UNIVERSITY OF CALIFORNIA

Los Angeles

# Improving the Accuracy and Inference Efficiency for Low-resource Automatic Speech Recognition

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Ruchao Fan

2024

ABSTRACT OF THE DISSERTATION

Improving the Accuracy and Inference Efficiency for Low-resource
Automatic Speech Recognition

by

Ruchao Fan
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2024
Professor Abeer A. Alwan, Chair

Automatic speech recognition (ASR) systems have improved significantly in the last decade due to advances in deep learning algorithms and easier access to very large databases. ASR systems, however, face two major challenges. The first challenge is in low-resource situations, such as child speech, where the accuracy degrades significantly, and the second challenge concerns low inference efficiency due to the autoregressive mechanism and size of ASR models. In this dissertation, we address these challenges by introducing novel techniques to improve the accuracy and the inference efficiency for ASR tasks, especially child ASR.

To address the accuracy challenge, we introduce novel self-supervised learning (SSL) methods using un-annotated adult speech data and explore how these methods can improve the downstream child ASR tasks. Specifically, a bidirectional autoregressive predictive coding (Bi-APC) method is proposed for non-causal models pretraining with the usage of adult speech data. The pretrained model is then finetuned on supervised child speech-text pairs. We also propose a novel framework, domain responsible adaptation and finetuning (DRAFT), to reduce the domain shifting in pretrained speech models. The DRAFT framework is effective for APC that uses a causal transformer as the backbone, and for Bi-APC, Wav2vec2.0 and HuBERT methods, which use a non-causal transformer as the backbone.

To address the inference efficiency challenge, we introduce a novel Connectionist Temporal Classification (CTC) Alignment-based Single-Step Non-Autoregressive Transformer (CASS-NAT) for end-to-end ASR. A comprehensive evaluation of CASS-NAT is performed in this dissertation. In CASS-NAT, the word embeddings in an autoregressive transformer (AT) are substituted with token-level acoustic embeddings (TAE) that are extracted based on the encoder outputs and CTC alignments. TAE can be obtained simultaneously without recurrent operations, leading to a parallel generation of output sequences. In addition, an error-based alignment sampling method is proposed to reduce alignment mismatch between the training and inference. CASS-NAT achieves $\sim$20x speed up during inference without significant performance degradation compared to AT. We also propose a CASS-NAT variant (UniEnc-CASSNAT) that consists of only an encoder module. Together with the proposed multi-pass CTC training and iterative decoding, UniEnc-CASSNAT can perform as well as CASS-NAT with fewer model parameters.

Beyond these two challenges, and in order to facilitate the development of better child ASR, we build the first child ASR benchmark for the research community. The benchmark includes comparisons of widely-used techniques for ASR such as data augmentation, parameter efficient finetuning (PEFT), self-supervised (HuBERT and WavLM) and supervised models (such as Whisper). All codes developed in this dissertation will be available.

The dissertation of Ruchao Fan is approved.

Christina Panagio Fragouli

Jonathan Chau-Yan Kao

Achuta Kadambi

Abeer A. Alwan, Committee Chair

University of California, Los Angeles

2024

*To my family . . .*

*for their love and support*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

# ACKNOWLEDGMENTS

This dissertation would not have been possible without the endless support from many people. First and foremost, I would like to express my most earnest appreciation to my doctoral advisor, Professor Abeer Alwan. She has been a wonderful mentor and provided me with gracious support, especially during the most difficult time in my Ph.D. career when I stayed in hospital for months. She has given me great freedom to pursue the research topics that I would like to work on. She has encouraged me to tackle challenging problems and taught me useful academic writing skills. Under Prof. Alwan's supervision, I have learned how to be a good researcher.

I am immensely grateful to Professor Achuta Kadambi for his support on my UCLA-Amazon fellowship application and insightful suggestions to my research. I would also like to thank the members of my Ph.D. committee: Professor Christina Fragouli and Professor Jonathan Kao, for their invaluable advice and comments on my dissertation.

I have been honored and grateful to work with amazing mentors during my internships. Many thanks to Dr. Wei Chu, Dr. Peng Chang and Dr. Mei Han for the internship opportunity at PAII Inc. and the one-year research funding. Sincere appreciation also goes to Dr. Jie Yuan, Dr. Nilaksh Das, Dr. Thomas Muller, and Dr. Monica Sunkara for their supervision during my summer internship at AWS AI Lab. I have enjoyed and benefited a lot from various discussions with them during the internship. Special thanks to my manager Jinyu Li when I interned at Microsoft Azure AI speech team. His sound knowledge and nice personality made my internship and our further collaborations a great pleasure for me. I am also grateful to Dr. Guoli Ye, Dr. Yiming Wang, Dr. Yashesh Gaur, and Dr. Yifan Gong at Microsoft. Their mentorship has immensely impacted my thought process as a researcher.

My heartfelt thanks also go to my former and current labmates at SPAPL. I have had numerous and inspiring conversations with them that helped me keep up with the latest research and gave me ideas whenever I was stuck. Many thanks to Wei, Jinxi, Gary, Amber, Vijay, AJ, Jinhan, Vishwas, Eray, Yunzheng, Balaji, and Hariram for many enjoyable discussions on the research and life during my Ph.D. study. Special thanks to Wei, Gary, Amber, Jinhan, Yunzheng, Balaji for the collaborations directly related to this dissertation, without which this work would not have been possible.

I also extend my gratitude to the administrative staff in the department (Deeona Columbia, Ylena Requena, Tatyana Vigil, Jacquelyn Trang, Lorena Sanchez, Yadira Trejo, and Jose Cano) for their patience and tremendous help throughout my entire Ph.D. period. Life would have been a lot tougher without them.

I am truly blessed to have great roommates and friends (Yunzheng Zhu, Ruixuan Zhao, Jinhan Wang, Chunhe Ni) who were one call away whenever I needed them. They provided constant support, encouragement, and a hearing ear for all my breakdowns. I am thankful for their company, support and many enjoyable chats.

Last but more importantly, I am ever grateful to my mother Genmei Hu and my father Xiaogen Fan without whom I would not be the person I am today. I would like to thank my sister Yuanlan Fan, giving accompanying with my parents when I pursued my Ph.D. abroad. They have been my biggest cheerleader and made a lot of sacrifices to ensure that I could focus on my research and career goals. I am exceptionally grateful to my girlfriend Siying Guan who cheers my small and big moments and supports through day-to-day struggles. The time difference and visa problems did not stop the bond between us. Although we were living in the opposite side of the earth, she has been persisting in making me a phone call to me everyday in the last five years. Her support and encouragement have made everyday of my life full of energy.

I would not have a wonderful and successful Ph.D. career without all these people. Thank you all for being there through the ups and downs of my Ph.D. journey.

## VITA

| | |
|---|---|
| 2012–2016 | B.S., Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China |
| 2016–2019 | M.S., Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China |
| 2019-2024 | Graduate Student Researcher, Speech Processing and Auditory Perception, Laboratory, UCLA. |
| 2020 | Research Intern, PAII Inc., Palo Alto, California |
| 2020-2021 | One-year Research Funding from PAII Inc. |
| 2021,2022 | Research Intern, Microsoft Corporation, Redmond, Washington |
| 2022 | UCLA-Amazon Fellowship |
| 2023 | Applied Scientist Intern, Amazon Web Service, Santa Clara, California |

## SELECTED PUBLICATIONS

**Ruchao Fan**, Natarajan Balaji Shankar, and Abeer Alwan, "Benchmarking Children's ASR with Supervised and Self-supervised Speech Foundation Models", Interspeech 2024.

**Ruchao Fan**, Natarajan Balaji Shankar, and Abeer Alwan, "UniEnc-CASSNAT: An Encoder-only Non-autoregressive ASR for Speech SSL Models", IEEE Signal Processing Letters, 2024

**Ruchao Fan**, Yiming Wang, Yashesh Gaur and Jinyu Li, "CTCBERT: Advancing Hidden-Unit Bert with CTC Objectives," ICASSP 2023.

**Ruchao Fan**, Wei Chu, Peng Chang, and Abeer Alwan, "A CTC Alignment-based Non-autoregressive Transformer for End-to-end Automatic Speech Recognition", in IEEE Transactions on Audio, Speech and Language Processing, 2023.

**Ruchao Fan**, Yunzheng Zhu, Jinhan Wang, and Abeer Alwan, "Towards Better Domain Adaptation for Self-Supervised Models: A Case Study of Child ASR," in IEEE Journal of Selected Topics in Signal Processing, 2022.

**Ruchao Fan**, and Abeer Alwan, "DRAFT: A Novel Framework to Reduce Domain Shifting in Self-supervised Learning and Its Application to Children's ASR", Interspeech 2022.

**Ruchao Fan**, Wei Chu, Peng Chang, Jing Xiao, and Abeer Alwan, "An Improved Single Step Non-autoregressive Transformer for Automatic Speech Recognition," Interspeech 2021.

**Ruchao Fan**, Amber Afshan and Abeer Alwan, "BI-APC: Bidirectional Autoregressive Predictive Coding for Unsupervised Pre-training and its Application to Children's ASR," ICASSP 2021.

**Ruchao Fan**, Wei Chu, Peng Chang and Jing Xiao, "CASS-NAT: CTC Alignment-Based Single Step Non-Autoregressive Transformer for Speech Recognition," ICASSP 2021.

# CHAPTER 1

# Introduction

## 1.1 Motivation

In the last decade, automatic speech recognition (ASR) systems have evolved rapidly from hybrid to end-to-end models[1]. The technical evolution largely benefits from the increase in computational power and the collection of large-scale supervised databases. For example, end-to-end models typically require tens of thousands of hours of speech-text pairs to perform better than hybrid models [CSW18]. In other words, when large amounts of speech databases are not available, the performance of end-to-end ASR models might degrade significantly. Child speech falls into this category because of the difficulty of collecting speech-text pair data due to privacy and logistical issues. Additionally, the attention-based encoder decoder model (AED) [CSW18], one of the most popular end-to-end models, adopts an autoregressive mechanism for transcription generation, where each generated token is conditioned on all previous tokens, following the Markov assumption. However, such a mechanism slows down the inference speed, aka. real time factor (RTF), which is an essential metric for designing efficient ASR systems.

With the increasing use of voice-based educational technology, better child ASR systems are needed because speech is one of the only mechanisms young children have to interact with such devices due to their limited reading, writing, and typing abilities. However, child ASR is difficult due to, in part, the lack of large child speech databases, and can be treated as a low-resource task. The larger inter- and intra-speaker variability in child speech, compared to

[1]Hybrid models indicate the combination of a deep neural network-based acoustic model and Hidden Markov Model (HMM). End-to-end models are one single neural network to jointly optimize the acoustic, pronunciation, and language models.

adults, makes child ASR even more challenging and unique compared to other low-resource tasks. In addition, on-device deployment of an ASR system is becoming necessary for children because of privacy issues and accessibility. Decoding efficiency is one of the most important factors to make system deployment on chips feasible. The AED model might not be suitable in this case because of the autoregressive generation mechanism.

In this dissertation, we aim to address the above issues from two directions: a) improving the accuracy of a child ASR system by mitigating the data scarcity issue; and b) improving the inference efficiency of a child ASR system by proposing a novel non-autoregressive transformer framework. The proposed methods are eventually combined to achieve a good-performing ASR system for child speech in terms of both accuracy and efficiency. Moreover, child ASR is always lacking a benchmark for fair comparisons with different methods. To reduce this gap, we benchmark the widely used open-sourced speech foundation models to date and compare their performance on multiple child speech datasets. Other topics such as data augmentation and parameter efficient finetuning are also discussed in the benchmark. We hope the benchmark and the released code can facilitate the development of robust child ASR.

We will briefly introduce the related techniques in this chapter, including ASR models overview, self-supervised learning methods and non-autoregressive mechanisms.

## 1.2 Automatic Speech Recognition

Automatic Speech Recognition (ASR) transcribes human speech waveform with word sequences. Suppose the input speech is represented by $\boldsymbol{X} = \{x_1, x_2, ..., x_T\}$, ASR systems attempt to find the most probable word sequence $\boldsymbol{Y}$ for $P(\boldsymbol{Y}|\boldsymbol{X})$.

$$\hat{\boldsymbol{Y}} = \underset{\boldsymbol{Y}}{\text{argmax}} P(\boldsymbol{Y}|\boldsymbol{X}) \tag{1.1}$$

The pipeline of widely used ASR systems is shown in Figure 1.1. Typically, the input speech is first passed through a feature extractor to generate acoustic features, such as MFCC and log-mel filter-bank. The extracted features are used by ASR models for sequence classification.

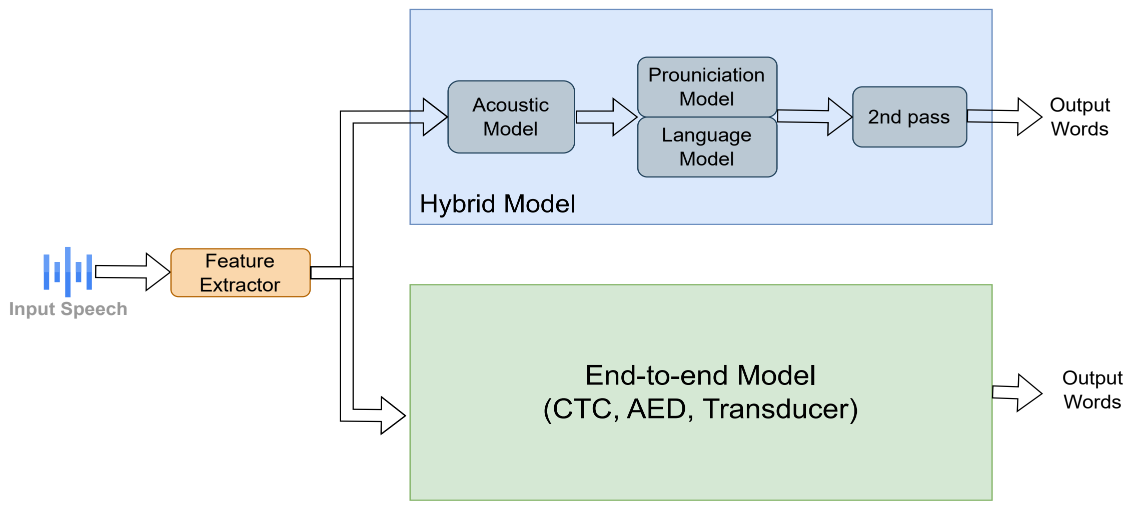


Figure 1.1: The pipeline of widely used ASR systems including hybrid model, Connectionist Temporal Classification (CTC), Attention-based Encoder Decoder(AED), and Transducer Model.

We will briefly introduce the relevant techniques to this section.

### 1.2.1 Hybrid and End-to-end Models

The earliest large-scale continuous ASR system is the HMM-GMM system [JR85]. In HMM-GMM, acoustic statistics of the input speech features are modelled by a Gaussian Mixture Model (GMM) conditioned on the spoken sound units, which are usually represented by states in the Hidden Markov Model (HMM). This statistical model is also known as the acoustic model (AM). After acoustic modeling, the HMM states are merged to a phoneme sequence through the decision tree and then to a word sequence through the lexicon (also known as pronunciation model, PM) that contains the mapping between word and phoneme sequence. The language model (LM) (typically n-gram models in the HMM system) is finally integrated into a decoding graph (known as Weighted Finite State Transducer, WFST) with the acoustic model score. The most probable word sequence is obtained by searching the WFST using dynamic programming algorithms. Formally, the posterior probability $P(Y|X)$ can be decomposed into two terms using Bayes Rule, represented by the acoustic and language model, respectively.

$$\hat{\boldsymbol{Y}} = \underset{\boldsymbol{Y}}{argmax} P_{AM}(\boldsymbol{X}|\boldsymbol{Y}) P_{LM}(\boldsymbol{Y}) \tag{1.2}$$

In HMM-GMM systems, GMM outputs frame-level conditional probability $P(x_t|s_t)$ and the HMM maintains sequential properties of the speech signal by constructing transition probabilities between HMM states $P(s_t|s_{t-1})$. This process can be described as follows:

$$\begin{aligned} P_{\boldsymbol{X}|\boldsymbol{Y}} &= \sum_{\boldsymbol{S}} P(\boldsymbol{X}, \boldsymbol{S}|\boldsymbol{Y}) \\ &= \sum_{\boldsymbol{S}} P(\boldsymbol{X}|\boldsymbol{S}, \boldsymbol{Y}) P(\boldsymbol{S}|\boldsymbol{Y}) \\ &= \sum_{\boldsymbol{S}} \prod_{t} P(x_t|s_t) P(s_t|s_{t-1}) P(\boldsymbol{S}|\boldsymbol{Y}) \end{aligned} \tag{1.3}$$

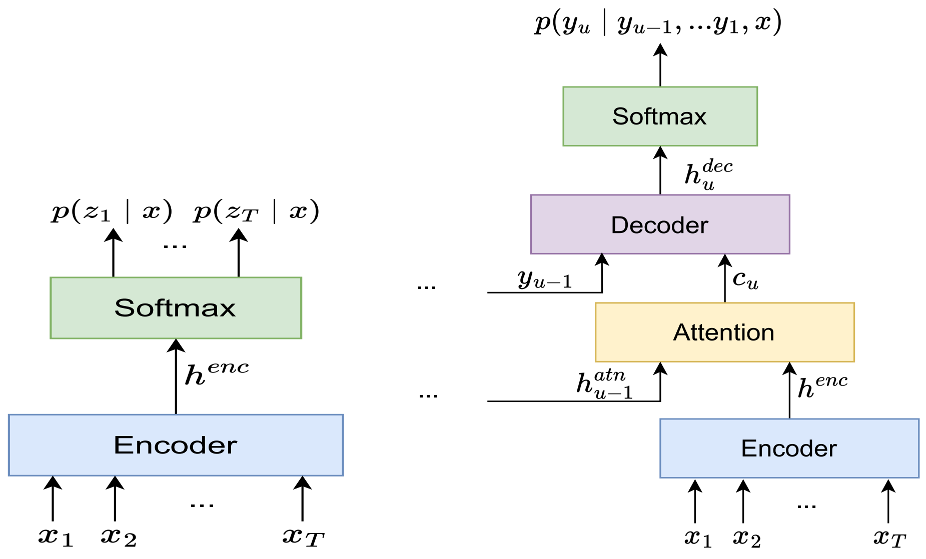
where $P(\boldsymbol{S}|\boldsymbol{Y})$ indicates the relations between the word sequence and the HMM state sequence , and is represented by the pronunciation model and decision tree.

With the development of deep neural networks, the acoustic model is found to perform better by using neural network models, e.g. a Deep Neural Network (DNN), Convolutional Neural Network (CNN), or Recurrent Neural Network (RNN). Such systems are known as hybrid models. Mathematically, the statistical modeling of $P(x_t|s_t)$ is replaced with a discriminative model that directly output the categorical distribution across all HMM states given the speech features as input, described as:

$$P(x_t|s_t) = \frac{P(s_t|x_t)P(x_t)}{P(s_t)} \tag{1.4}$$

where $P(s_t|x_t) = \mathrm{NN}(x_t)$, which is the neural network output.

More recently, it is shown that with enough training data, the components (AM, PM, and LM) in the hybrid model can be reduced to an end-to-end model, which directly models the posterior probability $P(\boldsymbol{Y}|\boldsymbol{X})$. There are three major methods for end-to-end modeling, including Connectionist Temporal Classification (CTC) [GFG06], Attention-based Encoder Decoder (AED) [CSW18] and Transducer models [Gra12]. Overall, AED is the best-performing end-to-end model. However, CTC has the advantages of fast generation of the entire sentence and the Transducer is suitable for the streaming use cases. We introduce CTC and AED because they are closely related to Chapter 4.1.



(a) Model Architecture of CTC (b) Attention-based Encoder Decoder

Figure 1.2: The model architecture of CTC and Attention-based Encoder Decoder (AED).

### 1.2.2 Connectionist Temporal Classification

End-to-end models are typically neural networks for sequential modeling such as long short-term memory (LSTM) and the transformer, where the length of an output sequence is the same as that of the input, as show in in Figure 1.2a. In CTC, the posterior probability of $Y$ given $X$ can be computed using the output probability at each frame. Let $Z = \{z_1, ..., z_t, ..., z_T\}$ be the output of the model, where $z_t$ stands for the output at time step $t$ corresponding to the input $x_t$. The length of $Z$, however, is always longer than that of $Y$ for a speech recognition task. To compute the loss, a special blank token $b$ is added in the vocabulary; $b$ can be predicted as an output of $Z$ at any time step. During inference, $b$ and repeated tokens are removed to obtain a shorter sequence $Y$, where this process can be defined as a mapping rule $\beta$. During training, there exist multiple $Z$s that can be mapped to $Y$ using the rule $\beta$. As a consequence, CTC loss is a summation of all such $Z$s, which is formulated as follows [GFG06]:

$$\begin{aligned} L_{CTC} &= -\log P(Y|X) = -\log \sum_{Z \in \beta^{-1}(Y)} P(Z|X) \\ &= -\log \sum_{Z \in \beta^{-1}(Y)} \prod_{t=1}^{T} P(z_t|X) \end{aligned} \tag{1.5}$$

where $\beta^{-1}$ is the inverse of the mapping rule. Note that output tokens are independent to each other in CTC, requiring external language model to be as competitive as the hybrid model.

### 1.2.3 Attention-based Encoder Decoder

To remove the output independent constraint in a CTC model, an attention-based encoder decoder (AED) is proposed, shown in Figure 1.2b. AED [CSW18] consists of three modules: encoder, decoder and attention. The AED encoder behaves the same as a CTC encoder for extracting high-level acoustic representations denoted as $h^{enc}$. The attention module selects the most relevant acoustic information in $h^{enc}$ at each generation step $u$ and generates content information $c_u$. The relevance is calculated through the similarities between the hidden token state $h_{u-1}$ and acoustic representations. Finally, the AED decoder generates the output sequence token by token, where each token is generated conditioned on all previous tokens and the content information $c_u$. This architecture design achieves sequence modelling by a chain of conditional probabilities, where each conditional probability constructs a classification problem. The AED model is trained through an objective function as follows:

$$\begin{aligned} L_{AT} &= -\log P(Y|X) = -\log \prod_{i=1}^{U} P(y_i|y_{<i}, X) \\ &= -\sum_{i=1}^{U} \log P(y_i|y_{<i}, X) \end{aligned} \tag{1.6}$$

where $P$ is the probability distribution of the AED model and $y_{<i}$ are all previous tokens before the $i^{th}$ token. The alignment between input speech features and the text sequence is implicitly modeled in the attention module. The encoder and decoder are similar to the acoustic and language model, respectively in the hybrid systems.

The most popular AED models are transformers [VSP17], where self-attention layers are used in the encoder and decoder module. The attention module is achieved by cross-attention layers where the query is token embedding and the key-value pairs are high-level acoustic representations. Please refer [VSP17] for more details of the transformer. The query, key and value are also introduced in Section 3.1.1.

### 1.2.4 Challenges for Child Speech and Corpus

Despite impressive advancement in developing automatic speech recognition (ASR) techniques in the last decade, children's ASR remains difficult. Challenges arise, in part, from difficulties in acoustic and language modeling of child speech. Due to different growth patterns of children and motor control issues, child speech has a higher degree of intra-speaker and inter-speaker acoustic variability than adult speech [LPN99]. Additionally, child speech is characterized by significant mispronunciations and disfluencies [YNF99, TTY20].

Another challenge is the lack of large-scale publicly-available child speech databases. However, some smaller-scale child speech databases are available to the public. The most common style of speech in these databases is read speech with single-word, phrase, or sentence transcripts such as in the OGI Kids Corpus [SHC00]. While databases of spontaneous styles of speech also exist, mispronunciations by children often make transcription unreliable or inconsistent, both within and across databases. Databases for English speakers include the OGI Kids' Speech Corpus (5-16 years old) [SHC00], CMU Kids Corpus (6-11 years old) [MJD21], PF-STAR Children's Speech Corpus (4-13 years old) [AMS05], and My Science Tutor (MYST, 8-10 years old) [WCP19]. However, unlike adult speech databases that have thousands of hours of speech data, child speech databases generally have at most tens or hundreds (less than three hundreds) of hours of speech data. Additionally, due to the lack of child speech data, when training ASR systems using child speech databases, it is usually necessary to consider specific tasks or applications, such as using speech data from children in educational settings to train ASR for classroom appropriate speech technology and thus child ASR can be treated as a low-resource task [WZF21].

## 1.3 Self-supervised Learning for Speech Recognition

Recently, self-supervised learning (SSL) from speech data has been investigated [CW21, ZPH22, WW21, CZ21, AWZ21, JL21, WTL20, LY20, LLL21] because of its great potential of improving low-resource tasks through learning prior knowledge from large amounts of data without annotations. SSL models can be used in two manners: 1) feature extraction to replace human-designed features [YC21, EN21, CM21]; and 2) model initialization for finetuning downstream tasks [VMB21, MH21]. The idea of SSL is to design pseudo tasks (with self-supervision) for training deep neural networks (DNN) and then transfer the learned knowledge to a downstream supervised task. For example, autoregressive predictive coding (APC) uses temporally-shifted sequences to perform prediction such that the model predicts future frames from previous frames [CH19, CG20a, RF20]. Wav2vec-based methods are implemented to include negative samples, and a contrastive loss is utilized to increase the distance from the output to negative samples and decrease that distance to the positive sample [OLV18, SBC19, BZM20, BSA19]. The positive sample is the frame being masked (to be predicted), and negative samples are the unmasked frames in the utterance. A more recent SSL framework, HuBERT [HT21, HBT21], creates the pseudo-label of each speech frame using clustering techniques like K-means. Other variants such as W2vBERT [CZ21], WavLM [CW21], BESTRQ [CQZ22] are proposed for learning better self-supervised representations. These methods have been shown to be effective for low-resource ASR tasks such as low-resource languages [RJM20, YWC20], noisy speech [WLW22] and accented speech [LMC21].

However, a weakness of SSL training is domain shifting that happens when the domain of the finetuning data is different than that of the pretraining data [MCL22, HS21]. Although a performance improvement can be observed when the magnitude of the pretraining data is large enough, previous work has shown that additional gains can be obtained by including target domain data in the ASR pretraining stage [HS21, HMH22]. But including target domain data would be impractical if we are not aware of the finetuning task at the pretraining stage. In addition, retraining a large-scale SSL model with both the source and target domain data to address domain shifting may not always be possible or computationally efficient. Hence, investigating adaptation methods for SSL is gaining attention for work involving out-of-domain low-resource tasks. Previous studies proposed to perform adaptation of supervised models either during or after the finetuning stage [KLG22,HHS21]. No additional adaptation stage of self-supervised models has been investigated before for domain shifting in SSL methods. We propose a novel method to the reduce the domain mismatch between the pretraining and finetuning stage in Section 2.1.3.

### 1.3.1 A General Self-supervised Learning Framework

Self-supervised learning (SSL) learns useful speech representations for downstream tasks without explicit supervision. After pretraining, the model can be used for model initialization for downstream tasks. We summarize a general framework for various SSL methods, which is illustrated in Figure 1.3.

Let $X = (x_1, ..., x_i, ..., x_n)$ denote the raw waveform of an utterance, where each $x_i$ is a sampled data point. Self-supervised learning methods first extract representations $Z = (z_1, ..., z_t, ..., z_T)$ for each frame $t$ using a function $h$, which in general can be either a human-designed function, like an MFCC extractor [Har78], or a learned deep network, like a convolution neural network [LB95]. There may be a special case when $X$ represents human-designed spectral features. Then $h$ is the module that maps the spectral features to latent representations for prediction. A backbone model $f$, parameterized with $\theta$, is then used to build contextualized representations. A generator $g$ finally converts the contextualized representations into a prediction space with a pre-defined dimension and outputs $Y = (y_1, ..., y_t, ..., y_T)$. An operation $O$ over speech representation $Z$ is designed to obtain a pseudo-label for the task. The key idea of SSL is to construct a loss function $L$ between $O(Z)$ and model output $Y$, ensuring that no information leaking appears in the forward computation so that trivial solutions are ignored during the optimization process. As a result, the SSL objective function is:

$$L_{\text{SSL}} = L(f(h(X)), O(h(X))) \tag{1.7}$$

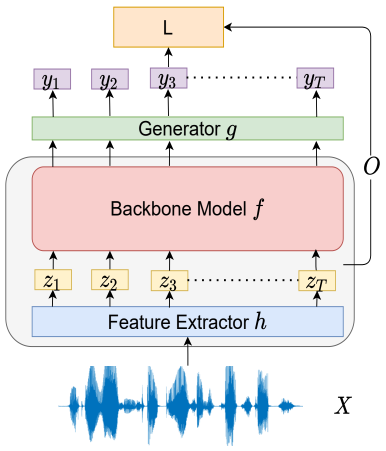


Figure 1.3: Proposed self-supervised learning framework. $h$ is a function to extract speech representation $Z$. $O$ is an operation over $Z$. $f$ is the backbone model for pretraining. $g$ is a generator that maps the output of the backbone model to have the same dimension as $O(Z)$ and outputs $Y$. $L$ computes the SSL loss using $Y$ and $O(Z)$.

