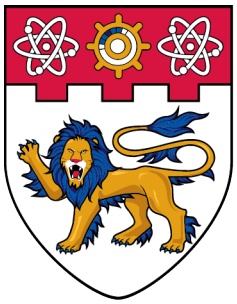


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

# IMPROVING END-TO-END TRANSFORMER MODEL ARCHITECTURE IN ASR

ZHAO YINGZHU
INTERDISCIPLINARY GRADUATE PROGRAMME
ALIBABA-NTU JOINT RESEARCH INSTITUTE

2023

# IMPROVING END-TO-END TRANSFORMER MODEL ARCHITECTURE IN ASR

**ZHAO YINGZHU**

INTERDISCIPLINARY GRADUATE PROGRAMME

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2023

## Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

10 Jan 2023

. . . . . .
Date

. . . . . .
Zhao Yingzhu

## Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

10 Jan 2023

. . . . . .

Date

. . . . . .

Assoc Prof Chng Eng Siong

## Authorship Attribution Statement

This thesis contains material from 4 paper(s) published in the following papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, “Speech transformer with speaker aware persistent memory,” in INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association, 2020, pp. 1261–1265. and Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, “A unified speaker adaptation approach for asr,” in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 2021, pp. 9339–9349.

Chapter 4 is published as Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, “Cross attention with monotonic alignment for speech transformer,” in INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association, 2020, pp. 5031–5035.

Chapter 5 is published as Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, “Universal speech transformer,” in INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association, 2020, pp. 5021–5025.

The contributions of the co-authors are as follows:

- I conducted literature review and brainstormed the new idea.
- I prepared the manuscript drafts. The manuscript was revised by Prof Joty, Prof Chng and Dr Ma.
- I co-designed the experiments with Chongjia and Cheung Chi. I conducted all the experiments in Alibaba Cloud. I also analyzed the data.

10 Jan 2023

. . . . . .
Date

. . . . . .
Zhao Yingzhu

# Acknowledgments

It has been my utmost privilege to be given this opportunity to pursue a Ph.D degree. First, I would like to express my sincere gratitude to my supervisor, Prof. Chng Eng Siong for his invaluable mentorship throughout my Ph.D candidature. The countless discussions that I have had with him have played a considerable role in orientating my research direction in the vast area of speech recognition technologies.

Next, I would like to thank my co-supervisor, Dr Ma Bin, for his support of my Ph.D project. With his long experience and deep expertise in the industry, Dr Ma's comments and suggestions have been crucial in improving the practical relevance of my work.

I would also like to thank Prof. Joty Shafiq Rayhan for his significant role in imparting natural language processing related techniques to me, which is very inspiring in my research work in speech technologies.

As a fair and honest critic of my work, Chongjia has enabled me to have a better understanding of my research area through his insights and encouragement. His expectation of high standard and rigour has also been instrumental in raising the quality of my work. I am also particularly grateful to Gary for the advice and encouragement dispensed in the course of the review process for all my papers, and the help in critiquing my work.

I want to extend my thanks to Alibaba Group and the Singapore Economic Development Board (EDB) for the support and funding of this Ph.D project, and also to Nanyang Technological University for the conducive environment in the Interdisciplinary Graduate School.

Last, but not least, this work could not have been achieved without the love and support of my family, my husband who helped me through in many ways, and our daughter who brings endless joy to our lives.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| AED | Attention Encoder-Decoder |
| ANN | Artificial Neural Networks |
| ASR | Automatic Speech Recognition |
| BLSTM | Bidirectional Long Short-Term Memory |
| CAT | Cluster Adaptive Training |
| CER | Character Error Rate |
| CNN | Convolutional Neural Network |
| CTC | Connectionist Temporal Classification |
| DNN | Deep Neural Network |
| E2E | End-to-end |
| FHL | Factorized Hidden Layer |
| fMLLR | feature-space Maximum Likelihood Linear Regression |
| HMM | Hidden Markov Model |
| KLD | Kullback-Leibler Divergence |
| LM | Language Modeling |
| LSTM | Long Short-Term Memory |
| MAP | Maximum A Posteriori |
| MT | Machine Translation |
| NCCFs | Normalized Cross-Correlation Functions |
| NLP | Natural Language Processing |
| NMT | Neural Machine Translation |
| ReLU | Rectified Linear Unit |
| RNA | Recurrent Neural Aligner |
| RNN | Recurrent Neural Network |
| RNN-T | RNN Transducer |
| SE | Speech Enhancement |
| ST | Speech Translation |
| SWBD | Switchboard |
| VC | Voice Conversion |
| WER | Word Error Rate |
| WSJ | Wall Street Journal |

# List of Publications

## International Conferences

1. Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq Joty, Eng Siong Chng, Bin Ma, "Speech Transformer with Speaker Aware Persistent Memory", in INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association, 2020, pp. 1261-1265.

2. Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq Joty, Eng Siong Chng, Bin Ma, "Universal Speech Transformer", in INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association, 2020, pp. 5021-5025.

3. Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq Joty, Eng Siong Chng, Bin Ma, "Cross Attention with Monotonic Alignment for Speech Transformer", in INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association, 2020, pp. 5031-5035.

4. Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq Joty, Eng Siong Chng, Bin Ma, "Preventing Early Endpointing for Online Automatic Speech Recognition", in 2021 International Conference on Acoustics, Speech and Signal Processing, 2021, pp. 6813-6817.

5. Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq Joty, Eng Siong Chng, Bin Ma, "A Unified Speaker Adaptation Approach for ASR", in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 9339-9349.

# Abstract

As a result of advancement in deep learning and neural network technology, end-to-end models have been introduced into automatic speech recognition (ASR) successfully and achieved superior performance compared to conventional hybrid systems. End-to-end models simplify the traditional GMM-HMM models by transcribing speech to text directly with fast computation speed and fast development time. Transformer model, the latest end-to-end model, has achieved a huge success not only in ASR but also in natural language processing and computer vision. In spite of its great performance, transformer model architecture can be further improved to better suit the characteristics of ASR.

To be more specific, ASR performance is greatly affected by the speaker mismatch between training and test data, and speaker adaptation is a long-standing problem in ASR. Additionally, ASR has the characteristic to have monotonic alignment between text output and speech input. Different phonemes along an utterance may require different level of computation due to varying complexity and noise level. How to better deploy the transformer model for ASR remains a challenge. In this thesis, three novel architectural changes to the transformer model are proposed for the above ASR characteristics. For each of the proposed methods, detailed experimental evaluation is carried out to compare the performance against the transformer baseline with analysis.

In the first study, to alleviate the performance drop in ASR due to speaker mismatch between training and test data, we present a unified framework for speaker adaptation, which consists of feature adaptation and model adaptation. A speaker-aware persistent memory model is presented to capture speaker knowledge through the persistent memory in order to generalize better to unseen test speakers, which belongs to the feature adaptation method. Furthermore, a model-based gradual pruning approach is deployed to free up partial model encoder parameters for target speaker adaptation without sacrificing the original model performance. Finally, instead of only adapting to general

speakers or specific target speaker, we design a multi-speaker adaptation model capable of adapting to multiple speakers simultaneously, which is practical in ubiquitous environment with multiple users. Our proposed approach brings relative 2.74-6.52% word error rate (WER) reduction on general speaker adaptation. On target speaker adaptation, our method outperforms the baseline with up to 20.58% relative WER reduction, and surpasses the finetuning method by up to relative 2.54%.

In the second study, we then address the text output speech input misalignment problem in transformer model which greatly affects the ASR accuracy during training and inference. An effective cross attention biasing technique in transformer is proposed that takes monotonic alignment between text output and speech input into consideration by making use of cross attention weights. Specifically, a Gaussian mask is applied on cross attention weights to limit the input speech context range locally given alignment information. A regularizer is further introduced for alignment regularization. Experiments on LibriSpeech dataset find that our proposed model can obtain improved output-input alignment for ASR, and yields 14.5%-25.0% relative WER reductions.

In the final study, to resolve the problem of static number of layers in transformer model, we present universal speech transformer. It generalizes the speech transformer with dynamic numbers of encoder/decoder layers, which can relieve the burden of tuning depth related hyperparameters. Universal transformer adds the depth and positional embeddings repeatedly for each layer, which dilutes the acoustic information carried by hidden representation, and it also performs a partial update of hidden vectors between layers, which is less efficient especially on the very deep models. For better use of universal transformer, we modify its processing framework by removing the depth embedding and only adding the positional embedding once at transformer encoder frontend. Furthermore, to update the hidden vectors efficiently, especially on the very deep models, we adopt a full update. Experiments on LibriSpeech, Switchboard and AISHELL-1 datasets show that our model outperforms a baseline by 3.88%-13.7%, and surpasses other model with less computation cost.

# Chapter 1

# Introduction

## 1.1 Motivation

Automatic speech recognition (ASR) refers to mapping speech signals to corresponding transcriptions automatically by machines. It has been studied for decades and the research has made great progress from recognizing only ten digits initially [16] to wide usage in many products such as Alexa and Google Assistant. The conventional methods use pipeline structure that includes acoustic model, pronunciation dictionary and language model for ASR. More recently, end-to-end models come into stage and are potential successors of conventional models. In particular, transformer model [17] performs well and appears in most latest models in natural language processing (NLP) and speech recognition.

Transformer model has the architecture similar to attention encoder-decoder (AED) model [5–10], where the encoder acts like an acoustic model to encode speech features and the decoder is responsible for generating label predictions sequentially. Transformer model is well-known for its fast speed with parallel computation capability. With the self-attention network, it can learn long-range dependencies in speech input as well. Transformer model has boosted the ASR performance greatly [13].

In spite of the good performance of the transformer model, its model architecture can be further improved to better suit the characteristics of ASR. Three concerns are highlighted below:

1. Speaker adaptation is a long standing issue in ASR, which arises due to speaker mismatch between training and test data [18]. ASR performance could drop drasti-

cally should the test speaker data deviates a lot from training speaker data. This is because the model trained on training speaker data does not adapt well on unseen speakers.

2. Different from other NLP applications such as machine translation, ASR has the characteristic to have monotonic alignment between text output and speech input. Without taking this into consideration, cross attention will disperse the attention to the entire speech utterance. This means the cross attention may not be able to focus on the corresponding speech input aligned with the current text output, which is both inefficient and imprecise.

3. Transformer model has static number of encoder layers, which needs careful parameter tuning to achieve the best performance. Compared with other models such as recurent neural network, transformer model loses the recurrent inductive bias. Moreover, the fixed number of encoder layers is unable to recognize phonemes with different complexity and noise level dynamically.

All these characteristics of ASR prompt us to improve the transformer model architecture for ASR. Keeping above in mind, we propose three innovative changes to the tranformer model.

## 1.2 Contributions

This thesis highlights three areas of contributions. The first proposed model focuses on speaker adaptation problem in ASR. The second proposed model aims at resolving text output speech input misalignment issue in speech transformer model. The third method improves on the static number of encoder layers in transformer. All of them target at improving the transformer model architecture for ASR.

### 1.2.1 Speaker Adaptation [1, 2]

We first focus on speaker mismatch between training and test data problem in ASR, which could deteriorate ASR accuracy significantly [18]. A speaker-aware persistent memory

model [1] is presented to capture speaker knowledge through the persistent memory in order to generalize better to unseen test speakers, which belongs to the feature adaptation method. Furthermore, a model-based gradual pruning approach is deployed to free up partial model encoder parameters for target speaker adaptation without sacrificing the original model performance [2]. Finally, instead of only adapting to general speakers or specific target speaker, we design a multi-speaker adaptation model capable of adapting to multiple speakers simultaneously, which is practical in ubiquitous environment with multiple users. Our proposed approach brings relative 2.74-6.52% word error rate (WER) reduction on general speaker adaptation. On target speaker adaptation, our method outperforms the baseline with up to 20.58% relative WER reduction, and surpasses the finetuning method by up to relative 2.54%. The thorough description and evaluation of the proposed approach will be provided in Chapter 3.

### 1.2.2 Text Speech Alignment [3]

We next focus on the text output speech input misalignment problem in transformer model. Speech recognition has a characteristic to have monotonic alignment between text output and speech input. Transformer model disperses the attention over the entire speech utterances and is not optimized from this perspective. Therefore, we propose to make use of the monotonic alignment characteristic and improve the transformer model for ASR [3]. Specifically, we take the highest attention score to locate the alignment position, and apply a Gaussian mask on attention weights centered at the alignment position. We further introduce a regularizer to encourage monotonicity. The motivation is that this method does not reference other models and therefore is not dependent on other models' accuracy. Besides, it does not increase parameter size by utilizing existing attention weights. Experiments on LibriSpeech dataset find that our proposed model can obtain improved output-input alignment for ASR, and yields 14.5%-25.0% relative WER reductions. The comprehensive analysis of the proposed method will be presented in Chapter 4.

### 1.2.3 Dynamic Number of Layers in Encoder and Decoder [4]

Lastly, to improve on the current architecture constrained by the static number of encoder layers, we propose the universal speech transformer with dynamic number of encoder layers [4]. The maximum number of encoder layers is defined beforehand, but how many layers to be used is learnt during model training, and is dynamic across different input timesteps. It suits the needs of recognizing phonemes with different complexity and noise level, and relieves the burden of tuning depth related hyperparameters. Universal transformer adds the depth and positional embeddings repeatedly for each layer, which dilutes the acoustic information carried by hidden representation, and it also performs a partial update of hidden vectors between layers, which is less efficient especially on the very deep models. For better use of universal transformer, we modify its processing framework by removing the depth embedding and only adding the positional embedding once at transformer encoder frontend. Furthermore, to update the hidden vectors efficiently, especially on the very deep models, we adopt a full update. Experiments on LibriSpeech, Switchboard and AISHELL-1 datasets show that our model outperforms a baseline by 3.88%-13.7%, and surpasses other model with less computation cost. Chapter 5 presents this work in full detail.

## 1.3 Thesis Organization

The remaining part of this thesis is organized as follows.

First, the background information of ASR systems, including the common corpora and toolkits for ASR, and how is ASR evaluated are discussed in Chapter 2. End-to-end models used for ASR over the years are then presented, with the analysis of each model strengths and weaknesses. Some areas for improvement with the latest transformer model and relevant existing works are introduced as well.

Next, a unified framework for speaker adaptation in the transformer model is proposed in Chapter 3, which consists of feature adaptation and model adaptation. This model is applicable for both general speaker adaptation and specific target speaker adaptation. Moreover, the proposed system is capable of adapting to multiple speakers simultaneously with the multi-speaker adaptation method.

In Chapter 4, text output speech input misalignment problem in transformer model is discussed. In particular, a cross attention biasing technique with misalignment regularization for the transformer model is proposed. Detailed experiments are conducted to verify its effectiveness and to compare with the baseline model.

Chapter 5 then proposes the universal speech transformer, which generalizes the speech transformer with dynamic numbers of encoder/decoder layers, and relieves the burden of tuning depth related hyperparameters. It suits the needs of recognizing phonemes with different complexity and noise level.

Finally, the thesis is concluded in Chapter 6, with discussions on relevant future directions.

# Chapter 2

# End-to-end Approaches for ASR: A Literature Review

This chapter introduces the key concepts and components of automatic speech recognition (ASR). We first introduce the background information of ASR systems, including the common corpora and toolkits for ASR, and how is ASR evaluated. Then, we review the end-to-end models used for ASR over the years. Afterwards, we identify some challenging problems with ASR or transformer model, and evaluate some existing works addressing these issues.

