapters 5 and 6. Additionally, the use of pre-trained acoustic models may offer avenues for further improvements. While constructing a paired ASR dataset involves substantial annotation costs, obtaining audio-only data can be achieved more easily and efficiently on a larger scale (Kahn et al., 2020b). This audio-only data is highly beneficial for learning robust acoustic features through self-supervised pre-training, which has been shown to boost the performance of end-to-end ASR models (Baevski et al., 2020; Hsu et al., 2021; Chung et al., 2021a; Chen et al., 2022a). Integrating such a pre-trained acoustic model into the encoder network can be a promising advancement, enabling the proposed models to benefit from both audio and text-only data.

7.3.2 Enhancement for Streaming Application

The models proposed in this dissertation have limitations in scenarios involving online streaming, where output tokens need to be predicted synchronously with sequential speech input. Specifically, these models were designed to utilize the full context of the output token sequence, aiming to effectively extract and integrate linguistic information that contributes to improving end-to-end ASR. It is not particularly problematic for utterance-level ASR tasks. However, in real-time applications like spoken dialogue systems, which require immediate interaction, the proposed models face challenges. A viable solution to overcoming this limitation is to develop a two-pass streaming system (Sainath et al., 2019), where a single model performs streaming ASR in the first pass, followed by offline ASR in the second pass for refining the initial output. The proposed models adopt similar two-pass decoding strategies that combine bottom-up and top-down modeling, and the bottom-up process can be implemented as the streaming first pass, employing advanced techniques such as time-restricted attention masking (Povey et al., 2018; Zhang et al., 2020a; Moritz et al., 2020; Chen et al., 2021d), and block-wise processing (Tsunoo et al., 2019; Wang et al., 2021b).

A
Corpus Details

A.1 Wall Street Journal (WSJ)

WSJ[1] (Paul and Baker, 1992) consists of read English texts sourced from Wall Street Journal news articles. The training set (*si284*) contains 80 hours of utterances. The official development (*dev93*) and evaluation (*eval92*) sets were used for tuning hyper-parameters and evaluating performance, respectively. Data preparation for this corpus was done using the recipes provided by the Kaldi[2] and ESPnet[3] toolkits. See Table A.1 for the detailed statistics.

A.2 VoxForge

VoxForge[4] is an open speech dataset created to support the development and testing of ASR systems. This corpus encompasses multiple languages, of which I utilized solely the 20-hour Italian subset for my experiments. The construction of the training, development, and evaluation splits adhered to the data preparation procedures provided by the ESPnet recipe, which resulted in 16, 2, and 2 hours of data, respectively. Data preparation for this corpus was done using the recipes provided by the Kaldi[5] and ESPnet[6] toolkits. See Table A.2 for the detailed statistics.

[1] LDC93S6B, LDC94S13B
[2] `https://github.com/kaldi-asr/kaldi/tree/master/egs/wsj`
[3] `https://github.com/espnet/espnet/tree/master/egs/wsj`
[4] `https://www.voxforge.org`
[5] `https://github.com/kaldi-asr/kaldi/tree/master/egs/voxforge`
[6] `https://github.com/espnet/espnet/tree/master/egs/voxforge`

Table A.1: Dataset description of WSJ.

Data Split		**#hours**	**#utterances**
Training	WSJ1 si284	80	37k
Development	dev93	1.1	503
Evaluation	eval92	0.7	333

Table A.2: Dataset description of VoxForge Italian.

Data Split		**#hours**	**#utterances**
Training	Train	16	8417
Development	Dev	2.0	1082
Evaluation	Test	2.0	1055

A.3 Corpus of Spontaneous Japanese (CSJ)

CSJ (Maekawa, 2003) is a Japanese ASR corpus that contains approximately 520 hours of academic lecture speech. The development set was constructed using the first 4k utterances in the training set, which follows the data preparation procedures used in the Kaldi and ESPnet recipes. The official evaluation sets, namely *eval1*, *eval2*, and *eval3*, were used to evaluate performance. See Table A.3 for the detailed statistics. Data preparation for this corpus was done using the recipe provided by the ESPnet[7] toolkit.

A.4 TED-LIUM

TED-LIUM is an ASR task consisting of English TED Talks[8]. There have been three releases of this corpus, each progressively increasing the amount of training data; here, the second and third versions are described. The second release, or TED-LIUM2 (Rousseau et al., 2014), contains 207 hours of training data. The third release, or TED-LIUM3 (Hernandez et al., 2018), contains 452 hours of training data. Both versions contain the same development and evaluation sets provided by TED-LIUM2, which were used for tuning hyper-parameters and evaluating performance, respectively. See Table A.4 for the detailed statistics. Data preparation for these corpora was done using the recipes provided by the Kaldi[9] and/or ESPnet[10] toolkit.

[7] https://github.com/espnet/espnet/tree/master/egs2/csj
[8] https://www.ted.com
[9] https://github.com/kaldi-asr/kaldi/tree/master/egs/tedlium
[10] https://github.com/espnet/espnet/tree/master/{egs,egs2}/{tedlium2,tedlium3}

Table A.3: Dataset description of CSJ.

Data Split		**#hours**	**#utterances**
Training	Train	420	403k
Development	Dev	6.5	4000
Evaluation	eval1	1.8	1272
	eval2	1.9	1292
	eval3	1.3	1385

Table A.4: Dataset description of TED-LIUM.

Data Split		**#hours**	**#utterances**
Training	Train (v2)	207	93k
	Train (v3)	452	268k
Development	Dev	1.6	507
Evaluation	Test	2.6	1155

A.5 AISHELL-1

AISHELL-1 (Bu et al., 2017) is a multi-domain Mandarin ASR corpus that covers a range of common applications, such as voice control for smart speakers. The training set consists of 170 hours of utterances. The official development and evaluation sets were used for tuning hyper-parameters and evaluating performance, respectively. Data preparation for this corpus was done using the recipe provided by the ESPnet[11] toolkit. See Table A.5 for the detailed statistics.

A.6 LibriSpeech

LibriSpeech (Panayotov et al., 2015) consists of utterances derived from read English audiobooks that are part of the LibriVox project[12]. The training set contains 960 hours of data, which can be divided into *train-clean-100*, *train-clean-360*, and *train-other-500*, each varying in size and data quality (i.e., "clean" and "other"). The official development (*dev-clean* and *dev-other*) and evaluation sets (*test-clean* and *test-other*) were used for tuning hyper-parameters and evaluating performance, respectively. For training external language models, the text-only data with 803M words was used. Data preparation for this corpus was done using the recipes provided by the

[11]https://github.com/espnet/espnet/tree/master/egs2/aishell
[12]https://librivox.org

Table A.5: Dataset description of AISHELL-1.

Data Split		**#hours**	**#utterances**
Training	Train	150	120k
Development	Dev	10	14326
Evaluation	Test	5	7176

Table A.6: Dataset description of LibriSpeech.

Data Split		**#hours**	**#utterances**
Training	Train	960	281k
	train-clean-100	100	28k
	train-clean-360	360	104k
	train-other-500	500	149k
	Train (LM)	–	40M
Development	dev-clean	5.4	2703
	dev-other	5.1	2864
Evaluation	test-clean	5.4	2620
	test-other	5.3	2939

Kaldi[13] and/or ESPnet[14] toolkit. See Table A.6 for the detailed statistics.