We omit $g$ for simplicity because it can be regarded as a part of $f$. The main differences between various SSL methods are the definition of $O$ for obtaining a supervision and of $L$ as an optimization objective.

In this section, we discuss how SSL methods can be used in our proposed framework. These methods include autoregressive predictive coding [CH19,CG20a], Wav2vec2.0 [BZM20] and hidden unit BERT (HuBERT) [HT21, HBT21]. Other variants are developed based on the three models and can be adapted to the framework accordingly.

### 1.3.2 Autoregressive Predictive Coding

Autoregressive predictive coding (APC) uses human-designed features as model input ($Z = h(X)$). Typically, 80-dimensional log-mel filter-bank features are used as $Z$. APC utilizes a temporally-shifted sequence to predict the frame $n$ steps ahead of the current frame given all previous frames. As a consequence, the operation $O$ with a temporal lag of $n$ up to the time step $T - n$ satisfies $O_n(\{z_1, z_2, ..., z_{T-n}\}) = \{z_{1+n}, z_{2+n}, ..., z_T\}$. Since $Z$ vectors are not latent representations in APC, the $L_p$ norm distance can be used as the loss function. The final objective function is formulated as follows:

$$L_{\mathrm{APC}} = L_p(f(Z), O_n(Z)) = \sum_{t=1}^{T-n}(|y_t - z_{t+n}|_p) \tag{1.8}$$

where $n$ is fixed as a hyper-parameter. APC essentially adopts neural language model style training using speech features instead of word embeddings. The mechanism is suitable for online speech recognition model pretraining because APC considers information from only one direction. It is, however, not suitable for bidirectional model pretraining. In this paper, we conduct experiments to explore whether APC can be extended to bidirectional model pretraining.

### 1.3.3 Wav2vec2.0

Wav2vec2.0 has evolved from contrastive predictive coding (CPC) [OLV18], Wav2vec [SBC19], and Vq-Wav2vec [BSA19, BAM20]. We only study Wav2vec2.0 because of its better performance.

Wav2vec2.0 uses raw waveforms as model inputs, which means $h$ is a parameterized model for learning feature extraction. $h$ consists of multiple blocks of temporal convolution layers with a total stride that decreases the sequence length from the number of sampled points to the number of frames. Different from APC, Wav2vec2.0 adopts masked language model (MLM) [DCL19] style training, where the backbone model $f$ tries to reconstruct masked speech representations. Let $M$ be the mask operation onto speech representation $Z$. We define $Z_1$ as the original tokens that will be masked by $M$, and $Z_2$ as the original tokens that will not be masked. Then, when applying $M$ to $Z$, we obtain $Z_{mask}$ and $Z_{obs}$ as the masked and unmasked tokens. The corresponding outputs are referred to as $Y_{mask}$ and $Y_{obs}$. Only $Y_{mask}$ contributes to the loss computation. Hence, the forward computation of $f$ could be formulated as $Y_{mask} = f(Z_{mask} \oplus Z_{obs}) - Y_{obs}$. Suppose the length of the masked proportion is $U$, we write $Y_{mask}$ as $\{y^1_{mask}, y^2_{mask}, ..., y^U_{mask}\}$. Since $Z$ are latent representations, the contrastive loss is preferred so that the true latent is distinguished from distractors. A vector-quantization (vq) layer is also inserted after $Z$ to obtain more compact representations for supervision so that the model can learn more efficiently. The operation $O$ can be summarized as $O(Z) = \text{vq}(Z_1) \oplus \text{Sample}(\text{vq}(Z_2)) = \{(q^1_{pos}, Q^1_{neg}), (q^1_{pos}, Q^1_{neg}), ..., (q^U_{pos}, Q^U_{neg})\}$, where $q_{pos}$ is a positive sample after vq layers, and $Q_{neg}$ is a set for negative samples as distractors in the contrastive loss. The objective function is formulated as:

$$\begin{aligned} L_{\text{wav2vec2.0}} &= L_{ctras}(Y_{mask}, O(Z)) \\ &= -\sum_u log \frac{\exp(sim(y^u_{mask}, q^u_{pos}))}{\sum_{q^u_{neg} \in Q^u_{neg}} \exp(sim(y^u_{mask}, q^u_{neg}))} \end{aligned} \tag{1.9}$$

where $sim(a, b)$ is the cosine similarity between context representation $Y_{mask}$ and quantized latent representations $O(Z)$. There is also an additional diversity loss in Wav2vec2.0. Since the observed sequence has information from both directions for most masked frames, Wav2vec2.0 is suitable for bidirectional model pretraining. However, Wav2vec2.0 always requires more training iterations than APC [HMH22]. This is because only a portion of the frames are masked for prediction in Wav2vec2.0 and the mask regions are different each time the sequence is trained.

### 1.3.4 HuBERT

Hidden unit BERT (HuBERT) uses the same masked language model style training as Wav2vec2.0. Differently, HuBERT does not require negative samples. Instead, it introduces an acoustic unit discovery process before the pretraining stage. For example, the most useful strategy in HuBERT [HBT21] is performing K-means on MFCC features or intermediate model outputs to obtain a pseudo-label category for each frame. HuBERT creates a learned embedding for each category. The embedding of the true category is equivalent to the positive sample in Wav2vec2.0 and all other embeddings are essentially negative samples. Thus, the operation $O$ is an unit discovery process for HuBERT. The loss computation is similar to the fine-tuning task. If we define $O(Z) = (c_1, c_2, ..., c_T)$, where $c_t$ is the pseudo-category for each frame, the objective function is computed as a weighted sum of both the masked and observed output sequences.

$$\begin{aligned}
L_{\text{HuBERT}} &= L(Y, O(Z)) \\
&= \alpha L(Y_{mask}, O(Z_{mask})) + (1-\alpha) L(Y_{obs}, O(Z_{obs})) \\
&= -\alpha \sum_{c_t \in O(Z_{mask})} \log P(c_t|Z) - \\
(1-\alpha) &\sum_{c_t \in O(Z_{obs})} \log P(c_t|Z)
\end{aligned} \tag{1.10}$$

where $\alpha$ is the task ratio. Similar to [HT21, HBT21], we use $\alpha = 1$ because we directly use the open-sourced pretrained HuBERT model [OEB19] as initialization for children's ASR training. More importantly, we apply the proposed domain adaptation technique on these models to show its general effectiveness.

## 1.4 Non-autoregressive Transformer for Speech Recognition

End-to-end models have proven successful for speech recognition because of their ability to include implicitly the role of the acoustic, pronunciation, and language models into one single neural network [LWG20, Li21]. Training the above components together leads to fewer intermediate errors and thus a lower word error rate (WER) for ASR systems. This training mechanism also requires fewer model parameters, which is suitable for on-device deployment. As we introduced earlier, CTC, AED and Transducers are the most widely used end-to-end models. CTC has a high decoding efficiency when using the best path decoding strategy, but it is restricted by its assumption of conditionally independent outputs. AED, such as the autoregressive transformer (AT) [VSP17, DXX18], models output dependencies by incorporating a language-model-style decoder. However, the decoding in AT adopts an autoregressive mechanism for joint probability factorization, leading to a step-by-step generation of output tokens as shown in Figure 1.4. Such a mechanism lowers the inference speed for ASR, which is an essential factor when designing an efficient ASR system. Recently, non-autoregressive mechanisms have received increasing attention for their decoding efficiency, enabled by generating output tokens in parallel [GBX18,LMC20,SCS20,QGJ21,CWV20,HWC20], as shown in Figure 1.4.

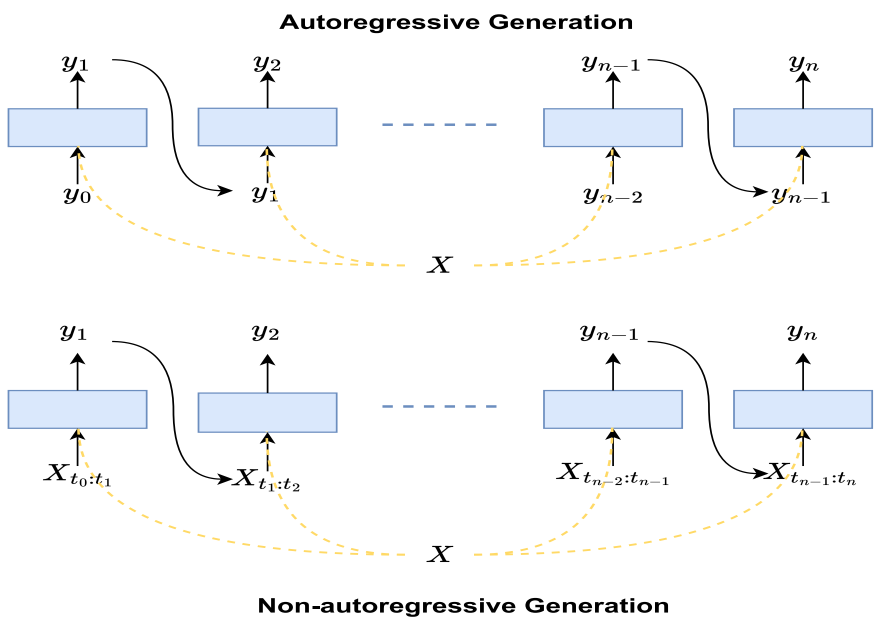


Figure 1.4: An illustration of autoregressive and non-autoregressive generation conditioned on acoustic features $X$. $t_0 : t_1$ indicates the first segment of the sequence $X$

More formally, non-autoregressive models have no strict dependencies between tokens. For example, CTC has a conditional independence assumption, while Mask-CTC [HWC20] trains the decoder as a masked language model to build a weak dependency between masked and unmasked tokens. By relaxing the dependency assumption, it is possible to generate all tokens in parallel, and thus increase inference speed. Single-step NATs use the encoder-decoder structure, making the output the same length as $Y$. The output tokens are still conditionally independent of each other, just like the CTC. But there is an implicit assumption that language semantics can be captured by high-level acoustic representations, which are similar to word embeddings. The model is updated using a cross entropy loss for each token, which can be formulated as:

$$L_{NAT} = -\log P(Y|X) = -\sum_{i=1}^{U} \log P(y_i|X) \tag{1.11}$$

Because of the strict output dependency modeling, the performance of the autoregressive models are theoretically the upper bound of the performance of their non-autoregressive counterparts.

There are two major types of non-autoregressive methods for the encoder-decoder-based Transformers (NAT): (i) iterative NATs, and (ii) single-step NATs or one-shot NATs. The prevailing iterative NATs relax the strict non-autoregressive condition and iteratively generate outputs with $K$ decoding passes. Thus, iterative NATs are sometimes called “semi-NAT”. Single-step NATs, however, can generate the output sequence in one iteration. Different from the methods in neural machine translation that extend encoder input as the decoder input, single-step NATs for speech recognition extract high-level acoustic representations as the decoder input, assuming that language semantics can be captured by the acoustic representations [BYT21, TYT20, YLG21]. As shown in Figure 1.4, the frame-level acoustic features are processed into higher level token-level acoustic representations using additional segment information. However, the acoustic representations in previous works are either implicit, extracted by an attention mechanism [BYT21] or incomplete usage of CTC spikes [TYT20], which make learning language semantics difficult. In Chapter 3, we propose a novel CTC-alignment based NAT model (CASS-NAT) to extract more accurate token-level acoustic representations and reduce the gap between the autoregressive and non-autoregressive models.

In addition to the encoder-decoder-based NAT models, non-autoregressive methods continue to be proposed based on encoder-based CTC models. Chi et al. proposed to train a refiner to iteratively improve CTC alignment based on the previous outputs of the refiner [CSK20]. In [CSH20], CTC alignment is enhanced with a mask token as a prior information for the decoder in each iteration. Nozaki et al. alleviate the output-independent problem of CTC by using intermediate predictions as additional inputs [NK21]. The advantages of the CTC models are their simplicity when using greedy search decoding [CZM21] and their convenience to be integrated with SSL pretraining. For example, [NCZ21] improves the WER performance of a pure CTC model by a large margin using Wav2vec2.0 pretraining techniques. We design an encoder-only CASS-NAT model to combine the advantage of the CTC and encoder-decoder NAT models in Chapter 4.

## 1.5 Other Related Works

### 1.5.1 Convolution-augmented self-attention

The self-attention module in transformers captures global information by a weighted summation of the whole sequence. However, local information is also important for sequence modelling. Taking speech features as an example, frequency details in each vowel or consonant helps the recognition of these sounds. In computer vision, convolution layers have proved to be successful at capturing local details within a kernel [BZV19]. Recent works adopt this idea and augment transformers with a convolution module [YDL18, GQC20, HZZ20] in ASR. Relative positional encoding is also used in each self-attention module. The convolution-augmented self-attention block can effectively improve performance, but the improvement is only significant if applied in the AT encoder. The AT decoder, on the other hand, adopts a causal structure (upper triangular mask matrix for attention) that captures less local details with a convolution module, and thus the improvement would not be significant.

### 1.5.2 Intermediate Loss

Deep transformers always suffer from gradient vanishing, especially for parameters that are distant from the output layers. Intermediate loss has previously been proposed to add additional loss functions after each layer to boost the gradient update [SLJ15, TLZ20]. It has been proven useful to use intermediate CTC loss for improving the performance of the CTC model [LW21, LKW21]. In [WML20], intermediate CE loss is used for training deep transformer-based acoustic models for an HMM-based hybrid ASR system.

## 1.6 Dissertation Overview

This dissertation makes several contributions to the development of child ASR systems. First, we explore the effectiveness of the self-supervised learning methods in child ASR with un-annotated adult speech data. Specifically, we propose a novel SSL pretraining method

(Bi-APC) and a better finetuning framework (DRAFT) to improve the accuracy of child ASR systems. Second, we propose a novel non-autoregressive transformer model (CASS-NAT) and its variant with a single encoder (Unienc-CASSNAT) to improve the inference efficiency for the ASR systems. The improved inference efficiency is especially important to child ASR for on-device deployment. Lastly, a child ASR benchmark on the widely used child speech databases, OGI [SHC00] and MyST [WCP19], is created with the comparisons of different data augmentation techniques, parameter efficient tuning methods, and various open-sourced large speech foundation models. The benchmark will facilitate broader child ASR research and show the potential directions in the future.

The remainder of the dissertation is organized as follows. Chapter 2 describes the proposed SSL methods for improving the accuracy of child ASR including Bi-APC pretraining and DRAFT. Chapter 3 introduces the novel CTC-alignement based single step non-autoregressive transformer (CASS-NAT) to improve the inference efficiency for the child ASR. Chapter 4 further introduces an uni-encoder variant of CASS-NAT, which can better integrate the SSL pretraining and the non-autoregressive generation. Chapter 5 includes the child ASR benchmark we created and insights we obtained from the benchmark. Finally, Chapter 6 concludes the dissertation with a brief summary, and directions for future work.

# CHAPTER 2

# Improving the Accuracy with SSL Pretraining

Due to low-resource limitations, the word error rate (WER) of child ASR systems is usually higher than that of adult ASR systems. In this chapter, we introduce novel self-supervised learning (SSL) techniques to improve the accuracy of child ASR systems. The SSL models are typically trained on adult speech and then finetuned on child speech. Specifically, we explore methods to improve SSL models in both the pretraining and finetuning stages.

In the pretraining stage, a well-known method, autoregressive predictive coding (APC), predicts the speech features of the next frame given previous frames (left-to-right prediction). Considering the contextual property of a speech sequence, a bidirectional APC (Bi-APC) method is proposed to also include a right-to-left prediction with multiple temporally-shifted sequences as targets for multi-task training. To examine the effectiveness of the proposed method, the Bi-APC is applied on both the Bi-LSTM and non-causal transformer models because of their different organization of parameters in the forward and reversed paths. We investigate whether parameter organization in the forward and reversed computations affect Bi-APC performance.

In the finetuning stage, we propose a domain responsible adaptation and finetuning (DRAFT) framework to address the domain shifting problem in SSL models that are pre-trained on adult speech data. In DRAFT, residual adapters are placed between transformer blocks and are responsible for learning domain-related information at an additional adaptation stage with the child speech data. Only residual adapters are updated during the adaptation stage so that the knowledge learned from source domain data can be retained. The effectiveness of the proposed DRAFT are examined on Bi-APC, and the widely used Wav2vec2.0 [BZM20] and HuBERT [HT21] models. The works in this chapter were published in [FAA21, FA22, FZW22].

## 2.1 Methods

In this section, we first review the basics of APC and then introduce the proposed Bi-APC and their extended versions with multiple temporally-shifted target sequences. We end the section by introducing DRAFT, the proposed adaptation framework for self-supervised pretrained models.

### 2.1.1 Bi-APC: Bidirectional Autoregressive Predictive Coding

As introduced in Section 1.3.2, APC adopts a left-to-right prediction from $Z_1$ to $Z_T$, and hence is well suited for uni-directional structures such as uni-LSTM or causal transformers. However, bidirectional models usually provide better WER performance than their unidirectional counterparts because they learn from both directions [ZDV17, GJM13]. Therefore, we propose a bidirectional APC (Bi-APC), which extends APC to exploit its potential for bidirectional pretraining. The idea of Bi-APC is to add a reversed version (right-to-left prediction) of APC, where we predict the frame $n$ steps behind the current frame given all future frames. However, the mechanism and the applications of BiAPC on LSTM and transformer for bidirectional models are different.

Figure 2.1 shows how to use Bi-APC for BLSTM pretraining. To prevent equivalent mapping in the network, the outputs of the BLSTM should not contain information about the corresponding supervisions. We, therefore, split the BLSTM into forward-related and reverse-related parts as shown in red parts and blue parts in Figure 2.1, respectively, including the parameters (arrows) and outputs (rectangles) at each layer. When computing the outputs $Y^{fwd} = \{y_1^{fwd}, y_2^{fwd}, \ldots, y_T^{fwd}\}$ in the forward direction, the values of the blue rectangles are set to zero to exclude the information that are extracted from the frames on the right side. The reversed-related parameters are also not updated. The same strategies are used in the computation of outputs $Y^{rev} = \{y_1^{rev}, y_2^{rev}, \ldots, y_T^{rev}\}$ in the reversed direction. The parameters represented by black arrows are not used in pretraining since they allow for

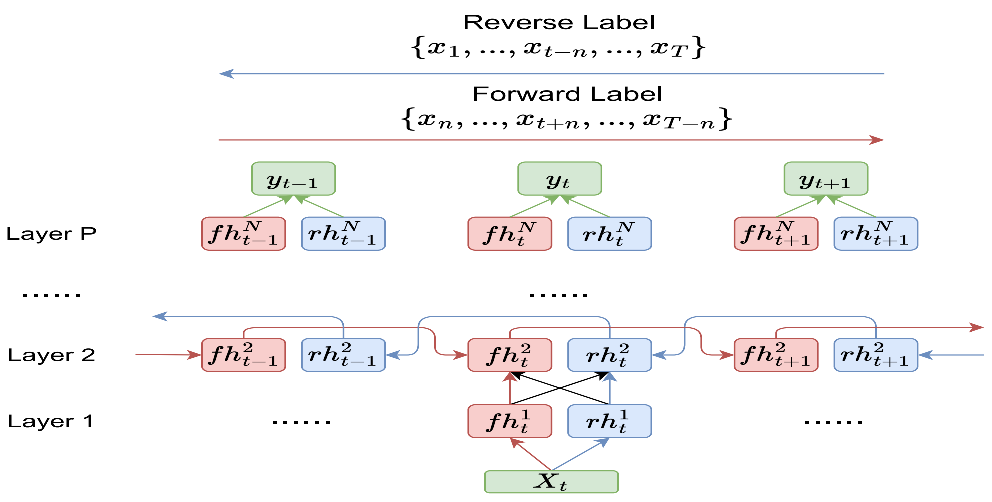


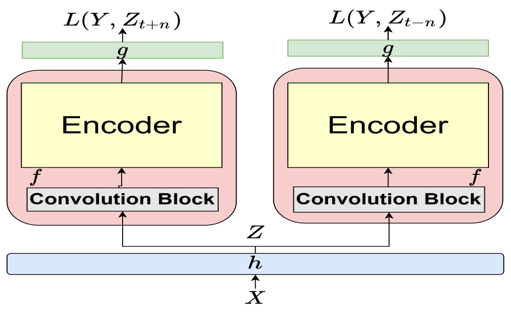
Figure 2.1: Illustration of Bi-APC pretraining for BLSTM. Red and blue parts are the forward-related and reversed-related parameters and computations, respectively. $fh$ and $rh$ indicate the hidden states of the forward and reversed calculations, respectively. Green boxes are model input and outputs, respectively. Black arrows represent the forward computations in a BLSTM model but are not used in the pretraining.

an illegal information exchange from different directions. The green arrows are the shared parameters which are not used in finetuning. The BLSTM is then pretrained by optimizing the APC from both directions as:

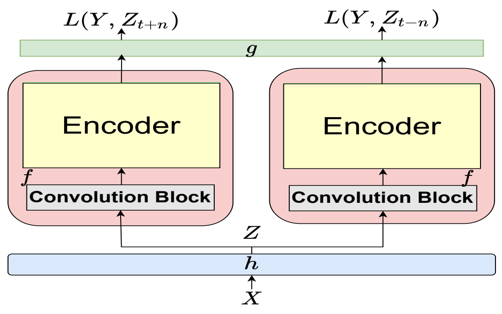
$$L_{\text{Bi-APC}} = 0.5 \cdot \sum_{t=1}^{T-n} |h_{t+n} - y_t^{fwd}| + 0.5 \cdot \sum_{t=n+1}^{T} |x_{t-n} - y_t^{rev}| \tag{2.1}$$

where task ratios are set to 0.5 as both directions have the same importance. Note that we can also train an APC with uni-LSTM and only initialize the parameters of the red parts in Figure 2.1 for BLSTM. We still denote such pretraining as APC in the experimental results.

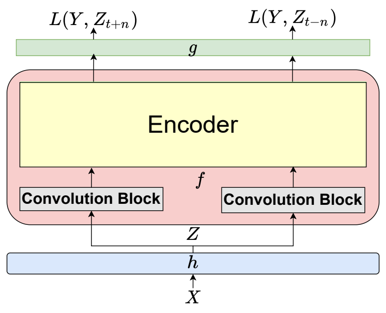
The parameters in non-causal transformers, however, are not separated for contextual modeling from both directions. It is unknown whether the parameters would affect the learning of individual APC loss in two directions. In the widely-used pretrained transformer model [HT21, BZM20], there are three major modules: convolution block, encoder, and generator. We can assume each module to have two copies, where one trains a causal transformer using a left-to-right APC and the other trains a causal transformer using a



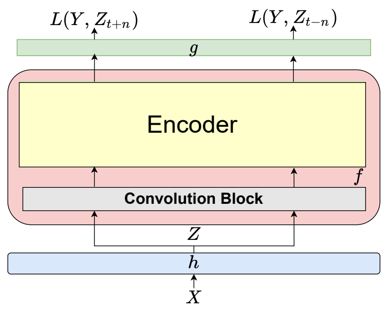
(a) No Sharing



(b) Share Generator $g$



(c) Share $g$ and Encoder



(d) Share All

Figure 2.2: Various solutions for training non-causal transformers with Bi-APC. Notations and blocks are consistent with those described in Section 1.3.1.

right-to-left APC. The trained modules are then averaged to be the final model initialization for the finetuning task. Parameters of the two copies could also be shared so that averaging is not needed after pretraining. As a result, we explore four various Bi-APC pretraining schemes for non-causal transformers as shown in Figure 2.2: 1) no modules are shared; 2) only the generator is shared; 3) only the convolution block is not shared and 4) all modules are shared, which is similar to [CZS20]. The four solutions are selected based on the number of modules shared from top to bottom during Bi-APC pretraining. By exploring the four Bi-APC pretraining schemes, we can understand how the shared parameters affect APC pretraining in two directions and whether Bi-APC framework is suitable for models with shared parameters for bidirectional contextual modelling. Note that we use causal convolution layers in the convolution block and a causal mask in each self-attention layer in the encoder.

Another way of incorporating bidirectional contextual information from pretraining could be averaging or concatenating outputs from two pretrained causal transformers as investigated in [LLS20]. However, averaging or concatenating outputs will result in doubling model parameters with the goal of finding good speech representations for downstream tasks. In our case, we aim to find a good initialization of non-causal transformers for finetuning.

### 2.1.2 Extensions to APC and Bi-APC

The original APC technique [CH19] uses one temporally-shifted sequence $Z_{t+n}$ during pretraining, as shown in Eq.1.8. A model may learn differently with different temporal lags, aka. different values of $n$. For example, the model learns to exploit local smoothness of the signal with a small value of $n$, while it learns a global structure with a large value of $n$. Hence, it is intuitive to include multiple temporally shifted sequences with different lags during pretraining and reformulate APC as a multi-task training loss. If we regard Eq.1.8 as $L^n_{\text{APC}}$, the extension of APC (E-APC) has the following objective function:

$$L_{\text{E-APC}} = \sum_{n=s}^{s+k} L^n_{\text{APC}} = \sum_{n=s}^{s+k} \sum_{t=1}^{T-n} (|y_t - z_{t+n}|_p) \tag{2.2}$$

where $s$ is the temporally-shifted sequence with the smallest value of $n$ and $k$ is the number of consecutive temporally-shifted sequences used in pretraining. During implementation, the backbone model $f$ is shared across different tasks while each task has its own generator $g$.

A recent paper also considered multiple targets for better APC pretraining [CG20b] where the auxiliary predictive loss with the same temporal lag as the original APC loss is proposed based on an additional RNN for regularization. In our work, however, only one model (an RNN or transformer) is used to learn various temporal lags.

The parameters of BLSTM are designed to be separated into a left-to-right and right-to-left context modelling LSTMs. When the extension of APC is also applied to Bi-APC, we can re-write the objective function of the Bi-APC in Eq. 2.1 as follows:

$$\begin{aligned} L_{\text{E-BiAPC}} &= 0.5 \cdot \sum_{n=s}^{s+k} L^n_{\text{APC}} + 0.5 \cdot \sum_{n=s}^{s+k} L^{-n}_{\text{APC}} \\ &= 0.5 \cdot \sum_{n=s}^{s+k} \sum_{t=1}^{T-n} (|y_t - z_{t+n}|_p) + 0.5 \cdot \sum_{n=s}^{s+k} \sum_{t=n+1}^{T} (|y_t - z_{t-n}|_p) \end{aligned} \tag{2.3}$$

### 2.1.3 DRAFT: Reducing Domain Shifting in SSL Pretrained Models

Finetuning a well-pretrained self-supervised model can provide good WER performance for low-resource speech recognition, e.g. child ASR. Howeve, performance improvement could potentially be limited due to domain mismatch between the pretraining (adult speech) and finetuning (child speech) data. Previous work [HS21] has shown that including data from the target domain in the pretraining stage can improve the performance of the target task. However, that approach requires the knowledge of the target domain and then re-training the self-supervised model with a larger amount of data, which is time-consuming and computationally expensive. It would be more practical to adapt the pretrained models with target data only when the target domain is unknown in the pretraining stage. In this section, we propose DRAFT, a domain responsible adaptation and finetuning framework, to alleviate domain shifting in the conventional self-supervised pretraining and finetuning paradigm. DRAFT is a three-stage training paradigm with residual adapters inserted in the backbone model $f$. The residual adapters are designed to learn knowledge from the target domain data.

#### 2.1.3.1 Simple Adaptation for Finetuning (SAFT)

Before introducing DRAFT, we would like to discuss the most direct way of doing adaptation by re-training the pretrained models, which we refer to as simple adaptation and finetuning (SAFT). In SAFT, an adaptation stage is inserted between the pretraining and finetuning stage, and the model is adapted with an SSL loss and with the finetuning data. All model parameters are updated at the adaptation stage with a smaller learning rate than the pretraining stage to prevent overfitting. The model after the adaptation stage is used as initialization for the finetuning stage with an ASR loss.