## 2.1 Background

Automatic speech recognition refers to the process to map speech signals to its text transcription. It started with a program called Audrey in Bell Labs in the 1950s, which could transcribe simple numbers only. The early research on ASR traced back to the mid-1970, where hidden Markov model (HMM) was deployed to determine the correct words with probability function [19, 20]. The conventional methods use pipeline structures to perform ASR task, consisting of acoustic model, pronunciation dictionary and language model. Acoustic model converts the speech input into phonemes, which is the smallest unit of speech distinguishing one word from another. Pronunciation dictionary is a collection of rules to connect phonemes into words, and language model connects words into semantically meaningful sentences. On top of the conventional methods, artificial neural

networks (ANN) was then combined with HMM to improve flexibility and ASR performance [21–29]. This is to take advantage of the benefits of both models. The output of ANN is trained to predict the posterior probability of an HMM state with the acoustic input. Later on, a major breakthrough in ASR occurs with the use of deep neural network (DNN) [30–34]. Some works explore different types of neuron activation function and different network architectures. For example, convolutional neural network (CNN) was well studied and applied in computer vision successfully. It is used in ASR across frequency, with the HMM capturing time variation. Some works investigate the multi-task learning in the DNN framework, such as multilingual or cross-lingual speech recognition. Recently, end-to-end models are potential successors of conventional methods, which integrate the pipeline components into one neural network architecture. End-to-end models directly map the speech input to text output. Connectionist Temporal Classification (CTC) model uses one encoder only for the mapping, while the more popular model architecture is encoder and decoder combination in attention encoder-decoder (AED) model or transformer. Watanabe et al. [35] proposed the joint CTC/AED architecture, including the advantages of both models. We will focus on end-to-end model architectures in this work. These latest end-to-end models will be reviewed in Section 2.2.

### 2.1.1 Corpora for ASR

Common open source corpora for ASR task are summarized in Table 2.1, which vary in language, duration and context, etc. We only include the most commonly used corpora, and haven't included the multilingual cases or minor languages. In this thesis, we mainly use LibriSpeech, Switchboard and AISHELL-I corpus, which contain both English and Mandarin languages. LibriSpeech and AISHELL-I are of sampling rate 16k, while Switchboard has a sampling rate of 8k.

### 2.1.2 Toolkits for ASR

There are a number of open source toolkits for ASR. We do not discuss proprietary speech recognition toolkit here. The Kaldi toolkit [44] is a C++ based speech recognition toolkit that builds on traditional GMM-HMM and hybrid DNN-HMM models [19, 20].

Table 2.1: Common corpora for ASR task

| **Corpus** | **Description** |
|---|---|
| AISHELL-I [36] | 150 hours Chinese Mandarin speech corpus by 400 speakers with various accents across China |
| CSJ [37] | 661 hours spontaneous Japanese speech of about 7 million words |
| HKUST [38] | 200 hours HKUST Mandarin Telephone Speech corpus by more than 2100 speakers in mainland China |
| LibriSpeech [39] | approximately 1000 hours of 16kHz read English speech of audiobooks |
| Switchboard (SWBD) [40] | 300 hours of conversational English telephone speech from around the US |
| Ted-Lium [41, 42] | speech from the TED talks with its accompanying aligned transcripts |
| Wall Street Journal (WSJ) [43] | 81 hours of transcribed audio data |

It provides the complete recipes for building speech recognition systems. Espnet [45] is developed on top of Kaldi style data processing, feature extraction, etc. and supports almost all end-to-end model architectures used in ASR. Besides ASR, Espnet also supports text-to-speech (TTS), speech translation (ST), machine translation (MT) and voice conversion (VC) tasks. Espnet2 is proposed afterwards which gets rid of Kaldi and Chainer. We use Espnet toolkit to conduct experiments in Chapter 3, Chapter 4 and Chapter 5. Recently, Espresso toolkit [46] is developed with the aim to integrate with systems from natural language processing (NLP). Compared with Espnet, it decoders 4-11× faster. Speechbrain [47] is an all-in-one speech toolkit on PyTorch that is capable of speech recognition, speaker recognition, speech enhancement (SE), etc. With the development of self-supervised learning, speech representation learning is integrated in the Fairseq toolkit [48], which was meant for text generation tasks like language modeling (LM) or summarization. The learned speech representation can be used for any downstream tasks, like ASR, phoneme recognition, speech emotion recognition, etc. State-of-the-art speech

Table 2.2: Toolkits for End-to-end ASR

| **Toolkits** | **Year** | **Language** | **Recipes** | **Other Applications** |
|---|---|---|---|---|
| ESSEN [53] | 2015 | C++ | WSJ, SWBD, HKUST, LibriSpeech, TedLium | - |
| Espnet [45] | 2018 | Python | various | TTS, ST, MT, VC |
| Espresso [46] | 2019 | Python | WSJ, LibriSpeech, SWBD | NMT |
| E2E LF-MMI [54] | 2018 | C++ | various | - |
| Fairseq [48] | 2019 | Python | various | NMT, LM, etc. |
| LINGVO [55] | 2019 | Python | LibriSpeech | NMT |
| OpenSeq2Seq [56] | 2018 | Python | LibriSpeech | NMT, TTS |
| RETURNN [57] | 2018 | Python | WSJ, LibriSpeech, SWBD | NMT |
| SpeechBrain [47] | 2021 | Python | various | TTS, SE, etc. |
| Wav2Letter++ [58] | 2019 | C++ | WSJ, LibriSpeech | - |

representation learning models like wav2vec [49], wav2vec2.0 [50], HuBERT [51] are all included in the fairseq toolkit. Whisper is a model developed by OpenAI, with a general purpose to train a rubost ASR model on a large amount of data (680k hours) [52]. It can be used for multilingual speech recognition, speech translation and language identification as well. Popular end-to-end ASR toolkits are summarized in Table 2.2.

### 2.1.3 ASR Evaluation

The most commonly used metric for evaluating ASR performance is WER [59]. By comparing a ground truth text transcription and a hypothesis, WER is computed by:

$$\mathrm{WER} = \frac{\mathrm{S} + \mathrm{D} + \mathrm{I}}{\mathrm{N}} \in [0, +\infty), \tag{2.1}$$

where $S$, $D$ and $I$ cover all possible error cases, i.e. substitution, deletion and insertion. $N$ represents the number of words in the ground truth text transcription. If the hypothesis is hundred percent correct, WER equals to zero. WER can be arbitrarily large as the hypothesis can insert any number of wrong words over the ground truth text transcription. Character error rate (CER) is another metric that is similar to WER, except that WER evaluates on the word unit while CER evaluates on the character unit. CER is mostly used in Mandarin and Japanese speech recognition corpora.

More recently, with the advance of speech representation learning, a new benchmark named SUPERB [60] is raised to evaluate speech representation across a wide range of speech processing tasks. SUPERB assesses the usability of pretrained models on numerous downstream tasks. The evaluation is targeted in four domains below.

1. **Content** This is to evaluate the content in speech representation. Phoneme recognition, ASR, keyword spotting and query by example spoken term detection are four tasks in this domain.

2. **Speaker** Speaker modeling is analyzed by speaker identification, automatic speaker verification and speaker diarization, which are all common tasks in relation to speaker.

3. **Semantics** High level semantics is directly inferred from raw audio input, and gauged in intent classification and slot filling. These are tasks belonging to spoken language understanding.

4. **Paralinguistics** Emotion recognition is used to predict an emotion class for each utterance.

## 2.2 An Overview of End-to-end Models for ASR

In this section, we review end-to-end models used for ASR and analyze their strengths and weaknesses. End-to-end models transform input acoustic features into output tokens (phonemes, characters, words, etc.) directly. Traditional method uses multiple pipelined components including acoustic model, pronunciation dictionary and language model. They perform speech recognition by passing the information in sequence. This will bring the error propagation, as errors in the earlier stage are passed to the next component. In comparison, end-to-end models have the advantage to learn the mapping holistically without error propagation from multiple pipelined components. Furthermore, end-to-end models save the effort of handcrafting the pronunciation dictionary by linguists beforehand, and greatly reduce system complexity. This section summarizes four main types of end-to-end models: connectionist temporal classification (CTC) model [61–63], attention encoder-decoder (AED) model [5–10], recurrent neural network transducer (RNN-T) model [11, 12] and speech transformer model [13]. Variants of these four main models are briefly introduced here [64].

### 2.2.1 Connectionist Temporal Classification (CTC)

CTC model [61–63] transforms speech signals to texts directly by encoder only. It has an encoder structure with a softmax layer at last to make a label prediction (character or phoneme) for each speech time step, i.e. it predicts all labels for each time step concurrently with a non-autoregressive nature. This has the advantage of fast inference speed. A blank label is added to the vocabulary, which indicates non-speech signals. Given an input speech $x$ with length $T$, the model generates a normalized probability of emitting the $k^{th}$ label at $t^{th}$ time step:

$$\Pr(k, t|x) = \frac{\exp(y_t^k)}{\sum_{k'} \exp(y_t^{k'})}, \tag{2.2}$$

where $y_t^k$ is the element $k$ of output at timestep $t$ ($y_t$). The prediction is a probability distribution of labels with blank label included. The probability of output sequence

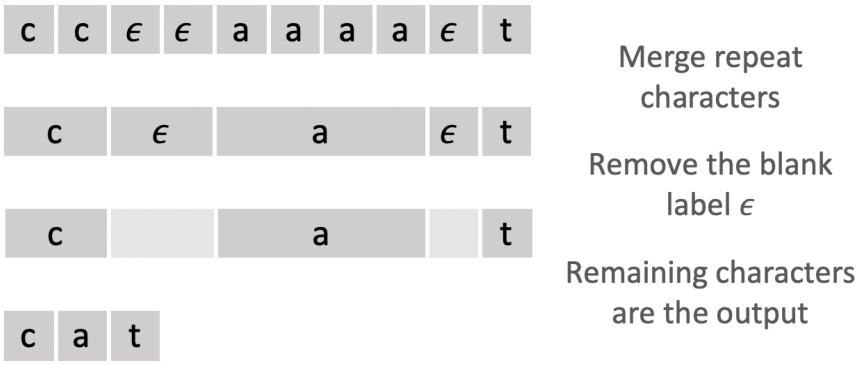



Fig. 2.1: CTC compression rule.

$a = \{a_1, a_2, ..., a_T\}$ is the product of output probability of all time steps:

$$\Pr(a|x) = \prod_{t=1}^{T} \Pr(a_t, t|x). \tag{2.3}$$

Given an output sequence with length $T$, which has same length as the input sequence, it derives the final output by compressing the continuous repeated labels into one label, and then removing the blank label, as illustrated in Figure 2.1. In this case, several output sequences may correspond to one final output. It further uses the forward-backward algorithm for the normalized probability of the output text given input speech:

$$\Pr(y|x) = \sum_{\alpha \in \beta^{-1}(y)} \Pr(a|x), \tag{2.4}$$

where $\beta$ represents the operation to compress the repeated labels and discard the blank label, i.e. it sums over all sequences which correspond to the same output after $\beta$ operation. The CTC loss is defined in Eq. (2.5) given a target output sequence $y^*$.

$$\text{CTC}(x) = -\log\Pr(y^*|x). \tag{2.5}$$

CTC model has a simple model architecture to deploy one encoder and map speech signals to texts for each time step [12] as shown in Figure 2.2. Besides, it has fast inference speed due to the parallel computation nature. However, the most criticized problem of CTC model is that it assumes each time step speech frame is independent from the speech frames of other time steps, and it does not consider the speech contextual information.

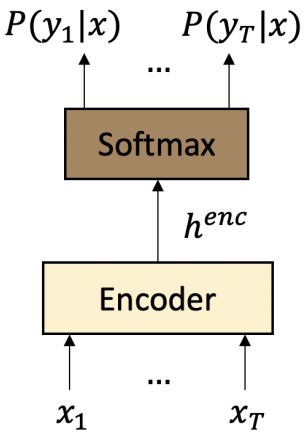



Fig. 2.2: CTC model architecture.

To address this issue, language model integration like rescoring or shallow fusion are common techniques to bring in the contextual knowledge. AED model improves from CTC model by deploying recurrent neural network (RNN) in encoder and decoder, thus considering all past speech frames during encoding. AED model is introduced in the next section.

### 2.2.2 Attention Encoder-Decoder (AED)

AED model [5–10] consists of encoder and decoder network with attention computation in between. The attention computation is to attend to the encoder output for text decoding. For both encoder and decoder, it deploys RNN structure to process sequential information. Most of the time, the encoder uses bidirectional long short-term memory (BLSTM) RNN to encode both past and future speech contexts. Given a speech sequence $x = (x_1, ..., x_T)$, the BLSTM network computes the hidden state vectors $h = (h_1, ..., h_T)$ in the forward direction with $t$ from 1 to $T$ by:

$$i_t = \sigma(W_{xi}x_t + W_{ci}c_{t-1} + W_{hi}h_{t-1} + b_i), \tag{2.6}$$

$$f_t = \sigma(W_{xf}x_t + W_{cf}c_{t-1} + W_{hf}h_{t-1} + b_f), \tag{2.7}$$

$$c_t = i_t \tanh(W_{hc}h_{t-1} + W_{xc}x_t + b_c) + f_t c_{t-1}, \tag{2.8}$$

$$o_t = \sigma(W_{xo}x_t + W_{co}c_{t-1} + W_{ho}h_{t-1} + b_o), \tag{2.9}$$

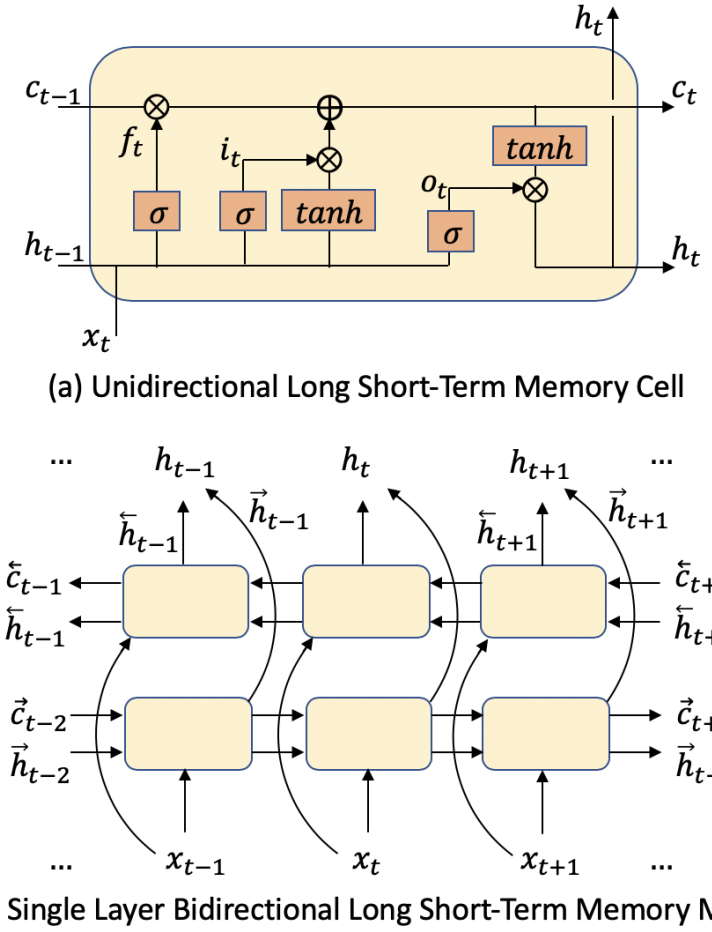



Fig. 2.3: Long short-term memory cell and bidirectional LSTM model architecture.

$$h_t = o_t \tanh(c_t), \tag{2.10}$$

where $i_t$ represents the input gate at time step $t$, $f_t$ and $o_t$ are the forget gate and the output gate respectively, $c_t$ is the cell state vector and has the same dimension with hidden vectors, $\sigma$ is the sigmoid function. The computation from Eq. (2.6) to Eq. (2.10) is denoted as $\mathcal{H}$, and is presented in Figure 2.3(a).