A.7 Libri-Light

Libri-Light (Kahn et al., 2020b) is another collection of spoken English audio, which is assembled from open-source audiobooks derived from the LibriVox project. The primary objective of this corpus is to assess ASR systems under a setting with limited or no supervision. Libri-Light offers 60k hours of large-scale unlabeled speech, along with a small training set (ranging from 10 hours, 1 hour, or 10 minutes) for limited supervision. I mainly used the limited training sets for the experiments conducted in this dissertation. The development and evaluation sets are identical to those from LibriSpeech. Data preparation for this corpus was done using the recipe provided by the ESPnet[15] toolkit. See Table A.7 for the detailed statistics.

Table A.7: Dataset description of Libri-Light. †Include six different versions.

Data Split		#hours	#utterances
Training	train-10h	10	2763
	train-1h	1	286
	train-10m†	10min	{48, 46, 45, 46, 49, 52}

Table A.8: Dataset description of CoVoST2 (En→X).

Data Split		#hours	#utterances
Training	Train	407	272k
Development	Dev	22	13k
Evaluation	Test	25	16k

A.8 CoVoST2

CoVoST2 (Wang et al., 2021a) is a corpus designed for speech translation tasks, derived from the Common Voice project[16] (Ardila et al., 2020). Primarily, it was used as an English ASR task by exclusively extracting source speech-text data from the "En→X" task, resulting in 430 hours of training data. The unnormalized transcriptions provided by CoVoST2 were used for training models in a setting that preserved punctuation and casing. Data preparation for this corpus was done using the recipe provided by the ESPnet[17] toolkit. See Table A.8 for the detailed statistics.

A.9 SpeechStew

SpeechStew (Chan et al., 2021) is a multi-domain ASR task that combines public English corpora, including AMI (Carletta et al., 2005), Common Voice (Ardila et al., 2020), English Broadcast News[18], LibriSpeech (Panayotov et al., 2015), Switchboard/Fisher[19] (Godfrey et al., 1992), TED-LIUM3 (Hernandez et al., 2018), and WSJ. The training set contains approximately 5,140 hours of data. Each corpus provides official development and evaluation sets, all of which were considered for tuning hyper-parameters and evaluating overall performance, respectively.

[13] `https://github.com/kaldi-asr/kaldi/tree/master/egs/librispeech`
[14] `https://github.com/espnet/espnet/tree/master/{egs,egs2}/librispeech`
[15] `https://github.com/espnet/espnet/tree/master/egs2/librilight_limited`
[16] `https://commonvoice.mozilla.org`
[17] `https://github.com/espnet/espnet/tree/master/egs2/covost2`
[18] LDC97S44, LDC97T22, LDC98S71, LDC98T28
[19] LDC2004T19, LDC2005T19, LDC2004S13, LDC2005S13, LDC97S62

Table A.9: Dataset description of Fisher-CallHome Spanish.

Data Split		**#hours**	**#utterances**
Training	Train	171	139k
Development	Fisher-dev	4.6	3937
	Fisher-dev2	4.7	3957
	CallHome-devtest	4.5	3956
Evaluation	Fisher-test	4.7	3638
	CallHome-evltest	1.8	1825

Table A.10: Dataset description of SLURP.

Data Split		**#hours**	**#utterances**
Training	Train	83	12k
Development	Dev	6.9	2033
Evaluation	Test	10.3	2974

A.10 Fisher-CallHome Spanish

The Fisher and CallHome Spanish-English Speech Translation Corpus[20] (Post et al., 2013) is a corpus designed for speech translation tasks, featuring conversational telephone speech with the corresponding transcriptions in both Spanish and English. The training set consists of 171 hours of utterances. The official development (*Fisher-*{*dev, dev2*} and *CallHome-devtest*) and evaluation (*Fisher-test* and *CallHome-evltest*) sets were used for tuning hyper-parameters and evaluating performance, respectively. The translation performance was evaluated using the case-sensitive detokenized BLEU. All punctuation marks, except for apostrophes, were removed from both transcriptions and translations, following the standard practice of this corpus. Data preparation for this corpus was done using the recipe provided by the ESPnet-ST[21] toolkit. See Table A.9 for the detailed statistics.

A.11 SLURP

SLURP (Bastianelli et al., 2020) is a corpus designed for spoken language understanding tasks, involving single-turn conversations between a user and a home assistant. These conversations are annotated with intent and entities, in addition to ASR transcriptions. I mainly focused on the intent

[20]LDC2014T23

[21]`https://github.com/espnet/espnet/tree/master/egs/fisher_callhome_spanish/st1`

classification task, which comprises 69 distinct classes. In the experiments, I used the 40-hour training set, augmented with an additional 43 hours of official synthetic data. Data preparation for this corpus was done using the recipe provided by the ESPnet-SLU[22] toolkit. See Table A.10 for the detailed statistics.

[22]https://github.com/espnet/espnet/tree/master/egs2/slurp/slu1

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Shintaro Ando and Hiromasa Fujihara. Construction of a large-scale Japanese ASR corpus on TV recordings. In *Proceedings of ICASSP*, pages 6948–6952, 2021.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. In *Proceedings of LREC*, pages 4218–4222, 2020.

Siddhant Arora, Siddharth Dalmia, Pavel Denisov, Xuankai Chang, Yushi Ueda, Yifan Peng, Yuekai Zhang, Sujay Kumar, Karthik Ganesan, Brian Yan, Ngoc Thang Vu, Alan W Black, and Shinji Watanabe. ESPnet-SLU: Advancing spoken language understanding through ESPnet. In *Proceedings of ICASSP*, pages 7167–7171, 2022.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *Proceedings of ICLR*, 2019.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Proceedings of NeurIPS*, pages 12449–12460, 2020.

Tom Bagby, Kanishka Rao, and Khe Chai Sim. Efficient implementation of recurrent neural network transducer in tensorflow. In *Proceedings of SLT*, pages 506–512, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*, 2014.

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

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Proceedings of NeurIPS*, pages 688–699, 2019.

Ye Bai, Jiangyan Yi, Jianhua Tao, Zhengkun Tian, Zhengqi Wen, and Shuai Zhang. Fast end-to-end speech recognition via non-autoregressive models and cross-modal knowledge transferring from BERT. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1897–1911, 2021.

Jeong-Uk Bang, Min-Kyu Lee, Seung Yun, and Sang-Hun Kim. Improving end-to-end speech translation model with BERT-based contextual information. In *Proceedings of ICASSP*, pages 6227–6231, 2022.

Murali Karthick Baskar, Andrew Rosenberg, Bhuvana Ramabhadran, Yu Zhang, and Pedro Moreno. Ask2Mask: Guided data selection for masked speech modeling. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1357–1366, 2022.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. SLURP: A spoken language understanding resource package. In *Proceedings of EMNLP*, pages 7252–7262, 2020.

Yonatan Belinkov and James Glass. Analyzing hidden representations in end-to-end automatic speech recognition systems. In *Proceedings of NeurIPS*, pages 2441–2451, 2017.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of NeurIPS*, pages 1171–1179, 2015.

Rami Botros, Tara N. Sainath, Robert David, Emmanuel Guzman, Wei Li, and Yanzhang He. Tied & reduced RNN-T decoder. In *Proceedings of Interspeech*, pages 4563–4567, 2021.

Herve A. Bourlard and Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, 1994.

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. In *Proceedings of ASRU*, pages 16–23, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proceedings of NeurIPS*, pages 1877–1901, 2020.

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. AISHELL-1: An open-source Mandarin speech corpus and a speech recognition baseline. In *Proceedings of O-COCOSDA*, pages 1–5, 2017.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. The AMI meeting corpus: A pre-announcement. In *International workshop on machine learning for multimodal interaction*, pages 28–39, 2005.

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

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. KERMIT: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*, 2019.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. Imputer: Sequence modelling via imputation and dynamic programming. In *Proceedings of ICML*, pages 1403–1413, 2020.