#### 2.1.3.2 Domain Responsible Adapters for Finetuning (DRAFT)

SAFT updates the parameters of the entire model, and thus overfitting might occur, leading to a catastrophic forgetting for the self-supervised model [Fre99,KTK21,CLL21]. Knowledge

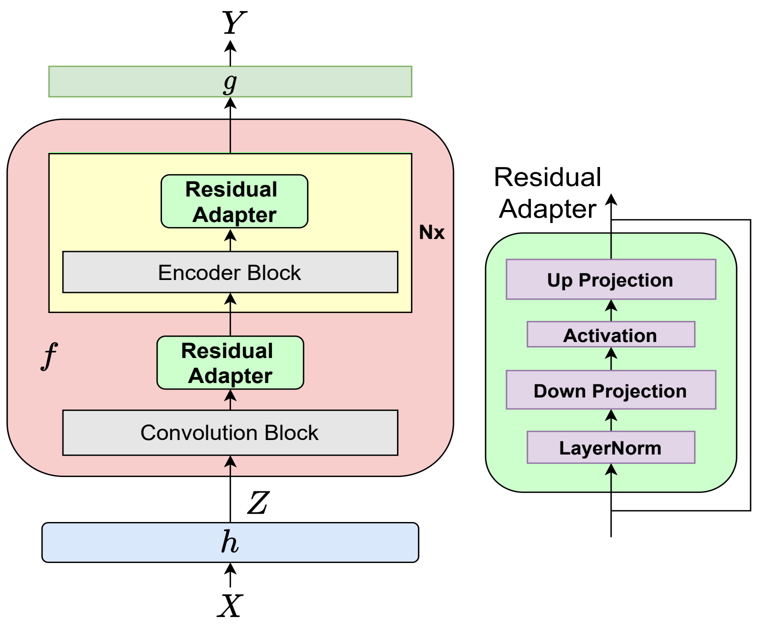


Figure 2.3: Structure of the backbone model $f$ with residual adapters inserted after each block. $N$x indicates that the module can be repeated $N$ times (proportional to the number of encoder blocks). $X, h, Z, f, g$ and $Y$ are the same as those in Figure 1.3. The right side of the figure shows the components in a residual adapter.

learned at the pretraining stage may diminish because of an aggressive learning strategy. Not only does the domain shifting problem remain unsolved, but also it leads to performance degradation. To address this issue, we propose domain responsible adapters for finetuning (DRAFT) framework that uses residual adapters in the backbone model to learn from target domain data while retaining the source domain knowledge.

We start with a description of residual adapters (RAs) (shown in Figure 2.3) since they are the most important modules in DRAFT. Specifically, an RA consists of two feed-forward layers with a residual connection. The activation function between the two feed-forward layers makes the adapter non-linear. A layer normalization is inserted at the beginning, which is similar to a self-attention block. We define the dimension of the first feed-forward layer output as $d_{ada}$, which determines the number of parameters of the RA. The effect of $d_{ada}$ on performance will be analysed in Section 2.2.3.2. Note that RA can be placed anywhere in the model. In our case, we insert one residual adapter after the convolution block and one after each encoder block as shown in Figure 2.3 and Figure 2.4. We assume that the output of each block needs to be transformed to be similar to that of the target domain data so that the model can easily converge.

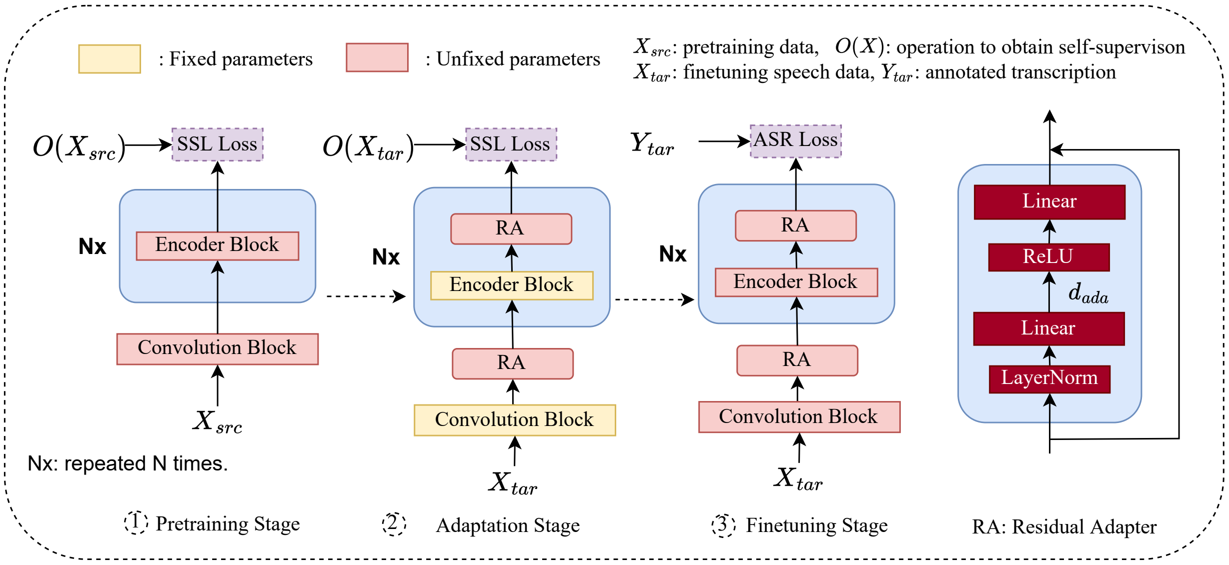


Figure 2.4: An overview of DRAFT. $d_{ada}$ is the output dimension of the first linear layer in the residual adapter (RA).

Residual adapters have been previously used for domain adaptation of supervised models [TZ21, HGJ19]. However, we develop a way of using residual adapters for adaptation of self-supervised models via a three-stage training paradigm as shonw in Figure 2.4. The motivation is to prevent catastrophic forgetting that happens when finetuning the entire pretrained model, and to address the domain shifting problem in self-supervised learning. We also do not learn different residual adapters for different domains. Our goal is to find a better model initialization of the downstream low-resource tasks. In [KSS21], residual adapters are used to re-pretrain and finetune the target domain data with the purpose of parameter efficiency in a natural language processing application. Hence, only residual adapters are updated at the finetuning stage, while we update the entire model, and the adaptation stage uses only finetuning data.

To better understand the algorithm, we detail the novel three training stages of DRAFT as shown in Figure 2.4. Let $\theta_{ada}$ be the parameters in residual adapters, $\theta_f$ the parameters in the backbone model (without residual adapters), $\theta_g$ the parameters in the generator for the self-supervised task, and $\theta'_g$ the parameters in the generator for the ASR task. Suppose source domain data are $S_{src}$ and target domain data are $S_{tgt}$, the three-stage training paradigm can be described as:

- Stage 1: Initialize a model $\{\theta_f^0, \theta_g^0\}$, update the parameters using data $S_{src}$ and self-supervised loss $L_{ssl}$, and obtain a pretrained model $\{\theta_f^1, \theta_g^1\}$.
- Stage 2: From model $\{\theta_f^1, \theta_g^1\}$, insert residual adapters after each block initialized with $\theta_{ada}^0$, freeze $\{\theta_f^1, \theta_g^1\}$ and update $\theta_{ada}^0$ using data $S_{tgt}$ and the same self-supervised loss $L_{ssl}$, and obtain an adapted model $\{\theta_f^1, \theta_{ada}^1, \theta_g^1\}$.
- Stage 3: From model $\{\theta_f^1, \theta_{ada}^1, \theta_g^1\}$, replace $\theta_g^1$ with a new generator that can map the embedding space to the token space as $\theta_{g'}^0$, update the entire model with data $S_{tgt}$ and an ASR loss such as connectionist temporal classification (CTC), to obtain the final ASR model $\{\theta_f^2, \theta_{ada}^2, \theta_{g'}^1\}$.

Note that the superscript in each $\theta$ is the number of times the parameters are updated. For example, $\theta_f^2$ means that the backbone model has been updated twice, once in stage one and the other in stage three. DRAFT is universal to all self-supervised pretrained models. We verify the effectiveness of DRAFT on E-APC, Bi-APC, Wav2vec2.0 and HuBERT models.

A similar concurrent work to the proposed approach is [KTK21] where residual adapters are also used to prevent catastrophic forgetting, but for continually learning representations from various languages. Different from [KTK21], finetuning data are used in an additional stage with the purpose of adapting SSL-pretrained models in our method. Furthermore, we update all the parameters during the finetuning stage instead of fixing the backbone model parameters because fixing the parameter does not perform well in pilot experiments.

## 2.2 Experimental Results

### 2.2.1 Datasets

Because of the availability of large databases of adult speech, we explore how SSL methods trained on adult speech data can help the development of child ASR systems. In this section, we introduce the data and the experimental settings for the pretraining, adaptation and finetuning stages.

#### 2.2.1.1 Librispeech 960-hour adult speech corpus

Librispeech is a widely-used adult speech corpus [PCP15]. It contains 960 hours of read speech extracted from audio books. We use this dataset during the pretraining stage. A 10h subset of the data introduced in [KR20] is often used for evaluating SSL methods on low-resource tasks, and is referred to as Libri-10h.

#### 2.2.1.2 OGI 50-hour child speech corpus

For the finetuning experiments, the scripted part of the OGI Kids' Speech Corpus [SHC00] is used. It contains speech from approximately 100 speakers per grade (from kindergarten to grade 10) saying single words, sentences and digit strings. The utterances are randomly split into train (70%), development (15%) and test (15%) sets without speaker overlap. As a result, nearly 50 hours of child data are used to train the child ASR system.

#### 2.2.1.3 My Science Tutor (MyST) 240-hour child speech corpus

Another corpus used for finetuning is the MyST children speech corpus [WC11, WCP19]. MyST consists of 499 hours with 244,069 utterances of conversational speech between children and a virtual tutor from 1,372 students between third and fifth grades. However, only 42% of the corpus (240 hours) is annotated for ASR. We use the annotated part of the corpus to verify the effectiveness of our proposed methods. The corpus also contains a development set and test set for evaluation.

### 2.2.2 Bi-APC for BLSTM Pretraining

#### 2.2.2.1 Acoustic Model Settings

We use the HMM-based hybrid system for the BLSTM experiments, where the acoustic model is the BLSTM. The initial experiments are HMM-GMM model training with the usage of Librispeech and JHU OGI recipes [WGK19] in Kaldi for pretraining and finetuning. The GMM models are then used to obtain frame-level alignment for BLSTM-based acoustic model training. The HMM states are 5776 and 1360 for adult and child models, respectively.

Uni-LSTM and BLSTM are chosen as acoustic models to compare pretraining methods. 80-dimensional Mel-filter bank features are extracted from each 25ms window with a 10ms frame shift as the input. No frame stacking or skipping is applied. Hence, the output dimension for the unsupervised pretraining task is 80. The uni-LSTM model consists of 4 uni-LSTM layers with 800 hidden units, while the BLSTM model has 4 BLSTM layers with 512 hidden units in each direction. Batch normalization and dropout layers with a 0.2 dropout rate are applied after each LSTM layer. The outputs of the LSTMs are then transferred into either the state space for classification or the feature space for prediction with a single feed-forward layer.

All models are trained with a multi-step schedule, where the learning rate is held in the first 2 epochs and then is exponentially decayed to a factor ($\lambda$) of the initial learning rate in the remaining epochs. For pretraining tasks, 8 epochs are used with the initial learning rate of 0.001 and $\lambda = 0.1$. For the finetuning tasks, we train the models with 15 epochs. The learning rate starts from 2e-4 to 2e-6. The last three model checkpoints are averaged as the final model for evaluation. For both APC and Bi-APC training, the time shift $n$ is heuristically set to 2. Sequence discriminative training is not applied in our experiments since our goal is to compare different pretraining methods. Results on the OGI dataset are reported. Note that the development and test sets are merged and named as OGI-test for the BLSTM experiments.

#### 2.2.2.2 Language Model Setup

All experiments use the same lexicon and language models from the original Librispeech corpus. Specifically, the 14M tri-gram (tgsmall) language model is used for first pass decoding, and the 725M tri-gram (tglarge) language model is used for second-pass rescoring. We report the results after the rescoring.

#### 2.2.2.3 Results and Discussion

Table 2.1: WERs of baseline systems, including uni-LSTM, BLSTM and TDNN-F trained with Librispeech and OGI data, respectively.

| WERs(%) | Libri-adult | | Children |
|---|---|---|---|
| | test-clean | test-other | OGI-test |
| Adult Model - Librispeech | | | |
| uni-LSTM | 5.71 | 15.15 | 65.90 |
| BLSTM | 4.90 | 12.59 | 59.12 |
| Child Model - OGI Corpus | | | |
| TDNN-F [WGK19] | - | - | 10.71 |
| uni-LSTM | 95.77 | 97.28 | 12.58 |
| BLSTM | 86.82 | 92.15 | 9.16 |

**Baseline:** We first show the results of the baseline models in Table 2.1. Here we compare two models–(a) adult model trained using Librispeech and (b) child model trained using the OGI speech corpus. We evaluate these models on test-clean and test-other from Librispeech and also on the OGI test. We compare uni-LSTM and BLSTM acoustic model architectures for both setups. For the adult model, we obtain performances similar to previously published results [PCP15]. Adult models are also used to test on ogi-test that has an acoustic domain mismatch resulting in high WERs for LSTM models.

For child models, the performance on Librispeech degrades drastically with both uni-LSTM and BLSTM models. To compare with existing results in the literature, we evaluate the TDNN-F acoustic model trained with the OGI corpus [WGK19]. We see that the uni-LSTM performed worse than TDNN-F but BLSTM outperformed TDNN-F, thus motivating us to explore model pretraining for the BLSTM system.

Table 2.2: Performance comparison of supervised pretraining (SPT) and unsupervised pretraining (UPT) in terms of WER (%) for both LSTM and BLSTM acoustic model architectures. The results are for OGI-test. We also provide word error rate reduction (WERR) compared to the baseline.

| WERs(%) | | uni-LSTM | WERR | BLSTM | WERR |
|---|---|---|---|---|---|
| Baseline | | 12.58 | - | 9.16 | - |
| SPT | | 11.85 | 5.8% | 8.46 | 7.6% |
| UPT | MPC [JLL19] | - | - | 9.02 | 1.5% |
| | APC | 11.76 | 6.5% | 8.85 | 3.4% |
| | Bi-APC | - | - | 8.57 | 6.5% |

**Comparison of Pretraining Methods for Child ASR:** Next, we aim to explore the performance of supervised (SPT) and unsupervised pretraining (UPT)[1] for children's ASR. As mentioned in Section 2.2.1, we use Librispeech data for pretraining and OGI data for finetuning. Table 2.2 presents results of finetuning on both uni-LSTM and BLSTM architectures, evaluated on the OGI test data. Note that, different from [SG20], all layers are updated during finetuning since this is the best setting for our experiments. Table 2.2 shows that SPT improved the performance of the uni-LSTM model to 11.85% and was better than the baseline without pretraining. Interestingly, unsupervised pretraining using APC also provides improvement (11.76%) similar to that of SPT with the uni-LSTM model.

[1]Here we use the terminology of unsupervised pretraining as opposed to the supervised pretraining for a better understanding

As mentioned earlier, BLSTM has better performance than uni-LSTM. SPT results in the best performance (WER, 8.46%) among the pretraining methods applied to BLSTM. Note that, to perform UPT, we first use APC to pretrain only the forward path parameters of BLSTM, resulting in a WER of 8.85%. We then compare it to a widely used bidirectional pretraining method, the masked predictive coding (MPC) [JLL19] and show that MPC (9.02%) performs worse than APC (8.85%). The reason may be that MPC has fewer frames to be predicted (only 15% of the frames were randomly masked) although MPC can learn from both directions. The proposed Bi-APC achieves a WER of 8.57% that is comparable to the SPT. This can be valuable when there is a large amount of data without transcriptions. Since the pretraining task is a 960-hour dataset, UPT could possibly benefit from more unlabeled data.

Table 2.3: BLSTM-based ASR performance breakdown based on age groups of kindergarten to grade 2, grade 3-6 and grade 7-10.

| WERs(%) | K0-G2 | G3-G6 | G7-G10 |
|---|---|---|---|
| Baseline | 18.87 | 7.24 | 5.51 |
| +SPT | 17.43 | 6.66 | 5.11 |
| +APC | 18.07 | 7.03 | 5.40 |
| +Bi-APC | 17.23 | 6.91 | 5.26 |

**Performance Breakdown based on Age Groups:** To obtain an insight into the influence of the speaker's age on the performance of pretraining methods, in Table 2.3, we present results based on age groups in the OGI dataset. Similar to [SNH16], three different age groups are selected–kindergarten to grade 2, grade 3-6, and grade 7-10. We present the results using the BLSTM model. For younger children (kindergarten- grade 2), the Bi-APC provides slightly better results compared to SPT. In contrast, we do not observe any such improvement in the older age groups for children. This trend could mean that UPT may be capturing a representation crucial to the performance of very young child speech, whose speech is more variable and difficult to recognize than older children [YA18]. Further research is required to explore the usage of the approach more effectively for children's ASR.

### 2.2.3 Bi-APC for Transformer Pretraining and DRAFT Finetuning

More recently, transformers have shown better performance than the LSTM-based ASR models. Hence, investigating the application of Bi-APC for transformer models is important. This section also includes the proposed DRAFT experiments on widely used transformer-based SSL models in addition to the Bi-APC.

#### 2.2.3.1 Experimental Settings

**Pretraining Stage Settings:** Four self-supervised learning algorithms are investigated: E-APC, Bi-APC, Wav2vec2.0 and HuBERT. All models at this stage are trained with the Librispeech dataset.

For E-APC and Bi-APC, we use 80-dimensional log-mel filter-bank features ($Z$ in Figure 1.3) without any concatenation or frame skipping. The features are extracted using a 25ms Hamming window and a frame rate of 10ms. Padding is used for the shorter utterances to make the length be the maximum length of the utterances in a batch. The backbone model $f$ consists of a two-layer convolution block with a sub-sampling of four along the time axis, 12 transformer encoder blocks and a generator for each temporally-shifted sequence. We predict four consecutive frames at each step because of the sub-sampling in the convolution block, resulting in a 320-dimensional output of the generator. Adam optimizer is used with a noam-based scheduler, where the noam factor is 5 and the warmup step is 15k. The model is updated for 130k steps with a batch size of 256. Various starting temporal lags ($s$) and various numbers of consecutive temporally-shifted sequences ($k$) in the E-APC are compared and the best settings in the E-APC are used for the subsequent Bi-APC pretraining.

For Wav2vec2.0 and HuBERT, we directly use the open-sourced pretrained models in the Fairseq[1] toolkit [OEB19]. We choose the base model that has about 95M parameters

[1]Our code modified on Fairseq is available at `https://github.com/Diamondfan/fairseq`.

to evaluate the effectiveness of the proposed DRAFT framework. Note that the number of parameters in the E-APC and Bi-APC pretraining models is about 39M.

**Adaptation Stage Settings:** Residual adapters are added to the pretrained model at this stage and only the parameters of residual adapters are updated. We use Xavier uniform initialization [GB10] for all RA parameters.

For E-APC and Bi-APC, we adapt the model from the pretraining stage with either the OGI or MyST datasets according to the finetuning task. For the OGI data, residual adapters are updated in 55k steps with a noam factor of 8, warmup steps of 10k. For the MyST data, residual adapters are updated in 74k steps with a noam factor 4 and a warmup step of 15k. The batch size is set to 64 for both datasets.

For Wav2vec2.0 and HuBERT, the residual adapters of Wav2vec2.0/HuBERT are updated in 200k/100k steps with learning rate ramping up from 0 to the peak learning rate in 32k/8k steps, and then decays linearly back to 0, where the peak learning rate is 5e-4. The batch size is set to 16.

We also run experiments using SAFT with the above configurations but all the parameters are updated. The learning rate is lower than the one used in the pretraining stage (e.g. peak learning rate of 1e-4 on Wav2vec2.0 and HuBERT). The learning rate is determined empirically.

**Finetuning Stage Settings:** The CTC loss function is used for the finetuning ASR task training. We train two types of models: 1) a causal transformer with a causal convolution block and encoder blocks with upper-triangular matrices for attention. 2) a non-causal transformer with a regular convolution block and encoder blocks with all-one matrices for attention. The causal transformer is initialized with E-APC pretrained models, while Bi-APC, Wav2vec2.0 and HuBERT use the non-causal transformer.

In E-APC and Bi-APC, we finetune the model from the pretraining stage or the adaptation stage. The model is updated in 240k steps with a batch size of 32, a noam factor is 2, and a warmup step of 10k steps for the OGI data. For the MyST data, the model is updated in 340k steps with a batch size of 64, a noam factor of 2, and a warmup step of 15k steps.

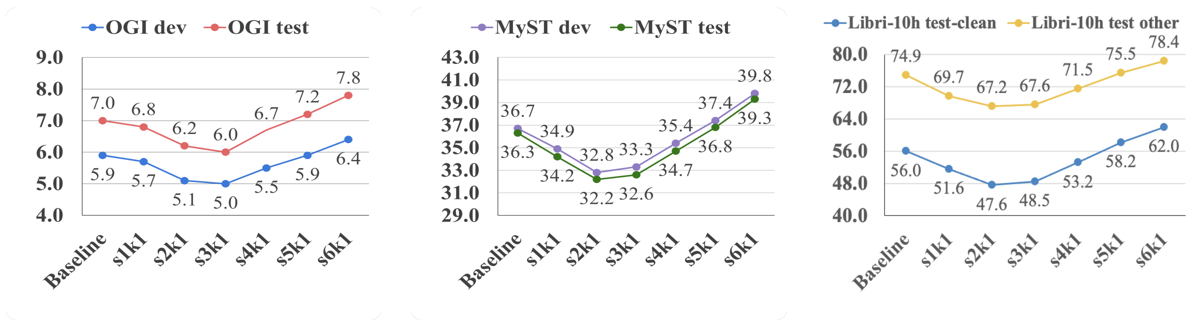
In Wav2vec2.0/HuBERT, the model is updated with a batch size of 64 in 40k steps with a multi-step scheduler where the warmup steps are set to 4k. The peak learning rate of 3e-5/7e-5 holds for the next 16k steps, then exponentially decays to a factor ($\lambda$) of the initial learning rate, where $\lambda$ is set to 0.05.

The data augmentation methods, speed perturbation [KPP15] and SpecAug [PCZ19], are used for all the experiments at the finetuning stage. Greedy search decoding is used during evaluation. We conduct experiments to decide empirically on the final hyper-parameters. We start the training with a large training epoch (e.g. 100 epochs). Then, if the WER on the validation set does not decrease for several epochs, we stop the training and decide the number of training steps based on the convergence of the training phase.

#### 2.2.3.2 Results and Discussion

The base model (95M) that was pretrained on Librispeech 960 hours data using the Wav2vec2.0 method in the Fairseq toolkit [OEB19] achieves a WER of 3.4% on Librispeech test-clean data [BZM20]. However, the model results in WERs of 41.67% and 30.77% for the OGI and MyST test data, respectively, showing a large mismatch between the adult and child speech. In this section, we present experimental results of the proposed methods.

**APC for Causal Transformers and Its Extension:** Figure 2.5 and Table 2.4 show ASR results using APC and its extension on the OGI, MyST, and Librispeech 10h [KR20] datasets. We experimented with the Libri-10h data to show that our methods are not limited to children's speech. The results for the Libri-10h data are similar to those in [KR20]. The reason for its poor performance compared to finetuning on child speech is because of the amount of finetuning data (10 hours for Librispeech, 50 hours for OGI, and 240 hours for MyST). In addition, OGI is an easier task because of similar distributions for the train and test sets. In Figure 2.5, we use only one temporally-shifted sequence ($k = 1$) and explore the effect of the starting temporal lag $s$. A baseline that does not use any pretraining methods



(a) OGI (b) MyST (c) Libri-10h

Figure 2.5: WER results on the OGI, MyST, and Libri-10h datasets for different temporal lags $s$ in APC. Only one temporally-shifted sequence ($k = 1$) is used in these experiments. Baseline is the causal transformer trained from scratch. Note that we use adult speech in the pretraining stage. Therefore, (a) and (b) are domain mismatched case, while (c) is the domain matched case.

is also included in Figure 2.5. We can see from the figure that when the prediction lag is more than 4 (16 frames), the WERs increase for all tasks. This result is similar to the results in [CG20a] when considering the sub-sampling in the convolution block. The best choice of $s$ is 3 (12 frames) for OGI, and 2 (8 frames) for MyST and Libri-10h. The lags are approximately the duration of an acoustic unit (a vowel or a short syllable). Hence, the model can learn local smoothness of the spectral features by predicting frames within an acoustic unit, and acoustic unit transitions (global structure) by predicting frames in the next acoustic unit, resulting in learning more meaningful speech representations. Then, multiple temporally-shifted sequences are combined to construct E-APC. Based on the results in Figure 2.5, we experiment with two settings: s1k4, combining four consecutive temporally-shifted sequences starting from 1, and s2k2, combining two consecutive temporally-shifted sequences starting from 2, because they yielded better results than other settings. The results are shown in Table 2.4. From the table, the performance of $s2k2$ has a $\sim 1.8\%$ relative WER improvement on both tasks compared to the best setting in APC. We also experimented with $L_1$, $L_2$, and a combination of $L_1$ and $L_2$ distance measures as the basic loss function for APC, and found that $L_1$ performs the best.

Table 2.4: WER results of APC with $s2k1$ and E-APC with settings of $s1k4$ and $s2k2$. $s1k4$ stands for using four consecutive temporally-shifted sequences starting with a temporal lag of one. A similar meaning applies to $s2k1$ and $s2k2$. $L_p$ norm is the basic loss function used as shown in Eq.1.8. Baseline is the causal transformer trained from scratch.

| APC | $L_p$ | OGI | | MyST | | Libri-10h | |
|---|---|---|---|---|---|---|---|
| | | dev | test | dev | test | clean | other |
| Baseline | - | 5.9 | 7.0 | 36.7 | 36.3 | 56.0 | 74.9 |
| APC-s2k1 | $L_1$ | 5.1 | 6.2 | 32.8 | 32.2 | 47.6 | 67.2 |
| EAPC-s1k4 | $L_1$ | 5.1 | 6.0 | 35.5 | 34.8 | 45.7 | 65.1 |
| EAPC-s2k2 | $L_1$ | 5.0 | 6.1 | 32.2 | 31.6 | 45.6 | 65.1 |
| | $L_2$ | 5.4 | 6.3 | 32.9 | 32.2 | 48.5 | 67.1 |
| | $L_1 + L_2$ | 5.2 | 6.3 | 32.4 | 31.7 | 45.6 | 65.1 |

Table 2.5: WER results of four solutions (shown in Figure 2.2) for using Bi-APC for a non-causal transformer. Baseline is the non-causal transformer trained from scratch.

| | Share? | | | OGI | | MyST | | Libri-10h | |
|---|---|---|---|---|---|---|---|---|---|
| | Conv. | Enc. | G. | dev | test | dev | test | clean | other |
| Baseline | - | - | - | 2.9 | 3.3 | 28.0 | 27.8 | 51.9 | 70.2 |
| Bi-APC | | | | 3.2 | 3.9 | 27.8 | 27.3 | 60.7 | 77.0 |
| | | | ✓ | 3.3 | 4.1 | 32.1 | 31.8 | 58.6 | 75.7 |
| | | ✓ | ✓ | 2.8 | 3.4 | 26.2 | 25.7 | 40.1 | 58.9 |
| | ✓ | ✓ | ✓ | 2.8 | 3.3 | 25.5 | 25.0 | 40.3 | 58.9 |

**Bi-APC with A Non-causal Transformer:** Using the best settings of E-APC ($s2k2$ with $L_1$ distance), we discuss the four solutions of applying Bi-APC to the non-causal transformer mentioned in Section 2.1.1. Results are shown in Table 2.5. As can be seen from the table, with more modules shared, the performance tends to be better except that sharing only the generator causes an increase in WERs. However, there is only a 0.1% absolute WER improvement on the OGI development set when sharing all parameters. The reason may be because the used OGI data (the scripted part) contains shorter utterances (3.5s for OGI, 8.3s for MyST, and 12.8s for Libri-10h), and thus benefit less from Bi-APC. We note a larger improvement with the MyST and Libri-10h data. For example, the WER for MyST test data using the sharing-all solution is decreased from 27.8% to 25.0%, and the WER is

Table 2.6: WER results of different values of $d_{ada}$ in residual adapters. SAFT is the sample adaptation for finetuning that updates the entire model at the adaptation stage. DRAFT is the proposed domain responsible adapter for finetuning that updates only residual adapters at the adaptation stage. The number of updated parameters are also shown in absolute and relative values (compared to the baseline of a causal transformer).

| | $d_{ada}$ | OGI | | MyST | | Updated Params | |
|---|---|---|---|---|---|---|---|
| | | dev | test | dev | test | total | relative |
| Baseline | 0 | 5.9 | 7.0 | 36.7 | 36.3 | 39.2M | 100% |
| +EAPC | 0 | 5.0 | 6.1 | 32.2 | 31.6 | 39.2M | 100% |
| +SAFT | 0 | 5.0 | 5.9 | 33.4 | 32.9 | 39.2M | 100% |
| +DRAFT | 64 | 4.9 | 5.7 | 31.9 | 31.0 | 0.9M | 2% |
| | 128 | 4.7 | 5.6 | 31.6 | 30.9 | 1.7M | 4% |
| | 256 | 4.6 | 5.3 | 31.1 | 30.4 | 3.4M | 9% |
| | 512 | 4.4 | 5.2 | 30.9 | 30.2 | 6.8M | 17% |
| | 1024 | 4.4 | 4.9 | 30.1 | 29.4 | 13.7M | 35% |
| | 2048 | 4.4 | 4.9 | 30.0 | 29.3 | 27.3M | 70% |

decrease from 51.9% to 40.3% on Librispeech test clean data. The parameters in the separated modules learn very different distributions when they are not shared during Bi-APC pretraining. Averaged parameters may lose information from both sides. As a result, the sharing-all solution outperforms other solutions for Bi-APC pretraining. However, Bi-APC is still worse than methods like Wav2vec2.0 and HuBERT as shown in the next section. When compared to our previous Bi-APC for LSTMs study [FAA21], Bi-APC for transformers performs worse for the child speech databases in terms of relative WERs. We assume that Bi-APC may not be suitable for bidirectional pretraining of models that have only one set of parameters. From this perspective, it is interesting to see whether the transformer can be reformulated as a model that has forward-related and reverse-related parameters.