The BLSTM network computes hidden state in both forward direction and backward direction to encode both past and future information. It then computes the encoder final output as Eq. (2.13) using hidden state vectors from both directions.

$$\overrightarrow{h}_t = \mathcal{H}(W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + W_{x\overrightarrow{h}} x_t + b_{\overrightarrow{h}}), \tag{2.11}$$

$$\overleftarrow{h}_t = \mathcal{H}(W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + W_{x\overleftarrow{h}} x_t + b_{\overleftarrow{h}}), \tag{2.12}$$

$$h_t = W_{\overrightarrow{h}h}\overrightarrow{h}_t + W_{\overleftarrow{h}h}\overleftarrow{h}_t + b_h, \tag{2.13}$$

where $\overrightarrow{h}$ is the hidden sequence in the forward direction and $\overleftarrow{h}$ is the hidden sequence in the backward direction. Figure 2.3(b) gives an illustration of a single layer BLSTM model structure.

Usually the encoder has a few BLSTM layers stacked together to build a deeper model. It has more model capacity and is able to encode complex information for better representations. The output from previous layer is fed to the next layer as input, and the first layer receives the speech input. For layer $n \in [2, N]$ (except the first layer), the hidden state is calculated by:

$$\overrightarrow{h}_t^n = \mathcal{H}(W_{h^{n-1}\overrightarrow{h}^n}h_t^{n-1} + W_{\overrightarrow{h}^n\overrightarrow{h}^n}\overrightarrow{h}_{t-1}^n + b_{\overrightarrow{h}^n}), \tag{2.14}$$

$$\overleftarrow{h}_t^n = \mathcal{H}(W_{\overleftarrow{h}^n\overleftarrow{h}^n}\overleftarrow{h}_{t+1}^n + W_{h^{n-1}\overleftarrow{h}^n}h_t^{n-1} + b_{\overleftarrow{h}^n}), \tag{2.15}$$

$$h_t^n = W_{\overrightarrow{h}^n h^n}\overrightarrow{h}_t^n + W_{\overleftarrow{h}^n h^n}\overleftarrow{h}_t^n + b_h^n. \tag{2.16}$$

For decoder, it typically deploys a unidirectional long short-term memory (LSTM) network, whose algorithm is depicted in Figure 2.3(a). The context vector in decoder comes from the attention computation between encoder output hidden sequence $h^N = (h_1^N, ..., h_T^N)$ and decoder hidden state vector $s_i$, so as to extract useful information from encoder to generate the next token in decoder. For context vector $c_i$, the attention computation is:

$$e_{i,t} = s_i W_s * (h_t^N W_h)^T, \tag{2.17}$$

$$\alpha_{i,t} = \frac{\exp(e_{i,t})}{\sum_{t=1}^{T}\exp(e_{i,t})}, \tag{2.18}$$

$$c_i = \sum_{t=1}^{T}\alpha_{i,t}h_t^N. \tag{2.19}$$

With the context vector $c_i$, LSTM network in decoder and the prediction probability is defined as:

$$s_i = \mathcal{H}(W_{ys}y_{i-1} + W_{ss}s_{i-1} + W_{cs}c_{i-1} + b_s), \tag{2.20}$$

$$P(y_i|x, y_{<i}) = \text{softmax}(W_{sy}s_i + W_{cy}c_i + b_y). \tag{2.21}$$

Figure 2.4 shows the overall architecture of AED model. AED model could resolve the frame independence issue of CTC explained in the last section by using the BLSTM

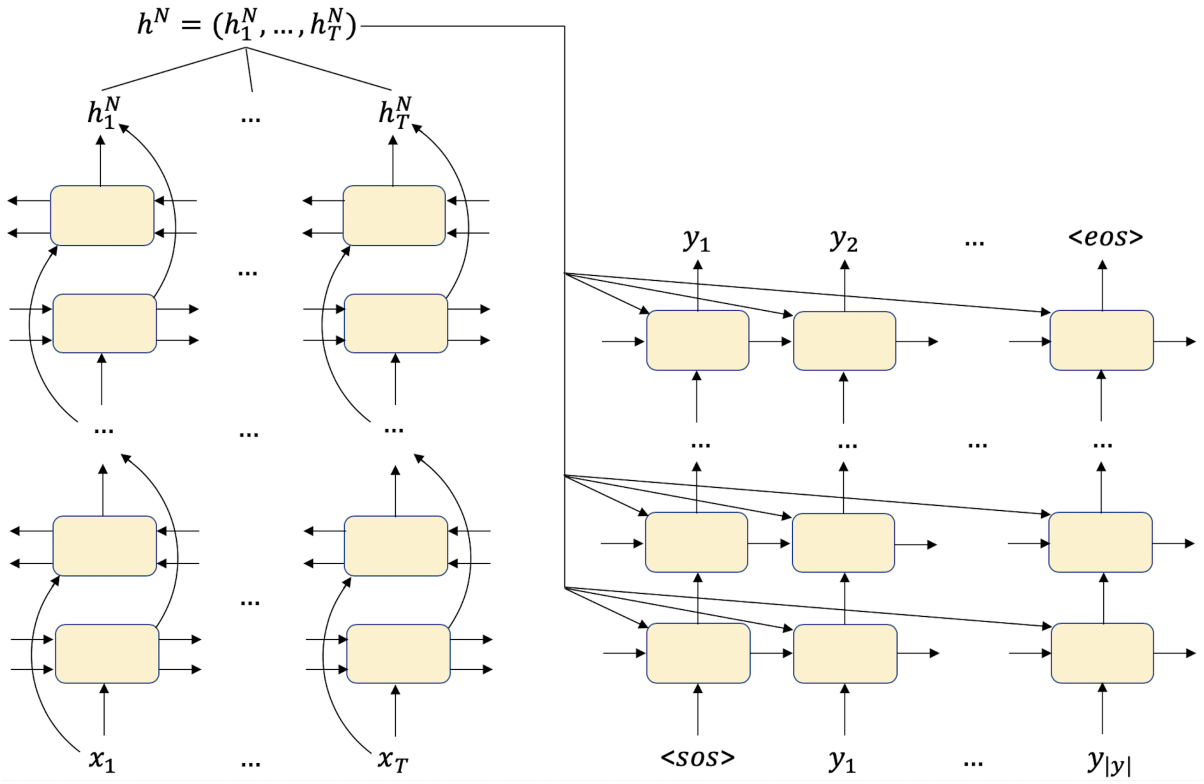



Fig. 2.4: Attention Encoder-Decoder model architecture [5–10].

network to encode contextual information from both past and future input at the same time. Furthermore, compared with the standard encoder-decoder network without the attention in between, it uses the attention computation to relate useful information from encoder output at each time step, rather than depending only on the last hidden state vector of encoder for decoding. This method is more robust and could provide more knowledge to the decoder for making accurate prediction. However, the RNN encoder and decoder structure still suffers from the vanishing gradient problem, i.e., propagating the gradient from the output back to the input. Besides, the attention computation is processed over the entire input speech sequence, making it impossible to use AED model for online speech recognition, whose speech input comes in real time. RNN transducer (RNN-T) model is suitable for both offline and online speech recogntion, and is detailed in Section 2.2.3.

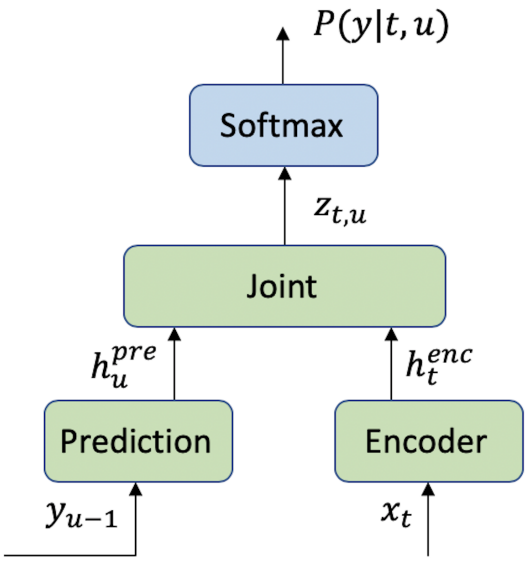



Fig. 2.5: RNN Transducer model architecture [11, 12].

### 2.2.3 RNN Transducer (RNN-T)

RNN-T model [11, 12] has encoder, prediction and joint network as presented in Figure 2.5. The encoder network encodes the acoustic features, and the prediction network acts as a language model. The joint network operates on the encoder and prediction network outputs for the final text transcription. The encoder uses LSTM or BLSTM network, which is the same as the encoder network of AED model in Section 2.2.2. The encoder output at time step $t$ is:

$$h_t^{enc} = f^{enc}(x_t), \tag{2.22}$$

where $x_t$ is the speech input at timestep $t$, $f^{enc}$ represents the encoder network in Figure 2.5. The prediction network also uses RNN model to predict the next label given current output $y_{u-1}$ of RNN-T model:

$$h_u^{pre} = f^{pre}(y_{u-1}), \tag{2.23}$$

where $f^{pre}$ represents the prediction network in Figure 2.5. Lastly, the joint network combines outputs $h_t^{enc}$ and $h_u^{pre}$, and adds a softmax layer behind. Same as CTC model, it then uses the forward-backward algorithm to obtain the normalized probability of the

output text given input speech.

$$h_{t,u} = W_y \psi(U h_t^{enc} + V h_u^{pre} + b_z) + b_y, \tag{2.24}$$

$$P(k|t,u) = \text{softmax}(h_{t,u}^k), \tag{2.25}$$

where $k$ is the output token, $\psi$ represent a non-linear function, $U$, $V$, $b_z$ and $b_y$ are the learnable model parameters.

RNN-T model has the advantage to be applied for online speech recognition. The joint network can easily process encoder output and prediction network output concurrently, while the encoder network processes the incoming speech signals. Besides, it also relaxes the frame independence assumption of CTC model. However, RNN-T model is very computationally expensive due to the forward-backward computation. $P(k|t,u)$ in Eq. (2.25) has three dimension $k, t, u$, and it requires more computation resources compared with previous reviewed end-to-end models in Section 2.2.1 and Section 2.2.2 [12].

### 2.2.4 Speech Transformer

More recently, transformer model [17] is proposed and plays a role in almost all latest models in NLP and speech recognition. Transformer model can learn pairwise relationship between any two elements directly through the self-attention network. It does not incur the vanishing gradient problem of RNN or is limited by the kernel size of convolutional neural network (CNN), thus is able to capture longer range context. Besides, its capability for parallel computation also enables batched operation and fast computation speed. Speech transformer [13] builds based on the transformer model for ASR. Figure 2.6 gives the overall framework of speech transformer model. There are $N_e$ encoder layers and $N_d$ decoder layers in the model. For a speech input sequence, it first applies two convolution layers with stride two to reduce the hidden representation length, as the speech input length is much longer than its corresponding text length in the decoder. A sinusoidal positional encoding is added to encode position information as Eq. (5.8) and Eq. (5.9).

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}), \tag{2.26}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}), \tag{2.27}$$

where $pos$ is the position, $d_{model}$ is the positional and depth embedding dimension, and $i$ is the i$^{\text{th}}$ dimension.

Transformer encoder then computes hidden state representation of each position in parallel with a self-attention network. For the three inputs key, query and value, which are three distinct transformation of input sequence used to extract the input information, the multi-head attention network, which concatenates self-attention network $h$ times as introduced in [17], is:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O, \tag{2.28}$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{2.29}$$

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.30}$$

where $h$ is the head number, $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$, $d_k = d_q = d_v = d_{model}/h$ in our experiments.

Multi-head attention allows learning input representation from different subspaces concurrently. Layer normalization and residual connection are applied before and after multi-head attention network. Afterwards, there is a position-wise feedforward network with rectified linear unit (ReLU) activation:

$$\text{FFN}(x) = b_2 + W_2\text{max}(0, b_1 + W_1 x), \tag{2.31}$$

where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$, and the biases $b_1 \in \mathbb{R}^{d_{ff}}$, $b_2 \in \mathbb{R}^{d_{model}}$.

Each encoder layer has multi-head attention network and position-wise feedforward network. For speech transformer decoder, there is a third sublayer between multi-head attention and feedforward network. This is a multi-head cross attention computed over encoder and decoder output, where key and value vectors come from encoder, and query vector comes from decoder. To prevent self-attention network from attending to future positions in the decoder, a masking is applied.

Transformer architecture is widely used due to its ability to capture long range relationship from inputs, as well as its fast computation speed. However, due to its nature to attend to the entire utterance, transformer is not suitable for online speech recognition.

A masking could be applied to let the attention focuses on partial history input frames only [65], so as to apply for online speech recognition.

Despite transformer model's capability to learn long-range relationship between inputs and to enable parallel computation with fast speed [66], there is room for improvement. First, speaker adaptation is a long standing problem in ASR, which arises due to speaker mismatch between training and test data. ASR performance could drop drastically should the test speaker data deviates a lot from training speaker data. There is less work to improve the transformer model for speaker adaptation. Second, speech recognition has a characteristic to have monotonic alignment between text output and speech input. This is different from other tasks such as text summarization, which highly depends on the past and future contexts. Transformer model disperses the attention distribution over the entire speech input for information extraction, and does not consider the monotonic alignment. Third, transformer model has static number of encoder layers, which needs careful tuning to achieve the best performance. The fixed number of encoder layers is unable to recognize phonemes with different complexity and noise level dynamically. We review some existing works with regards to speaker adaptation in end-to-end ASR models in Section 2.3.

## 2.3 Prior Speaker Adaptation Works

In this section, we review some prior works related to speaker adaptation. Given our motivation to improve speaker adaptation in the latest Transformer model, we mainly introduce prior efforts on neural network-based speech recognition systems, which can be broadly categorized into feature adaptation and model adaptation.

### 2.3.1 Feature Adaptation

Feature adaptation acts on the acoustic features to conduct speaker-aware training. The simplest form is feature transformation. One method is through normalization to standardize the acoustic features to deal with different speakers. Another method is augmenting the acoustic features to supplement speaker-related knowledge [67–69]. Let $\mathrm{x} \in \mathbb{R}^d$

denotes the acoustic features, and normalization is characterized by an affine transformation or standardization:

$$x_t = Ax + b, \tag{2.32}$$

$$x_t = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{2.33}$$

where $\mu \in \mathbb{R}^d$ is the mean value, $\sigma^2 \in \mathbb{R}^d$ is the variance term, and $\epsilon$ is a small constant for numerical stability. This is a straightforward and common method to regulate the acoustic features.

Augmentation aims to provide speech variants with characteristics resembling the target speaker or acoustic environment. Speech variants include different volumes or speaking rates of the target speaker [70–72]. Some works use label-preserving speech with perturbed vocal tract length warp factors [73, 74]. Augmentation can also help in creating a richer adaptation dataset. For example, [74–76] use stochastic feature mapping to perform voice conversion [77] between speakers, which generates more acoustic information for the target speaker not seen in the training dataset. Text-to-speech (TTS) technique is also employed to expand the target speaker voice set and could potentially cover unknown acoustic conditions [78–80].