William Chan, Daniel Park, Chris Lee, Yu Zhang, Quoc Le, and Mohammad Norouzi. SpeechStew: Simply mix all available speech recognition data to train one large neural network. *arXiv preprint arXiv:2104.02133*, 2021.

Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, et al. GigaSpeech: An evolving, multi-domain ASR corpus with 10,000 hours of transcribed audio. *arXiv preprint arXiv:2106.06909*, 2021a.

Nanxin Chen, Shinji Watanabe, Jesús Villalba, Piotr Żelasko, and Najim Dehak. Non-autoregressive Transformer for speech recognition. *IEEE Signal Processing Letters*, 28:121–125, 2021b.

Nanxin Chen, Piotr Żelasko, Laureano Moro-Velázquez, Jesús Villalba, and Najim Dehak. Align-Denoise: Single-pass non-autoregressive speech recognition. In *Proceedings of Interspeech*, pages 3770–3774, 2021c.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. WavLM: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022a.

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. Developing real-time streaming Transformer transducer for speech recognition on large-scale dataset. In *Proceedings of ICASSP*, pages 5904–5908, 2021d.

Yang Chen, Weiran Wang, and Chao Wang. Semi-supervised ASR by end-to-end self-training. In *Proceedings of Interspeech*, pages 2787–2791, 2020.

Zhehuai Chen, Yu Zhang, Andrew Rosenberg, Bhuvana Ramabhadran, Pedro J. Moreno, Ankur Bapna, and Heiga Zen. MAESTRO: Matched speech text representations through modality matching. In *Proceedings of Interspeech*, pages 4093–4097, 2022b.

Ethan A Chi, Julian Salazar, and Katrin Kirchhoff. Align-Refine: Non-autoregressive speech recognition via iterative realignment. In *Proceedings of NAACL-HLT*, pages 1920–1927, 2021a.

Po-Han Chi, Pei-Hung Chung, Tsung-Han Wu, Chun-Cheng Hsieh, Yen-Hao Chen, Shang-Wen Li, and Hung-yi Lee. Audio ALBERT: A lite BERT for self-supervised learning of audio representation. In *Proceedings of SLT*, pages 344–350, 2021b.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *Proceedings of ICASSP*, pages 4774–4778, 2018.

Shih-Hsuan Chiu and Berlin Chen. Innovative BERT-based reranking language models for speech recognition. In *Proceedings of SLT*, pages 266–271, 2021.

Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. In *Proceedings of Interspeech*, pages 523–527, 2017.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Proceedings of NeurIPS*, pages 577–585, 2015.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Yung-Sung Chuang, Chi-Liang Liu, Hung-yi Lee, and Lin-shan Lee. SpeechBERT: An audio-and-text jointly learned language model for end-to-end spoken question answering. In *Proceedings of Interspeech*, pages 4168–4172, 2020.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. w2v-BERT: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *Proceedings of ASRU*, pages 244–250, 2021a.

Yu-An Chung, Chenguang Zhu, and Michael Zeng. SPLAT: Speech-language joint pre-training for spoken language understanding. In *Proceedings of NAACL-HLT*, pages 1897–1907, 2021b.

CMUdict. The CMU pronouncing dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`, 1993. [Online; Accessed on January-1-2024].

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2011.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of ACL*, pages 2978–2988, 2019.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of ICML*, pages 933–941, 2017.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal Transformers. In *Proceedings of ICLR*, 2019.

Keqi Deng, Songjun Cao, Yike Zhang, and Long Ma. Improving hybrid CTC/attention end-to-end speech recognition with pretrained acoustic and language models. In *Proceedings of ASRU*, pages 76–82, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

Linhao Dong, Shuang Xu, and Bo Xu. Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *Proceedings of ICASSP*, pages 5884–5888, 2018.

Linhao Dong, Feng Wang, and Bo Xu. Self-attention aligner: A latency-control end-to-end model for ASR using self-attention network and chunk-hopping. In *Proceedings of ICASSP*, pages 5656–5660, 2019.

Ruchao Fan, Guoli Ye, Yashesh Gaur, and Jinyu Li. Acoustic-aware non-autoregressive spell correction with mask sample decoding. *arXiv preprint arXiv:2210.08665*, 2022.

Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. Is ChatGPT a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746*, 2023.

William Fedus, Ian Goodfellow, and Andrew M Dai. MaskGAN: Better text generation via filling in the _. In *Proceedings of ICLR*, 2018.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proceedings of IJCAI*, pages 774–779, 2007.

Yuya Fujita, Shinji Watanabe, Motoi Omachi, and Xuankai Chang. Insertion-based modeling for end-to-end automatic speech recognition. In *Proceedings of Interspeech*, pages 3660–3664, 2020.

Hayato Futami, Hirofumi Inaguma, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara. Distilling the knowledge of BERT for sequence-to-sequence ASR. In *Proceedings of Interspeech*, pages 3635–3639, 2020.

Hayato Futami, Hirofumi Inaguma, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara. ASR rescoring and confidence estimation with ELECTRA. In *Proceedings of ASRU*, pages 380–387, 2021.

Hayato Futami, Hirofumi Inaguma, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara. Non-autoregressive error correction for CTC-based ASR with phone-conditioned masked LM. In *Proceedings of Interspeech*, pages 3889–3893, 2022.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of ICML*, pages 1050–1059, 2016.

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

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

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

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arXiv:2001.08785*, 2020.

Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein. RNN-transducer with stateless prediction network. In *Proceedings of ICASSP*, pages 7049–7053, 2020.

John J Godfrey, Edward C Holliman, and Jane McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, pages 517–520, 1992.

Alex Graves. Sequence transduction with recurrent neural networks. In *Proceedings of ICML Representation Learning Workshop*, 2012.

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

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

Alex Graves, Abdelrahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, pages 6645–6649, 2013.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. In *Proceedings of ICLR*, 2018.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein Transformer. In *Proceedings of NeurIPS*, pages 11181–11191, 2019.

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 *Proceedings of Interspeech*, pages 5036–5040, 2020.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.

Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. Incorporating BERT into parallel sequence decoding with adapters. In *Proceedings of NeurIPS*, pages 10843–10854, 2020.

Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, et al. Recent developments on ESPnet toolkit boosted by Conformer. In *Proceedings of ICASSP*, pages 5874–5878, 2021.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. ContextNet: Improving convolutional neural networks for automatic speech recognition with global context. In *Proceedings of Interspeech*, pages 3610–3614, 2020.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014a.

Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. *arXiv preprint arXiv:1408.2873*, 2014b.

Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Shubham Toshniwal, and Karen Livescu. Pre-trained text embeddings for enhanced text-to-speech synthesis. In *Proceedings of Interspeech*, pages 4430–4434, 2019.

Abdelwahab Heba, Thomas Pellegrini, Jean-Pierre Lorré, and Régine Andre-Obrecht. Char+CV-CTC: Combining graphemes and consonant/vowel units for CTC-based ASR using multitask learning. In *Proceedings of Interspeech*, pages 1611–1615, 2019.

François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève. TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation. In *Proceedings of SPECOM*, pages 198–208, 2018.

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 *Proceedings of Interspeech*, pages 3655–3659, 2020.

Yosuke Higuchi, Nanxin Chen, Yuya Fujita, Hirofumi Inaguma, Tatsuya Komatsu, Jaesong Lee, Jumon Nozaki, Tianzi Wang, and Shinji Watanabe. A comparative study on non-autoregressive modelings for speech-to-text generation. In *Proceedings of ASRU*, pages 47–54, 2021a.