**Effect of $d_{ada}$ in DRAFT:** We conducted experiments with different values of $d_{ada}$ in the residual adapters to examine the impact of the number of adapter parameters, because this number influences both WERs and adaptation efficiency. The experiments are conducted on the OGI and MyST datasets using the E-APC method. Specifically, $d_{ada}$ values are selected from 64 to 2048 and the results are shown in Table 2.6. For reference, we also include the results for the baseline, pretraining from E-APC and SAFT. Both WER results and the number of parameters that need to be updated during the adaptation stage are shown in the table. First, when compared to SAFT, DRAFT achieves a better performance with fewer parameters to be updated. In addition, we observe that the WER drops when we increase the number of parameters in the residual adapters. However, the cost is increased training time at the adaptation stage because more parameters need to be updated. We can even achieve an improvement from a WER of 5.9% to 5.7% on the OGI test set with only 2% of the parameters being updated. As a result, the choice of $d_{ada}$ in DRAFT can be adjusted according to different scenarios. For example, one can use a small value of $d_{ada}$ to achieve a fast adaptation of the self-supervised model when computational resources are limited. A large value of $d_{ada}$ can be used to achieve a better performance for the finetuning task. All subsequent DRAFT experiments will use 1024 for $d_{ada}$ since it results in a good trade-off between performance and efficiency.

**Results of DRAFT with Non-causal Transformers:** DRAFT is shown to be effective for E-APC with a causal transformer. Here, we continue to evaluate the DRAFT framework for Bi-APC and other two widely used SSL methods: Wav2vec2.0 and HuBERT. Both the models are adopting a non-causal transformer architecture. We conduct DRAFT experiments on the OGI and MyST datasets for E-APC, Bi-APC, Wav2vec2.0 and HuBERT. Results are shown in Table 2.7. The table shows that SAFT yields a small improvement or even a negative effect on the WERs compared to the pretraining baselines (without adaptation). The reason may be that updating the entire model causes a catastrophic forgetting of the knowledge learned from adult speech. However, when the proposed DRAFT framework is used, the WERs of the four SSL methods have an improvement on the OGI dataset compared to the pretraining baselines (without adaptation). Specifically, we achieve relative WER improvements of 19.7%, 3.0%, 7.4%, and 16.0% on the OGI test set for E-APC, Bi-APC, Wav2vec2.0, and HuBERT, respectively. HuBERT achieves the best WER of 2.1% on the OGI test data. WER improvements are even larger when compared to the baselines with-

Table 2.7: WER results of using DRAFT for E-APC, Bi-APC, Wav2vec2.0 and HuBERT on the OGI and MyST datasets. Baseline indicates that models are trained from scratch. Improvements of DRAFT are statistically significant ($p < 0.05$) compared to the pretraining results. The convergence of Wav2vec2.0 and HuBERT is difficult to achieve without pretraining. AF indicates Adapter Finetuning.

| | E-APC | | | | Bi-APC | | | | Wav2vec2.0 | | | | HuBERT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OGI | | MyST | | OGI | | MyST | | OGI | | MyST | | OGI | | MyST | |
| | dev | test | dev | test | dev | test | dev | test | dev | test | dev | test | dev | test | dev | test |
| Baseline | 5.9 | 7.0 | 36.7 | 36.3 | 2.9 | 3.3 | 28.0 | 27.8 | - | - | - | - | - | - | - | - |
| + Finetune | 5.0 | 6.1 | 32.2 | 31.6 | 2.8 | 3.3 | 25.5 | 25.0 | 2.3 | 2.7 | 17.84 | 17.16 | 2.1 | 2.5 | 17.40 | 16.71 |
| + AF [TKK22] | 8.6 | 10.1 | 47.4 | 47.3 | - | - | - | - | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Self-Transfer | | | | | | | | | | | | | | | | |
| +SAFT | 5.0 | 5.9 | 33.4 | 32.9 | - | - | - | - | 2.22 | 2.67 | 17.85 | 17.28 | 2.02 | 2.43 | 17.52 | 16.89 |
| +DRAFT | 4.4 | 4.9 | 30.1 | 29.4 | 2.7 | 3.2 | 24.8 | 24.3 | 2.11 | 2.51 | 17.21 | 16.70 | 1.85 | 2.05 | 16.79 | 16.53 |
| Cross-Transfer | | | | | | | | | | | | | | | | |
| +SAFT | 5.2 | 6.2 | 37.8 | 37.3 | - | - | - | - | 2.33 | 2.85 | 21.24 | 20.28 | 2.11 | 2.30 | 17.67 | 17.20 |
| +DRAFT | 4.7 | 5.5 | 31.4 | 30.8 | - | - | - | - | 2.13 | 2.63 | 17.95 | 17.36 | 2.03 | 2.28 | 17.13 | 16.65 |

out using pretraining methods. For example, we achieve 30% and 19% WER improvements on the OGI and MyST data for the E-APC method, respectively. We also observe similar improvements on the MyST dataset, although the relative improvements are smaller than those using the OGI data (child read speech). The reason could be the mismatch in style between MyST data (child spontaneous speech) and pretraining data (adult read speech). Note that we do not try SAFT for Bi-APC because we have already shown its overfitting behaviour in E-APC. For Wav2vec2.0 and HuBERT, the baselines without pretraining are not available because their convergence is difficult to achieve without pretrained models. Note that the results of using HuBERT and Wav2vec2.0 are much better than those of E-APC is because of a better performance of non-causal transformers than causal transformers. However, improving and examining APC with our proposed framework is valuable for causal transformers (streaming models).

We also examine the cross knowledge transfer ability of the residual adapters (RAs) learned from one dataset to another as they learn domain-related information. Specifically, the RAs trained with OGI data during the adaptation stage are used for training MyST data at the finetuning stage and vice versa. The results are shown in Table 2.7 under cross transfer row. Results show that when using the redisual adapters learned from MyST data, DRAFT can improve the WER performance on the OGI dataset consistently with the three SSL methods. This shows that DRAFT can be used for efficiently learning knowledge from one domain (child spontaneous speech), and then achieve better finetuning for another related domain (child read speech). However, the use of RAs learned from OGI data does not help the performance for MyST data. It might be because OGI has only 50-hour speech data, which is not large enough for adapting to the MyST data (240 hours). However, DRAFT always performs better than SAFT in cross knowledge transfer settings, which again shows DRAFT's ability at preventing catastrophic forgetting. We are also interested in determining whether the learned residual adapters from different domains can be fused together to further improve the performance. However, preliminary experiments did not show performance improvements.

Experimental results using the adapter finetuning method [TKK22] are also presented

Table 2.8: Experiments showing the behaviour of residual adapters in DRAFT in terms of WER. “RA Initialization” and “Update RA?” are describing the finetuning stage. “+ RA” indicates adding randomly initialized RA at the finetuning stage for a fair comparison to DRAFT. $\theta^1_{ada}$ indicates the RA learned in the adaptation stage and $\theta^0_{ada}$ is the RA with random initialization.

| E-APC | RA Initialization | Update RA? | OGI | |
|---|---|---|---|---|
| | | | dev | test |
| Baseline | None | No | 5.9 | 7.0 |
| + RA | $\theta^0_{ada}$ | Yes | 5.5 | 6.4 |
| DRAFT | $\theta^0_{ada}$ | Yes | 4.8 | 5.6 |
| | $\theta^1_{ada}$ | Yes | 4.4 | 4.9 |
| | $\theta^1_{ada}$ | No | 4.7 | 5.4 |

in Table 2.7. Although adapter finetuning has been shown effective in [TKK22], it does not perform well in child ASR tasks, maybe because fixing the backbone model (pretrained on adult speech) is not appropriate when the domain mismatch (finetuning on child speech) exists. The results of adapter finetuning also show the importance and effectiveness of DRAFT in reducing domain mismatch. Note that the results of adapter finetuning on Wav2vec2.0 and HuBERT are 100% because of a slow convergence during training. We may get a reasonable WER for the adapter finetuning method with a longer training schedule. However, the results of adapter finetuning and DRAFT are comparable because they use the same amount of training steps.

**What do Residual Adapters Learn?** Finally, we explore the behaviour of residual adapters (RA) by using a random initialization of RA or freezing RA parameters during the finetuning stage. By initializing RA parameters randomly ($\theta^0_{ada}$), the WERs in comparison to that of DRAFT which has pretrained RA ($\theta^1_{ada}$), can give us an insight into whether the

RA learn the knowledge from the target domain data as expected. Results for E-APC on the OGI data are shown in Table 2.8. As shown in the table, the performance of DRAFT with learned RA is much better than that when RA are randomly initialized (4.9% v.s. 5.6%), showing the successful learning of the target domain knowledge. The table also shows the result when RA parameters are frozen during the finetuning stage. We can see from the table that the performance of freezing RA (5.4%) is better than the experiment with randomly initialized and updated RA (5.6%, third row in the table). The results imply that the RA may learn a domain-related transformation from adult to child speech after each block in the transformer. The learned transformation ability from the SSL task might be directly used in the ASR task without further finetuning. One might argue that WER improvements are the results of increasing model parameters. Hence, we conduct an experiment that directly adds residual adapters ("+RA" in the table) in the same way they are added to DRAFT but without any pretrained parameters. The results in Table 2.8 show that DRAFT outperforms the new baseline as well, which could address the concern that the improvements are from increased model capacity.

## 2.3 Chapter Summary

In this chapter, we developed self-supervised learning (SSL) based techniques to improve the accuracy of children's ASR using adult speech data in the pretraining stage. In the context of autoregressive predictive coding (APC), which is a neural language model style pretraining method for causal transformers, we proposed an extension to APC (E-APC) by learning from multiple temporally-shifted sequences because they contain different levels of information in the structured data. E-APC had a $\sim 1.8\%$ relative WER improvement on the OGI and MyST data compared to APC. Furthermore, a bidirectional APC (Bi-APC) framework was proposed for BLSTM pretraining, which addressed the problem that APC is not suitable for bidirectional model pretraining. In addition, we discussed the possibility of using Bi-APC for non-causal transformers. Various solutions were investigated and results showed that Bi-APC framework can have a slight improvement over the transformer baseline without pretraining, but the results are worse than other bidirectional pretraining methods like Wav2vec2.0 and HuBERT. Finally, a domain responsible adaptation and finetuning (DRAFT) framework was proposed to alleviate the domain shifting problem between adult and child speech. The DRAFT framework performed well on E-APC, Bi-APC, Wav2vec2.0 and HuBERT methods, showing that it can improve the performance of pretraining methods for both causal (E-APC) and non-causal transformers (the other techniques). When compared to the conventional pretraining baselines without adaptation, we achieved relative WER improvements of up to 19.7% on the two child ASR tasks. The relative WER improvements are even larger (30% and 19% for E-APC on the OGI and MyST data, respectively) when compared to the models without using pretraining methods. In the cross transfer experiments, we observe the potential that residual adapters learned from one dataset could benefit for the other dataset. As a result, a future direction of research could be to investigate the fusion of residual adapters learned from various domains for greater WER improvements on child and other low-resource ASR tasks.

# CHAPTER 3

# Improving the Inference Efficiency with Non-autoregressive Models

Besides the WER (accuracy) performance, the real time factor (RTF) is another important metric for ASR systems, especially when the system is required to be deployed on a device. As mentioned in Section 1.2.3, Autoregressive Encoder Decoder (AED) models, e.g. transformers, achieve good WER performance, but they adopt an autoregressive generation mechanism, slowing down the inference speed. In this chapter, we propose a novel CTC-alignment based Single Step Non-Autoregressive Transformer (CASS-NAT) that can generate the output sequence simultaneously in one step, greatly reducing the inference time for ASR. The trade-off is that non-autoregressive models perform worse than their autoregressive counterparts because of the weakened dependencies between output tokens. To reduce the gap between the autoregressive and non-autoregressive models, an Error-based Sampling Alignment method (ESA) is further proposed. Additionally, We improve the performance of CASS-NAT by applying various training strategies and conduct a comprehensive study of the proposed CASS-NAT. For example, we examine the effect of decoder structures on the WER and investigate the impact of the hyper-parameters on the proposed error-based sampling alignment (ESA) method, such as the sampling threshold, the number of sampled alignments, and the scoring model for ranking the sampled alignments. Various encoder initialization schemes (including AT encoder, CTC encoder, and random initialization) for CASS-NAT training are explored with an analysis of why the proposed CASSN-NAT works with ESA decoding. This work was published in [FCC21a, FCC21b, FCC23].

## 3.1 Method: CTC-alignment based Single-Step Non-autoregressive Transformer

We present a comprehensive study of a non-autoregressive transformer (NAT) framework by utilizing alignments over the CTC output space. The framework can generate the output sequence within one iteration, so we refer to it as CTC Alignment-based Single Step NAT (CASS-NAT). In CASS-NAT, there are four major modules: encoder, token-level acoustic embedding extractor (TAEE), self-attention decoder (SAD), and mixed-attention decoder (MAD). The encoder is used to extract a high-level acoustic representation for each frame. The TAEE extracts a more meaningful token-level acoustic embedding (TAE) using the information given by alignments over the CTC output space. The SAD and MAD model the dependencies between TAEs, where MAD considers encoder outputs directly for the purpose of source-attention while SAD does not. However, SAD indirectly uses the information from the encoder through the TAEs for self-attention. Since TAEs can be obtained in parallel, no recurrence in output sequence generation exists. Meanwhile, the two decoder modules can model the dependencies between TAEs in the latent space.

### 3.1.1 System Overview

In this section, we introduce the proposed CTC Alignment-based Single Step Non-autoregressive Transformer (CASS-NAT) in detail.

The proposed CASS-NAT system architecture builds upon the CTC/Attention hybrid architecture [WHK17] to be non-autoregressive using CTC alignments. Figure 3.1 shows the four major modules in CASS-NAT: encoder, token acoustic embedding extractor (TAEE), self-attention decoder (SAD) and mixed-attention decoder (MAD).

#### 3.1.1.1 Mask in attention

The attention mechanism is important for a transformer. The most basic computation in the attention mechanism is scaled dot-product self-attention using a sequence as input.

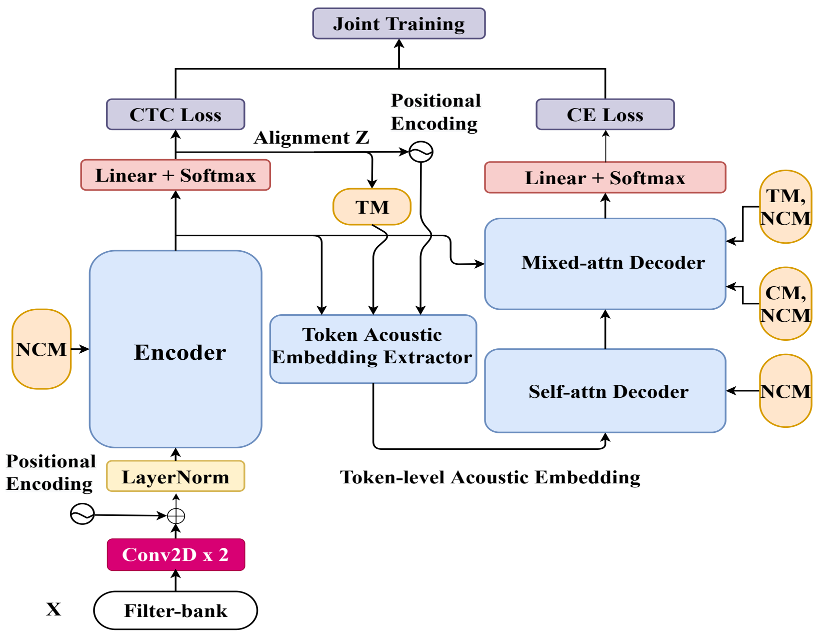


Figure 3.1: An overview of the proposed CASS-NAT architecture. CM and NCM represent a causal and non-causal mask, respectively. TM stands for trigger mask.

Conventional self-attention utilizes information across the whole sequence. But each output of the self-attention mechanism is not necessarily dependent on all of the input sequence. For example, a neural language model uses an upper triangular matrix (mask matrix) to gather information from only past tokens in a sequence. In Figure 3.1, we show three mask matrices for different purposes in the four major modules. Since we do not consider a streaming ASR model at this moment, non-causal mask (NCM) is used in the encoder. The NCM is a matrix where the paddings are zeros to prevent the padded tokens or padded frames from attention computation. In TAEE, a trigger mask (TM) is used to extract accurate token-level acoustic embeddings. TM masks the triggered frames, such that the positions of used frames are marked as ones, while other positions are marked as zeros. Examples of TM can be found in Section 3.1.1.3. MAD contains both self-attention and cross-attention layers, which are similar to an AT decoder. In such cases, either a causal mask (CM) or NCM can be used for the self-attention layer, and either an NCM or TM can be used for the cross-attention layer. The different choices of the mask matrices in MAD are explored experimentally in Section 3.3.1. Eventually, the self-attention computation can be augmented with a mask matrix as follows:

$$Attention(Q,K,V,M) = \left(Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \otimes M\right) V \tag{3.1}$$

where $Q \in R^{n_q \times d_q}$, $K \in R^{n_k \times d_k}$, $V \in R^{n_v \times d_v}$, and $M \in R^{n_q \times n_k}$ are the query, key, value and mask matrices, respectively.

#### 3.1.1.2 Encoder

The encoder extracts high-level acoustic representations $H$ from speech features $X$. A linear layer with a CTC loss function is added after the encoder as shown in Figure 3.1. The role of CTC is to obtain an alignment over the CTC output space to offer auxiliary information for the token acoustic embedding extractor (TAEE). During training, Viterbi-alignment is used. During inference, various methods for sampling from the CTC output space are explored experimentally.

#### 3.1.1.3 Token Acoustic Embedding Extractor

The token acoustic embedding extractor is designed to extract token-level acoustic embedding (TAE) with the auxiliary information offered by the CTC alignment. For example, given an alignment $Z = \{z_1, ..., z_t, ..., z_T\}$, we can estimate an acoustic segment for each token $u$ as $\{t_{u-1}+1, ..., t_u\}$ (note that 1 here refers to one frame), and the number of tokens in $Z$.

First, CTC alignments offer an acoustic boundary for each token $y_u$, which is then transformed into the trigger mask. Specifically, we define a mapping rule from alignment to trigger mask, and fix the rule in both the training and inference phases. We regard the first non-blank index of each token in the alignment as its end boundary. The intuition is our assumption that the model will not output a token until it observes all the acoustic information.of the token. Using the first non-blank index is for simplicity and consistency in training and decoding. For example, if an alignment is $Z = \{\_, C, C, \_, A, \_, \_, T, \_\}$ for the ground truth $Y = \{C, A, T\}$, where $\_$ is the blank symbol, the end boundary for $C$ and $A$ is $Z_2$ and $Z_5$, respectively, and thus the trigger mask for token $A$ is $[0,0,1,1,1,0,0,0,0]$.

The mapping rule might not be accurate for acoustic segmentations, but it should be consistent during the training and decoding. The trigger mask here is different from that used in [MHL20] for streaming purposes. In [MHL20], previous acoustic representations could be reused for each token, and thus the trigger mask in the previous example for token $A$ is $[1, 1, 1, 1, 1, 0, 0, 0, 0]$.

Second, CTC alignments provide the number of tokens for the decoder input. After removing blank symbols and repetitions, the number of tokens in an alignment $Z$ is used as the predicted length of sinusoidal positional encoding (decoder input length). As shown in Figure 3.1, TAE for each token is then extracted with the trigger mask and the sinusoidal positional encoding using a one-layer source-attention block. The TAEs replace the word embeddings in AT to achieve parallel generation of each sequence.

#### 3.1.1.4 Self-attention Decoder

TAEs extracted from TAEE (see Section 3.1.1.3) have the desirable property of parallel generation and thus are used as a substitution of word embeddings for the decoder input. Since there is no need to create recurrence in the decoder, we use a non-causal mask (NCM) in the self-attention decoder (SAD) to model the relationships between TAEs.

#### 3.1.1.5 Mixed-attention Decoder

We assume that TAEs have a similar capability of learning language semantics compared to word embeddings. Hence, we design a mixed-attention decoder (MAD) to retrace the encoder information for better decision making at the output layer. Similar to an AT decoder, MAD has a self-attention layer that uses either CM or NCM, and a source-attention layer that uses either TM or NCM. A linear layer is added after MAD, followed by a cross-entropy loss. Since we use the Viterbi-alignement during training, the output has the same length as the ground truth $Y$.

### 3.1.2 Training Details

The training criterion is presented in this section, followed by various training strategies used to improve the performance of CASS-NAT.

#### 3.1.2.1 Training Criterion

In our framework, CTC alignments $Z$ are introduced as latent variables. Given $X$ and $Y$ in Section 1.4, the log-posterior probability can be decomposed into:

$$\log\, P(Y|X) = \log\, \mathbb{E}_{Z|X}[P(Y|Z,X)], \quad Z \in q. \tag{3.2}$$

where $q$ is the set of alignments that can be mapped to $Y$. For those alignments that do not belong to $q$, we assume that $P(Y|Z,X)$ is equal to zero. To reduce computational cost, the maximum approximation [ZMS20] is applied:

$$\begin{aligned} \log\, P(Y|X) &\geq \mathbb{E}_{Z|X}[\log\, P(Y|Z,X)] \\ &\approx \max_{Z} \log \prod_{u=1}^{U} P(y_u|z_{t_{u-1}+1:t_u}, x_{1:T}) \end{aligned} \tag{3.3}$$

where $\mathbb{E}$ represents the expectation and $t_u$ is the end boundary of token $u$ ($t_0 = 0$). All $t_u$s can be estimated from the alignment $Z$. We expect that TAEs can capture language semantics to a certain degree (see Section 3.3.5 for analysis), which helps alleviate performance degradation caused by the independence of the output tokens.

The framework is trained by jointly optimizing a CTC loss function on the encoder side ($L_{CTC}$) and a cross entropy (CE) loss function on the decoder side ($L_{dec}$) with a task ratio $\lambda$ [WHK17], and thus the final loss function ($L_{\text{joint}}$) is defined as:

$$L_{\text{joint}} = -\lambda \cdot \log \sum_{Z \in q} \prod_{i=1}^{T} P(z_i|X) - \log \prod_{u=1}^{U} P(y_u|z^*_{t_{u-1}+1:t_u}, X) \tag{3.4}$$

where P is the probability distribution, $Z^*$ is the most probable alignment (Viterbi-alignment). The second term is a maximum approximation for the log-posterior probability as computed by Eq. 3.3.

#### 3.1.2.2 Convolution Augmented Self-attention Block

The self-attention computation in Eq.3.1 considers global information across the sequence, but ignores local details. To alleviate this problem, convolution augmented self-attention blocks are proposed to emphasise the modelling of local dependencies of the input sequence in the encoder [GQC20, YWW19]. Different from previous work, we apply the convolution augmented self-attention blocks in the SAD and MAD as well. Specifically, the feed-forward layer is decomposed into two sub-layers to be placed at the beginning and the end of the block. A convolution layer similar to that in [GQC20] is inserted after the self-attention layer except that we empirically use layer normalization instead of batch normalization. The final computation in the $i^{th}$ MAD can be formulated as:

$$\hat{s}_i = s_i + \frac{1}{2}\text{FFN}(s_i) \tag{3.5}$$

$$s_i^{'} = \hat{s}_i + \text{LN}(\text{Attn}(\hat{s}_i, \hat{s}_i, \hat{s}_i, \text{NCM})) \tag{3.6}$$

$$s_i^{''} = s_i^{'} + \text{Conv}(s_i^{'}) \tag{3.7}$$

$$s_i^{'''} = s_i^{''} + \text{LN}(\text{Attn}(s_i^{''}, H, H, \text{NCM})) \tag{3.8}$$

$$o_i = \text{LN}(s_i^{'''} + \frac{1}{2}\text{FFN}(s_i^{'''})) \tag{3.9}$$

where LN indicates layer normalization and FFN is the feed-forward network. NCM is non-causal mask. $s_i$ and $o_i$ are the input and output of block $i$, respectively. The convolution-augmented self-attention block can be used for other NAT models.

Different from the usage of relative positional encoding in [DYY19, GQC20], we consider a maximum length of the relative position k as in [ZFC19]. Therefore, $2k + 1$ position embedding are learned to represent the relative position between $[-k, k]$.

#### 3.1.2.3 Intermediate Loss

Since CTC and CE loss functions are jointly optimized in the CASS-NAT framework, we incorporate intermediate CTC and intermediate CE loss functions into Eq.3.4 so that the parameters in different layers can be updated at the same scale.

Let $L_{dec} = -\log \prod_{u=1}^{U} P(y_u | z^*_{t_{u-1}+1:t_u}, X)$ and $L_{CTC} = -\log \sum_{Z \in q} \prod_{i=1}^{T} P(z_i | X)$, the objective function is re-written as:

$$\begin{aligned} L_{\text{joint}} &= \lambda_{CE} L_{dec}^{final} + (1 - \lambda_{CE}) L_{dec}^{mid} \\ &+ \lambda_{CTC} L_{CTC}^{final} + (1 - \lambda_{CTC}) L_{CTC}^{mid} \end{aligned} \tag{3.10}$$

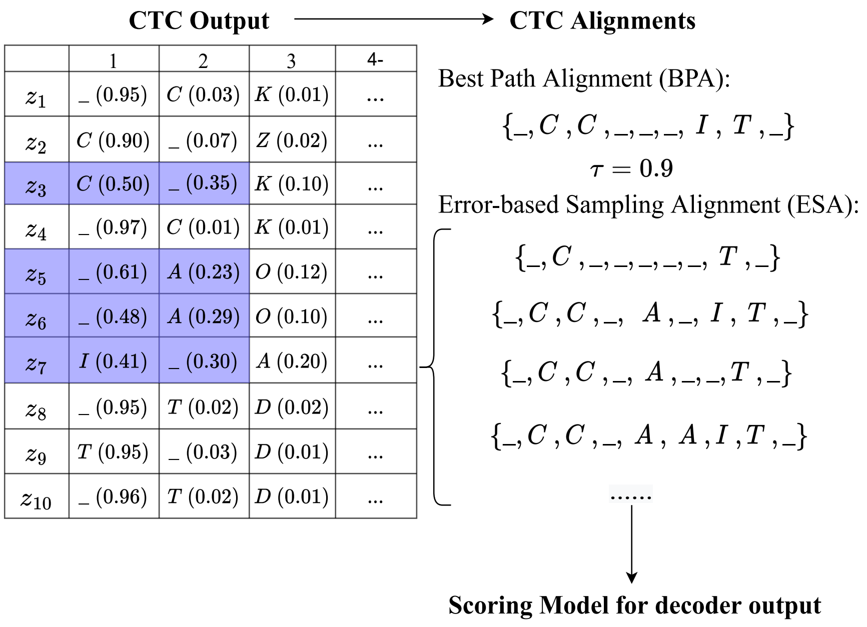
where $\lambda_{CE}$ and $\lambda_{CTC}$ are task ratios. $mid$ and $final$ indicate the layer position of the inserted loss functions. We found intermediate loss to be more effective for CASS-NAT than AT models [FCC21b]. The intermediate loss can be added to other NAT models as well.