Not just feature transformation, [81] supplements acoustic features with feature-space maximum likelihood linear regression (fMLLR) transformed features for improving the network robustness against speaker information mismatch. Adapting the learnable filterbanks with SincNet could compensate for varying vocal tract lengths among speakers [82]. Alternatively, dynamically generating the scaling and shifting parameters in layer normalization has resulted in adaptive acoustic modeling, which does not require extra adaptation data [83].

Another form of feature adaptation makes use of i-vectors or ⋆-vectors [84–90]. I-vector is extracted based on a data-driven approach by mapping frames of an utterance to a low-dimensional vector space using a factor analysis technique [91–93]. One usage of i-vector is taking attention with i-vectors from multiple speakers as speaker embedding and appending to the acoustic features to generalize to unseen test speakers [94, 95]. ⋆-vectors are network embeddings trained with the objective to distinguish between speakers [96]. In particular, x-vectors are transformed from variable-length utterances during speaker

recognition applying data augmentation [97, 98]. D-vectors are extracted from the long short-term memory recurrent neural networks for modeling speaker variability [99, 100]. [101] generates H-vectors with a hierarchical attention network to capture speaker related information at both frame level and segment level. Other than i-vectors or ⋆-vectors, summary vector computed from the sequence summary network is added to the encoder input to represent speaker characteristics [83, 102–104]. Figure 2.7 gives an example of a sequence summary network. [105] uses a time-delay neural network and an online averaging layer to predict learning hidden unit contribution (LHUC) features for speaker adaptation.

The advantage of feature adaptation methods is that it does not require extra adaptation data. The adaptation is performed with model transformation or with the help of ⋆-vectors to account for different target speakers. However, introducing additional ⋆-vectors is the drawback of feature adaptation methods, as it requires pre-processing and the model is not a fully end-to-end model. Furthermore, the model performance is dependent on the quality of ⋆-vectors.

### 2.3.2 Model Adaptation

Model adaptation aims to train the speaker-dependent model from speaker-independent model parameters with additional adaptation data. It can adapt the entire acoustic model or specific layers [106]. Structured parameterisation is another option to transform model parameters with a linear network [107–109]. The major challenge in model adaptation based methods is to prevent overfitting to the finite adaptation data and catastrophic forgetting [110, 111] on the existing training data, and the related works can be divided into three categories from this perspective.

The first group works on imposing a restricted set of speaker-dependent parameters on the model. learning hidden unit contribution (LHUC) is a typical example by modifying the combination of hidden units while fixing the network's basis functions [112–114]. Small speaker code is another method, which is learnt for each speaker via the back-propagation algorithm to transform the speaker features into a general speaker-independent feature space, in parallel with learning the large generic neural network [115, 116]. [117, 118] adapt the linear transformations in batch normalization to

match the hidden layer input distribution between training and test data, so as to account for different speaker input distribution. Other than batch normalization, sigmoid and rectified linear unit (ReLU) parameters are generalised for adaptation as well [119]. [120] evaluates various affine transformations of the top hidden layer and softmax layer by batch update and stochastic gradient ascent. A low-rank plus diagonal (LRPD) decomposition method is proposed in [121, 122] by reconstructing the adaptation matrix with a variable amount of adaptation parameters to equip the model with the scalability for adaptation. Factorized hidden layer (FHL) approach uses a linear combination of bases to represent speaker-dependent hidden layers, which can be initialized with i-vectors and trained in an unsupervised fashion [123]. This factorization idea is utilized similarly in [124], which estimates an interpolation vector of multiple weight matrices in the cluster adaptive training (CAT). Likewise, [125] factorizes a context adaptive neural network and learns the weighting coefficients jointly with the main network. To reduce the number of adaptable parameters, singular value decomposition (SVD) technique can be applied directly to shrink the model size [126, 127].

The second class of methods avoids overfitting by regularization, which is a common technique to address the overfitting issue. L2 regularization [106] and Kullback-Leibler divergence (KLD) regularization [128–131] have been studied for this purpose. Since KLD is an asymmetric measure in distribution and it may not reach the global optimum, [132] applies adversarial learning to regularize the speaker-dependent hidden representations with a discriminator network. Meta-learning technique is used in [133, 134], which formulate speaker adaptation as a meta-learning task and investigate the use of meta-learner to embed speaker adaptation via gradient descent directly into the training of the acoustic model. Maximum a posteriori (MAP) adaptation [135] is a Bayesian approach by obtaining a prior density from the training data and re-estimating speaker-dependent parameters [136]. Another bayesian learning method is bayesian learning of hidden unit contributions (BLHUC), which adopts a variational inference based approach to estimate the posterior distribution of LHUC parameters and models the uncertainty of the speaker-dependent parameter space [137].

Lastly, adjusting the training objective function can tackle the overfitting problem as well. Multitask learning [138, 139] is a typical option where the main objective is for

ASR and the related tasks are learnt jointly for dealing with overfitting. For instance, auxiliary mono-phone/senone-cluster classification task can help build up the coverage of acoustic units and enhance the discrimination capability of the adapted models with limited adaptation data [140–142]. Similarly, word and letter classification are proposed in [129, 130] with different granularity for CTC and AED models, which also helps in enlarging the coverage of acoustic units. Conservative training is another technique which compensates for the lack of adaptation data by replacing the target of missing units with the outputs computed by the original network [108, 143].

The strength of model adaptation approaches is that it is independent of knowledge of $\star$-vectors, and there is no pre-processing needed. However, one challenge with model adaptation methods is overfitting to adaptation data. As seen in this section, all methods are proposed to tackle overfitting issue from different perspective, by restricting parameters, regularization or adjusting the training objective. Besides, it is difficult to determine which model parameters to adapt for target speaker. Some works focus on the batch normalization layer, while some works modify the sigmoid and rectified linear unit (ReLU) parameters. Therefore, there is no consistent guideline on which model parameters to adapt, and choosing certain sub-layer(s) intuitively may not be optimal.

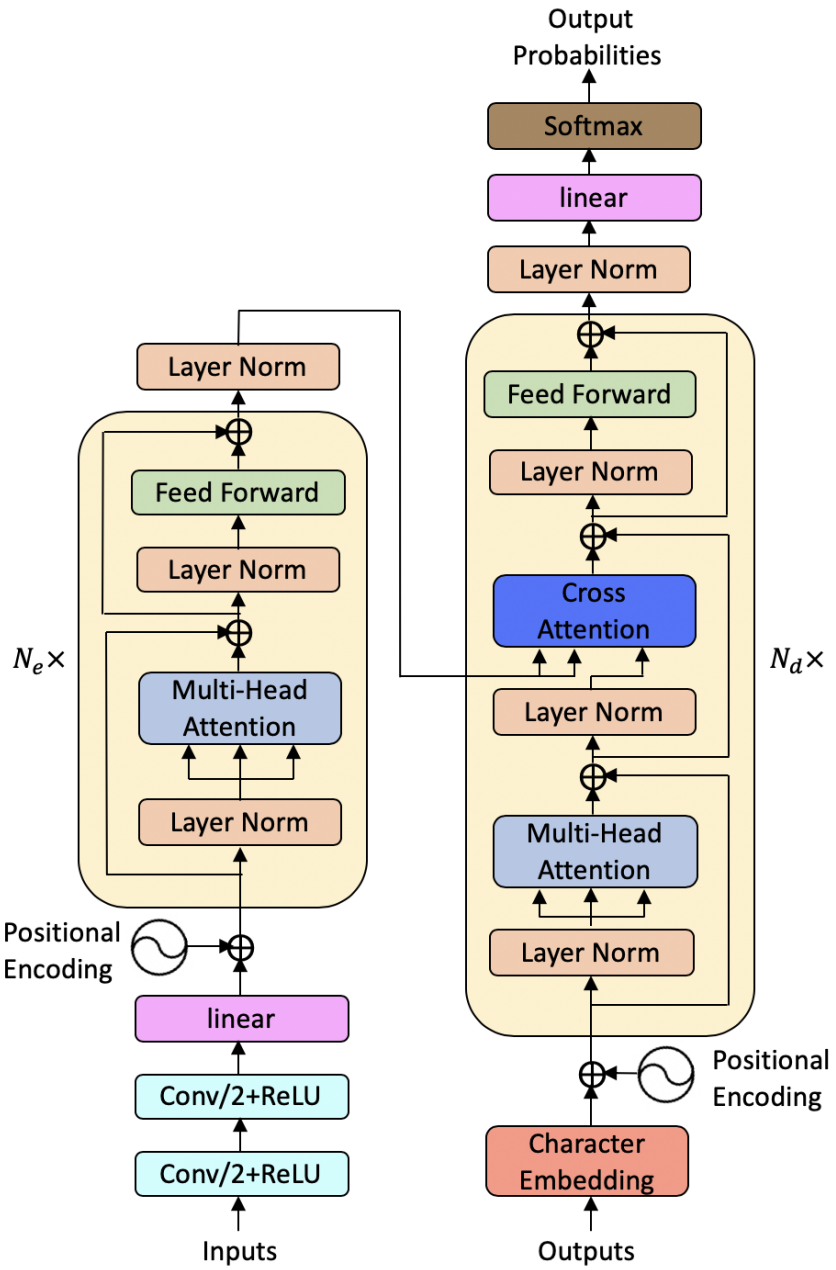



Fig. 2.6: Speech Transformer model architecture [13].

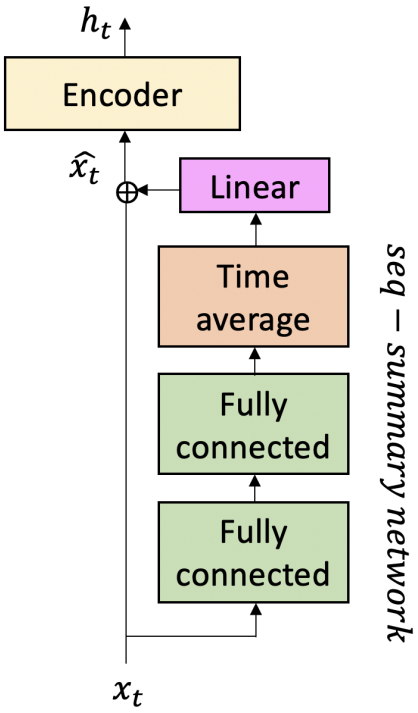



Fig. 2.7: Schematic diagram of the sequence summary network.

# Chapter 3

# Unified Speaker Adaptation

In this chapter, a unified speaker adaptation method is proposed to tackle the speaker adaptation issue in the transformer model. Speaker adaptation is a long-standing problem for ASR, and has been under study from traditional GMM-HMM models to the latest end-to-end models. Our focus is working on the transformer model and improve the ASR performance.

As reviewed in Section 2.3, previous speaker adaptation works can be divided into featuer adaptation and model adaptation. They are applicable to different scenarios, i.e., feature adaptation works directly on the model input and does not require additional target speaker data, while model adaptation usually works with some extra target speaker data. Our proposed method uses both feature adaptation and model adaptation, with the aim to cover all the use cases. Firstly, our method can be used for general speaker adaptation with no certain target speaker. Secondly, under the scenario of limited target speaker data, our method could be utilized to adapt to that target speaker. Lastly, our method applies to adapting to multiple speakers at the same time.

This chapter is organized as follows. Section 3.1 first provides the motivation for speaker adaptation. Section 3.2 then gives a full picture of our proposed method including feature adaptation and model adaptation. We further extend our method to multi-speaker adaptation scenario. Experiments are conducted in Section 3.3 to analyze the performance of the proposed approach under different use cases. We then conduct some analysis of the method in Section 3.4. Section 3.5 concludes this chapter.

## 3.1 Introduction

### 3.1.1 Motivation

As detailed in Chapter 2, although end-to-end ASR models are reaching promising performance especially with the latest transformer model, speaker adaptation in end-to-end models can be further improved, which arises due to *speaker mismatch* between training and test data. The model trained using training data speakers may not apply well in unknown target speakers, due to variation in speaker characteristics. Adpating ASR model is a challenging task as it involves the change of large and complex models. There are some prior published efforts targeting this issue review in Section 2.3, including feature transformation [67, 70], using additional information such as i-vectors [84], structured parameterisation [107], regularization [128], variant objective functions [138], etc. However, most methods have their dedicated application scenarios, either for general speaker adaptation or some specific target speaker(s). There is less universal technique that can be applied for any speaker adaptation scheme.

### 3.1.2 Our Method

To surmount this obstacle, in this chapter we propose a unified speaker adaptation approach that can be deployed for general speaker adaptation, specific target speaker adaptation and multi-speaker adaptation. First, we propose the speaker-aware persistent memory model to generalize better to unseen test speakers. In particular, speaker i-vectors from the training data are sampled and concatenated to speech utterances in each encoder layer, and the speaker knowledge is learnt through attention computation with speaker i-vectors. Second, we use the gradual pruning method to gradually prune less contributing parameters on model encoder, and then use the pruned parameters for target speaker adaptation while freeze the unpruned parameters to retain the model performance on general speaker data. Third, we distribute the pruned free parameters among target speakers equally, randomly, and simultaneously for multi-speaker adaptation.

## 3.2 Model Architecture

We present our proposed unified speaker adaptation approach including both feature adaptation and model adaptation, in line with the historical speaker adaptation categories reviewed in Chapter 2. Our algorithm is applicable for both general speaker adaptation and specific target speaker adaptation, with or without target speaker data. Additionally, we refine our method for multi-speaker adaptation scenario, which is described in this section as well.

### 3.2.1 Feature Based Adaptation Method

Persistent memory is a concept first raised in [144] to replace the feedforward network (Eq. (2.31)) in the transformer model with an *all-attention* network. To replace the feedforward network in the transformer model, the attention was computed on a set of key and value vectors which are randomly initialized, so as to simulate $\mathrm{W}_1$ and $\mathrm{W}_2$ vectors in Eq. (2.31), which are also randomly initialized. The attention computed on random vectors was meant to extract information independent of adjacent contexts, such as general knowledge of the given input. The term "persistent memory" originated from the sharing of the random key and value vectors among the training and test data, in a way carrying the memory along.