Yosuke Higuchi, Hirofumi Inaguma, Shinji Watanabe, Tetsuji Ogawa, and Tetsunori Kobayashi. Improved Mask-CTC for non-autoregressive end-to-end ASR. In *Proceedings of ICASSP*, pages 8363–8367, 2021b.

Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. Momentum pseudo-labeling for semi-supervised speech recognition. In *Proceedings of Interspeech*, pages 726–730, 2021c.

Yosuke Higuchi, Keita Karube, Tetsuji Ogawa, and Tetsunori Kobayashi. Hierarchical conditional end-to-end ASR with CTC and multi-granular subword units. In *Proceedings of ICASSP*, pages 7797–7801, 2022a.

Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. Advancing momentum pseudo-labeling with Conformer and initialization strategy. In *Proceedings of ICASSP*, pages 7672–7676, 2022b.

Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. Momentum pseudo-labeling: Semi-supervised ASR with continuously improving pseudo-labels. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1424–1438, 2022c.

Yosuke Higuchi, Brian Yan, Siddhant Arora, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. BERT meets CTC: New formulation of end-to-end speech recognition with pre-trained masked language model. In *Findings of EMNLP*, pages 5486–5503, 2022d.

Yosuke Higuchi, Tetsuji Ogawa, and Tetsunori Kobayashi. Harnessing the zero-shot power of instruction-tuned large language model in end-to-end speech recognition. *arXiv preprint arXiv:2309.10524*, 2023a.

Yosuke Higuchi, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. BECTRA: Transducer-based end-to-end ASR with BERT-enhanced encoder. In *Proceedings of ICASSP*, 2023b.

Yosuke Higuchi, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. InterMPL: Momentum pseudo-labeling with intermediate CTC loss. In *Proceedings of ICASSP*, 2023c.

Yosuke Higuchi, Andrew Rosenberg, Yuan Wang, Murali Karthick Baskar, and Bhuvana Ramabhadran. Mask-Conformer: Augmenting Conformer with mask-predict decoder. In *Proceedings of ASRU*, 2023d.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Takaaki Hori, Shinji Watanabe, and John R Hershey. Joint CTC/attention decoding for end-to-end speech recognition. In *Proceedings of ACL*, pages 518–529, 2017a.

Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan. Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. In *Proceedings of Interspeech*, pages 949–953, 2017b.

Wei-Ning Hsu, Ann Lee, Gabriel Synnaeve, and Awni Hannun. Semi-supervised speech recognition via local prior matching. *arXiv preprint arXiv:2002.10336*, 2020.

Wei-Ning Hsu, Yao-Hung Hubert Tsai, Benjamin Bolte, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: How much can a bad teacher benefit ASR pre-training? In *Proceedings of ICASSP*, pages 6533–6537, 2021.

Ke Hu, Tara N Sainath, Ruoming Pang, and Rohit Prabhavalkar. Deliberation model based two-pass end-to-end speech recognition. In *Proceedings of ICASSP*, pages 7799–7803, 2020.

Ke Hu, Tara N Sainath, Bo Li, Nan Du, Yanping Huang, Andrew M Dai, Yu Zhang, Rodrigo Cabrera, Zhifeng Chen, and Trevor Strohman. Massively multilingual shallow fusion with large language models. In *Proceedings of ICASSP*, 2023.

Wen-Chin Huang, Chia-Hua Wu, Shang-Bao Luo, Kuan-Yu Chen, Hsin-Min Wang, and Tomoki Toda. Speech recognition by simply fine-tuning BERT. In *Proceedings of ICASSP*, pages 7343–7347, 2021.

Kyuyeon Hwang and Wonyong Sung. Character-level language modeling with hierarchical recurrent neural networks. In *Proceedings of ICASSP*, pages 5720–5724, 2017.

Hirofumi Inaguma and Tatsuya Kawahara. Alignment knowledge distillation for online streaming attention-based speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1371–1385, 2021.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe. ESPnet-ST: All-in-one speech translation toolkit. In *Proceedings of ACL: System Demonstrations*, pages 302–311, 2020.

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Proceedings of NeurIPS*, pages 1942–1950, 2017.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Language modeling with deep Transformers. In *Proceedings of Interspeech*, pages 3905–3909, 2019.

Frederick Jelinek. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of Workshop on Pattern Recognition in Practice, 1980*, 1980.

Frederick Jelinek. *Statistical methods for speech recognition*. MIT press, 1998.

Jae-Jin Jeon and Eesung Kim. Multitask learning and joint optimization for Transformer-RNN-transducer speech recognition. In *Proceedings of ICASSP*, pages 6793–6797, 2021.

Kam-Chuen Jim, C Lee Giles, and Bill G Horne. An analysis of noise in recurrent neural networks: convergence and generalization. *IEEE Transactions on neural networks*, 7(6):1424–1438, 1996.

B-H Juang. Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains. *AT&T technical journal*, 64(6):1235–1249, 1985.

Jacob Kahn, Ann Lee, and Awni Hannun. Self-training for end-to-end speech recognition. In *Proceedings of ICASSP*, pages 7084–7088, 2020a.

Jacob Kahn, Morgane Rivière, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, et al. Libri-Light: A benchmark for ASR with limited or no supervision. In *Proceedings of ICASSP*, pages 7669–7673, 2020b.

Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar. An analysis of incorporating an external language model into a sequence-to-sequence model. In *Proceedings of ICASSP*, pages 5824–5828, 2018.

Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on Transformer vs RNN in speech applications. In *Proceedings of ASRU*, pages 449–456, 2019.

Tom Kenter, Manish Kumar Sharma, and Rob Clark. Improving the prosody of RNN-based english text-to-speech synthesis by incorporating a BERT model. In *Proceedings of Interspeech*, pages 4412–4416, 2020.

Juntae Kim and Jeehye Lee. Generalizing RNN-transducer to out-domain audio via sparse self-attention layers. In *Proceedings of Interspeech*, pages 4123–4127, 2022.

Kwangyoun Kim, Felix Wu, Yifan Peng, Jing Pan, Prashant Sridhar, Kyu J Han, and Shinji Watanabe. E-Branchformer: Branchformer with enhanced merging for speech recognition. In *Proceedings of SLT*, pages 84–91, 2023.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *Proceedings of ICASSP*, pages 4835–4839, 2017.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of EMNLP*, pages 1317–1327, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Proceedings of Interspeech*, pages 3586–3589, 2015.

Tatsuya Komatsu. Non-autoregressive ASR with self-conditioned folded encoders. In *Proceedings of ICASSP*, pages 7427–7431, 2022.

Tatsuya Komatsu and Yusuke Fujita. Interdecoder: Using attention decoders as intermediate regularization for CTC-based speech recognition. In *Proceedings of SLT*, pages 46–51, 2023.

Jan Kremer, Lasse Borgholt, and Lars Maaløe. On the inductive bias of word-character-level multi-task learning for speech recognition. *arXiv preprint arXiv:1812.02308*, 2018.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. QuartzNet: Deep automatic speech recognition with 1D time-channel separable convolutions. In *Proceedings of ICASSP*, pages 6124–6128, 2020.

Kalpesh Krishna, Shubham Toshniwal, and Karen Livescu. Hierarchical multitask learning for CTC-based speech recognition. *arXiv preprint arXiv:1807.06234*, 2018.

Yotaro Kubo, Shigeki Karita, and Michiel Bacchiani. Knowledge transfer from large-scale pretrained language models to end-to-end speech recognizers. In *Proceedings of ICASSP*, pages 8512–8516, 2022.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of ACL*, pages 66–75, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of ICLR*, 2020.

Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Julius — an open source real-time large vocabulary recognition engine. In *Proceedings of EUROSPEECH*, pages 1691–1694, 2001.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Proceedings of ICML Workshop on Challenges in Representation Learning*, 2013.