#### 3.1.2.4 Trigger Mask Expansion

The quality of TAE relies on the accuracy of the trigger mask (TM), which is mapped from the CTC alignment. Although the CTC loss function is used to optimize the alignment, there are still errors when doing forced alignment over the CTC output space, leading to an inaccurate TM. To address this issue, we expand TM to include contextual frames for each token. For example, suppose the contextual frame size is one, the acoustic boundary of token U becomes $\{z_{t_{u-1}}, ..., z_{t_u+1}\}$. The trigger mask will then be expanded by one in the subsequent acoustic embedding extraction. Note that trigger mask expansion is used only for CASS-NAT.

### 3.1.3 Inference: Error-based Sampling Decoding

During decoding, it is essential to obtain a CTC alignment that is close to the hypothetical Viterbi-alignment used in training. The transcription, however, is not available. We propose to use three different alignment generation methods for inference: (1) best path alignment (BPA); (2) beam search alignment (BSA); and (3) error-based sampling alignment (ESA). We also present the results of using Viterbi-alignment as an upper bound of WER, assuming that transcriptions are available during decoding, which is referred to as oracle alignment. **BPA** is similar to CTC greedy decoding that selects the token with the highest probability at each time step, but without removing blank and repetitive tokens in the final sequence. **BSA** is similar to beam search decoding over the CTC output space, which is the most


CTC Output
CTC Alignments
Best Path Alignment (BPA):
{_, C, C, _, _, _, I, T, _}
τ = 0.9
Error-based Sampling Alignment (ESA):
{_, C, _, _, _, _, _, T, _}
{_, C, C, _, A, _, I, T, _}
{_, C, C, _, A, _, _, T, _}
{_, C, C, _, A, A, I, T, _}
......
Scoring Model for decoder output

| | 1 | 2 | 3 | 4- |
|---|---|---|---|---|
| $z_1$ | _ (0.95) | $C$ (0.03) | $K$ (0.01) | ... |
| $z_2$ | $C$ (0.90) | _ (0.07) | $Z$ (0.02) | ... |
| $z_3$ | $C$ (0.50) | _ (0.35) | $K$ (0.10) | ... |
| $z_4$ | _ (0.97) | $C$ (0.01) | $K$ (0.01) | ... |
| $z_5$ | _ (0.61) | $A$ (0.23) | $O$ (0.12) | ... |
| $z_6$ | _ (0.48) | $A$ (0.29) | $O$ (0.10) | ... |
| $z_7$ | $I$ (0.41) | _ (0.30) | $A$ (0.20) | ... |
| $z_8$ | _ (0.95) | $T$ (0.02) | $D$ (0.02) | ... |
| $z_9$ | $T$ (0.95) | _ (0.03) | $D$ (0.01) | ... |
| $z_{10}$ | _ (0.96) | $T$ (0.02) | $D$ (0.01) | ... |

Figure 3.2: Illustration of obtaining CTC alignments from CTC outputs, including best path alignment (BPA) and error-based sampling alignment (ESA). $C(0.90)$ indicates $P(z_i = C|X) = 0.90$. "_" is a blank token. The threshold $\tau$ for sampling is set to 0.9.

probable alignment during decoding. Compared to BPA, BSA is supposed to generate an alignment that is closer to the oracle alignment, which could lead to a lower WER, but the parallelism of the CASS-NAT would be destroyed, resulting in a significant increase of the real time factor (RTF).

Considering the expectation in Eq. 3.3, we propose a third alignment generation approach by sampling based on the potential errors in BPA. The method is referred to as error-based sampling alignment (ESA). To generate ESA, a threshold $\tau$ determines whether sampling is required at each time step. If the highest probability of the output distribution is larger than $\tau$, the rule of BPA holds. Otherwise, we randomly sample from the tokens with the first two largest probabilities. As shown in shaded blue in Figure 3.2, CTC outputs at $z_3$, $z_5$, $z_6$, and $z_7$ need a sampling because of their low top-1 probability. According to the proposed sampling rule, four sampled alignments are shown as examples on the right side of Figure 3.2. The reason for sampling within the top-2 tokens is that the trigger mask is sensitive to blank tokens and most mistaken outputs in BPA contain blanks in the top-2 tokens. In addition, sampling within 2 tokens is efficient because of the small sampling space. ESA aims at correcting the output which is prone to errors. Sampling in the decoding stage will not affect much inference speed because it can be done in parallel. It is possible that ESA-generated alignments are closer to the oracle alignment than BPA. Compared to BSA, ESA can be implemented in parallel, avoiding any increase in the RTF. Finally, either an AT baseline or language model can be used to score and identify the best overall alignment. Note that ESA can have different lengths for decoder input compared to BPA. As shown in Figure 3.2, the decoder length is 4 for BPA (including an EOS token), while the lengths are 3, 5, 4, and 5 in the case of ESA. This fluctuation of the token numbers allows ESA to possibly sample an alignment that is of the same length as the oracle alignment, which is important for the performance of NAT models as will be shown experimentally. Note that ESA decoding is proposed specifically for CASS-NAT.

## 3.2 Experimental Setups

### 3.2.1 Data Preparation

To examine the effectiveness of the proposed framework, we conduct several ASR tasks, including read and spontaneous speech, English and Mandarin speech, and adult and child speech. Four datasets are selected: (1) the 960-hour LibriSpeech English corpus [PCP15] with read speech, (2) the 178-hour Aishell1 Mandarin corpus [BDN17] with adult read speech, (3) the 210-hour TEDLIUM2 (TED2) English corpus [RDE14] with TED talk speech, and (4) My Science Tutor (MyST) Kids English corpus [WC11] with spontaneous speech. We use the annotated part of MyST, which accounts for 42% (240 hours) of the corpus.

All experiments use 80-dim log-Mel filter-bank features, computed every 10ms with a 25ms Hamming window. Features of every 3 consecutive frames are concatenated to form a 240-dim feature vector as the input. The sets of output labels consist of 5k word pieces for LibriSpeech, and 500 word pieces for TED2 and for MyST. All sub-words are obtained by SentencePiece [KR18] using the training set of each dataset. 4230 Chinese characters are used as vocabulary for the Aishell1 dataset. To avoid overfitting, we applied speed perturbation [KPP15] and SpecAugment [PCZ19] to the filter-bank features from TED2, Aishell1 and MyST. Speed perturbation is not used for LibriSpeech because of limited computational resources.

### 3.2.2 Network Architecture

#### 3.2.2.1 Autoregressive Models

A CTC/Attention AT baseline is first trained with the architecture ($N_e = 12$, $N_d = 6$, $d_{ff} = 2048$, $nh = 8$, $d_{att} = 512$) for LibriSpeech, and ($N_e = 12$, $N_d = 6$, $d_{ff} = 2048$, $nh = 4$, $d_{att} = 256$) for the other three datasets, where $d_{ff}$ is the dimension of the FFN module, $d_{att}$ stands for the dimension of the attention module, $nh$ is the number of attention heads, $N_e$ and $N_d$ are the number of encoder and decoder blocks, respectively. Prior to the encoder, two convolution layers with 64 filters, a kernel size of 3, and a stride of 2 is adopted, leading to a 4x frame-rate reduction. When using a conformer structure to the AT encoder, we reduce the $d_{ff}$ to be 1024 to keep the number of parameters in the model the same as in the transformer baseline, and the maximum length of relative position k is set to 20. The kernel size in the convolution module is 31 for LibriSpeech and 15 for the other three datasets.

#### 3.2.2.2 CASS-NAT

During training, CASS-NAT encoder is initialized with an AT encoder for faster convergence as in [FZC19]. The decoder in AT baseline is replaced by 1-block TAEE, m-block SAD and n-block MAD. $m$ and $n$ are investigated using the LibriSpeech dataset, and the best setting is applied to the other three datasets. For convolution-augmented decoder, the dimension of feed-forward layers is also halved and the maximum length of relative position is set to 8 for the tasks with word piece units and 4 for the Aishell1 data. The contextual frame of trigger mask expansion is set to 1 because we do not see further improvements with a larger expansion. The intermediate loss functions are inserted in the middle layer of the encoder and MAD with $\lambda_{CE}$ of 0.99 and $\lambda_{CTC}$ of 0.5. The inserted projection layers are discarded during inference. These settings were chosen empirically.

#### 3.2.2.3 Training and Decoding Setup

All experiments are implemented with Pytorch [PGM19] [1]. The Double schedule in [PCZ19] is adopted as the learning schedule for the LibriSpeech and Aishell datasets, where the learning rate is ramped up to and held at 0.001, then be exponentially decayed to 1e-5. The transformer-based scheduler in [VSP17] with warm-up steps of 25k and noam factor of 5 is used for the TED2 and MyST datasets. Layer normalization, dropout with rate of 0.1 and label smoothing with a penalty of 0.1 are all applied as the common strategies for training a transformer. We compute WERs of development sets for early stopping (no improvement for 11 epochs). The last 12 epochs are averaged for final evaluation. Most experiments end within 90 epochs. In order to investigate the impact of different models for scoring alignments in ESA, a transformer-based language model is trained with the provided text in LibriSpeech. The provided n-gram models in the dataset are also compared in the experiments. In terms of the pretrained HuBERT encoders, we use the Fairseq model that is trained from LibriSpeech 960-hour corpus for finetuning the English data model and the Tencent model that is trained from 10000-hour WenetSpeech [ZLG22] for finetuning the Mandarin data model.

During AT decoding, the beam size is set to 20 for LibriSpeech, and 10 for the other three datasets. No external language models are used during beam search decoding. The evaluation of the real time factor (RTF) is conducted using a V100 GPU with a batch size of one. For CASS-NAT decoding with ESA, the threshold, number of sampled alignments and scoring models are investigated in Section 3.3.2.

[1] Our code is available at https://github.com/Diamondfan/cassnat_asr

Table 3.1: WERs for different block numbers (m and n) of the self-attention decoder (SAD) and mixed-attention decoder (MAD) using LibriSpeech, shown as mSAD+nMAD. Various mask matrices used in MAD are also explored, where CM, NCM and TM represent causal mask, non-causal mask and trigger mask, respectively. For example, NCM + TM indicates that NCM is used for self-attention in MAD and TM is used for source-attention in MAD. Bold numbers represent the best results.

| Decoder Structure | Mask in MAD | dev-clean | dev-other | test-clean | test-other |
|---|---|---|---|---|---|
| 7SAD + 0MAD | - | 5.3 | 11.1 | 5.4 | 11.1 |
| 5SAD + 2MAD | NCM + NCM | **4.7** | **10.4** | **4.8** | **10.3** |
|  | NCM + TM | 4.7 | 10.5 | 4.9 | 10.5 |
|  | CM + NCM | 4.8 | 10.7 | 4.9 | 10.6 |
| 3SAD + 4MAD | NCM + NCM | 4.7 | 10.4 | 4.8 | 10.4 |
| 1SAD + 6MAD | NCM + NCM | 4.6 | 10.5 | 4.7 | 10.4 |

## 3.3 Results and Discussion

The first set of ASR experiments used the LibriSpeech dataset to explore various decoder structures, impact factors in ESA decoding, CTC alignment behaviour, and the effectiveness of the proposed training strategies. Using the best settings found with the LibriSpeech task, experiments are run with the other three datasets.

### 3.3.1 The Structure of CASS-NAT Decoder

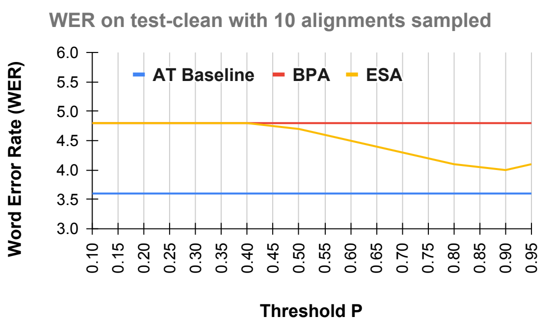
To investigate the structure of the CASS-NAT decoder, we use various combinations of $m$ SAD blocks and $n$ MAD blocks ($m + n = 7$) in the decoder to keep the number of model parameters similar to that of the AT baseline. We use best path alignment during inference and discuss other decoding strategies in the next section. Results are shown in Table 3.1. As shown in the table, not using MAD (0 MAD in the first row) causes a big performance degradation compared to other configurations, indicating that token-level speech representations might have to retrace the fine-grained frame-level information (encoder outputs) for better contextual modelling. The best performance is achieved when using 5 SADs and 2 MADs (WER of 10.3% on the test-other set).

Because there are two attention layers (self-attention and source-attention) in each MAD, we try different mask matrices for the two layers. For the self-attention layer, either a causal mask (CM) or non-causal mask (NCM) is considered. For the source-attention layer, either a trigger mask (TM) or NCM is used. The results in Table 3.1 show that using NCM on both attention layers results in the best WER. Although TAE is regarded as a substitution of word embedding, CM seems not necessarily required in the CASS-NAT decoder for contextual modelling of token-level acoustic embeddings. The settings that achieve the best performance in this section (5 SADs and 2 MADs with NCM in both attention layers) are selected as default for subsequent experiments.

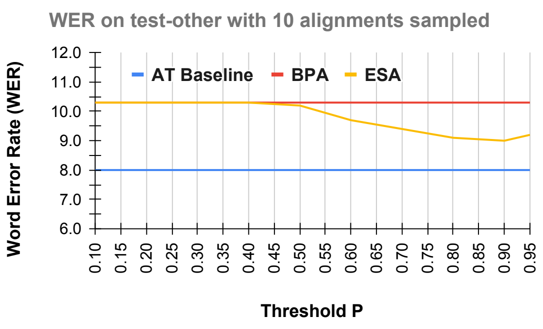
### 3.3.2 Error-based Sampling Alignment (ESA) Decoding

Viterbi alignment is used in training, but not available during inference. To reduce the alignment mismatch between training and inference, we propose error-based sampling alignment (ESA). There are three important factors that can affect the WER performance of ESA: sampling threshold $P$, the number of sampled alignments $S$ and scoring model for ranking alignments. We use the best decoder structure in the previous section (5SAD + 2MAD) to evaluate the performance of ESA decoding with different configurations.

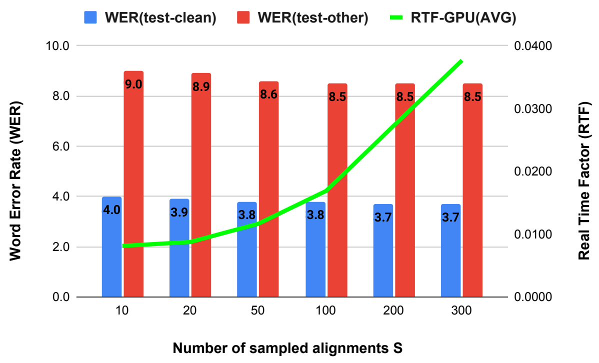
Figures 3.3a and 3.3b show the results when the threshold $P$ varies from 0.10 to 0.95 using the LibriSpeech test-clean and test-other data, respectively. The number of sampled alignments here is 10 and the scoring model is the AT baseline. We also include the WERs of the AT baseline and CASS-NAT decoding with best path alignment (BPA) as comparisons. As shown in the figures, ESA reaches the best performance when $P = 0.9$. A higher threshold indicates fewer sampled alignments, and thus no further improvement is observed. The threshold $P$ is set to 0.9 in subsequent experiments.



(a) Effect of $P$ using test-clean



(b) Effect of $P$ using test-other



(c) Effect of the number of sampled alignments $S$

Figure 3.3: WER performance of different values of the threshold $P$ and the number of sampled alignments $S$ in error-based sampling alignment (ESA) decoding. Real time factor (RTF) is evaluated using a V100 GPU with a batch size of one.

Figure 3.3c shows the effect of the number of sampled alignments $S$ in terms of WER and RTF on the test-clean and test-other data. As observed in the figure, by increasing the number of sampled alignments, the WER of ESA decoding improves but the improvement is small when $S$ is greater than 50. Meanwhile, the RTF increases rapidly as $S$ increases. Overall, $S = 50$ might be the most appropriate value to use as default in subsequent experiments.

Finally, various scoring models are compared. We consider using the AT baseline, neural language model (NLM) and n-gram language model for ranking the sampled alignments. NLM is a transformer-based model trained with the provided text in the LibriSpeech corpus. The n-gram models are also the ones provided in the LibriSpeech corpus. The results of ESA

Table 3.2: Comparisons, in terms of WER and GPU speedup, of BPA, BSA, and ESA decoding. NLM is the transformer-based neural language model. 3-gram and 4-gram models are the ones offered in LibriSpeech. Bold numbers are the best WER performance other than the Oracle and AT baseline.

| | Scoring Model | dev-clean | dev-other | test-clean | test-other | GPU Speedup |
|---|---|---|---|---|---|---|
| AT baseline | - | 3.4 | 8.1 | 3.6 | 8.0 | 1.00x |
| Oracle | - | 2.1 | 5.5 | 2.2 | 5.3 | 39.6x |
| BPA | - | 4.7 | 10.4 | 4.8 | 10.3 | 90.0x |
| BSA | - | 3.8 | 8.8 | 3.9 | 8.8 | 0.90x |
| ESA+ | AT baseline | **3.6** | **8.8** | **3.8** | **8.6** | 28.4x |
| | NLM | 3.6 | 8.9 | 3.9 | 8.7 | 31.6x |
| | 3-gram | 5.4 | 11.4 | 5.8 | 11.4 | 32.6x |
| | 4-gram | 5.4 | 11.3 | 5.7 | 11.3 | 31.5x |

decoding together with that of BPA and ESA decoding are shown in Table 3.2. As analyzed in Section 3.1.3, BPA has an impressive speedup but the WER is much worse than the AT baseline, and BSA can obtain a better WER but it is even slower than the AT baseline. By using neural network based scoring models, ESA can retain both the advantages of BPA and BSA. For example, using NLM as a scoring model, ESA can achieve a WER of 8.7% on the LibriSpeech test-other data and 31.6x speedup. The degradation in speedup compared to BPA originates from the ranking process. Using n-gram models for ranking alignments leads to worse WER compared to NLM in ESA because n-gram models are worse than NLM for language modelling. Another possible reason might be that the probability distribution of n-gram models is different from that of CASS-NAT decoder outputs, while for NLM it is similar. The reason why n-gram models have similar GPU speedup compared to NLM is that the scores of the sampled alignments cannot be obtained simultaneously. The best performing scoring model is the AT baseline, and thus the AT baseline is used in ESA decoding as default in the following experiments.

Note that because of the sampling process in ESA decoding, WER may be slightly different for different seeds of the random number generator. Fortunately, the randomness is small with a variance of around 0.5% relatively in our experiments.

Table 3.3: Comparisons of different decoding methods for CASS-NAT decoding. **Oracle**: Viterbi alignment with ground truth. **MR**: mismatch rate; **LPER**: length prediction error rate (using word-piece as the modeling unit). **S**: the number of alignments sampled in ESA. Bold numbers are the best performance other than the Oracle.

| Decoding Method | S | WER (%) | | MR (%) | | LPER (%) | |
|---|---|---|---|---|---|---|---|
| | | test-clean | test-other | test-clean | test-other | test-clean | test-other |
| Oracle | 1 | 2.2 | 5.3 | - | - | - | - |
| BSA | 1 | 4.0 | 9.2 | 2.5 | 6.0 | 28.09 | 48.83 |
| BPA | 1 | 4.8 | 10.3 | **2.4** | **5.2** | 34.92 | 51.68 |
| ESA | 10 | 4.0 | 9.0 | 3.6 | 6.0 | 26.41 | 44.91 |
| | 50 | 3.8 | 8.6 | 3.8 | 6.3 | 25.46 | 43.08 |
| | 100 | 3.8 | 8.5 | 3.8 | 6.2 | 25.61 | 42.70 |
| | 300 | **3.7** | **8.5** | 3.8 | 6.3 | **25.53** | **42.67** |

### 3.3.3 Why does ESA Work? - An Analysis of the CTC Alignments

In this section, we analyze the CTC alignments obtained from different decoding strategies to understand why ESA can improve the performance of CASS-NAT. Two metrics are evaluated on the LibriSpeech test sets: mismatch rate (MR) and length prediction error rate (LPER). MR and LPER are measured between the alignments used in decoding and the oracle alignment, in which the blank and repetitive tokens are removed. The MR is the ratio of deletion and insertion errors compared to the oracle alignment, and substitution errors are not included because they do not change either the acoustic boundary for each token or the number of predicted tokens. If the number of tokens in an alignment is different from that in the oracle alignment, this alignment is considered as a case of length prediction error. LPER is the percentage of the utterances with length prediction errors.

Results of MR and LPER are presented in Table 3.3. Note that the lower bound of WER (using oracle alignment) is 2.2% on the test-clean data assuming the transcriptions are available during decoding; this indicates that the framework is promising. When comparing

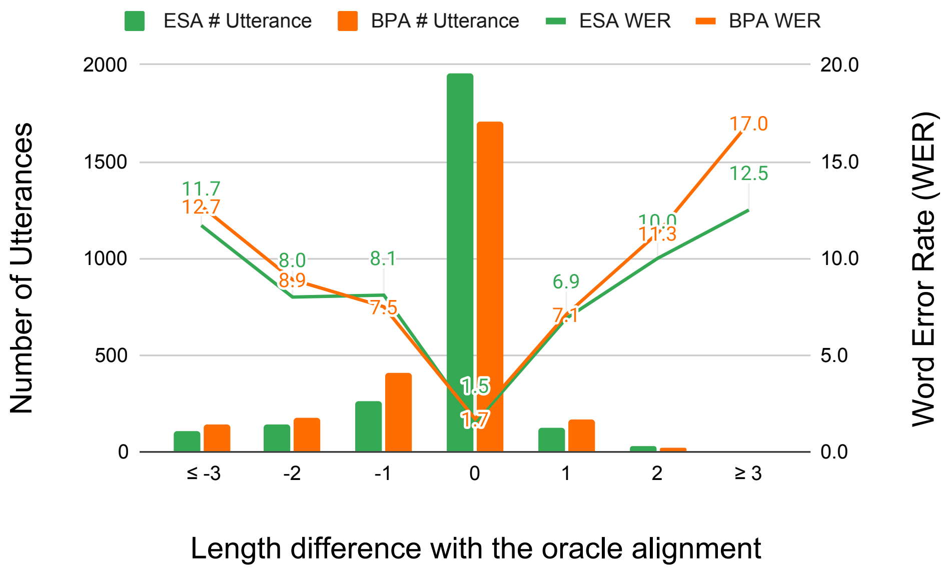


Figure 3.4: The length prediction error distribution and their corresponding WERs using ESA(s=50) decoding on the test-clean dataset.

the two metrics for different decoding strategies, we observe that the WER is more correlated with LPER than MR. This suggests that a correct estimation of the output length (the length of decoder input) is very important for NAT, which is also mentioned in [CWV20]. The reason may be because CASS-NAT decoders have a stronger ability of correcting mistakes in CTC alignment by contextual token-level acoustic modelling when the length prediction is accurate. To further validate this assumption, we plot the length prediction error distribution and their corresponding WERs in Figure 3.4. As seen from the figure, CASS-NAT achieves WERs that are lower than 2% when the length of the decoder input is estimated correctly, while the WERs are higher than 10% for those utterances with an absolute difference in length of more than 3 compared to the oracle alignment.

### 3.3.4 Improving the Training of CASS-NAT

Despite the use of ESA decoding method, the WER performance of CASS-NAT (8.6%) is still worse than that of the AT baseline (8.0%). In this section, we attempt to improve the WER performance of CASS-NAT by using various training strategies that include convolution-

Table 3.4: WERs of using the proposed training strategies on the LibriSpeech data. SpecAug is used in all configurations. WERR is the relative WER improvement compared to their corresponding baselines on the test-other data. KD represents using knowledge distillation from AT decoder outputs. Other strategies are the ones introduced in Section 3.1.2. Bold faced numbers indicate the best WER results.

| Model w/o LM | ConvEnc. | InterCTC | ConvDec. | InterCE | Mask Exp. | KD | dev-clean | dev-other | test-clean | test-other | *WERR* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AT | | | | | | | 3.4 | 8.1 | 3.6 | 8.0 | - |
| | ✓ | | | | | | 2.7 | 6.9 | 2.9 | 6.9 | 13.8% |
| | ✓ | ✓ | | | | | 2.7 | 6.8 | 2.8 | 6.7 | 16.3% |
| CASS-NAT | | | | | | | 3.6 | 8.8 | 3.8 | 8.6 | - |
| | ✓ | | | | | | 3.0 | 7.3 | 3.2 | 7.5 | 12.8% |
| | ✓ | ✓ | | | | | 2.9 | 7.4 | 3.1 | 7.3 | 15.1% |
| | | | ✓ | | | | 3.4 | 8.3 | 3.6 | 8.4 | 2.3% |
| | | | ✓ | ✓ | | | 3.3 | 8.2 | 3.6 | 8.3 | 3.5% |
| | ✓ | ✓ | ✓ | ✓ | | | 2.8 | 7.1 | 2.9 | 7.1 | 17.4% |
| | | | | | ✓ | | 3.6 | 8.9 | 3.8 | 8.7 | -1.2% |
| | | | ✓ | ✓ | ✓ | | 3.3 | 8.3 | 3.6 | 8.1 | 5.8% |
| | ✓ | ✓ | ✓ | ✓ | ✓ | | **2.7** | **7.1** | **2.9** | **7.0** | 18.6% |
| | | | | | | ✓ | 3.6 | 8.8 | 3.8 | 8.5 | 1.2% |

augmented encoder (ConvEnc.), intermediate CTC loss (InterCTC), convolution augmented decoder (ConvDec.), intermediate CE loss (InterCE) and trigger mask expansion (Mask Exp.). The results of various combinations of the training strategies are shown in Table 3.4.

First, we apply ConvEnc. and InterCTC to the autoregressive transformer to create a new AT baseline for fair comparisons. Note that, we have applied ConvDec. and InterCE to the AT baseline as well, but no improvements are observed, and thus their results are not included. When ConvEnc. and InterCTC are applied to CASS-NAT, similar improvements compared to their use for AT are observed. The improvements stem from their ability of capturing local details in the acoustic representations, and thus a stronger encoder is learned and it offers an accurate CTC alignment for implicit token-level acoustic embedding modelling in the decoder.

Second, when using ConvDec. and InterCE, the WER of CASS-NAT drops further to 7.1% on the test-other data, which is a 17.4% relative WER improvement over the CASS-NAT baseline. The use of ConvDec. and InterCE, however, is not as effective as ConvEnc and InterCTC, indicating that a stronger modelling of frame-level acoustic representations is more important than that of token-level acoustic representations (TAEs) in CASS-NAT. This is reasonable because TAEs are extracted from frame-level acoustic representations from the encoder. Interestingly, we observe a -1.2% relative WER reduction when the trigger mask expansion method is applied by itself. However, the method leads to WER improvements when it is used together with ConvDec. related strategies. The reason could be that expanding the acoustic boundary for each token might cause confusion for the decoder when the contextual modelling is not strong enough.