Inspired by [144], we propose to perform speaker aware training by learning speaker specific knowledge in a similar manner as capturing general knowledge by the persistent memory model. We modify the speech transformer encoder layers as seen in Fig. 3.1. $N_e$ represents the number of encoder layers, and $h$ is the number of attention head. Instead of constructing a set of random key and value vectors for capturing general knowledge, we take advantage of speaker i-vectors [145] which contain speaker information by nature, and hence our method falls under the feature adaptation category. To be more specific, $N$ speaker i-vectors $m_1, ..., m_N \in \mathbb{R}^{d_k}$ are randomly sampled from the training data, which form the speaker space. Persistent memory vectors $\mathrm{M}_k$ and $\mathrm{M}_v$ are learned transformation of the speaker space:

$$\mathrm{M}_k = \mathrm{Concat}([\mathrm{U}_k m_1, ..., \mathrm{U}_k m_N]) \in \mathbb{R}^{N \times d_k}, \tag{3.1}$$

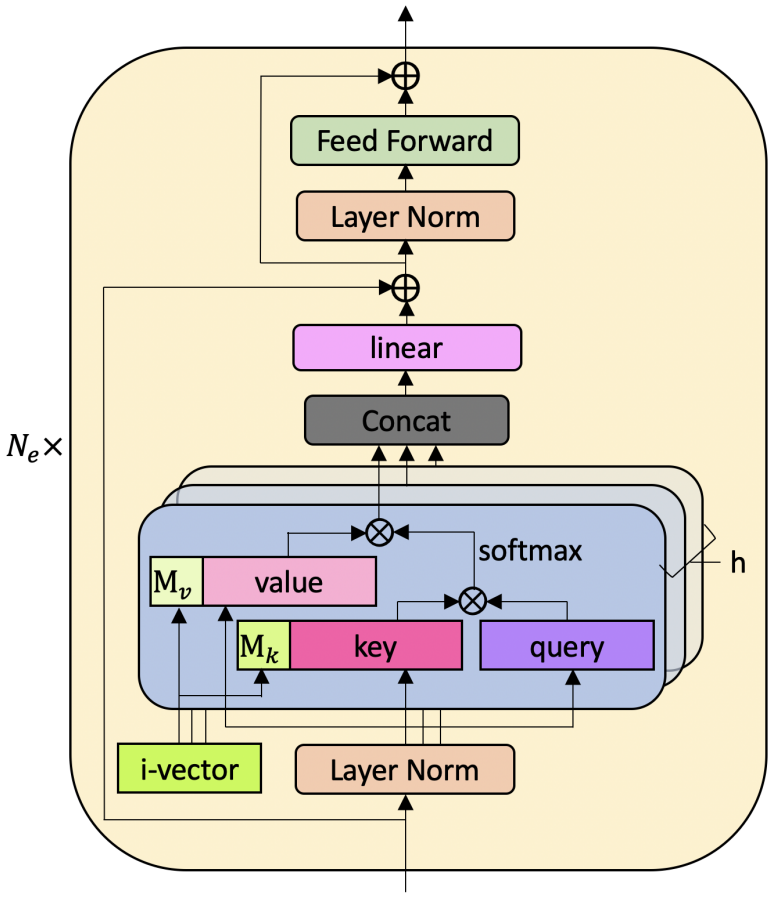



Fig. 3.1: Proposed speaker adaptation model.

$$\mathrm{M}_v = \mathrm{Concat}([\mathrm{U}_v m_1, ..., \mathrm{U}_v m_N]) \in \mathbb{R}^{N \times d_k}, \tag{3.2}$$

where $\mathrm{U}_k \in \mathbb{R}^{d_k \times d_k}$, $\mathrm{U}_v \in \mathbb{R}^{d_k \times d_k}$. With the persistent memory vectors, speaker knowledge is learnt through the self-attention computation in Eq. (2.30). This is built on the assumption that the linear combinations of speaker space are enough to cover the speaker information space, i.e., a weighted sum of persistent memory vectors can represent any specific speaker knowledge. This is also the reason we term it as *speaker space*. The input vectors of the self-attention network $\mathrm{X} = [x_1, ..., x_t]$ and the persistent memory vectors $\mathrm{M}_k$ and $\mathrm{M}_v$ are then concatenated together to form the new key and value vectors for attention computation:

$$\mathrm{K}_m = [k_1, ..., k_{t+N}] = \mathrm{Concat}([\mathrm{W}_k x_1, ..., \mathrm{W}_k x_t], \mathrm{M}_k), \tag{3.3}$$

$$\mathrm{V}_m = [v_1, ..., v_{t+N}] = \mathrm{Concat}([\mathrm{W}_v x_1, ..., \mathrm{W}_v x_t], \mathrm{M}_v), \tag{3.4}$$

$$\mathrm{Attention}(\mathrm{Q}, \mathrm{K}_m, \mathrm{V}_m) = \mathrm{softmax}(\frac{\mathrm{Q}\mathrm{K}_m^T}{\sqrt{d_k}})\mathrm{V}_m. \tag{3.5}$$

Different from the randomly initialized learnable key and value vectors in [144], the $N$ speaker i-vectors we sample are fixed, and are supposed to form the basis for speaker knowledge extraction. Only the weight matrices associated with the attention mechanism $\mathrm{U}_k$ and $\mathrm{U}_v$ are learnable. Furthermore, instead of replacing the feedforward network, we aim to get the relevant speaker information with the persistent memory essentially, so we leave the feedforward network untouched. Since the set of speaker i-vectors are shared across all the training data, and in a manner of carrying the persistent memory, we name the model as speaker aware persistent memory model.

Our proposed speaker aware persistent memory model has a few distinct advantages over other similar feature adaptation techniques. Firstly, by randomly sampling $N$ speaker i-vectors from the training data which form the speaker space and learning various speaker data from them, the model is meant to capture any speaker knowledge. It effectively addresses the problem of having unknown speakers in the test data. Thus, it is suitable for general speaker adaptation. It also relieves us from computing all the speaker i-vectors from the training data. Secondly, most similar to our work, [94] takes the attention with speaker i-vectors and appends it to the acoustic features for each input time step, which actually obtains time dependent speaker information. In this step, temporal information in speech is not taken into consideration. In comparison, we concatenate the persistent memory vectors to the speech input along the time dimension, and the attention is computed together with the entire utterance. By doing so, it takes the entire utterance into consideration while capturing speaker information, and thus our method learns utterance level speaker knowledge. Since speaker information is more global and could be studied along the utterance, we believe our method is more reasonable. Lastly, [94] concatenates speaker embedding with the encoder final layer output, and downscales the vector to the original dimension at the decoder side with a linear layer. It doubles the parameter size of linear transformation in each decoder layer. In contrast, attention with persistent memory vectors is integrated into the self-attention network in our model, and there is no extra parameters of linear transformation in the decoder.

### 3.2.2 Model Based Adaptation Method

The speaker aware persistent memory model we propose could adapt well to unseen speakers in the test data, and is more feasible for general speaker adaptation (unknown target speaker). When the target speaker profile is available, for instance having a few utterances of the target speaker, model adaptation techniques suit this scenario better. Nevertheless, overfitting to the limited adaptation data and catastrophic forgetting on the existing training data are critical issues for this group of methods as reviewed in Section 2.3.2. To this end, we use a novel gradual pruning method to adapt to the target speaker in a fast manner, and retain the model performance on the general speaker data at the same time. Compared with previous model adaptation works to choose a subset of model parameters to adapt, our method could let the model itself determine which model parameters to adapt, which is a more viable solution.

To illustrate our design clearly, we first set up the context of the problem. Given an input speech $\mathrm{Y} = \{y_1, ..., y_n\}$ and an output text $\mathrm{Z} = \{z_1, ..., z_m\}$, ASR models the conditional probability of the output text over the input speech as follows:

$$\Pr(\mathrm{Z}|\mathrm{Y};\theta) = \prod_{i=1}^{m} \Pr(z_i|\mathrm{Y}, z_{<i};\theta), \tag{3.6}$$

where $\theta$ represents the model parameters. Given a training dataset $\mathrm{D} = \{\mathrm{Y}_{\mathrm{D}}^j, \mathrm{Z}_{\mathrm{D}}^j\}_{j=1}^{L}$, $\theta$ is trained with the training objective to maximize the following log-likelihood of the conditional probability:

$$\mathcal{J}(\theta) = \sum_{j=1}^{L} \log \Pr(\mathrm{Z}_{\mathrm{D}}^j|\mathrm{Y}_{\mathrm{D}}^j;\theta). \tag{3.7}$$

Given a target speaker dataset $\mathrm{D}_t = \{\mathrm{Y}_{\mathrm{D}_t}^j, \mathrm{Z}_{\mathrm{D}_t}^j\}_{j=1}^{L_t}$, speaker adaptation is carried out by re-training the model for a few epochs using the new dataset with the following objective:

$$\mathcal{J}(\theta_{\mathrm{D}_t}) = \sum_{j=1}^{L_t} \log \Pr(\mathrm{Z}_{\mathrm{D}_t}^j|\mathrm{Y}_{\mathrm{D}_t}^j;\theta_{\mathrm{D}_t}), \tag{3.8}$$

where $\theta_{\mathrm{D}_t}$ is initialized with the trained parameters $\theta$ in Eq. (3.7), i.e., continue to train the model to adapt to the target speaker.

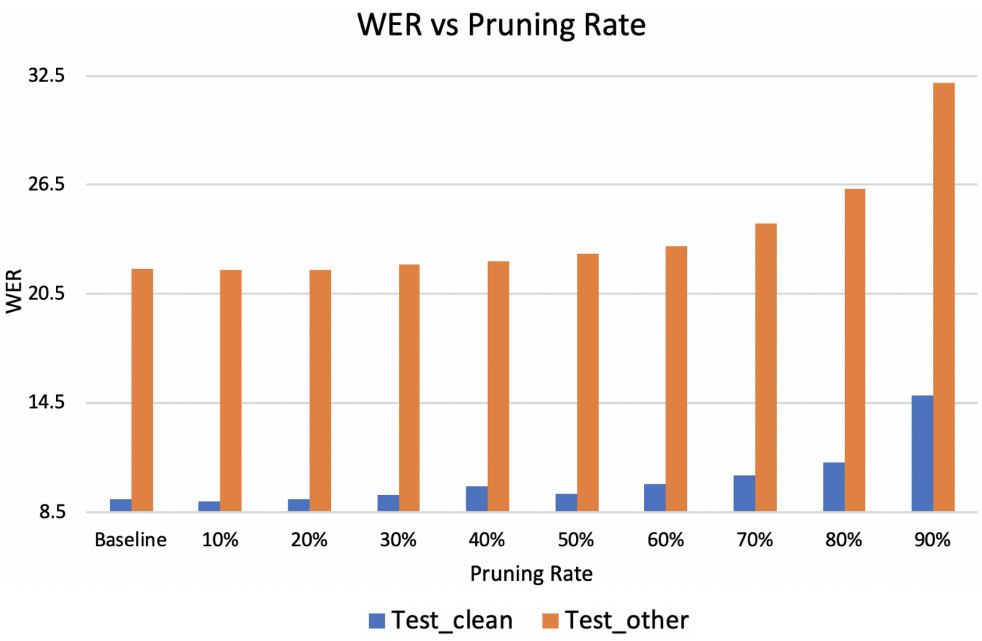



Fig. 3.2: Experiment results with varying pruning rates.

Many latest studies have shown that not all the parameters in a neural network model contribute to the training objective [146–148]. Pruning the redundant parameters leads to negligible performance degradation [149,150] or may even outperform the original model [147] due to less overfitting and better generalization. To verify this experimentally, we examine the ASR performance with different pruning rates on the model encoder using the LibriSpeech dataset as seen in Fig. 3.2. From the graph, up to 50% of encoder parameters can be pruned in ASR model with insignificant performance drop. With this enlightening finding, we propose to make use of these dispensable parameters for the speaker adaptation objective.

In particular, we first train and prune the model with training data gradually to the predetermined sparsity level by zeroing out low magnitude parameters for every 10k training steps, i.e., only retain a certain percentage of high magnitude unpruned parameters $\theta_{UP}$ after $n$ epochs of training. This is to unearth an *informative subnetwork* whose performance well matches the original model (Fig. 3.3(b)). We choose to prune the low magnitude parameters as we think they contribute less to the training objective by their magnitude. Instead of pruning on a well-trained model [151], we train and prune concurrently (with warm-up training first) to reduce the total number of training

steps and thus save computation resources. We also prove experimentally in the later section that this gradual pruning method is better than the one-time pruning method at different training stages. Given that it is the speech transformer encoder which encodes the speech or speaker information, we only prune the encoder parameters including embedding network, self-attention network and feedforward network in all encoder layers. Afterwards, we use the free encoder parameters being pruned to encode speaker knowledge specifically.

Subsequently, unpruned parameters in the informative subnetwork $\theta_{UP}$ are frozen to retain the performance on existing speakers and avoid catastrophic forgetting problem, as represented by the light grey connections in Fig. 3.3(c). We only finetune the pruned free parameters $\theta_P$ for target speaker adaptation (blue connections in Fig. 3.3(c)). The training objective will be:

$$\mathcal{J}(\theta_P) = \sum_{j=1}^{L_t} \log \Pr(\mathrm{Z}_{\mathrm{D}_t}^{j} | \mathrm{Y}_{\mathrm{D}_t}^{j}; \theta_{UP}, \theta_P), \tag{3.9}$$

where $\theta_{UP}$ represents frozen parameters and $\theta_P$ represents updated parameters. Since the informative sub-network is already capable of performing ASR task very well, we believe further finetuning the free parameters with target speaker data is an added value to the speaker-specific model.

Our proposed method has a few advantages over prior model adaptation approaches. First, as opposed to finetuning the entire model (Eq. (3.8)), we only need to finetune a small number of parameters $\theta_P$, around 10% of the model encoder parameters only. Second, compared with previous model adaptation approaches in Section 2.3.2, our method saves the trouble of intuitively adapting a certain subset of model parameters to prevent overfitting, e.g., batch normalization layer in [117] or sigmoide and ReLU parameters in [119]. Instead, we leave the model to learn by itself which parameters to keep (for maintaining the model performance) and which parameters to discard (for speaker adaptation in our case). This is more tenable as the decision is supported by the objective function. Third, our method neither change the model architecture nor add additional tasks, which is common in multitask learning [138, 139]. Last, fixing the informative subnetwork makes our model retain past knowledge with no catastrophic forgetting issue.

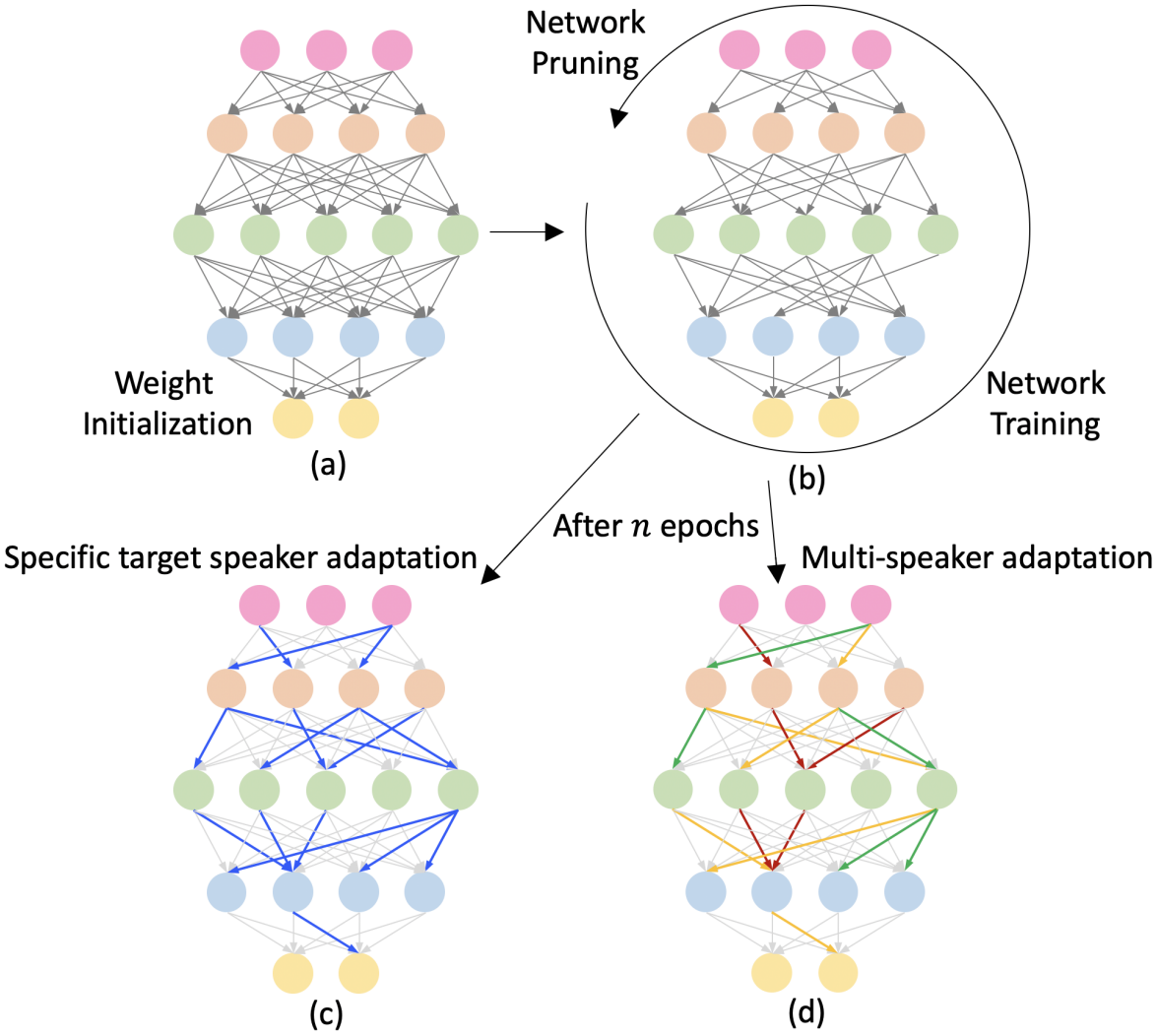



Fig. 3.3: Illustration of the proposed model adaptation methods. We first initialize model parameters (a), train and prune the model alternately for $n$ epochs (b), and finetune the pruned parameters with target speaker(s) data (c,d). Frozen parameters are indicated by light grey connections in (c,d), while finetuned parameters are colored ones, which are the parameters pruned at the earlier stage (b). A specific target speaker is adapted in (c), while (d) gives an example of three-speaker adaptation. Free parameters are equally and randomly assigned to three target speakers (represented by three colors), and are adapted accordingly.