Jaesong Lee and Shinji Watanabe. Intermediate loss regularization for CTC-based speech recognition. In *Proceedings of ICASSP*, pages 6224–6228, 2021.

Jaesong Lee, Lukas Lee, and Shinji Watanabe. Memory-efficient training of RNN-transducer with sampled softmax. In *Proceedings of Interspeech*, pages 4441–4445, 2022.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of EMNLP*, pages 1173–1182, 2018.

Bo Li, Tara N Sainath, Ruoming Pang, and Zelin Wu. Semi-supervised training for end-to-end models via weak distillation. In *Proceedings of ICASSP*, pages 2837–2841, 2019a.

Bo Li, Anmol Gulati, Jiahui Yu, Tara N Sainath, Chung-Cheng Chiu, Arun Narayanan, Shuo-Yiin Chang, Ruoming Pang, Yanzhang He, James Qin, et al. A better and faster end-to-end model for streaming ASR. In *Proceedings of ICASSP*, pages 5634–5638, 2021.

Jinyu Li. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

Jinyu Li, Guoli Ye, Rui Zhao, Jasha Droppo, and Yifan Gong. Acoustic-to-word model without OOV. In *Proceedings of ASRU*, pages 111–117, 2017.

Jinyu Li, Rui Zhao, Hu Hu, and Yifan Gong. Improving RNN transducer modeling for end-to-end speech recognition. In *Proceedings of ASRU*, pages 114–121, 2019b.

Mohan Li and Rama Doddipatla. Non-autoregressive end-to-end approaches for joint automatic speech recognition and spoken language understanding. In *Proceedings of SLT*, pages 390–397, 2023.

Sheng Li, Raj Dabre, Xugang Lu, Peng Shen, Tatsuya Kawahara, and Hisashi Kawai. Improving Transformer-based speech recognition systems with compressed structure and speech attributes augmentation. In *Proceedings of Interspeech*, pages 4400–4404, 2019c.

Tatiana Likhomanenko, Qiantong Xu, Vineel Pratap, Paden Tomasello, Jacob Kahn, Gilad Avidov, Ronan Collobert, and Gabriel Synnaeve. Rethinking evaluation in ASR: Are our models robust enough? *arXiv preprint arXiv:2010.11745*, 2020.

Tatiana Likhomanenko, Qiantong Xu, Jacob Kahn, Gabriel Synnaeve, and Ronan Collobert. slim-IPL: Language-model-free iterative pseudo-labeling. In *Proceedings of Interspeech*, pages 741–745, 2021.

Yi Chieh Liu, Eunjung Han, Chul Lee, and Andreas Stolcke. End-to-end neural diarization: From Transformer to Conformer. In *Proceedings of Interspeech*, pages 3081–3085, 2021.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pre-training approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ke-Han Lu and Kuan-Yu Chen. A context-aware knowledge transferring strategy for CTC-based ASR. In *Proceedings of SLT*, pages 60–67, 2022.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving Transformer from a multi-particle dynamic system point of view. In *Proceedings of ICLR*, 2020.

Jeff Ma and Richard Schwartz. Unsupervised versus supervised training of acoustic models. In *Proceedings of Interspeech*, pages 2374–2377, 2008.

Rao Ma, Mark J. F. Gales, Kate M. Knill, and Mengjie Qian. N-best T5: Robust ASR error correction using multiple input hypotheses and constrained decoding space. In *Proceedings of Interspeech*, pages 3267–3271, 2023a.

Rao Ma, Mengjie Qian, Potsawee Manakul, Mark Gales, and Kate Knill. Can generative large language models perform ASR error correction? *arXiv preprint arXiv:2307.04172*, 2023b.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of EMNLP-IJCNLP*, pages 4273–4283, 2019.

Kikuo Maekawa. Corpus of spontaneous Japanese: Its design and evaluation. In *Proceedings of ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

Somshubra Majumdar, Jagadeesh Balam, Oleksii Hrinchuk, Vitaly Lavrukhin, Vahid Noroozi, and Boris Ginsburg. Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition. *arXiv preprint arXiv:2104.01721*, 2021.

Ryo Masumura, Mana Ihori, Akihiko Takashima, Takafumi Moriya, Atsushi Ando, and Yusuke Shinohara. Sequence-level consistency training for semi-supervised end-to-end automatic speech recognition. In *Proceedings of ICASSP*, pages 7054–7058, 2020.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using Kaldi. In *Proceedings of Interspeech*, pages 498–502, 2017.

James L McClelland and Jeffrey L Elman. The TRACE model of speech perception. *Cognitive psychology*, 18(1):1–86, 1986.

Quentin Meeus, Marie-Francine Moens, and Hugo Van Hamme. Bidirectional representations for low-resource spoken language understanding. *Applied Sciences*, 13(20):11291, 2023.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Proceedings of ASRU*, pages 167–174, 2015.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of Interspeech*, pages 1045–1048, 2010.

Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2011.

Niko Moritz, Takaaki Hori, and Jonathan Le. Streaming automatic speech recognition with the Transformer model. In *Proceedings of ICASSP*, pages 6074–6078, 2020.

Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Semi-supervised speech recognition via graph-based temporal classification. In *Proceedings of ICASSP*, pages 6548–6552, 2021.

Takafumi Moriya, Sei Ueno, Yusuke Shinohara, Marc Delcroix, Yoshikazu Yamaguchi, and Yushi Aono. Multi-task learning with augmentation strategy for acoustic-to-word attention-based encoder-decoder speech recognition. In *Proceedings of Interspeech*, pages 2399–2403, 2018.

Edwin G. Ng, Chung-Cheng Chiu, Yu Zhang, and William Chan. Pushing the limits of non-autoregressive speech recognition. In *Proceedings of Interspeech*, pages 3725–3729, 2021.

Mengxi Nie, Ming Yan, Caixia Gong, and D Chuxing. Prompt-based re-ranking language model for ASR. In *Proceedings of Interspeech*, pages 3864–3868, 2022.

Dennis Norris. Shortlist: A connectionist model of continuous speech recognition. *Cognition*, 52 (3):189–234, 1994.

Jumon Nozaki and Tatsuya Komatsu. Relaxing the conditional independence assumption of CTC-based ASR by conditioning on intermediate predictions. In *Proceedings of Interspeech*, pages 3735–3739, 2021.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Proceedings of NeurIPS*, pages 27730–27744, 2022.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *Proceedings of ICASSP*, pages 5206–5210, 2015.

Sankaran Panchapagesan, Daniel S Park, Chung-Cheng Chiu, Yuan Shangguan, Qiao Liang, and Alexander Gruenstein. Efficient knowledge distillation for rnn-transducer models. In *Proceedings of ICASSP*, pages 5639–5643, 2021.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Proceedings of Interspeech*, pages 2613–2617, 2019.

Daniel S Park, Yu Zhang, Chung-Cheng Chiu, Youzheng Chen, Bo Li, William Chan, Quoc V Le, and Yonghui Wu. SpecAugment on large scale datasets. In *Proceedings of ICASSP*, pages 6879–6883, 2020a.

Daniel S. Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu, and Quoc V. Le. Improved noisy student training for automatic speech recognition. In *Proceedings of Interspeech*, pages 2817–2821, 2020b.

Douglas B Paul and Janet M Baker. The design for the wall street journal-based CSR corpus. In *Proceedings of Workshop on Speech and Natural Language*, pages 357–362, 1992.

Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe. Branchformer: Parallel MLP-attention architectures to capture local and global context for speech recognition and understanding. In *Proceedings of ICML*, pages 17627–17643, 2022.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of RepL4NLP*, pages 7–14, 2019.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. Improved speech-to-text translation with the Fisher and Callhome Spanish–English speech translation corpus. In *Proceedings of IWSLT*, 2013.