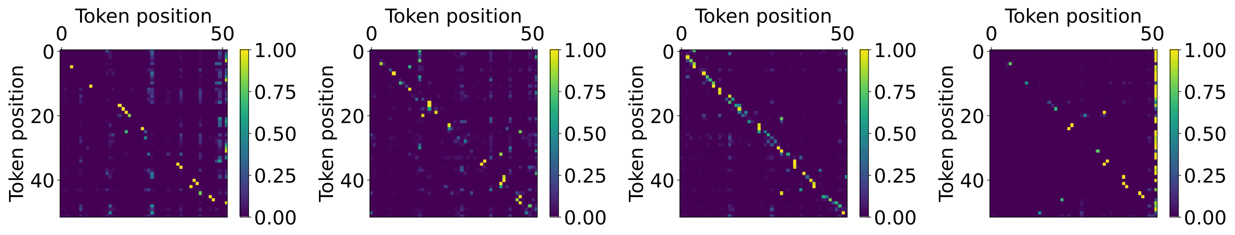
In addition, as shown in the last line of Table 3.4, applying knowledge distillation, which takes the AT output as a target for NAT training, does not result in WER improvements. Although knowledge distillation has been proven useful in non-autoregressive neural machine translation in [ZGN20], it does not seem to work for speech recognition possibly because there are no different outputs corresponding to the same input, while multiple translations of the source language exist in the target language for machine translation.

Finally, using all the training strategies, we achieve an 18% relative WER improvement compared to CASS-NAT baseline; this is only a 3% WER degradation compared to the best version of the AT baseline. In summary, we obtain a non-autoregressive speech transformer that performs close to its autoregressive counterpart with a significant GPU speedup.

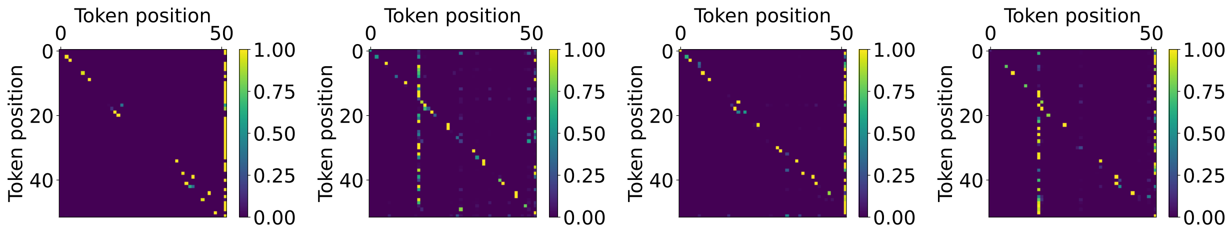
### 3.3.5 Why does CASS-NAT Work? - An Analysis of the Decoder

In this section, we explain why CASS-NAT can achieve a good performance that is close to its autoregressive counterpart by analysing the following elements in CASS-NAT decoder: (1) attention weight distribution; and (2) acoustic-level acoustic representations.

First, to analyse the behaviour of the attention mechanism in CASS-NAT decoder, we choose the first utterance in the LibriSpeech train-clean-100 subset and plot the attention weight distribution in the last SAD and MAD blocks. The weights of the last 4 heads (8 heads in total) are shown in Figure 3.5. For self-attention weight distributions in SAD, we notice from the figure that most of the heads learn a monotonic alignment between token-level acoustic representations, indicating that each token relies on adjacent tokens. This is

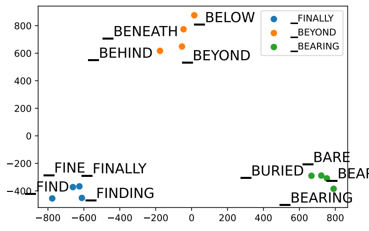


(a) The 5th $\sim$ 8th head of self-attention in the last SAD

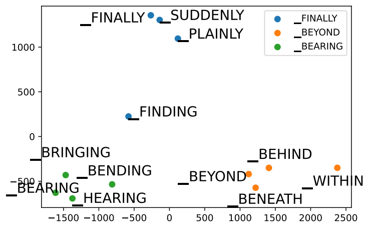


(b) The 5th $\sim$ 8th head of self-attention in the last MAD

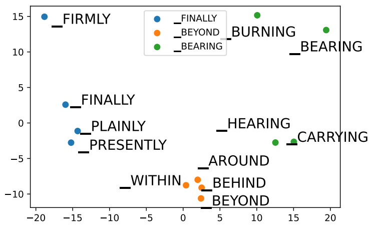
Figure 3.5: Attention weight distributions of the 5th $-$ 8th head in the last self-attention decoder (SAD) and mixed-attention decoder (MAD). The first utterance in the LiriSpeech train-clean-100 subset is used. The y-axis represents output tokens



(a) TAEE output



(b) SAD output



(c) MAD output

Figure 3.6: Visualization of token-level acoustic embeddings of three word pieces from outputs of TAEE, SAD and MAD, respectively. The embeddings are plotted using principle component analysis (PCA).

similar to the idea of word embedding using a continuous bag of words (CBOW) and skip-gram [MCC13]. The monotonic alignment also shows the usefulness of the relative positional encoding because distant tokens with close semantic similarity have low attention weights. For the attention weights in MAD, there exists a different behaviour in several subspaces of the attention computation, where outputs may rely on the same input (vertical line in Figure 3.5b).

As discussed in Section 3.1.2.1, we assume that language semantics can be captured by CASS-NAT decoder using token-level acoustic embeddings. To validate this assumption, we calculate three different token-level acoustic embeddings from TAEE, SAD and MAD for each token (word piece in our case). Specifically, token-level acoustic representations are first extracted from the first 5000 utterances in the train-clean-100 Librspeech subset, where each representation has its label in the form of a word piece. The embedding of each word piece is the average of the corresponding representations. After that, we randomly choose three word pieces and find the four closest ones (measured with cosine similarity distance) for each word piece. Using the same idea of visualizing word embeddings, the 12 selected embeddings are reduced to a 2-dimensional space using principal component analyses (PCA) and are then plotted. Examples of embeddings at different levels of the CASS-NAT decoder are shown in Figure 3.6. The figure suggests that the token-level acoustic embeddings learn not only acoustic similarities (_**BEAR** vs. _**BARE**), but also token-level semantic similarities (_**BENEATH** vs. _**BELOW**). This occurs, even though embeddings at different levels of the decoder provide different information. For example, higher layer (MAD) embeddings focus more on grammatical similarities (e.g. _**FIRMLY** and _**FINALLY**), which is similar to word embedding. This finding suggests the possibility of learning a joint speech and text embeddings in a common space. Even though there is no explicit language modelling, the CASS-NAT decoder is able to learn meaningful embeddings, which may explain why it has a similar performance to its AT counterpart.

### 3.3.6 Results on other datasets

Finally, we apply the proposed CASS-NAT to the other three datasets: Aishell (Mandarin speech), TED2 (English spontaneous speech), and MyST (child speech), to show the generalization ability of CASS-NAT. In addition, we explore the effect of different initialization schemes for the CASS-NAT encoder and measure the speedup given by CASS-NAT com-

Table 3.5: WER Results of CASS-NAT using different encoders as a starting point, including random initialization (Rand. Enc.), initialization from an encoder trained with CTC (CTC enc.), and initialization from a AT encoder (AT Enc.). AT baselines with greedy and beam search decoding are included together with SOTA results using non-autoregressive methods in the literature. "-" indicates that WER results have not been reported in the literature. RTF is evaluated with batch size of one on GPU. Bold-faced numbers are the best results for CASS-NAT and SOTA without self-supervised pretraining.

| Model | Conditions | Inference Speed | | LibriSpeech | | | | Aishell1 | | TED2 | | MyST | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RTF | Speed | dev clean | dev other | test clean | test other | dev | test | dev | test | dev | test |
| AT | greedy search | 0.0387 | 1.00x | 2.7 | 7.0 | 2.9 | 7.0 | 5.1 | 5.9 | 9.8 | 7.8 | 18.1 | 17.0 |
| | beam search | 0.3348 | 0.12x | 2.7 | 6.8 | 2.8 | 6.7 | 4.7 | 5.1 | 8.2 | 7.6 | 16.5 | 16.2 |
| Previous NAT | w/o SSL | | | 2.8 | 7.3 | 3.1 | 7.2 [FCC21b] | 4.5 | 4.9 [YLG21] | 8.7 | 8.0 [HCF21] | 28.0 | 27.8 [FZW22] |
| | w/ SSL | | | 1.7 | 3.6 | 1.8 | 3.6 [NCZ21] | 4.1 | 4.5 [DYW22] | - | - | 16.8 | 16.5 [FA22] |
| CASS-NAT (this work) | Rand. Enc. | | | 3.3 | 8.4 | 3.6 | 8.3 | **4.7** | **5.0** | 8.4 | **7.5** | **16.8** | **16.4** |
| | CTC Enc. | 0.0134 | 2.89x | 2.8 | 7.2 | 3.1 | 7.3 | 5.2 | 5.6 | **8.0** | 7.7 | 17.1 | 16.7 |
| | AT Enc. | | | **2.7** | **7.1** | **2.9** | **7.0** | 5.1 | 5.4 | 8.2 | 7.8 | 17.1 | 16.7 |
| | HuBERT Enc. | | | - | - | - | - | 4.2 | 4.5 | - | - | 16.0 | 15.6 |

pared to AT baselines. All results are summarized in Table 3.5. As can be observed from the table, a randomly initialized encoder for CASS-NAT achieves the best performance on the Aishell1, TED2 and MyST datasets, while the LibriSpeech dataset performs better with a pretrained encoder. The reason could be that we force the training iterations of CASS-NAT to be the same as the AT baseline for time considerations, as well as for a fair comparison, but LibriSpeech, as a large database compared to the others, needs more training iterations to obtain accurate alignments for the decoder. Hence, we suggest using a pretrained encoder for training CASS-NAT on large datasets.

With an appropriate encoder initialization, we now compare the results of CASS-NAT to the AT baselines and SOTA results using NATs. First, when compared to the AT baseline with greedy search decoding, CASS-NAT achieves better (on Aishell1, TED2, and MyST) or similar (on LibriSpeech) WERs performance, and has a 2.89x RTF speedup. When compared to the AT baseline with beam search decoding, the WERs performance on Aisehll and TED2 are still better, while the WERs on LibriSpeech and MyST are close to the baseline but with a 24x RTF speedup. Second, when compared to other NAT methods in the literature, we observe from the table that the proposed CASS-NAT achieves new state-of-the-art results on LibriSpeech, TED2, and MyST when no pretrained acoustic (e.g. Wav2vec2.0 [BZM20]) or language models (e.g. BERT [DCL19]) are used. To compare with the results that used extra training data (e.g. SSL with un-annotated data), we use the self-supervised pretrained models as the encoder for CASS-NAT to see whether it can outperform the SOTA results in Table 3.5. Due to time and computational constraints, we obtain only the results on Aishell1 and MyST datasets. The results show that CASS-NAT can achieve comparable results to the best NATs in the literature [DYW22, FA22] with SSL pretraining.

## 3.4 Chapter Summary

In this chapter, we presented a novel non-autoregressive transformer (CASS-NAT) that can increase the inference efficiency for an ASR system. The foundation behind CASS-NAT is that it utilizes the alignments obtained from CTC output for token-level acoustic embeddings (TAEs) extraction, and regard the TAEs as substitutions of the word embeddings in autoregressive transformers (AT) to achieve parallel forward computation. During training, Viterbi-alignment was used to estimate the posterior probability, and various training strategies were applied to improve the WER performance of the CASS-NAT. During inference, we investigated in depth an error-based sampling alignment (ESA) method that we introduced to reduce the alignment mismatch between training and inference.

The extensive experiments on CASS-NAT have shown that: (1) a mixed-attention decoder (MAD) in CASS-NAT was important for reducing the WER; (2) the reason why ESA decoding works well was because it has a lower length prediction error rate; (3) convolution augmented encoder and decoder, intermediate loss and mask expansion can improve the WER of CASS-NAT, while knowledge distillation can not; and (4) TAEs had similar functionality to word embeddings, such as representing grammatical structures, indicating the possibility of learning semantics without a language model. As a result, CASS-NAT achieved new state-of-the-art results for NAT models on several ASR tasks with and without SSL pre-training. The datasets included English and Mandarin speech, read and spontaneous speech, and child and adult speech, showing the generaliztion ability of the proposed method. The performances of CASS-NAT were comparable, in relative terms, to AT with beam search decoding, but maintain a $\sim$20x speed up. Future work includes model compression and distillation to further reduce the parameters and improve the efficiency.

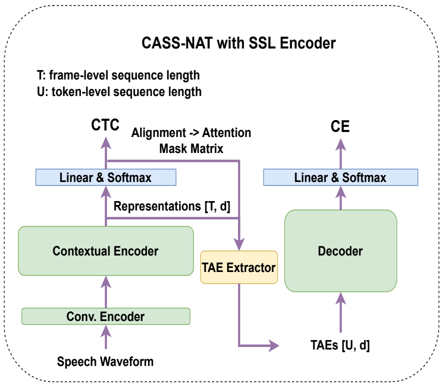
# CHAPTER 4

# Integration of SSL Pretraining with Non-autoregressive Models

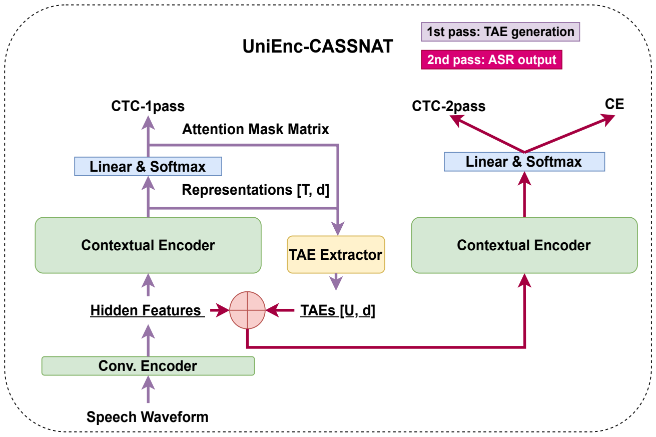
Previously, we have shown that self-supervised learning pretraining and proposed non-autoregressive models are effective for improving the accuracy and inference efficiency, respectively. However, the challenge of applying SSL to CASS-NAT is the model discrepancy between SSL and CASS-NAT (encoder for SSL versus encoder-decoder structure for CASS-NAT). This motivates us to propose an encoder-only CASS-NAT model that can benefit from SSL pretraining. Hence, in this chapter, we present a new encoder-only non-autoregressive model, UniEnc-CASSNAT, that can function in a way that is similar to CASS-NAT encoder and decoder. Additionally, we propose a multi-pass CTC (MP-CTC) training and iterative decoding method to improve the WER performance. Experiments on Librispeech 100-hour [PCP15], MyST [WC11], and Aishell1 [BDN17] datasets show that the proposed methods can achieve better or comparable WERs to CASS-NAT, and contain fewer parameters. This work was published in [FSA24].

## 4.1 An Encoder-only Non-autoregressive ASR for Speech SSL Models

In this section, we describe the newly proposed encoder-only NASR (UniEnc-CASSNAT). Like CTC, UniEnc-CASSNAT can be initialized from speech foundation models (HuBERT base model [HBT21] is used). To behave as both the CASS-NAT encoder and decoder, UniEnc-CASSNAT has two forward passes and accepts two types of inputs for each. In the first pass, speech features (output of HuBERT Conv. encoder) are fed into the contex-



(a) CASS-NAT



(b) UniEnc-CASSNAT

Figure 4.1: (a) Diagram of CASS-NAT. (b) Proposed UniEnc-CASSNAT with HuBERT conv. and contextual encoders. The TAE extractor is a self-attention module that transforms the acoustic representations with length T to TAEs with length U. The generation of TAEs and second pass forward computation are repeated during iterative decoding.

tual encoder to generate token-level acoustic embeddings (TAEs). In the second pass, the concatenation of speech features and the TAEs (along the time dimension) are used as the contextual encoder inputs. The TAE corresponding outputs are selected for ASR loss computation. The outputs in the second pass can generate better quality TAEs than those in the first pass and hence lead to better ASR performance. We, therefore, propose a multi-pass CTC (MP-CTC) training and iterative decoding method to improve performance.

### 4.1.1 Proposed Framework: UniEnc-CASSNAT

**CASS-NAT Review:** CASS-NAT consists of an encoder, a token-level embedding extractor (TAEE), and a decoder as shown in Figure 4.1(a). The connectionist temporal classification (CTC) [GFG06] loss is added to learn the alignment between the acoustic and token sequences. The alignment can provide segmentation information for each token. TAEE extracts an embedding for each token from encoder outputs (with a shape of $[T, d]$, where $T$ is the frame length and $d$ is the hidden dimension) using the segmentation information. The extracted token-level acoustic embeddings (TAEs) (with a shape of $[U, d]$, where $U$ is the token sequence length) are fed into the decoder which models the relationship between tokens. During decoding, we use error-based sampled alignments (ESA) (see details in 3.1.3), where the multiple alignments $Z$ are sampled based on the CTC greedy search output with low confidence scores. The TAEs computed from the sampled alignments are fed into the decoder to obtain multiple ASR outputs. The autoregressive transformer provides a ranking score for each ASR output (one ASR output corresponds to one alignment).

**An Encoder-only CASS-NAT (UniEnc-CASSNAT):** Speech foundation models are proven to be useful in downstream ASR tasks. The CASS-NAT encoder can be initialized from a speech foundation model and extracts better acoustic representations [FCC23] when compared to training from scratch. However, the CASS-NAT decoder has to be trained from scratch. Inspired by the success of recent work of speech-text joint pre-training [TGD22, WKB23] with a shared encoder, we rethought the necessity of the CASS-NAT decoder and propose an encoder-only CASS-NAT, denoted as UniEnc-CASSNAT to match the size of speech foundation models.

The UniEnc-CASSNAT is shown in Figure 4.1(b) with two forward passes. In the first pass, the hidden features extracted from the conv. encoder are fed into the contextual encoder for CTC modeling and the token-level acoustic embeddings (TAEs) are extracted using the alignment information from CTC outputs. In the second pass, the extracted TAEs ($[U, d]$) are concatenated with the hidden features ($[T, d]$) (along the time dimension) to be the input to the contextual encoder. The self-attention layer in the contextual encoder enables frame-frame, frame-token, and token-token interactions between hidden features and TAEs. Note that the goal of the first pass is to obtain TAEs, whose quality is highly related to the ASR performance. The better the speech foundation model, the better the quality of the TAEs extracted by UniEnc-CASSNAT. The second pass is similar to the role of the CASS-NAT decoder for modeling the relationships between TAEs and frame-level hidden features. We investigate whether the encoder is capable of both frame-level acoustic representation learning and contextual modeling between tokens.

### 4.1.2 MP-CTC Training and Iterative Decoding

The output of the second pass is a sequence of $T + U$ vectors, where the first $T$ vectors correspond to hidden features, and the $U$ vectors correspond to TAEs. Since the quality of TAEs is essential to the performance of the CASS-NAT decoder, we propose to add another CTC loss to the first $T$ outputs of the second pass and formulate a multi-pass CTC (MP-CTC) training. With the CE loss used on the $U$ outputs, the final objective function of UniEnc-CASSNAT can be written as:

$$L_{\text{unienc-cassnat}} = L_{\text{dec}} + \lambda_1 \cdot L_{CTC-1pass} + \lambda_2 \cdot L_{CTC-2pass} \tag{4.1}$$

We share the final feed-forward layer for the two CTC losses. Theoretically, the second-pass CTC loss would have better performance than the first pass because it accepts additional input information (TAEs). An intuitive idea is to iteratively improve the quality of TAEs by repeating the second pass with newly extracted TAEs. Hence, we propose an iterative decoding method for UniEnc-CASSNAT. Specifically, we define the hidden features as $H$, and the first pass of UniEnc-CASSNAT encoder as $\text{Iter}_0$. $\text{Iter}_0$ would generate $\text{TAE}_0$. The second pass uses $H+\text{TAE}_0$ as input and generates $\text{TAE}_1$, which we define as $\text{Iter}_1$. Generally, for iteration $n$, the contextual encoder accepts $H$ and $\text{TAE}_{n-1}$ as input and generates $\text{TAE}_n$ for the iteration $n+1$. In each iteration, ESA generates multiple TAEs for the next iteration. We define the number of sampled alignments in each iteration as $S_n$. The total number of the sampled alignments would be $\prod_{n=0}^{N-1} S_n$, where $N$ is the number of iterations used in the decoding. We empirically found that two iterations are sufficient for a desirable word error rate (WER).

## 4.2 Experimental Setup

### 4.2.1 Data Settings

The experiments were conducted with three datasets: the 100-hour subset of LibriSpeech English corpus [PCP15], the 240-hour (annotated section) My Science Tutor (MyST) children's speech corpus [WC11], and 170-hour Aishell1 Mandarian corpus [BDN17]. We chose the 100-hour subset of Librispeech to enable comparisons with previous work on non-autoregressive automatic speech recognition (NASR). We conducted pre-processing on MyST dataset to get a better baseline compared to [FCC23]. For example, we mapped filling pauses, non-speech events, and truncated words to ⟨UNK⟩. The ⟨UNK⟩ is not considered when computing WER.

The sets of output labels consist of 1024 word-pieces for Librispeech 100h and 500 word-pieces for the MyST, obtained by the SentencePiece method [KR18]. For Aishell, 4230 characters are used as the vocabulary.

### 4.2.2 Model Settings

A CTC/Attention autoregressive transformer (AT) baseline was first trained with an architecture of a 12-block encoder and a 6-block decoder. Suppose the tuple of a transformer setting is represented by (model dimension, feed-forward layer dimension, number of heads in self-attention), we define three settings: $d_{512}$ for (512, 2048, 8), $d_{768}$ for (768, 3072, 12), and $d_{256}$ for (256, 2048, 4). $d_{512}$ is used for the two English datasets, and $d_{256}$ is used for the Aishell1 dataset. Later on, we follow the same setting as in [FCC23] for CASS-NAT training. For a fair comparison, we also include a CTC baseline as an encoder-only NASR architecture. When training with the speech foundation models, the 12-block encoder was replaced with a HuBERT-base model, either the English[1] version for Librispeech and MyST, or the Chinese version[2] for Aishell1. We also conducted experiments on the TAE extractor in UniEnc-CASSNAT to examine the trade-off between performance and model size.

All models are optimized using a noam scheduler [VSP17] with warmup steps of 15k (10k for Librispeech 100h), a peak learning rate of 5e-5 for the encoder, and 1e-3 (5e-4 for MyST) for uninitialized modules. The models were trained using a batch size of 80s audio samples (40s for MyST because it contains longer utterances). The training either stops when the

[1]`https://dl.fbaipublicfiles.com/hubert/hubert_base_ls960.pt`

[2]`https://huggingface.co/TencentGameMate/chinese-hubert-base`

WER of the valid set doesn't improve for 10 epochs or is terminated at 30 epochs. For MP-CTC training, the task ratio of CTC loss in the second pass is set to one.

All results are decoded without the usage of the external language model. For the AT baseline, the beam search decoding is applied with a beam size of 20 for Librispeech and MyST, and 10 for Aishell1. For CASS-NAT, the number of sampled alignments is 50 and the threshold is 0.9. We explore the effects of the number of sampled alignments in two iterations, and the threshold for each iteration is set to 0.9 as well.

## 4.3 Results and Discussion

### 4.3.1 Main Results

The main WER results of UniEnc-CASSNAT on the Librispeech 100h, MyST, and Aishell1 datasets are shown in Table 4.1. We first train two autoregressive transformer baselines with or without the usage of self-supervised learning. The results in the table again show the effectiveness of the speech foundation models. CASS-NAT achieves close performance to their AT counterpart, which is consistent with previous work. We also present the results of CTC on the three datasets. Due to the output-independent assumption, CTC is worse than the AT baseline and CASS-NAT although it requires fewer parameters. Note that the motivation of UniEnc-CASSNAT is to investigate whether the encoder can jointly model the frame-level and token-level acoustic embedding without the use of the decoder and thus has fewer model parameters. We expect to obtain a model with similar parameters compared to CTC but close performance to CASS-NAT. As shown in Table 4.1, the proposed UniEnc-CASSNAT achieves comparable results than CASS-NAT, for example, a WER of 11.0% for UniEnc-CASSNAT vs. 11.2% for CASS-NAT on the Librispeech test-other set, but is superior to CASS-NAT in terms of model size (99.3M vs. 130.5M). A smaller model size can be helpful for on-device deployment. Compared to CTC, the UniEnc-CASSNAT achieves much better performance than CTC with a similar model size. The additional 3M parameters compared to CTC (95.7M) are from the TAE extractor. The limitation of

Table 4.1: WER performance of UniEnc-CASSNAT and comparisons to previous methods on Librispeech-100h, MyST, and Aishell1 datasets. State-of-the-art (SOTA) results with the usage of speech foundation models that are pre-trained with the same amount of unannotated data to ours are reported. The real-time factor (RTF) of each method on Librispeech test-other data is presented for speed comparison. All bold-faced improvements are statistically significant.

| Model Type | Model Size | Librispeech-100h | | | | | MyST | | Model Size | Aishell1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | dev-clean | dev-other | test-clean | test-other | RTF | dev | test | | dev | test |
| AT-w/o SSL | 85.1M | 6.6 | 18.2 | 6.9 | 18.2 | 0.325 | 13.5 | 14.9 | 33.6M | 4.6 | 5.0 |
| AT-w/ SSL | 121.6M | 4.8 | 11.0 | 4.8 | 10.8 | 0.486 | 11.4 | 13.1 | 107.3M | 4.0 | 4.3 |
| Non-autoregressive ASR | | | | | | | | | | | |
| Previous SOTA | w/ SSL | 4.6 | 11.3 | 4.8 | 11.3 [FWG23] | - | 16.0 | 15.6 [FCC23] | w/ SSL | 3.6 | 3.8 [ZXG21] |
| BERT-CTC [HYA22] | - | 7.0 | 16.3 | 7.2 | 16.6 | - | - | - | 143M | 3.9 | 3.9 |
| CTC | 95.7M | 6.1 | 13.8 | 6.2 | 13.8 | 0.005 | 12.9 | 14.5 | 95.7M | 4.5 | 4.9 |
| CASS-NAT | 130.5M | **4.7** | 11.4 | 4.9 | 11.2 | 0.014 | 11.9 | **13.5** | 109.7M | **4.0** | **4.3** |
| UniEnc-CASSNAT | 99.3M | 4.9 | **11.0** | **4.8** | **11.0** | 0.093 | **11.8** | **13.5** | 102.7M | 4.2 | 4.5 |

UniEnc-CASSNAT could be its slower inference than CTC and CASSNAT because of the multiple forward computations of the encoder with a longer input sequence (concatenation of frames and tokens). The RTF values in Table 4.1 show that the UniEnc-CASSNAT is still 3-5x faster than the AT models although it is 6x slower than CASS-NAT.

The proposed UniEnc-CASSNAT achieves the best-performing NASR results so far in the literature [FWG23, FCC23] on Librispeech 100h and MyST. One can find better WER performance on the Librispeech 100h data, for example, in [ZZA22, WKW23]. However, in that work, the authors either use a larger model trained with Libri60k hours of data or extra text data. We compare the UniEnc-CASSNAT results to a similar work BERT-CTC [HYA22], which also uses an encoder-only structure. Differently, UniEnc-CASSNAT generates ASR outputs with CE loss instead of CTC loss in BERT-CTC and does not require a pre-trained BERT module (smaller in size than BERT-CTC). In addition, UniEnc-CASSNAT achieves the best performance with two iterations only instead of more than 10 iterations in BERT-CTC (faster inference). Based on the results in Table 4.1, UniEnc-CASSNAT is better on Librispeech-100h but worse on Aishell1 than BERT-CTC. The reason could be that Aishell1 contains simple sentences where a pre-trained BERT model is more beneficial [HYA22] and the BERT-CTC has 143M parameters versus 102M in UniEnc-CASSNAT.