It also prevents the model from easily overfitting on low-resource target speaker data to some extent.

### 3.2.3 Multi-Speaker Adaptation

In previous sections, we have described the proposed unified speaker adaptation approach including feature adaptation and model adaptation, applicable for both general speaker adaptation and specific target speaker adaptation. However, in a ubiquitous environment especially with multiple users, the more crucial task is to adapt the ASR model for multiple speakers and improve the ASR model accuracy overall. There is less research in relation to multi-speaker adaptation in ASR [152], especially for end-to-end systems. In this section, we introduce our multi-speaker adaptation scheme, i.e., one model adapted for multiple speakers at the same time.

Inspired by the idea of prompt tuning [153], we refine the gradual pruning method in Section 3.2.2 and apply it to multiple target speakers. Prompt tuning is recently proposed for adapting language models. It is a simple yet effective mechanism via learning task-specific prompts to perform numerous downstream tasks conditioned on a frozen language model. Given input tokens $x$ and output tokens $y$, the language model assigns the output probability $\Pr_{\xi}(y|x)$, where $\xi$ represents the language model parameters. Prompting refers to adding additional information to the model during its text generation or classification, usually done by adding a series of tokens $p$ to the model input. The model objective now becomes maximizing the probability $\Pr_{\xi}(y|[p;x])$. Instead of having a frozen language model (frozen $\xi$) and carefully designing prompt tokens $p$, prompt tuning assigns tunable parameters $\xi_p$ to automatically chosen prompt tokens $p$, and updates $\xi_p$ via backpropagation. The model is trained to maximize $\Pr_{\xi;\xi_p}(y|[p;x])$, where $\xi$ is frozen and $\xi_p$ is updated.

Similar to prompt tuning [153] which learns task-specific prompt embeddings conditioned on a frozen language model, we update the speaker-specific free parameters $\theta_P$ conditioned on the frozen *informative subnetwork* (frozen $\theta_{UP}$) defined in Section 3.2.2. Prompt tuning can perform multiple downstream tasks concurrently, likewise our multi-speaker adaptation method can support multiple speakers at the same time. The difference to prompt tuning is that prompt tokens $p$ are prepended to the input tokens $x$ and fed to the input layer of the model in prompt tuning, while our approach specifies the number of target speakers and distributes among the free parameters in the entire model encoder, in the sense that the prompts are fed to each layer of the model encoder.

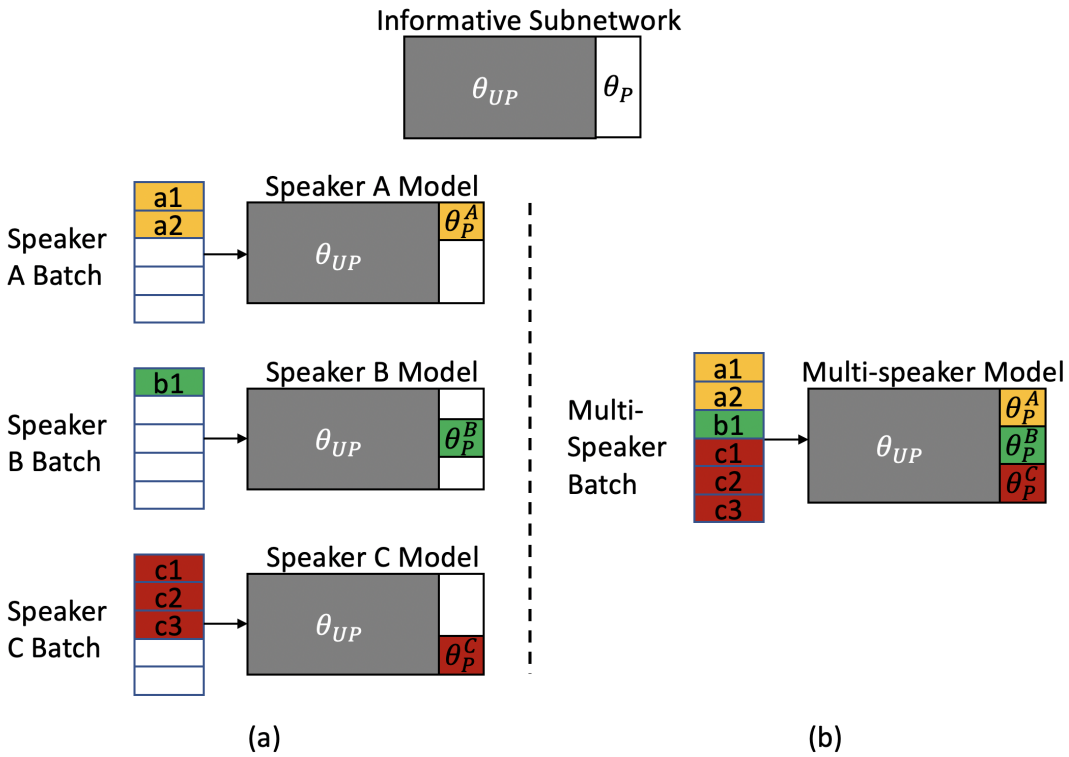



Fig. 3.4: Comparison of whether zeroing out other speakers' free parameters when training the current speaker's data. (a) shows the result of zeroing out these parameters (shown by the blank space), which is essentially the same as training a separate model for each speaker. (b) represents a multi-speaker model adapted for speakers A, B and C.

There are two design decisions to be made. One consideration is regarding the assignment of free parameters $\theta_P$ among multiple speakers. We can assign the parameters iteratively in sequence by following the train prune cycle, i.e., assign all free parameters to the first speaker and train the model, then prune the redundant parameters, freeze the unpruned ones and assign the rest free parameters to the second speaker, and so on and so forth until all speakers are finished. However, the computation cost and time of this method are proportional to the number of target speakers. Besides, the sequence of speakers to be adapted plays a role in affecting how each speaker is adapted. Therefore, we adopt a simple yet efficient approach to distribute the free parameters $\theta_P$ among all target speakers equally, randomly, and simultaneously ($\theta_P = \{\theta_P^1, \theta_P^2, ..., \theta_P^\kappa\}$ for $\kappa$ speakers), as represented in Fig. 3.3(d). This basically ensures fairness among all target speakers and one does not overwhelm another.

Another design consideration is that during adaptation of the speaker at hand, how to deal with the free parameters assigned to other target speakers. If we choose to zero out those parameters, this is more like keeping a separate copy of the model for each target speaker (as seen in Figure 3.4(a)), which defeats the purpose of developing a multi-speaker model. Hence, we decide to freeze other speakers' free parameters when training the current speaker's data, and it gives us a real multi-speaker model (Figure 3.4(b)).

Given a target multi-speaker dataset $\left\{\mathrm{D}_t^k\right\}_{k=1}^{\kappa}$ with $\kappa$ number of speakers, where each speaker $k$ has $L_t^k$ pairs of input speech and output text ($\mathrm{D}_t^k = \{\mathrm{Y}^j_{\mathrm{D}_t^k}, \mathrm{Z}^j_{\mathrm{D}_t^k}\}_{j=1}^{L_t^k}$), multi-speaker adaptation is carried out by re-training the model with the following objective:

$$\mathcal{J}(\theta_P) = \sum_{k=1}^{\kappa} \sum_{j=1}^{L_t^k} \log \Pr(\mathrm{Z}^j_{\mathrm{D}_t^k} | \mathrm{Y}^j_{\mathrm{D}_t^k}; \theta_{UP}, \theta_P), \tag{3.10}$$

where $\theta_{UP}$ is the same as that in Eq. (3.9) and is frozen, $\theta_P = \left\{\theta_P^1, \theta_P^2, ..., \theta_P^\kappa\right\}$ for $\kappa$ speakers and is updated.

## 3.3 Experiments

### 3.3.1 Datasets

We evaluate the effectiveness of the proposed speaker adaptation approaches on the open-source LibriSpeech dataset [39]. LibriSpeech consists of 16kHz read English speech from audiobooks. We use the given train/development/test splits of the LibriSpeech dataset. Test_clean data is clean and Test_other data has noise in speech. See Table 3.1 for the statistics of the LibriSpeech dataset used in our experiments. We use Espnet toolkit [45] for the experiments.

### 3.3.2 Training Setup

#### 3.3.2.1 Inputs

We use 83-dimensional features, including 80-dimensional filter banks, pitch, deltapitch and Normalized Cross-Correlation Functions (NCCFs). Input features are generated with

Table 3.1: Statistics of the LibriSpeech Dataset Used for Experiments

| **LibriSpeech** | |
|---|---|
| Training Set | 100h (251 speakers) |
| Dev_clean Set | 5.4h (20 males, 20 females) |
| Dev_other Set | 5.3h (17 males, 16 females) |
| Test_clean Set | 5.4h (20 males, 20 females) |
| Test_other Set | 5.1h (16 males, 17 females) |

a window size of 25ms shifted every 10ms. The acoustic features are mean and variance normalized. We exclude utterances longer than 3000 frames or 400 characters to keep memory manageable. Speed perturbation [71] is applied with factor 0.9, 1.0 and 1.1 to improve the model robustness.

#### 3.3.2.2 Outputs

For the model output, we use the unigram sub-word algorithm [154] with the vocabulary size capped to be 5000.

#### 3.3.2.3 Speech Transformer

For the speech transformer architecture as introduced in Section 2.2.4, the convolutional frontend before transformer encoder is two 2D convolutional neural network layers with filter size (3,2) and stride 2, each followed by a ReLU activation. Both CNN-block have 256 filter kernels for each CNN layer. For the multi-head attention network and the feedforward network in both encoder and decoder following the CNN-blocks, the attention dimension $d_{model}$ is 256, the feedforward network hidden state dimension $d_{ff}$ is 2048, the number of attention heads $h$ is 4, with $d_k = d_q = d_v = 64$ for each head, the number of encoder layers $N_e$ is 12, the number of decoder layers $N_d$ is 6. The total number of model parameters is 31 million. We train the model for 100 epochs ($n = 100$ in Fig. 3.3(b)). The training and decoding parameters are listed in Table 3.2.

Table 3.2: Experimental Hyperparameters for Speech Transformer

| **Parameter** | |
|---|---|
| Batch Size | 12 |
| CTC Weight for Training | 0.3 |
| Dropout Rate | 0.1 |
| Gradient Clipping Norm | 5.0 |
| Learning Rate | 5.0 |
| Warmup Steps | 25000 |
| CTC Weight for Decoding | 0.4 |
| Decoding Beam Size | 60 |
| Language Model Weight | 0.6 |

#### 3.3.2.4 External Language Model

For the external language model, we adopt the transformer architecture as well. The attention dimension is 512, the feedforward network hidden state dimension is 2048, the number of attention heads is 8, the number of layers is 16. We train the model for 50 epochs with batch size being 32. The text corpus contains the LibriSpeech training transcripts and 14500 public domain books [39]. The language model is trained separately from the ASR model.

#### 3.3.2.5 I-vector

For i-vector generation, we follow the SRE08 recipe in Kaldi [44] toolkit on the training data. I-vectors extracted are of dimension 100. They are then transformed to have the same dimension as speech vectors for concatenation.

Table 3.3: State-of-the-art Results of Different Speaker Adaptation Algorithms on LibriSpeech Test Data

| **Model** | **Test_clean** | **Test_other** |
|---|---|---|
| Baseline | 9.2 | 21.9 |
| DFSMN-SAN [14] | 8.9 | 21.6 |
| SAST [94] | 8.9 | 21.4 |
| Our method | 8.6 | 21.3 |

### 3.3.3 General Speaker Adaptation

We first test on the adaptation for general speakers without knowing the target speaker profile. Speaker-aware persistent memory model introduced in Section 3.2.1 achieves this objective. 64 speaker i-vectors are randomly sampled from the training data and are concatenated to speech vectors on all the encoder layers in the speech transformer. 64 i-vectors are tested to be a good choice to provide diverse speaker information in Section 3.4.1, and ablation study in Section 3.4.2 shows that applying on all the encoder layers helps capture speaker knowledge from both low-level phonetic features and high-level global information. Table 3.3 shows that our method brings 2.74-6.52% relative improvement over the baseline, and surpasses [94] who also makes use of speaker i-vectors. Furthermore, here we also compare our model with the first persistent memory model used in ASR [14], in which persistent memory vectors are randomly initialized and meant to capture general knowledge. Different from them, our model is to address the speaker mismatch issue. Our method achieves the best results.

### 3.3.4 Specific Target Speaker Adaptation

If the target speaker profile is known beforehand, the gradual pruning method discussed in Section 3.2.2 could adapt to the target speaker. Directly finetuning the entire model takes high computation resources by updating all the model parameters, and could overfit easily if the amount of target speaker data is limited. We are interested to see the

performance of the gradual pruning method especially on low-resource data, as well as how much it alleviates the catastrophic forgetting problem. Therefore, we randomly choose a speaker from the LibriSpeech Test_other data as the target speaker, and only select 10 utterances of the target speaker as adaptation data. The remaining utterances of the target speaker are chosen as the test data. We do this four times and report the average performance to see the generalizability of the proposed approach. The average baseline WER of four speakers is 20.5, and is slightly smaller than the average WER of Test_other speakers, which is 21.9, so further improving the target speaker performance is a bit more challenging. The pruning rate is set as 10% here. We compare the performance of 1) Finetune: directly finetuning the entire model as Eq. (3.8), 2) I-vec: speaker-aware persistent memory method by adding i-vectors, 3) Pruning: gradual pruning, 4) Pruning+I-vec: combining the feature adaptation and model adaptation methods proposed.