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 *Proceedings of ASRU*, 2011.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for ASR based on lattice-free MMI. In *Proceedings of Interspeech*, pages 2751–2755, 2016.

Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur. A time-restricted self-attention layer for ASR. In *Proceedings of ICASSP*, pages 5874–5878, 2018.

Rohit Prabhavalkar, Kanishka Rao, Tara N Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly. A comparison of sequence-to-sequence models for speech recognition. In *Proceedings of Interspeech*, pages 939–943, 2017.

Rohit Prabhavalkar, Tara N Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan. Minimum word error rate training for attention-based sequence-to-sequence models. In *Proceedings of ICASSP*, pages 4839–4843, 2018.

Rohit Prabhavalkar, Yanzhang He, David Rybach, Sean Campbell, Arun Narayanan, Trevor Strohman, and Tara N Sainath. Less is more: Improved RNN-T decoding using limited label context and path merging. In *Proceedings of ICASSP*, pages 5659–5663, 2021.

Rohit Prabhavalkar, Takaaki Hori, Tara N Sainath, Ralf Schlüter, and Shinji Watanabe. End-to-end speech recognition: A survey. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:325–351, 2024.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training, 2018. [Online; Accessed on January-13-2024].

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners, 2019. [Online; Accessed on January-13-2024].

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of ICML*, pages 28492–28518, 2023.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *Proceedings of ICLR*, 2016.

Kanishka Rao and Haşim Sak. Multi-accent speech recognition with hierarchical grapheme based models. In *Proceedings of ICASSP*, pages 4815–4819, 2017.

Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer. In *Proceedings of ASRU*, pages 193–199, 2017.

Anthony Rousseau, Paul Deléglise, and Yannick Estève. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *Proceedings of LREC*, pages 3935–3939, 2014.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. In *Proceedings of EMNLP*, pages 1098–1108, 2020.

Tara N Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, et al. Two-pass end-to-end speech recognition. In *Proceedings of Interspeech*, pages 2773–2777, 2019.

Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. Masked language model scoring. In *Proceedings of ACL*, pages 2699–2712, 2020.

Ramon Sanabria and Florian Metze. Hierarchical multitask learning with CTC. In *Proceedings of SLT*, pages 485–490, 2018.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Proceedings of NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. In *Proceedings of Interspeech*, pages 132–136, 2017.

George Saon, Zoltán Tüske, Daniel Bolanos, and Brian Kingsbury. Advancing RNN transducer technology for speech recognition. In *Proceedings of ICASSP*, pages 5654–5658, 2021.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. BLOOM: A 176B-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725, 2016.

Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie. Component fusion: Learning replaceable language model component for end-to-end speech recognition system. In *Proceedings of ICASSP*, pages 5361–5635, 2019.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*, 2019.

Zhiqiang Shen, Zechun Liu, and Eric Xing. Sliced recursive Transformer. In *Proceedings of ECCV*, pages 727–744, 2022.

Kyuhong Shim, Jungwook Choi, and Wonyong Sung. Understanding the role of self attention for efficient speech recognition. In *Proceedings of ICLR*, 2021.

Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. Effective sentence scoring method using BERT for speech recognition. In *Proceedings of ACML*, pages 1081–1093, 2019.

Harsh Shrivastava, Ankush Garg, Yuan Cao, Yu Zhang, and Tara Sainath. Echo state speech recognition. In *Proceedings of ICASSP*, pages 5669–5673, 2021.

Hagen Soltau, Hank Liao, and Hasim Sak. Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition. In *Proceedings of Interspeech*, pages 3707–3711, 2017.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. In *Proceedings of Interspeech*, pages 387–391, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Lukas Stappen, Fabian Brunn, and Björn Schuller. Cross-lingual zero- and few-shot hate speech detection utilising frozen Transformer language models and AXEL. *arXiv preprint arXiv:2004.13850*, 2020.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion Transformer: Flexible sequence generation via insertion operations. In *Proceedings of ICML*, pages 5976–5985, 2019.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *Proceedings of Interspeech*, pages 194–197, 2012.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proceedings of NeurIPS*, pages 3104–3112, 2014.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of NeurIPS*, pages 1195—-1204, 2017.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of ACL*, pages 4593–4601, 2019.

Andros Tjandra, Chunxi Liu, Frank Zhang, Xiaohui Zhang, Yongqiang Wang, Gabriel Synnaeve, Satoshi Nakamura, and Geoffrey Zweig. DEJA-VU: Double feature presentation and iterated loss in deep Transformer networks. In *Proceedings of ICASSP*, pages 6899–6903, 2020.

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *arXiv preprint arXiv:1704.01631*, 2017.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Anshuman Tripathi, Jaeyoung Kim, Qian Zhang, Han Lu, and Hasim Sak. Transformer transducer: One model unifying streaming and non-streaming speech recognition. *arXiv preprint arXiv:2010.03192*, 2020.

Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. Transformer ASR with contextual block processing. In *Proceedings of ASRU*, pages 427–433, 2019.

Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe. Streaming Transformer ASR with blockwise synchronous beam search. In *Proceedings of SLT*, pages 22–29, 2021.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of ACL*, pages 76–85, 2016.

Zoltán Tüske, George Saon, and Brian Kingsbury. On the limit of english conversational speech recognition. In *Proceedings of Interspeech*, pages 2062–2066, 2021.

Takuma Udagawa, Masayuki Suzuki, Gakuto Kurata, Nobuyasu Itoh, and George Saon. Effect and analysis of large-scale language model rescoring on competitive ASR systems. In *Proceedings of Interspeech*, pages 3919–3923, 2022.

Ehsan Variani, David Rybach, Cyril Allauzen, and Michael Riley. Hybrid autoregressive transducer (HAT). In *Proceedings of ICASSP*, pages 6139–6143, 2020.

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

Apoorv Vyas, Pranay Dighe, Sibo Tong, and Hervé Bourlard. Analyzing uncertainties in speech recognition using dropout. In *Proceedings of ICASSP*, pages 6730–6734, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of ICLR*, 2018.

Changhan Wang, Anne Wu, Jiatao Gu, and Juan Pino. CoVoST 2 and massively multilingual speech translation. In *Proceedings of Interspeech*, pages 2247–2251, 2021a.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep Transformer models for machine translation. In *Proceedings of ACL*, pages 1810–1822, 2019.

Tianzi Wang, Yuya Fujita, Xuankai Chang, and Shinji Watanabe. Streaming end-to-end ASR based on blockwise non-autoregressive models. In *Proceedings of Interspeech*, pages 3755–3759, 2021b.

Weiran Wang, Ke Hu, and Tara N Sainath. Deliberation of streaming RNN-transducer by non-autoregressive decoding. In *Proceedings of ICASSP*, pages 7452–7456, 2022.

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.

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 *Proceedings of Interspeech*, pages 2207–2211, 2018.

Shinji Watanabe, Florian Boyer, Xuankai Chang, Pengcheng Guo, Tomoki Hayashi, Yosuke Higuchi, Takaaki Hori, Wen-Chin Huang, Hirofumi Inaguma, Naoyuki Kamo, et al. The 2020 ESPnet update: New features, broadened applications, performance improvements, and future plans. In *Proceedings of DSLW*, pages 1–6, 2021.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *Proceedings of ICLR*, 2022a.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022b.