### 4.3.2 Ablation Study of UniEnc-CASSNAT

We present more results on the Librispeech 100h data to show the importance of the proposed MP-CTC training and iterative decoding. First, we set $\lambda_2$ in Eq. 4.1 to zero and train a UniEnc-CASSNAT with only first-pass CTC. The results in Table 4.2 show that the single-pass CTC (SP-CTC) training has a performance gap compared to the CASS-NAT. Additionally, SP-CTC training is not able to perform iterative decoding because $\text{TAE}_{n>1}$ is not constrained by CTC outputs. MP-CTC training is also worse than the CASS-NAT without iterative decoding (e.g. (50, NA)). When applying iterative decoding, we explore different combinations of the number of sampled alignments $S_n$ in each iteration. The total

Table 4.2: Ablation study of MP-CTC training, the size of the TAE module, and iterative decoding settings. $d_{256}$, $d_{512}$, and $d_{768}$ are defined in Section 4.2.2 and their model sizes (including encoder) are 96.1M, 99.3M, 104.2M, respectively. $S_n$ is the number of sampled alignments in iteration $n$.

| Model Type | $(S_1, S_2)$ | dev-clean | dev-other | test-clean | test-other |
|---|---|---|---|---|---|
| CASS-NAT | (50, NA) | 4.7 | 11.4 | 4.9 | 11.2 |
| | UniEnc-CASSNAT | | | | |
| SP-CTC | (50, NA) | 4.9 | 11.9 | 5.0 | 11.6 |
| MP-CTC-$d_{512}$ | (50, NA) | 5.0 | 11.7 | 5.2 | 11.8 |
| | (50, 1) | 5.0 | 11.1 | 4.9 | 11.1 |
| | (25, 2) | **4.9** | **11.0** | **4.8** | **11.0** |
| | (10, 5) | 4.9 | 11.1 | 4.9 | 11.1 |
| | (5, 10) | 4.9 | 11.1 | 4.9 | 11.2 |
| | (2, 25) | 5.0 | 11.4 | 5.1 | 11.4 |
| | (1, 50) | 5.2 | 11.5 | 5.3 | 11.6 |
| MP-CTC-$d_{256}$ | (25, 2) | 4.9 | 11.4 | 4.9 | 11.2 |
| MP-CTC-$d_{768}$ | (25, 2) | 4.7 | 11.2 | 4.8 | 11.0 |

number of sampled alignments is set to the same as that used in CASS-NAT for a fair comparison. As shown in Table 4.2, iterative decoding with a setting of $(25, 2)$ achieves the best WER performance and is better than the WER of CASS-NAT. Most of the combinations of $S_n$ achieve comparable WERs to CASS-NAT. It is also noted that the diversity of sampled alignments in the first iteration is more important than that in the second iteration.

Finally, since the TAE extractor introduces extra model parameters besides the foundation model, we conduct experiments of UniEnc-CASSNAT with different transformer settings ($d_{256}$, $d_{512}$, $d_{768}$) described in Section 4.2.2). The results are also shown in Table 4.2. We can see from the table that with a bigger TAEE module, the performance tends to be better. However, we select MP-CTC-$d_{512}$ as the final results to show in Table 4.1 because MP-CTC-$d_{768}$ did not achieve significant improvements with additional 5M parameters.

## 4.4 Chapter Summary

In this chapter, we presented a novel encoder-only non-autoregressive ASR (NASR) model, UniEnc-CASSNAT, which integrates the advantage of CTC and CASS-NAT. The encoder of UniEnc-CASSNAT acts as both the encoder and decoder in CASS-NAT to reduce the model parameters and can be well initialized from the speech foundation models (SSL pretraining). Furthermore, MP-CTC training and iterative decoding are proposed for UniEnc-CASSNAT to further improve the performance to be better or comparable to CASS-NAT. As a result, UniEnc-CASSNAT achieved similar performance to the CASS-NAT using iterative decoding and requires fewer parameters. We examined the effectiveness of the proposed methods on the Librispeech 100h, MyST, and Aishell1 datasets. To the best of our knowledge, we have achieved the best-performing WER results for NASR on the first two datasets with the same settings as those in the literature. Future work includes model compression and distillation to further reduce the parameters. Another interesting direction is to design SSL pretraining methods for the encoder-decoder NAT models.

# CHAPTER 5

# Benchmarking the Performance of Speech Foundation Models for Children's ASR

## 5.1 Motivation

Large foundation models have increasingly gained attention in the research community because of their impressive zero-shot and in-context learning ability [WTB22, LWL23, PT23]. Specifically in the speech area, the Whisper-large model [RKX23] has shown great robustness to diverse domains of speech data by learning from large-scale supervised data in a multi-task training setting. In addition to Whisper, another type of speech foundation models (SFM) is obtained through self-supervised learning, e.g. Wav2vec2.0 [BZM20], HuBERT [HT21], WavLM [Che22], and W2VBERT2.0 [CZ21]. Such models do not require annotations and learn to extract contextual representations based on data patterns in the speech signals [ZPH22]. State-of-the-art results of various speech recognition tasks can be achieved by finetuning these models or using them as feature extractors.

With the increasing use of voice-based educational technology, better child ASR systems are needed because speech is one of the mechanisms young children use to interact with devices due to their limited reading and writing abilities. However, as we mentioned in previous chapters, child ASR is difficult due to, in part, the lack of large child speech databases. To address this issue, researchers have developed a variety of data augmentation methods by perturbation [KPP15, YFA21b, YFA21b, KS20] or voice conversion [LV24, Sha20]. Another direction is to adopt the pretraining finetuning paradigm, which utilizes the unannotated data with self-supervised learning [ML22, FZW22] or the annotated adult data with transfer learning [SG20] in the pretraining stage. The knowledge learned in the pretrained models can greatly improve the performance for child ASR. With the recent advances of SFMs, several studies have compared the performance of widely used SFMs on child speech [JBY23, Att23, LSA22]. However, these studies provide only full finetuning experiments on SFMs. In addition, the speech corpora used in these studies are partitioned differently in each one, making direct performance comparisons difficult.

In this chapter, we initiate and present a comprehensive benchmark on the OGI [SHC00] read and MyST spontaneous [WC11] child speech corpora, studying the performance of various SFMs. More importantly, we investigate finetuning strategies for child speech by comparing various data augmentation and parameter-efficient finetuning (PEFT) methods, which are not discussed in previous works. We observe many interesting behaviors of finetuning different speech foundation models. For example, adapter finetuning is better than full finetuning for large models but vice versa for small models. Our benchmark study may offer guidance in selecting appropriate models, data augmentation and PEFT strategies to develop robust and accurate child ASR systems. The code and evaluation data list are released [1]. We hope the standard evaluation can lead to fairer comparisons for child ASR research.

## 5.2 Covered Topics in the Bechmark

We briefly introduce the methods that are compared in the benchmark, including used SFMs, data augmentation and parameter-efficient finetuning (PEFT) techniques.

### 5.2.1 Speech Foundation Model

Large models trained with large amounts of data have shown great potential to improve the performance of speech recognition tasks. There are two types of SFMs that are: 1) trained with supervised speech-text pairs, such as Whisper [RKX23] and Parakeet [RKK23], and 2) trained with un-annotated speech data using self-supervised learning, e.g. Wav2vec2.0

[1] `https://github.com/Diamondfan/SPAPL_KidsASR`

Table 5.1: Details of the speech foundation models used in the benchmark.

| **Model** | Model Architecture | Input Features | Model Size | Sup./Self-sup. | Training Data (hours) |
|---|---|---|---|---|---|
| Whisper-{tiny-large} | Encoder-Decoder | Fbank | 39M-1550M | Supervised | 680K |
| Whisper-largeV3 [RKX23] | Encoder-Decoder | Fbank | 1550M | Supervised | 1M |
| Canary [RKK23] | Encoder-Decoder | Fbank | 1B | Supervised | 85K |
| Parakeet-TDT [RKK23] | Transducer | Fbank | 1.1B | Supervised | 64K |
| Wav2vec2-Large [BZM20] | Encoder | Waveform | 311M | Self-supervised | 60k |
| HuBERT-Large [HT21] | Encoder | Waveform | 311M | Self-supervised | 60k |
| WavLM-Large [Che22] | Encoder | Waveform | 311M | Self-supervised | 94k |

[BZM20], HuBERT [HT21], and WavLM [Che22]. For supervised SFMs, the zero-shot ability of these models is compared as they can directly perform speech recognition tasks. We then conduct in-depth finetuning experiments on the Whisper series (tiny, base, small, medium, large and largeV3). For self-supervised SFMs, finetuning experiments are conducted. The models used in the benchmark are listed in Table 5.1 along with details including model architecture, input features, model size and training data. The open-sourced models can be accessed in the OpenASR leaderboard[2].

### 5.2.2 Data Augmentation

Data augmentation methods are commonly used in child ASR systems for alleviating the data scarcity problem, but no systematic comparison between them has been conducted before. Based on the Whisper-small model, we compare several widely-used methods including pitch perturbation [PNR11], speed perturbation [KPP15], vocal tract length perturbation (VTLP) [JH13], and SpecAugment [PCZ19]. Two augmented utterances were created for each utterance as has commonly been done in the literature.

- Pitch perturbation involves altering the fundamental frequency of speech signals while preserving other temporal /spectral features. The pitch is shifted to $n/12$ octave higher or lower for each utterance, where $n$ is randomly sampled from 1 to 12 twice.
- Speed perturbation modifies the speed of speech signals. Two copies of each utterance are created with the perturbation rate of 0.9 and 1.1.
- VTLP involves simulating the effects of variations in vocal tract length by applying frequency warping. The perturbation rate used (0.9 and 1.1) is the same as that used in speed perturbation.
- SpecAugment randomly masks consecutive frequency bands and time frames, which effectively increases the robustness of the model to time-frequency variations. We use the default SpecAug settings in the Whisper model.

[2]`https://huggingface.co/spaces/hf-audio/open_asr_leaderboard`

We look forward to incorporating new augmentation methods in the benchmark in the future.

### 5.2.3 Parameter Efficient Finetuning (PEFT)

Parameter-efficient finetuning techniques have become increasingly important when large foundation models are used as model initialization for various tasks [LQP24]. These techniques aim to adapt pretrained models to new tasks or domains while minimizing the computational resources required for training. We compare four widely-used PEFT techniques, which are Low Rank Adaptation (LoRA) [HW21], adapter tuning [HGJ19] (the one we used in Chapter 2.1.3), prompt tuning [LJ22], and prefix tuning [LL21].

- LoRA leverages the observation that the matrices of model layers often exhibit low-rank structures. By decomposing weight matrices into low-rank factors and updating only the low-rank factors during fine-tuning, LoRA reduces computational overhead while preserving model performance. We apply LoRA weights to both the query and value-related parameters in each attention layer, with a rank of eight [HW21] .
- Adapter tuning introduces lightweight adapter modules, which are small neural network components inserted between layers of the foundation models. By finetuning only the parameters within the adapters, efficient adaptation is achieved. We used the residual adapters with the bottleneck dimension of 32, which is similar to [HGJ19]. The residual adapters are inserted after each block in both the encoder and decoder.
- Prompt Tuning prepends randomly initialized prompt vectors to the input sequence and the prompts are optimized through gradient-based methods, allowing the model to directly learn task-specific input representations during fine-tuning. 100 and 20 prompts are inserted in the encoder and decoder inputs, respectively, in our experiments.
- Prefix tuning is similar to prompt tuning but prepends the prompts at each layer instead of at the input, bringing more flexibility during finetuning. In our experiments, 50 and 10 prompts are inserted at the beginning of each layer input to both the encoder

and decoder modules, respectively.

The number of prompts used in prompt tuning and prefix tuning are chosen empirically. By comparing various PEFT methods to full finetuning, we will discover the best finetuning strategy for child ASR when using speech foundation models.

## 5.3 Experiments

In this section, we present the speech datasets used, experimental setup and results.

### 5.3.1 Child Speech Datasets

The experiments are conducted on two child speech databases: My Science Tutor (MyST) spontaneous speech corpus [WC11], and CSLU OGI scripted read speech corpus [SHC00], as described in Chapter 2 and Chapter 3. The preprocessing of the MyST corpus is slightly different for the benchmark described in this chapter.

As mentioned in previous chapters, the MyST corpus consists of around 240 hours of transcribed conversational children's speech (from grade 3 to grade 5), recorded from virtual tutoring sessions in physics, geography, biology, and other topics. Different from previous chapters and similar to [Att23], we identify and filter low quality audio samples by passing the transcribed dataset through the Whisper-largeV2 model. Utterances with WER larger than 50% or with less than 3 words are removed, resulting in a 133 hours training set. When evaluating the Whisper model, we find that the results are unstable for the test samples that are longer than 30s (the maximum length for training Whisper). Hence, utterances longer than 30s are also removed in both the training and test sets. As a result, the original data splits in MyST corpus are as follows: train (133h), dev (21h), and test (25h).

The utterance ID list for the two corpora will be released as standard evaluations with the training code.

Table 5.2: Zero-shot performance of the supervised speech foundation models in terms of WER. Bold numbers are the best performance among the supervised SFMs.

| Model | Model Size | MyST | | OGI | |
|---|---|---|---|---|---|
| | | dev | test | dev | test |
| Whisper-tiny | 39M | 18.5 | 20.6 | 40.1 | 53.8 |
| Whisper-base | 74M | 15.6 | 16.8 | 36.8 | 38.0 |
| Whisper-small | 242M | 14.4 | 13.4 | 21.2 | 25.4 |
| Whisper-medium | 769M | 13.3 | 13.1 | 18.8 | 20.8 |
| Whisper-large | 1550M | 14.4 | 12.5 | 21.2 | 22.9 |
| Whisper-largeV3 | 1550M | 12.3 | 12.6 | 14.9 | 19.9 |
| Canary | 1B | **9.3** | **9.5** | 14.8 | 18.2 |
| Parakeet-rnnt | 1.1B | 10.7 | 11.1 | **14.3** | **16.7** |

### 5.3.2 Finetuning and Evaluation Setup

When finetuning the supervised SFMs, we use the same vocabulary and objective function as those used in the pretraining stage. When finetuning self-supervised SFMs, we use all characters in the transcriptions to create a vocabulary and apply a CTC loss to perform ASR. All results are reported by greedy search decoding without any external language model.

### 5.3.3 Zero-shot Performance for the Supervised SFMs

Since supervised speech foundation models are trained with ASR loss, we first compare their zero-shot abilities on child speech. Top performing models from the OpenASR benchmark for adult speech are selected for comparisons. The results are presented in Table 5.2. The Canary and Parakeet models have been shown to perform better than Whisper, on average, on the adult speech benchmark [SMK23]. The same conclusions can be drawn here for child speech, which is surprising because the Whisper models are trained with more data than Canary and Parakeet (training data sizes are shown in Table 5.1). Considering that many of the data for Whisper training is weakly-supervised, we conclude that data quality is sometimes more important than the size of data for obtaining a robust supervised speech foundation model, which has also been observed for large language models [Zho24].

### 5.3.4 Comparisons of Data Augmentation Methods

Table 5.3: WER comparisons of different data augmentation methods on MyST dataset using the Whisper-small model. PP, VTLP, SP, SA indicates pitch perturbation, vocal tract length perturbation, speed perturbation, and SpecAugment, respectively. PIF is perturbation invariant training. x3 indicates three copies of the original data for augmentation.

| Whisper-small | Augmentation | MyST-dev | MyST-test |
|---|---|---|---|
| Baseline | no | 14.4 | 13.4 |
| Finetuning | no | 8.4 | 9.3 |
| | PP (x3) | 8.6 | 8.8 |
| | VTLP (x3) | 8.6 | 9.0 |
| | SP (x3) | 8.1 | 8.9 |
| | SpecAug (SA) | 8.2 | 9.0 |
| | SA + PP | 8.2 | 8.9 |
| | SA + VTLP | 8.1 | 9.0 |
| | SA + SP | 8.3 | 8.9 |
| PIF | VTLP (x3) | 8.3 | 9.0 |
| | PitchP (x3) | 8.3 | 8.9 |

Data augmentation is an important technique to deal with low-resource tasks, such as child ASR. However, previous works either used private data or conducted experiments with their own settings, making the comparisons of different methods difficult. In addition, previous data augmentation methods are proposed based on training from scratch. It is unknown whether these methods improve the performance when using SFMs. To address this issue, we made a comparison of different data augmentation methods and explored their role in finetuning SFMs. The experimental results on MyST dataset using Whisper-small model are shown in Table 5.3. The reason we use the Whisper-small model is because it is computationally efficient given our limited number of GPUs, and achieves a reasonable WER on child speech. We can observe from the table that different augmentation methods achieve similar WER improvements compared to the finetuning baseline. Interestingly, the combination of two data augmentation methods does not provide further gains compared to using only one method. This is slightly different from the conclusion in [YFA21a] when the model is trained from scratch. This might be because the SFM itself is already robust to some variations created by the data augmentation methods. Note that F0-based data augmentation in [YFA21a] achieves similar performance to pitch perturbation.

The improvements of speed perturbation and SpecAugment are more stable (consistent improvements in WER on development and test sets) than pitch perturbation and VTLP. We therefore propose a perturbation invariant finetuning (PIF) technique to stabilize the VTLP and pitch perturbation. Specifically, an additional distance loss between the encoder outputs of original and perturbed utterance is added as a regularization for finetuning. The results in Table 5.3 show that PIF can lead to more consistent improvements of perturbation methods on the MyST-dev and MyST-test sets. We look forward to incorporating other data augmentation methods into the public codebase/benchmark in the future.

### 5.3.5 Comparisons of Parameter Efficient Finetuning

When speech foundation models are large, full finetuning with the entire model parameters would be difficult because of the high GPU memory costs. Parameter efficient finetuning can retain the performance of full tuning but update less parameters during the finetuning stage. We compare several widely used PEFT methods in the NLP area on Whisper-small model and present the results in Table 5.4. It can be seen from the table that adapter tuning achieves similar performance compared to the full finetuning while having only 1.29M parameters for updates. Note that the initialization of the adapters are important for good performance of adapter tuning. For example, the inserted adapter module should be equiva-

Table 5.4: WER comparisons of different parameter efficient finetuning (PEFT) methods on MyST dataset using the Whisper-small model. Params indicates the number of updated parameters during finetuning. Enc. and Dec. represents finetuning encoder and decoder only, respectively.

| Model | PEFT | MyST-dev | MyST-test | Params |
|---|---|---|---|---|
| Baseline | no | 14.4 | 13.4 | 0 |
| Full-FT | no | 8.4 | 9.3 | 242M |
| Whisper-small | Enc. | 9.0 | 9.2 | 88M |
| | Dec. | 8.9 | 9.5 | 154M |
| | Prompt [LWL23] | 10.4 | 10.4 | 92k |
| | Prefix [LL21] | 8.9 | 10.2 | 541k |
| | LoRA [HW21] | 9.1 | 9.6 | 917k |
| | Adapter [FA22] | 8.4 | 9.3 | 1.29M |

lent to the identity function at the start of the finetuning. However LoRA, the most popular PEFT method in the area of NLP, achieves worse performance than the full finetuning.

### 5.3.6 Supervised vs. Self-supervised Foundation Models with Finetuning

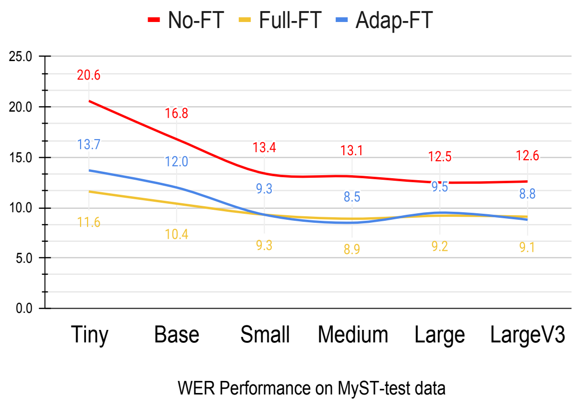
In addition to the supervised foundation models, self-supervised foundation models are also widely used for child ASR. We compare the full-finetuning performance between the two types of foundation models and present the results in Table 5.5. The results show that supervised SFMs can achieve better performance than the self-supervised SFMs after finetuning with similar model parameters (e.g. Whisper-small with 242M and WavLM with 311M parameters). Among the most widely used SSL models, WavLM achieves the best performance because it used more data and included a masked reconstruction from noisy and multi-talker speech data during pretraining. Note that an advantage of the SSL models are that they might also be robust to other speech tasks because the SSL loss is not specifically designed for ASR. We don't compare this ability of SSL models since we are mainly

Table 5.5: WER comparisons of finetuning supervised and self-supervised speech foundation models. Note the performance difference between OGI (read speech) and MyST (spontaneous speech).

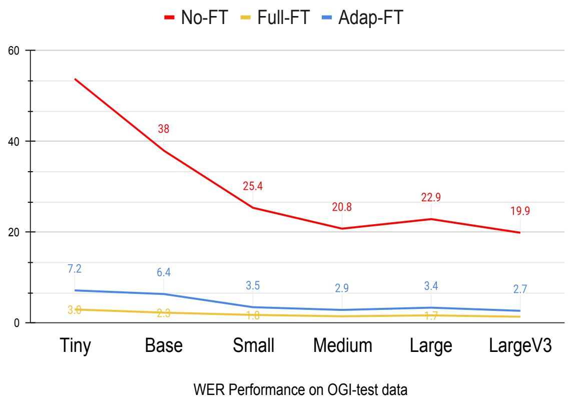
| Model | MyST | | OGI | |
|---|---|---|---|---|
| | Dev | Test | Dev | Test |
| Supervised SFM | | | | |
| Whisper-tiny | 11.6 | 11.6 | 2.7 | 3.0 |
| Whisper-base | 9.1 | 10.4 | 2.0 | 2.3 |
| Whisper-small | 8.4 | 9.3 | 5.0 | 1.8 |
| Whisper-medium | 8.4 | 8.9 | 1.6 | 1.5 |
| Whisper-large | **8.2** | 9.2 | 1.8 | 1.7 |
| Whisper-largeV3 | 8.5 | 9.1 | 1.6 | 1.4 |
| Self-supervised SFM | | | | |
| Wav2vec2.0 | 10.6 | 11.1 | 2.1 | 2.5 |
| HuBERT | 10.5 | 11.3 | 2.2 | 2.5 |
| WavLM | 9.6 | 10.4 | 1.7 | 1.8 |

focusing on the ASR task. The Canary and Parakeet-rnnt models were released after our finetuning experiments on the Whisper models. The finetuning of Canary and Parakeet will be reported in future publications.

### 5.3.7 The Impact of the Model size in PEFT



(a) MyST-test



(b) OGI-test

Figure 5.1: The impact of the Whisper model size for full and adapter finetuning (Adap-FT in the figure). The model size of each whisper model can be found in Table 5.1.

As shown in Table 5.5, the WER of the Whisper model decreases when the model size increases. We further explore whether the model size would affect the performance of PEFT, specifically adapter tuning because it behaves better than other PEFTs as shown in Table 5.4. The results of both full finetuning and adapter finetuning on MyST and OGI test data are plotted in Figure 5.1. We can observe from the figure that the adapter tuning does not work as well as the full finetuning for small models. However, when the model size increases, the gap between adapter tuning and full finetuning decreases. For example, the adapter tuning achieves even better performance than the full finetuning for the Whisper-largeV3 on the MyST-test data. This interesting behavior provides us with guidance on how to select the, appropriate finetuning strategy. That is, performing full finetuning for small models and PEFT for large models. It would also be interesting to investigate the impact of the model size for data augmentation methods, which will be included in future work.

## 5.4 Chapter Summary

In this chapter, we presented the first benchmark for child ASR with a comparison of various speech foundation models, such as Whisper, Canary, Parakeet, Wav2vec2.0, HuBERT, and WavLM. We found that the Canary and Parakeet models are better than Whisper models on child speech with much less training data, indicating the data quality is sometimes more important than the data quantity. As expected, supervised SFMs performed better than the self-supervised SFMs after finetuning. Moreover, we investigated finetuning strategies by comparing various data augmentation (pitch perturbation, speed perturbation, VTLP and SpecAugment) and parameter-efficient finetuning (PEFT) methods (prompt tuning, prefix tuning, adapter tuning, and LoRA). Similar performance was achieved for the data augmentation methods investigated. To stabilize the finetuning using the augmented data, we propose a perturbation invariant finetuning (PIF) loss as a regularization. Various parameter-efficient finetuning (PEFT) strategies were compared, and we observed that the behaviors of PEFT are different when the model size increases. For example, PEFT performed better than full finetuning for large models but worse for small models. This study may offer guidance in selecting appropriate models, data augmentation and PEFT strategies to develop robust child ASR systems.

# CHAPTER 6

# Conclusion

This dissertation explored methods to improve the accuracy and inference efficiency for children's automatic speech recognition (ASR). Additionally, a child ASR benchmark based on speech foundation models was created to facilitate further research on child ASR. In this chapter, we summarize the main results of the dissertation and discuss possible future work.

## 6.1 Summary

Chapter 2 presented the developed self-supervised learning (SSL) based techniques to improve the accuracy for children's ASR by leveraging adult speech data in the pretraining stage. Specifically, a bidirectional APC (Bi-APC) framework was proposed for BLSTM pretraining, which addressed the problem that APC is not suitable for bidirectional model pretraining. We discussed the possibility of using Bi-APC for non-causal transformers. Various solutions were investigated and results showed that Bi-APC framework can have a slight improvement over the transformer baseline without pretraining, but the results are worse than other bidirectional pretraining methods like Wav2vec2.0 and HuBERT. Additionally, a domain responsible adaptation and finetuning (DRAFT) framework was proposed to alleviate the domain shifting problem between adult and child speech. The DRAFT framework performed well on APC, Bi-APC, Wav2vec2.0 and HuBERT methods, showing that it can improve the performance of pretraining methods for both causal (APC) and non-causal transformers (the other three techniques). When compared to the conventional pretraining baselines without adaptation, we achieved relative WER improvements of up to 19.7% on the two child ASR tasks (OGI and MyST).

In Chapter 3, a novel non-autoregressive transformer (CASS-NAT) was proposed. CASS-NAT increases the inference efficiency for an ASR system. During training, Viterbi-alignment was used to estimate the posterior probability, and various training strategies were applied to improve the WER performance of CASS-NAT. During inference, we investigated in depth an error-based sampling alignment (ESA) method that we introduced to reduce the alignment mismatch between training and inference. The extensive experiments on CASS-NAT have shown that: (1) a mixed-attention decoder (MAD) in CASS-NAT was important for reducing the WER; (2) the reason why ESA decoding works well was because it has a lower length prediction error rate; (3) TAEs had similar functionality to word embeddings, such as representing grammatical structures, indicating the possibility of learning semantics without a language model. As a result, CASS-NAT achieved new state-of-the-art results for NAT models on several ASR tasks (Librispeech, Aishell1, TED2, and MyST) with and without SSL pretraining. The performances of CASS-NAT were comparable, in relative terms, to AT with beam search decoding, but maintain a ∼20x speed up.

Chpater 4 integrated the usage of SSL and non-autoregressive generation, and presented a novel encoder-only non-autoregressive ASR (NASR) model, UniEnc-CASSNAT. The UniEnc-CASSNAT acted as both the encoder and decoder in CASS-NAT to reduce the model parameters and can be well initialized from speech foundation models (SSL pretraining). Furthermore, MP-CTC training and iterative decoding were proposed for UniEnc-CASSNAT to further improve the performance so that it is comparable to CASS-NAT. Experimental results showed that UniEnc-CASSNAT achieved similar performance to the CASS-NAT using iterative decoding, and however, it required fewer parameters.

Finally, in Chapter 5, we introduced the first benchmark for child ASR with a comparison of various speech foundation models, such as Whisper, Canary, Parakeet, Wav2vec2.0, HuBERT, and WavLM. The Canary and Parakeet models are better than Whisper models on child speech with much less training data, indicating the data quality is sometimes more important than the data quantity. Supervised SFMs performed better than the self-supervised SFMs after finetuning. Moreover, various data augmentation (pitch perturbation, speed perturbation, VTLP and SpecAugment) and parameter-efficient finetuning (PEFT) methods (prompt tuning, prefix tuning, adapter tuning, and LoRA) were compared. We observed that different data augmentation methods performed similarly and PEFT performed better than full finetuning for large models but worse for small models.

## 6.2 Future work

In the proposed DRAFT framework, we observe that the residual adapters learned from one dataset could benefit another dataset. It could be interesting to investigate the fusion of residual adapters learned from various domains for greater WER improvements on child and/or other low-resource ASR tasks.

In addition to the proposed non-autoregressive generation methods, model compression and distillation are also promising directions for improving the inference efficiency by reducing model parameters.

The presented benchmark in this dissertation may offer guidance in selecting appropriate models, data augmentation and PEFT strategies to develop robust child ASR systems. Future work will include: 1) Evaluations on other child speech datasets; 2) Comparisons with new data augmentation methods; 3) Evaluations of other open-sourced speech foundation models, such as SeamlessM4T [BCM23], OWSM [PT23] and W2VBERT2.0 [CZ21]; 4) Migration of models not supported in Huggingface, e.g. the Canary and Parakeet models developed using the NeMo [KLN19] framework, since our finetuning code is implemented based on Huggingface.

Finally, the language modeling for child ASR is an under-explored topic, which might require more attention from the research community in the future.

## REFERENCES

[AMS05] Batliner A, Blomberg M, D'Arcy S, Elenius D, Giuliani D, Gerosa M, Hacker C, Russel M, Steidl S, and Wong M. "The PF-STAR Children's Speech Corpus." In *Proc. of INTERSPEECH*, p. 2761–2764, 2005.