For results on the target speaker in Figure 3.5(a), finetuning works better than the baseline. Adding i-vectors has the highest WER initially and the performance is worse than simply finetuning the trained model after 20 epochs. We believe speaker-aware persistent memory method works better on general speaker adaptation given that the sampled i-vectors form the speaker space to capture any speaker knowledge. It is not designed to adapt to some specific speakers. Using the gradual pruning method alone has lower WER than finetuning at the initial stage, but surprisingly it overfits more than the finetuning method after 20 epochs. Lastly, we combine the feature adaptation and model adaptation methods, and it achieves our best result. It outperforms the baseline with up to 20.58% relative WER reduction, and surpasses the finetuning method by up to relative 2.54%. We see that the feature adaptation method and the model adaptation method we propose complement each other, as the combined model result surpasses each individual one.

We want to analyze the performance of the rest non-target speaker data to see if catastrophic forgetting happens. From Figure 3.5(b), all target speaker adapted models perform slightly worse than the baseline, which is expected. Combining feature adaptation and model adaptation could alleviate catastrophic forgetting problem effectively. It generally outperforms finetuning in Figure 3.5(b).

Table 3.4: Baseline and Multi-speaker Adaptation Results of Four Target Speakers in WER

| | **Baseline** | **Multi-speaker Adaptation** |
|---|---|---|
| Speaker 1 | 19.9 | 18.5 |
| Speaker 2 | 13.0 | 14.0 |
| Speaker 3 | 34.9 | 29.6 |
| Speaker 4 | 14.3 | 16.0 |
| Average | 20.53 | 19.53 |

### 3.3.5 Multi-speaker Adaptation

We use the four speakers sampled in Section 3.3.4 to examine multi-speaker adaptation. The average performance is reported above, while each individual speaker performance is drawn in Fig. 3.6. Given the baseline of each speaker in Table 3.4, the unified speaker adaptation approach is very effective in target speaker adaptation. From the trendlines in Figure 3.6, the performance is relatively stable with increasing training epoch on the low-resource data (10 utterances), and it exemplifies the effectiveness of our method in reducing the overfitting issue.

Next, we test the multi-speaker adaptation technique discussed in Section 3.2.3. Since the same model is catered for multiple speakers, and the same amount of free parameters $\theta_P$ are now distributed among them, each target speaker gets less model capacity for adaptation in a sense, and it would need less training epoch before the overfitting signal appears. We verified this experimentally as well. Compared with the optimal results at training epoch 15 in Figure 3.6, we only train the multi-speaker data for 5 epochs. The results are listed in Table 3.4. Our multi-speaker adaptation technique is efficacious and outperforms the baseline with 4.87% relative improvement. On top of this, we have two other findings. First, the performance of each speaker in the multi-speaker adaptation model is inferior to that of uni-speaker adaptation in Figure 3.6, which is expected. As analyzed above, the same model is adapted for multiple speakers concurrently, so

each speaker gets less model capacity consequently. Second, we notice that the target speakers actually affect each other during training, and improving on one speaker will in turn brings worse performance to other speakers. It again confirms the difficulty and complexity of developing a multi-speaker adapted model. Overall, our proposed multi-speaker adaptation approach is effective.

## 3.4 Analysis

### 3.4.1 Number of Speaker I-vectors $N$

We investigate the influence of the number of speaker i-vectors $N$ in the speaker space. Less speaker i-vectors brings less speaker-related knowledge, and they may not be able to represent speaker space well. However, as tested by Fan et al. [94], more speaker i-vectors does not necessarily improve the model performance, or may even deteriorate the results. Fan [94] claims that the model could easily identify the speaker i-vector in the training data when more speaker i-vectors are provided, and it in turn hurts the model's learning ability to combine basic i-vectors.

Different numbers of speakers are randomly selected from the training data, ranging from 16 to 251 (all speakers). Speaker aware persistent memory is applied on all encoder layers. It can be seen from Fig. 3.7 that the performance improves with the number of i-vectors in the speaker space increasing from 16 to 64. This exemplifies the effectiveness of speaker aware persistent memory, that introducing speaker i-vector helps capture speaker knowledge. Furthermore, adding more speaker i-vectors gives more diverse speaker information. Afterwards, increasing the number of i-vectors does not affect much. We believe this is because 64 i-vectors are good enough to capture enough speaker variation. Thus, increasing the number of i-vectors will neither add value further nor deteriorate the performance.

### 3.4.2 Number of Layers Applied with Speaker Aware Persistent Memory

We take a look at the encoder layers to which speaker aware persistent memory is applied. For these layers, there is a component of speaker knowledge in the hidden layer vectors.

Table 3.5: Speaker Aware Persistent Memory Results (WER) with 64 Speaker I-vectors Applied on Different Layers

| **Encoder Layers Applied** | **Test_clean** | **Test_other** |
|---|---|---|
| Baseline | 12.0 | 29.7 |
| Lower Half (1-6) | 11.2 | 28.6 |
| Higher Half (7-12) | 10.9 | 28.1 |
| All (1-12) | 10.5 | 28.3 |

The results are summarized in Table 3.5. When we apply speaker aware persistent memory on all the encoder layers (layer 1-12), it achieves the best result on the clean test set (Test_clean), and overall it brings relative 4.7%-12.5% WER reductions. Applying it on higher encoder layers (layer 7-12) performs better than applying it on lower encoder layers (layer 1-6). Compared with lower layers, encoder output from higher layers has more abstract and global information. In contrast, encoder output from lower layers comes from raw features and has more phonetic information. Thus, introducing i-vector will be of less use in the lower encoder layers. We notice that the improvement on the noisy test set (Test_other) is moderate only. This is mainly because our 100h LibriSpeech training data is all clean data. The mismatch of training and test data deteriorates the performance.

### 3.4.3 Learnable Attention Score between Utterances and Persistent Memory

We would like to verify if our proposed speaker aware persistent memory really captures speaker knowledge using LibriSpeech dataset. WER reduction on all datasets could not prove or disprove from this perspective. Since we use the fixed speaker space, we plot the learnable attention scores between an utterance and persistent memory vector $\boldsymbol{M}_k$, which is the learned transformation of 64 speaker i-vectors, after softmax computation from Eq. (3.5), averaged over all time steps of the utterance. The plot on top of Figure 3.8 shows attention scores of ten different utterances from the same speaker. Given

the ten different utterances with the only common ground of coming from the same speaker, they show similar pattern of attention scores, which clearly shows that certain speaker knowledge is captured. On the other hand, in the middle of the figure, the attention scores of ten random utterances from ten different speakers are quite random. The dataset does not provide one common utterance from different speakers, so we have to select random utterances. We take average attention scores along the entire utterance, which will mitigate acoustic variation among different utterances to some extent. Here we also present the attention scores of ten utterances from the same speaker in a transformer model using randomly initialized $M_k$ as in [14] at the bottom of the figure, which is meant to capture general knowledge. From the irregular plot, it does not seem to capture speaker-related knowledge. From here, we conclude that speaker aware persistent memory captures speaker knowledge.

### 3.4.4 Pruning Rate

We test different pruning rates of the gradual pruning approach. Results are shown in Figure 3.9. Less pruning rate keeps more parameters for the general speaker data, and has less learning capability to target speaker. It is more suitable for simple adaptation tasks. Higher pruning rate generates a more sparse network and is more flexible for speaker adaptation, except that it retains less original model parameters, thus forgets more on the general speaker data. It can be seen from Figure 3.9 that pruning 10% of encoder parameters achieves the best result.

### 3.4.5 Pruning Time During Model Training

We use the gradual pruning method to prune to target sparsity for every 10k training steps. One-time pruning at the initial/middle/final stage of the overall training is tested for comparison as well. We train for 100 epochs, and initial/middle/final stage pruning is done at 0/50/100 epoch respectively. Gradual pruning and one-time pruning will reach the same sparsity level after the training. Here we use either gradual or one-time pruning at different stages during training, and show the best results of finetuning for 15 epochs. Table 3.6 shows that gradual pruning works better than one-time pruning, be it initial,

Table 3.6: WER Results of Gradual Pruning versus One-time Pruning

| **Model** | **Target Speaker** |
|---|---|
| Baseline | 19.9 |
| One-time Pruning at Initial Stage | 16.2 |
| One-time Pruning at Middle Stage | 17.6 |
| One-time Pruning at Final Stage | 16.0 |
| Gradual Pruning | 15.9 |

Table 3.7: Characteristics of Utterances Selected as the Extremely Low-resource Adaptation Data

| **Utterance(s)** | **1** | **5** | **10** |
|---|---|---|---|
| Total No. of Words | 32 | 104 | 174 |
| Total Duration (s) | 15.00 | 40.00 | 67.17 |

middle or final stage of the training. Compared with one-time pruning, gradual pruning could learn and prune at the same time. In particular, gradual pruning follows the train prune cycle, and is capable of iteratively learning the unpruned parameters after less contributing parameters are pruned. For the one-time pruning, pruning at an earlier stage has the advantage to let the model learn the unpruned parameters based on the pruned ones in the remaining training of the model, but pruning earlier has the risk to prune important parameters since the model is not well learnt yet, vice versa for pruning late. Hence, gradual pruning works the best.

### 3.4.6 Extremely Low-resource Adaptation Data

Lastly, we would like to see the extremely low-resource adaptation data scenarios. We reduce the amount of adaptation data and compare the performance with the baseline, where no adaptation is performed. The characteristics of the adaptation data selected are listed in Table 3.7. From Figure 3.10, when the amount of adaptation data is reduced

from 10 utterances to 5 utterances, the results are similar to that of 10 utterances at the initial training stage, and could outperform the baseline by up to relative 18.59%. With less adaptation data, the model overfits much faster, especially in the case of having only 1 utterance for adaptation. However, even with only 1 utterance, it could surpass the baseline by up to relative 6.53% with only 5 epochs of training. Therefore, even with extremely low-resource adaptation data such as 1 utterance, our method is effective with fast adaptation.

## 3.5 Chapter Summary

This work proposes a unified speaker adaptation approach consisting of feature adaptation and model adaptation. Speaker-aware persistent memory model makes use of speaker i-vectors to adapt at the feature level, and we use the gradual pruning approach to retrieve a subset of model parameters for adaptation at the model level. We further refine our method for multi-speaker adaptation scenario by distributing the pruned free parameters among target speakers equally, randomly, and simultaneously. We find that our proposed method is effective in terms of speech recognition accuracy in different speaker adaptation scenarios. Our method could effectively address the problem of having unknown speakers in the test data, and it learns utterance level speaker knowledge compared with simply appending the attention with speaker i-vectors. Compared with previous model adaptation approaches, our method saves the trouble of intuitively adapting a certain subset of model parameters to prevent overfitting. It can alleviate catastrophic forgetting problem as well by retaining the informative subnetwork. In the low-resource speaker adaptation aspect, the proposed method gives a great performance.


Fig. 3.5: WER results of target speaker (a) and non-target speakers (b). Finetune: directly finetuning the entire model as Eq. (3.8). I-vec: speaker-aware persistent memory method proposed in Section 3.2.1 by adding i-vectors. Pruning: gradual pruning proposed in Section 3.2.2. Pruning+I-vec: combining the feature adaptation and model adaptation methods proposed. The dotted lines are the second order polynomial trendlines.

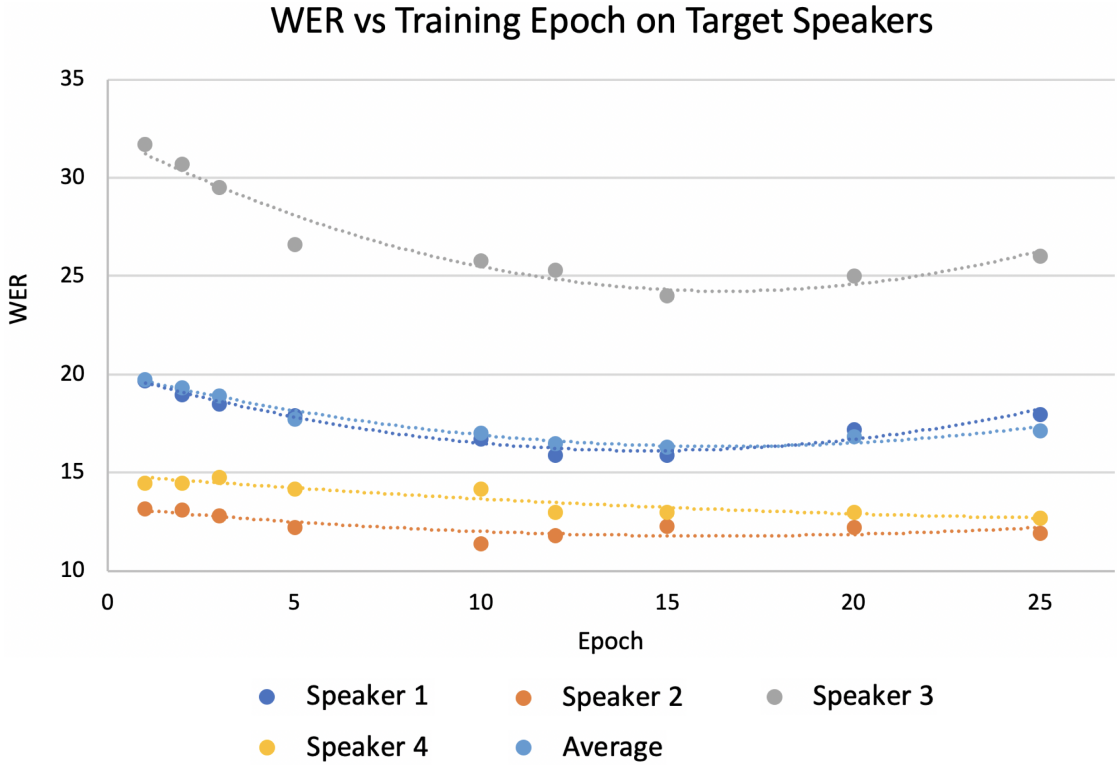



Fig. 3.6: WER results of target speakers versus training epoch. The dotted lines are the second order polynomial trendlines.

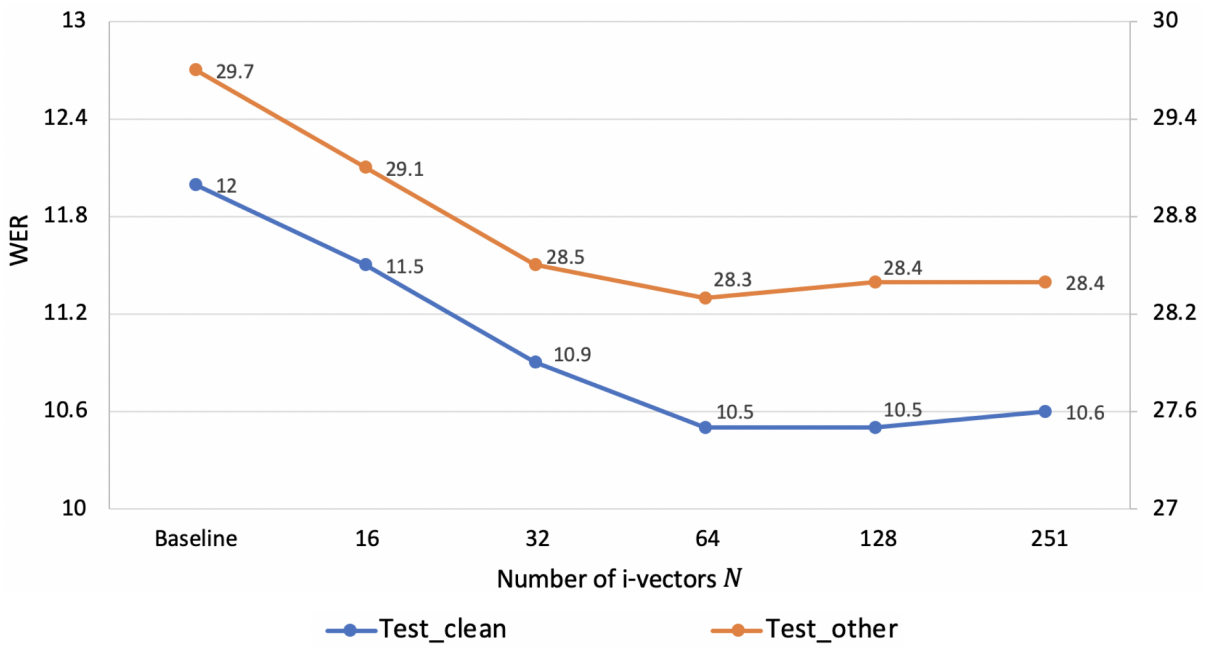



Fig. 3.7: Speaker aware persistent memory results (WER) applied on all the encoder layers for different number of speaker i-vectors.