Felix Weninger, Franco Mana, Roberto Gemello, Jesús Andrés-Ferrer, and Puming Zhan. Semi-supervised learning with data augmentation for end-to-end ASR. In *Proceedings of Interspeech*, pages 2802–2806, 2020.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*, pages 38–45, 2020.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. ChatGPT or Grammarly? evaluating ChatGPT on grammatical error correction benchmark. *arXiv preprint arXiv:2303.13648*, 2023.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of ECCV*, pages 3–19, 2018.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Deliberation networks: Sequence generation beyond one-pass decoding. In *Proceedings of NeurIPS*, pages 1784–1794, 2017.

Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. The Microsoft 2017 conversational speech recognition system. In *Proceedings of ICASSP*, pages 5934–5938, 2018.

Qiantong Xu, Tatiana Likhomanenko, Jacob Kahn, Awni Hannun, Gabriel Synnaeve, and Ronan Collobert. Iterative pseudo-labeling for speech recognition. In *Proceedings of Interspeech*, pages 1006–1010, 2020.

Cheng Yi, Shiyu Zhou, and Bo Xu. Efficiently fusing pretrained acoustic and linguistic encoders for low-resource speech recognition. *IEEE Signal Processing Letters*, 28:788–792, 2021.

Fu-Hao Yu, Kuan-Yu Chen, and Ke-Han Lu. Non-autoregressive ASR modeling using pre-trained language models for Chinese speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1474–1482, 2022.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proceedings of ECCV*, pages 818–833, 2014.

Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. A comparison of Transformer and LSTM encoder decoder models for ASR. In *Proceedings of ASRU*, pages 8–15, 2019.

Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, et al. WenetSpeech: A 10000+ hours multi-domain Mandarin corpus for speech recognition. In *Proceedings of ICASSP*, pages 6182–6186, 2022a.

Jicheng Zhang, Yizhou Peng, Haihua Xu, Yi He, Eng Siong Chng, and Hao Huang. Intermediate-layer output regularization for attention-based speech recognition with shared decoder. *arXiv preprint arXiv:2207.04177*, 2022b.

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. In *Proceedings of ICASSP*, pages 7829–7833, 2020a.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yao, Sanjeev Khudanpur, and James Glass. Highway long short-term memory RNNs for distant speech recognition. In *Proceedings of ICASSP*, pages 5755–5759, 2016.

Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. Pushing the limits of semi-supervised learning for automatic speech recognition. In *Proceedings of NeurIPS Workshop on Self-Supervised Learning for Speech and Audio Processing*, 2020b.

Yu Zhang, Daniel S Park, Wei Han, James Qin, Anmol Gulati, Joel Shor, Aren Jansen, Yuanzhong Xu, Yanping Huang, Shibo Wang, et al. BigSSL: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1519–1532, 2022c.

Yu Zhang, Wei Han, James Qin, Yongqiang Wang, Ankur Bapna, Zhehuai Chen, Nanxin Chen, Bo Li, Vera Axelrod, Gary Wang, et al. Google USM: Scaling automatic speech recognition beyond 100 languages. *arXiv preprint arXiv:2303.01037*, 2023.

Huaibo Zhao, Yosuke Higuchi, Tetsuji Ogawa, and Tetsunori Kobayashi. An investigation of enhancing CTC model for triggered attention-based streaming ASR. In *Proceedings of APSIPA ASC*, pages 477–483, 2021.

Huaibo Zhao, Yosuke Higuchi, Yusuke Kida, Tetsuji Ogawa, and Tetsunori Kobayashi. Mask-CTC-based encoder pre-training for streaming end-to-end speech recognition. In *Proceedings of EUSIPCO*, pages 56–60, 2023.

Guolin Zheng, Yubei Xiao, Ke Gong, Pan Zhou, Xiaodan Liang, and Liang Lin. Wav-BERT: Cooperative acoustic and linguistic representation learning for low-resource speech recognition. In *Findings of EMNLP*, pages 2765–2777, 2021.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. Incorporating BERT into neural machine translation. In *Proceedings of ICLR*, 2020.

Geoffrey Zweig, Chengzhu Yu, Jasha Droppo, and Andreas Stolcke. Advances in all-neural speech recognition. In *Proceedings of ICASSP*, pages 4805–4809, 2017.

Research Achievements

Journal Paper Publication

2022 Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. Momentum pseudo-labeling: Semi-supervised ASR with continuously improving pseudo-labels. In *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pages 1424-1438, October 2022.

International Conference Paper Publications (Peer Reviewed)

2024 Tomoki Ariga, Yosuke Higuchi, Kazutoshi Hayasaka, Naoki Okamoto, and Tetsuji Ogawa. Parody detection using source-target attention with teacher-forced lyrics. In *Proceedings of the 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2024. (accepted)

2023 Masao Someki, Nicholas Eng, Yosuke Higuchi, and Shinji Watanabe. Segment-level vectorized beam search based on partially autoregressive inference. In *Proceedings of the 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, December 2023.

2023 Yosuke Higuchi, Andrew Rosenberg, Yuan Wang, Murali Karthick Baskar, and Bhuvana Ramabhadran. Mask-Conformer: Augmenting Conformer with mask-predict decoder. In *Proceedings of the 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, December 2023.

2023 Tomoki Ariga, Yosuke Higuchi, Mitsunori Kanno, Rie Shigyo, Takato Mizuguchi, Naoki Okamoto, and Tetsuji Ogawa. Spotting parodies: Detecting alignment collapse between lyrics and singing voice. In *Proceedings of the 31st European Signal Processing Conference (EUSIPCO)*, pages 286–290, September 2023.

2023 Huaibo Zhao, Yosuke Higuchi, Yusuke Kida, Tetsuji Ogawa, and Tetsunori Kobayashi. Mask-CTC-based encoder pre-training for streaming end-to-end speech recognition. In *Proceedings of the 31st European Signal Processing Conference (EUSIPCO)*, pages 56–60, September 2023.

2023 Yosuke Higuchi, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. InterMPL: Momentum pseudo-labeling with intermediate CTC loss. In *Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2023.

2023 Yosuke Higuchi, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. BECTRA: Transducer-based end-to-end ASR with BERT-enhanced encoder. In *Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2023.

2023 Brian Yan, Siddharth Dalmia, Yosuke Higuchi, Graham Neubig, Florian Metze, Alan W Black, and Shinji Watanabe. CTC alignments improve autoregressive translation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1623–1639, May 2023.

2023 Yifan Peng*, Siddhant Arora*, Yosuke Higuchi, Yushi Ueda, Sujay Kumar, Karthik Ganesan, Siddharth Dalmia, Xuankai Chang, and Shinji Watanabe. A study on the integration of pre-trained SSL, ASR, LM and SLU models for spoken language understandings. In *Proceedings of the 2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 406–413, January 2023. (*equal contribution)

2022 Yosuke Higuchi, Brian Yan, Siddhant Arora, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. BERT meets CTC: New formulation of end-to-end speech recognition with pre-trained masked language model. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5486–5503, December 2022.

2022 Masao Someki, Yosuke Higuchi, Tomoki Hayashi, and Shinji Watanabe. ESPnet-ONNX: Bridging a gap between research and production. In *Proceedings of the 2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 420–427, November 2022.

2022 Keqi Deng*, Zehui Yang*, Shinji Watanabe, Yosuke Higuchi, Gaofeng Cheng, and Pengyuan Zhang. Improving non-autoregressive end-to-end speech recognition with pre-trained acoustic and language models. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8522–8526, May 2022. (*equal contribution)

2022 Yosuke Higuchi, Keita Karube, Tetsuji Ogawa, and Tetsunori Kobayashi. Hierarchical conditional end-to-end ASR with CTC and multi-granular subword units. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7797–7801, May 2022.

2022 Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. Advancing momentum pseudo-labeling with Conformer and initialization strategy. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7672–7676, May 2022.