[Att23] Ahmed Adel Attia et al. "Kid-Whisper: Towards Bridging the Performance Gap in Automatic Speech Recognition for Children VS. Adults." *arXiv preprint arXiv:2309.07927*, 2023.

[AWZ21] Junyi Ao, Rui Wang, Long Zhou, Shujie Liu, Shuo Ren, Yu Wu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, et al. "Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing." *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5723–5738, 2021.

[BAM20] Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. "Effectiveness of self-supervised pre-training for speech recognition." *ICASSP 2020*, pp. 7694–7698, 2020.

[BCM23] Loïc Barrault, Yu-An Chung, Mariano Coria Meglioli, et al. "Seamless: Multilingual Expressive and Streaming Speech Translation." *arXiv preprint arXiv:2312.05187*, 2023.

[BDN17] 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 *20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, pp. 1–5. IEEE, 2017.

[BSA19] Alexei Baevski, Steffen Schneider, and Michael Auli. "vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations." *International Conference on Learning Representation (ICLR)*, 2019.

[BYT21] 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 (TASLP)*, **29**:1897–1911, 2021.

[BZM20] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." *Advances in Neural Information Processing Systems*, **33**, 2020.

[BZV19] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. "Attention augmented convolutional networks." In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pp. 3286–3295, 2019.

[CG20a] Yu-An Chung and James Glass. "Generative pre-training for speech with autoregressive predictive coding." *ICASSP 2020*, pp. 3497–3501, 2020.

[CG20b] Yu-An Chung and James Glass. "Improved Speech Representations with Multi-Target Autoregressive Predictive Coding." *ACL 2020*, pp. 2353–2358, 2020.

[CH19] Yu-An Chung, Wei-Ning Hsu, et al. "An Unsupervised Autoregressive Model for Speech Representation Learning." *Interspeech 2019*, pp. 146–150, 2019.

[Che22] Sanyuan Chen 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, 2022.

[CLL21] Heng-Jui Chang, Hung-yi Lee, and Lin-shan Lee. "Towards Lifelong Learning of End-to-end ASR." *Interspeech 2021*, pp. 2551–2555, 2021.

[CM21] Xuankai Chang, Takashi Maekaku, et al. "An exploration of self-supervised pre-trained representations for end-to-end speech recognition." *ASRU, 2021*, pp. 228–235, 2021.

[CQZ22] Chung-Cheng Chiu, James Qin, Yu Zhang, Jiahui Yu, and Yonghui Wu. "Self-supervised learning with random-projection quantizer for speech recognition." In *International Conference on Machine Learning*, pp. 3915–3924. PMLR, 2022.

[CSH20] William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. "Imputer: Sequence modelling via imputation and dynamic programming." In *International Conference on Machine Learning (ICML)*, pp. 1403–1413. PMLR, 2020.

[CSK20] Ethan A Chi, Julian Salazar, and Katrin Kirchhoff. "Align-Refine: Non-Autoregressive Speech Recognition via Iterative Realignment." *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, pp. 1920–1927, 2020.

[CSW18] 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 *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4774–4778. IEEE, 2018.

[CW21] Sanyuan Chen, Chengyi Wang, et al. "WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing." *arXiv preprint arXiv:2110.13900*, 2021.

[CWV20] Nanxin Chen, Shinji Watanabe, Jesus Antonio Villalba, Piotr Zelasko, and Najim Dehak. "Non-Autoregressive Transformer for Speech Recognition." *IEEE Signal Processing Letters (SPL)*, 2020.

[CZ21] Yu-An Chung, Yu Zhang, et al. "W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training." *IEEE*

*Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 244–250, 2021.

[CZM21] Nanxin Chen, Piotr Zelasko, Laureano Moro-Velazquez, Jesus Villalba, and Najim Dehak. “Align-Denoise: Single-Pass Non-Autoregressive Speech Recognition.” In *Proc. Interspeech 2021*, pp. 3770–3774, 2021.

[CZS20] Xi Chen, Songyang Zhang, Dandan Song, Peng Ouyang, and Shouyi Yin. “Transformer with Bidirectional Decoder for Speech Recognition.” *Interspeech 2020*, pp. 1773–1777, 2020.

[DCL19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding.” *NAACL-HLT, 2019*, pp. 4171–4186, 2019.

[DXX18] Linhao Dong, Shuang Xu, and Bo Xu. “Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition.” In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5884–5888. IEEE, 2018.

[DYW22] 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.” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8522–8526, 2022.

[DYY19] 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 the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2978–2988, 2019.

[EN21] Solene Evain, Ha Nguyen, et al. “LeBenchmark: A Reproducible Framework for Assessing Self-Supervised Representation Learning from Speech.” *Interspeech 2021*, pp. 1439–1443, 2021.

[FA22] Ruchao Fan and Abeer Alwan. “DRAFT: A Novel Framework to Reduce Domain Shifting in Self-supervised Learning and Its Application to Children’s ASR.” In *Interspeech 2022*, pp. 4900–4904, 2022.

[FAA21] Ruchao Fan, Amber Afshan, and Abeer Alwan. “Bi-apc: Bidirectional autoregressive predictive coding for unsupervised pre-training and its application to children’s ASR.” *ICASSP 2021*, pp. 7023–7027, 2021.

[FCC21a] Ruchao Fan, Wei Chu, Peng Chang, and Jing Xiao. “Cass-nat: Ctc alignment-based single step non-autoregressive transformer for speech recognition.” In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5889–5893. IEEE, 2021.

[FCC21b] Ruchao Fan, Wei Chu, Peng Chang, Jing Xiao, and Abeer Alwan. "An Improved Single Step Non-Autoregressive Transformer for Automatic Speech Recognition." In *Proc. Interspeech 2021*, pp. 3715–3719, 2021.

[FCC23] Ruchao Fan, Wei Chu, Peng Chang, and Abeer Alwan. "A CTC Alignment-Based Non-Autoregressive Transformer for End-to-End Automatic Speech Recognition." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **31**:1436–1448, 2023.

[Fre99] Robert M French. "Catastrophic forgetting in connectionist networks." *Trends in cognitive sciences*, **3**(4):128–135, 1999.

[FSA24] Ruchao Fan, Natarajan Balaji Shankar, and Abeer Alwan. "UniEnc-CASSNAT: An Encoder-only Non-autoregressive ASR for Speech SSL Models." *IEEE Signal Processing Letters*, pp. 1–5, 2024.

[FWG23] Ruchao Fan, Yiming Wang, Yashesh Gaur, and Jinyu Li. "CTCBERT: Advancing Hidden-Unit Bert with CTC Objectives." In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[FZC19] Ruchao Fan, Pan Zhou, Wei Chen, Jia Jia, and Gang Liu. "An Online Attention-Based Model for Speech Recognition." *Interspeech 2019*, pp. 4390–4394, 2019.

[FZW22] Ruchao Fan, Yunzheng Zhu, Jinhan Wang, and Abeer Alwan. "Towards better domain adaptation for self-supervised models: A case study of child asr." *IEEE Journal of Selected Topics in Signal Processing*, **16**(6):1242–1252, 2022.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[GBX18] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. "Non-Autoregressive Neural Machine Translation." In *International Conference on Learning Representations (ICLR)*, 2018.

[GFG06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pp. 369–376, 2006.

[GJM13] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. "Hybrid speech recognition with Deep Bidirectional LSTM." *ASRU 2013*, pp. 273–278, 2013.

[GQC20] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. "Conformer: Convolution-augmented Transformer for Speech Recognition." *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

[Gra12] Alex Graves. “Sequence transduction with recurrent neural networks.” *International Conference of Machine Learning (ICML)*, 2012.

[Har78] Fredric J Harris. “On the use of windows for harmonic analysis with the discrete Fourier transform.” *Proceedings of the IEEE*, **66**(1):51–83, 1978.

[HBT21] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. “Hubert: Self-supervised speech representation learning by masked prediction of hidden units.” *TASLP*, **29**:3451–3460, 2021.

[HCF21] 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.” *Automatic Speech Recognition and Understanding (ASRU)*, pp. 47–54, 2021.

[HGJ19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. “Parameter-efficient transfer learning for NLP.” *ICML*, pp. 2790–2799, 2019.

[HHS21] Zhouyuan Huo, Dongseong Hwang, Khe Chai Sim, Shefali Garg, Ananya Misra, Nikhil Siddhartha, Trevor Strohman, and Françoise Beaufays. “Incremental Layer-wise Self-Supervised Learning for Efficient Speech Domain Adaptation On Device.” *arXiv preprint arXiv:2110.00155*, 2021.

[HMH22] Dongseong Hwang, Ananya Misra, Zhouyuan Huo, Nikhil Siddhartha, Shefali Garg, David Qiu, Khe Chai Sim, Trevor Strohman, Françoise Beaufays, and Yanzhang He. “Large-scale asr domain adaptation using self-and semi-supervised learning.” *ICASSP 2022*, pp. 6627–6631, 2022.

[HS21] Wei-Ning Hsu, Anuroop Sriram, et al. “Robust wav2vec 2.0: Analyzing Domain Shift in Self-Supervised Pre-Training.” *Interspeech 2021*, pp. 721–725, 2021.

[HT21] Wei-Ning Hsu, Yao-Hung Hubert Tsai, et al. “HuBERT: How much can a bad teacher benefit ASR pre-training?” *ICASSP 2021*, pp. 6533–6537, 2021.

[HW21] Edward J Hu, Phillip Wallis, et al. “LoRA: Low-Rank Adaptation of Large Language Models.” *International Conference on Learning Representations*, 2021.

[HWC20] Yosuke Higuchi, Shinji Watanabe, Nanxin Chen, Tetsuji Ogawa, and Tetsunori Kobayashi. “Mask CTC: Non-Autoregressive End-to-End ASR with CTC and Mask Predict.” *Proc. Interspeech 2020*, pp. 3655–3659, 2020.

[HYA22] 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*, pp.

5486–5503, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[HZZ20] 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." *Proc. Interspeech 2020*, pp. 3610–3614, 2020.

[JBY23] Rishabh Jain, Andrei Barcovschi, Mariam Yiwere, Peter Corcoran, and Horia Cucu. "Adaptation of Whisper models to child speech recognition." In *Proc. INTERSPEECH 2023*, pp. 5242–5246, 2023.

[JH13] Navdeep Jaitly and Geoffrey E Hinton. "Vocal tract length perturbation (VTLP) improves speech recognition." *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, **117**:21, 2013.

[JL21] Dongwei Jiang, Wubo Li, et al. "A Further Study of Unsupervised Pretraining for Transformer Based Speech Recognition." *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6538–6542, 2021.

[JLL19] Dongwei Jiang, Xiaoning Lei, Wubo Li, Ne Luo, Yuxuan Hu, Wei Zou, and Xiangang Li. "Improving transformer-based speech recognition using unsupervised pre-training." *arXiv preprint arXiv:1910.09932*, 2019.

[JR85] Biing-Hwang Juang and L. Rabiner. "Mixture autoregressive hidden Markov models for speech signals." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **33**(6):1404–1413, 1985.

[KLG22] Sameer Khurana, Antoine Laurent, and James Glass. "Magic dust for cross-lingual adaptation of monolingual wav2vec-2.0." *ICASSP 2022*, pp. 6647–6651, 2022.

[KLN19] Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, et al. "Nemo: a toolkit for building ai applications using neural modules." *arXiv preprint arXiv:1909.09577*, 2019.

[KPP15] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. "Audio augmentation for speech recognition." *Interspeech 2015*, pp. 3586–3589, 2015.

[KR18] Taku Kudo and John Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing." *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 66, 2018.

[KR20] Jacob Kahn, Morgane Riviere, et al. "Libri-light: A benchmark for asr with limited or no supervision." *ICASSP 2020*, pp. 7669–7673, 2020.

[KS20] Hemant Kumar Kathania, Mittul Singh, et al. "Data Augmentation Using Prosody and False Starts to Recognize Non-Native Children's Speech." In *Interspeech 2020*, pp. 260–264. ISCA, 2020.

[KSS21] Seungwon Kim, Alex Shum, Nathan Susanj, and Jonathan Hilgart. "Revisiting Pretraining with Adapters." *the 6th Workshop on Representation Learning for NLP*, pp. 90–99, 2021.

[KTK21] Samuel Kessler, Bethan Thomas, and Salah Karout. "Continual-wav2vec2: an application of continual learning for self-supervised automatic speech recognition." *arXiv preprint arXiv:2107.13530*, 2021.

[LB95] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series." *The handbook of brain theory and neural networks*, **3361**(10):1995, 1995.

[Li21] Jinyu Li et al. "Recent advances in end-to-end automatic speech recognition." *APSIPA Transactions on Signal and Information Processing*, **11**(1), 2021.

[LJ22] Xiao Liu, Kaixuan Ji, et al. "P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks." *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 61–68, 2022.

[LKW21] Jaesong Lee, Jingu Kang, and Shinji Watanabe. "Layer Pruning on Demand with Intermediate CTC." In *Proc. Interspeech 2021*, pp. 3745–3749, 2021.

[LL21] Xiang Lisa Li and Percy Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation." *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 4582–4597, 2021.

[LLL21] Andy T Liu, Shang-Wen Li, and Hung-yi Lee. "Tera: Self-supervised learning of transformer encoder representation for speech." *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, **29**:2351–2366, 2021.

[LLS20] Shaoshi Ling, Yuzong Liu, Julian Salazar, and Katrin Kirchhoff. "Deep contextualized acoustic representations for semi-supervised speech recognition." *ICASSP 2020*, pp. 6429–6433, 2020.

[LMC20] Jason Lee, Elman Mansimov, and Kyunghyun Cho. "Deterministic non-autoregressive neural sequence modeling by iterative refinement." In *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1173–1182, 2020.

[LMC21] Jialu Li, Vimal Manohar, Pooja Chitkara, Andros Tjandra, Michael Picheny, Frank Zhang, Xiaohui Zhang, and Yatharth Saraf. "Accent-Robust Automatic Speech Recognition Using Supervised and Unsupervised Wav2vec Embeddings." *arXiv preprint arXiv:2110.03520*, 2021.

[LPN99] Sungbok Lee, Alexandros Potamianos, and Shrikanth Narayanan. "Acoustics of children's speech: Developmental changes of temporal and spectral parameters." *Journal of the Acoustical Society of America (JASA)*, **105**(3):1455–1468, 1999.

[LQP24] Wei Liu, Ying Qin, Zhiyuan Peng, and Tan Lee. "Sparsely Shared LoRA on Whisper for Child Speech Recognition." *IEEE ICASSP 2024*, 2024.

[LSA22] Renee Lu, Mostafa Shahin, and Beena Ahmed. "Improving Children's Speech Recognition by Fine-tuning Self-supervised Adult Speech Representations." *arXiv preprint arXiv:2211.07769*, 2022.

[LV24] Matthew Le, Apoorv Vyas, et al. "Voicebox: Text-guided multilingual universal speech generation at scale." *Advances in neural information processing systems*, **36**, 2024.

[LW21] Jaesong Lee and Shinji Watanabe. "Intermediate Loss Regularization for CTC-Based Speech Recognition." In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6224–6228. IEEE, 2021.

[LWG20] Jinyu Li, Yu Wu, Yashesh Gaur, Chengyi Wang, Rui Zhao, and Shujie Liu. "On the Comparison of Popular End-to-End Models for Large Scale Speech Recognition." *Proc. Interspeech 2020*, pp. 1–5, 2020.

[LWL23] Yuang Li, Yu Wu, Jinyu Li, and Shujie Liu. "Prompting large language models for zero-shot domain adaptation in speech recognition." In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8. IEEE, 2023.

[LY20] Andy T Liu, Shu-wen Yang, et al. "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders." *ICASSP 2020*, pp. 6419–6423, 2020.

[MCC13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." *International Conference on Learning Representations (ICLR)*, 2013.

[MCL22] Yen Meng, Yi-Hui Chou, Andy T Liu, and Hung-yi Lee. "Don't speak too fast: The impact of data bias on self-supervised speech models." *ICASSP 2022*, pp. 3258–3262, 2022.

[MH21] Ananya Misra, Dongseong Hwang, et al. "A comparison of supervised and unsupervised pre-training of end-to-end models." *Interspeech 2021*, pp. 731–735, 2021.

[MHL20] Niko Moritz, Takaaki Hori, and Jonathan Le. "Streaming automatic speech recognition with the transformer model." In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6074–6078. IEEE, 2020.

[MJD21] Eskenazi M, Mostow J, and Graff D. "The CMU Kids Speech Corpus." In *LDC97S63.*, 2021.

[ML22] Abdelrahman Mohamed, Hung-yi Lee, et al. "Self-supervised speech representation learning: A review." *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[NCZ21] Edwin G Ng, Chung-Cheng Chiu, Yu Zhang, and William Chan. "Pushing the Limits of Non-Autoregressive Speech Recognition." *in Proc. Interspeech*, pp. 3725–3729, 2021.

[NK21] Jumon Nozaki and Tatsuya Komatsu. "Relaxing the Conditional Independence Assumption of CTC-Based ASR by Conditioning on Intermediate Predictions." In *Proc. Interspeech 2021*, pp. 3735–3739, 2021.

[OEB19] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. "fairseq: A Fast, Extensible Toolkit for Sequence Modeling." *NAACL-HLT (Demonstrations)*, pp. 48–53, 2019.

[OLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." *Conference on Neural Information Processing Systems (NIPS)*, 2018.

[PCP15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. "Librispeech: an asr corpus based on public domain audio books." *ICASSP 2015*, pp. 5206–5210, 2015.

[PCZ19] 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." *Interspeech 2019*, pp. 2613–2617, 2019.

[PGM19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library." *NIPS, 2019*, **32**:8026–8037, 2019.

[PNR11] Rupal Patel, Caroline Niziolek, Kevin Reilly, and Frank H Guenther. "Prosodic adaptations to pitch perturbation in running speech." *Journal of speech, language, and hearing research : JSLHR*, 2011.

[PT23] Yifan Peng, Jinchuan Tian, et al. "Reproducing whisper-style training using an open-source toolkit and publicly available data." In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8. IEEE, 2023.

[QGJ21] Weizhen Qi, Yeyun Gong, Jian Jiao, Yu Yan, Weizhu Chen, Dayiheng Liu, Kewen Tang, Houqiang Li, Jiusheng Chen, Ruofei Zhang, et al. "Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining." In *International Conference on Machine Learning (ICML)*, pp. 8630–8639. PMLR, 2021.

[RDE14] Anthony Rousseau, Paul Deléglise, Yannick Esteve, et al. "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks." In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, pp. 3935–3939, 2014.

[RF20] Vijay Ravi, Ruchao Fan, et al. "Exploring the Use of an Unsupervised Autoregressive Model as a Shared Encoder for Text-Dependent Speaker Verification." *Interspeech 2020*, pp. 766–770, 2020.

[RJM20] Morgane Riviere, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. "Unsupervised pretraining transfers well across languages." *ICASSP 2020*, pp. 7414–7418, 2020.

[RKK23] Dima Rekesh, Nithin Rao Koluguri, Samuel Kriman, Somshubra Majumdar, Vahid Noroozi, He Huang, Oleksii Hrinchuk, Krishna Puvvada, Ankur Kumar, Jagadeesh Balam, et al. "Fast conformer with linearly scalable attention for efficient speech recognition." *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8, 2023.

[RKX23] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. "Robust Speech Recognition via Large-Scale Weak Supervision." In *International Conference on Machine Learning, ICML 2023*, volume 202, pp. 28492–28518. PMLR, 2023.

[SBC19] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. "wav2vec: Unsupervised Pre-Training for Speech Recognition." *Interpseech 2019*, pp. 154–162, 2019.

[SCS20] Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. "Non-Autoregressive Machine Translation with Latent Alignments." In *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1098–1108, 2020.

[SG20] Prashanth Gurunath Shivakumar and Panayiotis Georgiou. "Transfer learning from adult to children for speech recognition: Evaluation, analysis and recommendations." *Computer Speech & Language*, **63**:101077, 2020.

[Sha20] S. Shahnawazuddin et al. "Voice Conversion Based Data Augmentation to Improve Children's Speech Recognition in Limited Data Scenario." In *Proc. Interspeech 2020*, pp. 4382–4386, 2020.

[SHC00] Khaldoun Shobaki, John-Paul Hosom, and Ronald A Cole. "The OGI kids' speech corpus and recognizers." *ICSLP 2000*, pp. 258–261, 2000.

[SLJ15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1–9, 2015.

[SMK23] Vaibhav Srivastav, Somshubra Majumdar, Nithin Koluguri, Adel Moumen, Sanchit Gandhi, Hugging Face Team, Nvidia NeMo Team, and SpeechBrain Team. "Open Automatic Speech Recognition Leaderboard." `https://huggingface.co/spaces/hf-audio/open_asr_leaderboard`, 2023.

[SNH16] Saeid Safavi, Maryam Najafian, Abualsoud Hanani, Martin J Russell, Peter Jancovic, and Michael J Carey. "Speaker recognition for children's speech." *arXiv preprint arXiv:1609.07498*, 2016.

[TGD22] Yun Tang, Hongyu Gong, Ning Dong, Changhan Wang, Wei-Ning Hsu, Jiatao Gu, Alexei Baevski, Xian Li, Abdelrahman Mohamed, Michael Auli, and Juan Pino. "Unified Speech-Text Pre-training for Speech Translation and Recognition." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, May 2022.

[TKK22] Bethan Thomas, Samuel Kessler, and Salah Karout. "Efficient Adapter Transfer of Self-Supervised Speech Models for Automatic Speech Recognition." *ICASSP 2022*, pp. 7102–7106, 2022.

[TLZ20] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6899–6903. IEEE, 2020.

[TTY20] Trang Tran, Morgan Tinkler, Gary Yeung, Abeer Alwan, and Mari Ostendorf. "Analysis of Disfluency in Children's Speech." *Interspeech 2020*, pp. 4278–4282, 2020.

[TYT20] Zhengkun Tian, Jiangyan Yi, Jianhua Tao, Ye Bai, Shuai Zhang, and Zhengqi Wen. "Spike-Triggered Non-Autoregressive Transformer for End-to-End Speech Recognition." *Proc. Interspeech 2020*, pp. 5026–5030, 2020.

[TZ21] Katrin Tomanek, Vicky Zayats, et al. "Residual Adapters for Parameter-Efficient ASR Adaptation to Atypical and Accented Speech." *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6751–6760, 2021.

[VMB21] Apoorv Vyas, Srikanth Madikeri, and Hervé Bourlard. "Comparing CTC and LFMMI for out-of-domain adaptation of wav2vec 2.0 acoustic model." *Interspeech 2021*, pp. 2861–2865, 2021.

[VSP17] 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 the 31st International Conference on Neural Information Processing Systems (NIPS)*, pp. 6000–6010, 2017.

[WC11] Wayne Ward, Ronald Cole, et al. "My science tutor: A conversational multimedia virtual tutor for elementary school science." *TASLP*, **7**(4):1–29, 2011.

[WCP19] Wayne Ward, Ron Cole, and Sameer Pradhan. “My science tutor and the myst corpus.” *NA*, 2019.

[WGK19] Fei Wu, Povey D García-PLP, and S Khudanpur. “Advances in automatic speech recognition for child speech using factored time delay neural network.” In *Interspeech*, pp. 1–5, 2019.

[WHK17] 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 (JSTSP)*, **11**(8):1240–1253, 2017.

[WKB23] Gary Wang, Kyle Kastner, Ankur Bapna, Zhehuai Chen, Andrew Rosenberg, Bhuvana Ramabhadran, and Yu Zhang. “Understanding Shared Speech-Text Representations.” In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

[WKW23] Felix Wu, Kwangyoun Kim, Shinji Watanabe, Kyu J Han, Ryan McDonald, Kilian Q Weinberger, and Yoav Artzi. “Wav2seq: Pre-training speech-to-text encoder-decoder models using pseudo languages.” In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

[WLW22] Yiming Wang, Jinyu Li, Heming Wang, Yao Qian, Chengyi Wang, and Yu Wu. “Wav2vec-switch: Contrastive learning from original-noisy speech pairs for robust speech recognition.” *ICASSP 2022*, pp. 7097–7101, 2022.

[WML20] Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, et al. “Transformer-based acoustic modeling for hybrid speech recognition.” In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6874–6878. IEEE, 2020.

[WTB22] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. “Emergent Abilities of Large Language Models.” *Transactions on Machine Learning Research*, 2022.

[WTL20] Weiran Wang, Qingming Tang, and Karen Livescu. “Unsupervised pre-training of bidirectional speech encoders via masked reconstruction.” *ICASSP 2020*, pp. 6889–6893, 2020.

[WW21] Chengyi Wang, Yu Wu, et al. “UniSpeech: Unified Speech Representation Learning with Labeled and Unlabeled Data.” *International Conference on Machine Learning (ICML)*, **139**:10937–10947, 18–24 Jul 2021.

[WZF21] Jinhan Wang, Yunzheng Zhu, Ruchao Fan, Wei Chu, and Abeer Alwan. "Low Resource German ASR with Untranscribed Data Spoken by Non-native Children–INTERSPEECH 2021 Shared Task SPAPL System." *Interspeech 2021*, pp. 1279–1283, 2021.

[YA18] Gary Yeung and Abeer Alwan. "On the Difficulties of Automatic Speech Recognition for Kindergarten-Aged Children." In *Interspeech*, pp. 1661–1665, 2018.

[YC21] Shu-wen Yang, Po-Han Chi, et al. "SUPERB: Speech processing Universal PERformance Benchmark." *Interspeech 2021*, pp. 1194–1198, 2021.

[YDL18] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension." In *International Conference on Learning Representations (ICLR)*, 2018.

[YFA21a] Gary Yeung, Ruchao Fan, and Abeer Alwan. "Fundamental frequency feature normalization and data augmentation for child speech recognition." In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6993–6997. IEEE, 2021.

[YFA21b] Gary Yeung, Ruchao Fan, and Abeer Alwan. "Fundamental frequency feature warping for frequency normalization and data augmentation in child automatic speech recognition." *Speech Communication*, **135**:1–10, 2021.

[YLG21] Fan Yu, Haoneng Luo, Pengcheng Guo, Yuhao Liang, Zhuoyuan Yao, Lei Xie, Yingying Gao, Leijing Hou, and Shilei Zhang. "Boundary and Context Aware Training for CIF-Based Non-Autoregressive End-to-End ASR." In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 328–334. IEEE, 2021.

[YNF99] J Scott Yaruss, Robyn M Newman, and Tracy Flora. "Language and disfluency in nonstuttering children's conversational speech." *Journal of Fluency Disorders*, **24**(3):185–207, 1999.

[YWC20] Cheng Yi, Jianzhong Wang, Ning Cheng, Shiyu Zhou, and Bo Xu. "Applying Wav2vec2. 0 to speech recognition in various low-resource languages." *arXiv preprint arXiv:2012.12121*, 2020.

[YWW19] Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. "Convolutional Self-Attention Networks." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4040–4045, June 2019.

[ZDV17] Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, and Hermann Ney. "A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition." *ICASSP 2017*, pp. 2462–2466, 2017.

[ZFC19] Pan Zhou, Ruchao Fan, Wei Chen, and Jia Jia. "Improving generalization of transformer for speech recognition with parallel schedule sampling and relative positional embedding." *arXiv preprint arXiv:1911.00203*, 2019.

[ZGN20] Chunting Zhou, Jiatao Gu, and Graham Neubig. "Understanding Knowledge Distillation in Non-autoregressive Machine Translation." In *International Conference on Learning Representations (ICLR)*, 2020.

[Zho24] Chunting Zhou et al. "Lima: Less is more for alignment." *Advances in Neural Information Processing Systems*, **36**, 2024.

[ZLG22] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6182–6186. IEEE, 2022.

[ZMS20] Albert Zeyer, André Merboldt, Ralf Schlüter, and Hermann Ney. "A New Training Pipeline for an Improved Neural Transducer." *Proc. Interspeech 2020*, pp. 2812–2816, 2020.

[ZPH22] 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*, 2022.

[ZXG21] 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 the Association for Computational Linguistics: EMNLP 2021*, pp. 2765–2777, 2021.

[ZZA22] Ziqiang Zhang, Long Zhou, Junyi Ao, Shujie Liu, Lirong Dai, Jinyu Li, and Furu Wei. "SpeechUT: Bridging Speech and Text with Hidden-Unit for Encoder-Decoder Based Speech-Text Pre-training." In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1663–1676, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.