2021 Huaibo Zhao, Yosuke Higuchi, Tetsuji Ogawa, and Tetsunori Kobayashi. An investigation of enhancing CTC model for triggered attention-based streaming ASR. In *Proceedings of the 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 477–483, December 2021.

2021 Yosuke Higuchi, Nanxin Chen, Yuya Fujita, Hirofumi Inaguma, Tatsuya Komatsu, Jaesong Lee, Jumon Nozaki, Tianzi Wang, and Shinji Watanabe. A comparative study on non-autoregressive modelings for speech-to-text generation. In *Proceedings of the 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 47–54, December 2021.

2021 Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. Momentum pseudo-labeling for semi-supervised speech recognition. In *Proceedings of the 22nd Annual Conference of the International Speech Communication Association (Interspeech)*, pages 726–730, August 2021.

2021 Shinji Watanabe, Florian Boyer, Xuankai Chang, Pengcheng Guo, Tomoki Hayashi, Yosuke Higuchi, Takaaki Hori, Wen-Chin Huang, Hirofumi Inaguma, Naoyuki Kamo, Shigeki Karita, Chenda Li, Jing Shi, Aswin Shanmugam Subramanian, and Wangyou Zhang. The 2020 ESPnet update: New features, broadened applications, performance improvements, and future plans. In *Proceedings of the 2021 IEEE Data Science & Learning Workshop (DSLW)*, pages 1–6, June 2021.

2021 Yosuke Higuchi, Shinji Watanabe, Hirofumi Inaguma, Tetsuji Ogawa, and Tetsunori Kobayashi. Improved Mask-CTC for non-autoregressive end-to-end ASR. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8363–8367, June 2021.

2021 Hirofumi Inaguma, Yosuke Higuchi, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. Orthros: Non-autoregressive end-to-end speech translation with dual-decoder. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7503–7507, June 2021.

2021 Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, Jing Shi, Shinji Watanabe, Kun Wei, Wangyou Zhang, and Yuekai Zhang. Recent developments on ESPnet toolkit boosted by Conformer. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5874–5878, June 2021.

2021 Yosuke Higuchi, Naohiro Tawara, Atsunori Ogawa, Tomoharu Iwata, Tetsunori Kobayashi, and Tetsuji Ogawa. Noise-robust attention learning for end-to-end speech recognition. In

Proceedings of the 28th European Signal Processing Conference (EUSIPCO), pages 311–315, January 2021.

2020 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 *Proceedings of the 21st Annual Conference of the International Speech Communication Association (Interspeech)*, pages 3655–3659, October 2020.

2020 Yosuke Higuchi, Masayuki Suzuki, and Gakuto Kurata. Speaker embeddings incorporating acoustic conditions for diarization. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7129–7133, May 2020.

2019 Yosuke Higuchi, Naohiro Tawara, Tetsunori Kobayashi, and Tetsuji Ogawa. Speaker adversarial training of DPGMM-based feature extractor for zero-resource languages. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 266–270, September 2019.

Domestic Conference/Workshop Paper Publications (in Japanese)

2024 楠奈穂美, 樋口陽祐, 小川哲司, 小林哲則. 再帰的フィードバックを用いた階層的マルチタスク学習によるEnd-to-End音声認識. 日本音響学会研究発表会講演論文集 *(ASJ)*, March 2024. (発表予定)

2023 牛尾貴志, 樋口陽祐, 久原卓, 藤原晴雄, 加藤 博司. 事前学習済み音声認識モデルを用いた笑い声検出. 言語・音声理解と対話処理研究会 *(SLUD)*, pages 59–65, September 2023.

2023 当間佐耶佳, 有賀智輝, 樋口陽祐, 早坂一寿, 岡本直紀, 小川哲司. 深層話者埋め込みを用いた歌唱者の照合に関する検討. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 1231–1232, September 2023.

2023 有賀智輝, 樋口陽祐, 早坂一寿, 岡本直紀, 小林哲則, 小川哲司. Teacher-Forcingにより歌詞を与えた際のAttentionの崩れに着目した替え歌検知. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 1101–1104, September 2023.

2023 樋口陽祐, 小川哲司, 小林哲則, 渡部晋治. 事前学習済みマスク言語モデルを用いたEnd-to-End音声認識. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 1023–1026, September 2023.

2023 有賀智輝, 樋口陽祐, 菅野光則, 執行里恵, 水口天都, 岡本直紀, 小川哲司. 歌詞と歌唱音声のアライメント崩れに基づく替え歌検知. 電子情報通信学会技術研究報告 *(SP)*, vol. 123, no. 88, SP2023-10, pages 48–53, June 2023.

2022 趙懐博, 樋口陽祐, 木田祐介, 小林哲則, 小川哲司. Transducer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.

2022 樋口陽祐, 軽部敬太, 小川哲司, 小林哲則. 粒度の異なるサブワード単位に基づく階層的条件付きEnd-to-End音声認識. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 955–956, March 2022.

2022 樋口陽祐, Moritz Niko, Le Roux Jonathan, 堀貴明. Momentum Pseudo-Labelingによる半教師ありEnd-to-End音声認識. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 881–882, March 2022.

2021 樋口陽祐, 軽部敬太, 小川哲司, 小林哲則. End-to-End音声認識のための粒度の異なるサブワード単位に基づく階層的な条件づけ. 情報処理学会研究報告 *(SLP)*, vol. 2021-SLP-139, no. 19, pages 1–8, December 2021.

2021 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.

2020 樋口陽祐, 渡部晋治, 稲熊寛文, 小川哲司, 小林哲則. CTCとマスク推定に基づく推論速度の速いEnd-to-End音声認識. 電子情報通信学会技術研究報告 *(SP)*, vol. SP2020-16, pages 1–6, December 2020.

2020 樋口陽祐, 渡部晋治, Chen Nanxin, 小川哲司, 小林哲則. Mask CTC: CTCとマスク推定に基づいた非自己回帰的なEnd-to-End音声認識. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 747–748, September 2020.

2020 樋口陽祐, 鈴木雅之, 倉田岳人. ダイアライゼーションのための音響的環境を考慮した話者エンベディング. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 689–690, September 2020.

2020 樋口陽祐, 俵直弘, 小川厚徳, 岩田具治, 小林 哲則, 小川哲司. Attentionに関する損失を利用したノイズに頑健なEnd-to-End音声認識. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 935–936, March 2020.

2019 樋口陽祐, 俵直弘, 小林哲則, 小川哲司. DPGMMと敵対的学習に基づく話者の違いに頑健な特徴抽出とゼロリソース音声認識での評価. 情報処理学会研究報告 *(SLP)*, vol. 2019-SLP-128, no. 6, pages 1–6, July 2019.

2019 樋口陽祐, 俵直弘, 小川哲司, 小林哲則. ゼロリソース言語音声認識のための発話者の違いに頑健な特徴抽出. 日本音響学会研究発表会講演論文集 *(ASJ)*, pages 923–924, March 2019.

Patents

2021 Metric learning of speaker diarization (U.S. Patent Application No. 16/807892)

2020 学習装置, 音声認識装置, 学習方法, および, 学習プログラム (Japan 2020-028869)

Awards

2023 Best Reviewer Award, from IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), December 2023.

2021 ISS Young Researcher's Award in Speech Field, from the Institute of Electronics, Information and Communication Engineers (IEICE), September 2021.

2020 Best Student Presentation Award, from the Acoustical Society of Japan (ASJ), November 2020.

2020 Yamashita SIG Research Award, from Information Processing Society of Japan (IPSJ), August 2020.

2020 Yahoo! JAPAN Award, from Information Processing Society of Japan (IPSJ) SIG-SLP, January 2020.