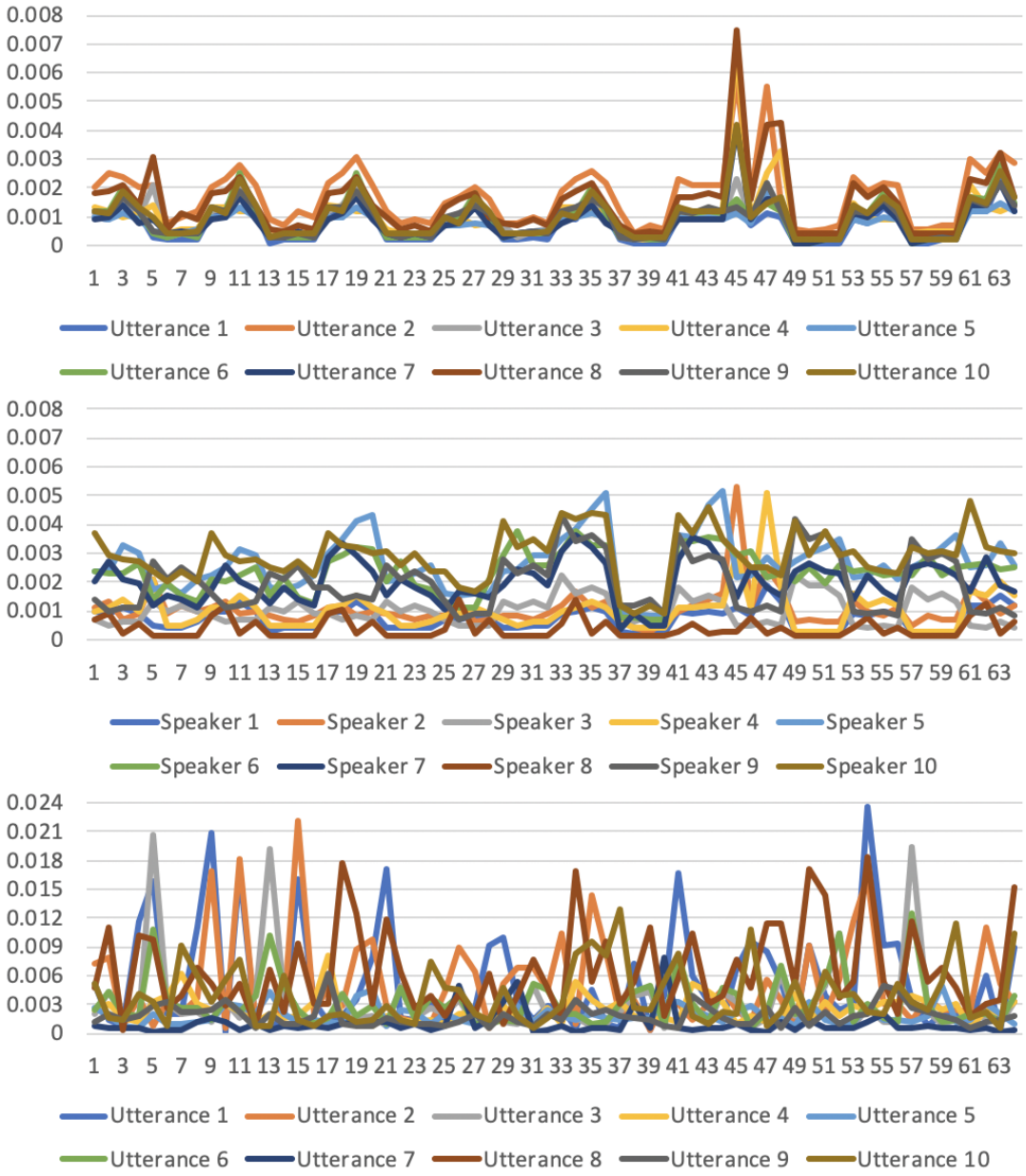



Fig. 3.8: Attention scores between ten utterances and persistent memory vector $M_k$. Top: All utterances from same speaker and $M_k$ is learned transformation of 64 speaker i-vectors. Middle: All utterances from different speakers and $M_k$ is learned transformation of 64 speaker i-vectors. Bottom: All utterances from same speaker and $M_k$ contains 64 randomly initialized vectors [14]. Attention scores shown are after softmax computation and averaged over all time steps of each utterance.

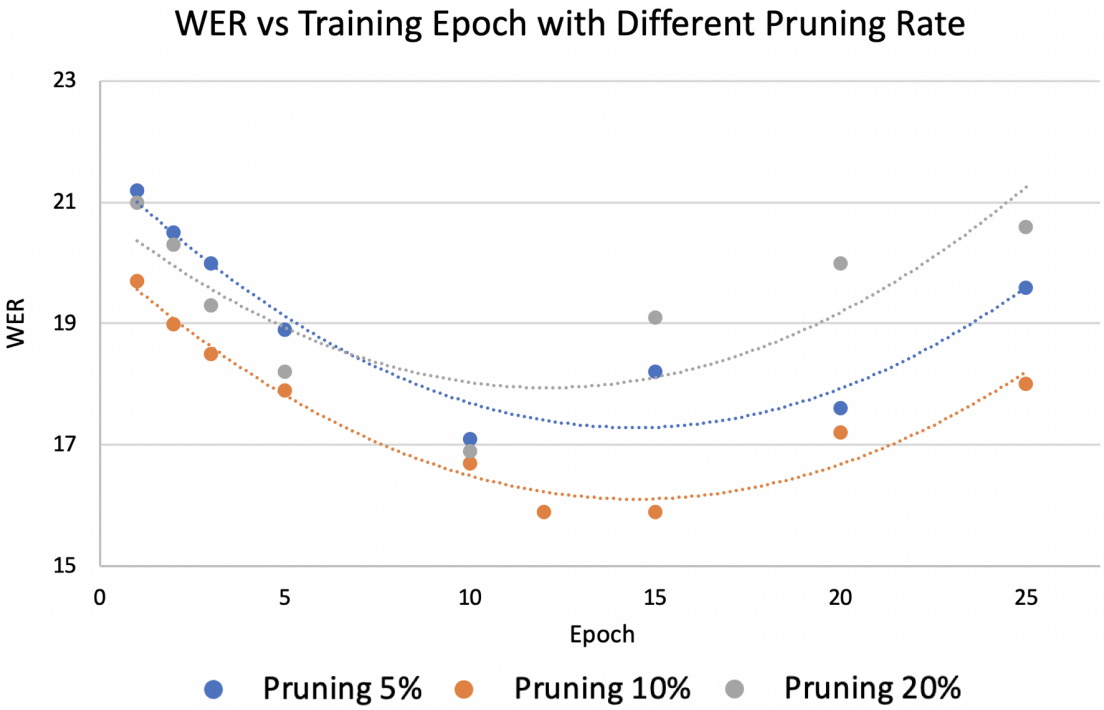



Fig. 3.9: WER results of target speaker with different pruning rates. The dotted lines are the second order polynomial trendlines.

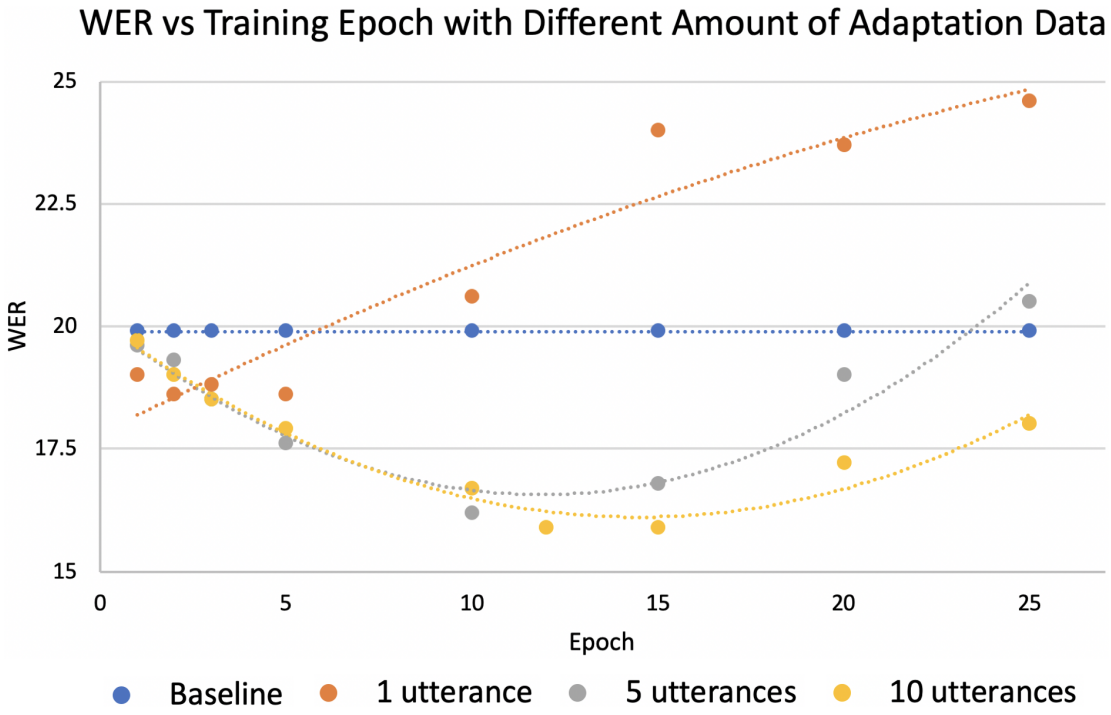



Fig. 3.10: WER results of target speaker with different amount of adaptation data. The dotted lines are the second order polynomial trendlines.

# Chapter 4

# Cross Attention with Monotonic Alignment

In this chapter, we aim to further improve the transformer model for ASR from the monotonic alignment perspective. Monotonic alignment between text output and speech input is a unique feature of ASR, unlike all other tasks like machine translation or summarization where the input and output are not matched in time dimension. Therefore, utilizing this feature and modifying the transformer architecture accordingly is a promising direction, as transformer was proposed to take all the contexts of input.

The approach taken in this chapter is motivated by the self-attentional acoustic models [155], which applies attention biasing in the encoder self-attention network to account for the context locality. Instead of applying on the acoustic encoder, our idea focuses on the cross attention module to consider the text-speech alignment. The attention biasing part is the same as the self-attentional acoustic models [155] to limit the attention range locally. However, one remarkable difference is that for cross attention module, locating the alignment position between text and speech is necessary and critical. Comparatively, for the self-attention network in the encoder, the attention biasing is centered at the input frame itself to capture local information. Imprecise alignment between text and speech could lead to inaccurate encoder information.

This chapter is organized as follows. Section 4.1 first provides the motivation for taking advantage of monotonic alignment. Some previous studies in this direction are also introduced. Section 4.2 then gives a full picture of our proposed algorithm. Experiments

are carried out in Section 4.3 to evaluate the performance of our approach as well as some analysis. Section 4.4 concludes this chapter.

# 4.1 Introduction

## 4.1.1 Motivation

Chapter 2 reviewed the model architecture of the transformer. In Figure 2.6, transformer model relies on self-attention network and cross attention network to capture direct pairwise relationships in the respective contexts. Its cross attention from decoder hidden states to encoder hidden states is the same as the cross attention in the long short-term memory (LSTM) based encoder-decoder model, i.e. attending to the entire speech utterances and obtain corresponding attention weights for decoding (Eq. (2.30)). This model architecture can disperse the attention distribution over the entire input, which is important for some sequence-to-sequence tasks, such as neural machine translation (NMT). However, when it comes to ASR, the same architecture may not work well, as monotonic alignment between text output and speech input is a characteristic of ASR. Dispersing the attention to input speech parts which are less relevant to current decoder output could lead to wrong encoder information and wasted attention weights.

This finding motivates us to utilize the monotonic alignment feature and modifying the transformer architecture accordingly. We review some prior works along this direction first and analyzing their drawbacks.

## 4.1.2 Prior Works

Several studies make use of the monotonic alignment feature. CTC [61–63] and its extensions (RNN-T [11], recurrent neural aligner (RNA) [156]) used monotonic alignment position to locate the local encoder representations for current token prediction, as introduced in Section 2.2.1. [15] proposed using the CTC output as alignment reference by getting both the first position of consecutive outputs and any non-consecutive output position, as illustrated in Figure 4.1. This is because of the compression rule of CTC output, which is explained in Section 2.2.1. However, their alignment depends heavily

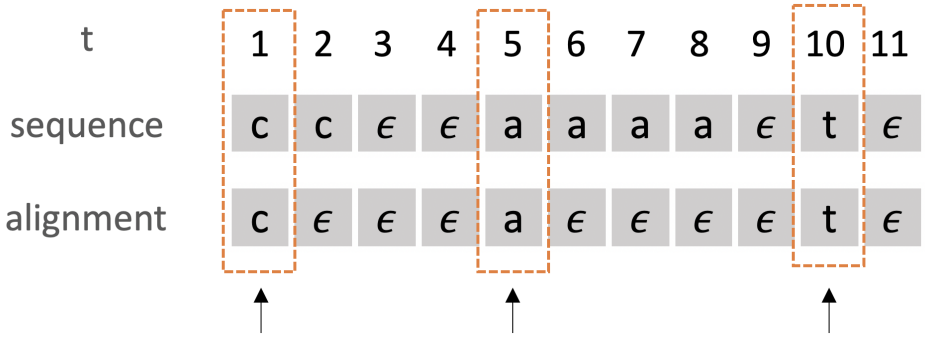



Fig. 4.1: Alignment obtained from CTC output in [15].

on CTC output accuracy, and the result cannot compete with state-of-the-art speech recognition results.

The recently proposed CIF model [157] integrated along input speech frames and triggered an output once an alignment boundary is found. They calculate a set of weights which indicate the amount of information contained in each hidden state. Afterwards, the weights are summed up along the time dimension, and a threshold is set to specify an alignment boundary. If the sum exceeds the boundary, current weight is split into two parts, where the left part belongs to previous label, and the right part is taken as the next label (Figure 4.2). They also introduce a quantity loss to rescale the weight based on the number of labels, so as to have a correct number of labels split by the boundary. However, their soft and monotonic alignment is affected by background noise greatly, since noise term will influence the integration and alignment boundary location, affecting the final accuracy.

### 4.1.3 This Work

In order to achieve better alignments between output and input for speech recognition under the sequence-to-sequence framework, here we propose a straightforward and effective cross attention biasing method for the transformer model that takes output-input alignments into consideration. The reason why we propose this approach is because our method does not reference CTC output, which is highly dependent on CTC output accuracy. Furthermore, our method does not add additional parameters on encoder hidden states.

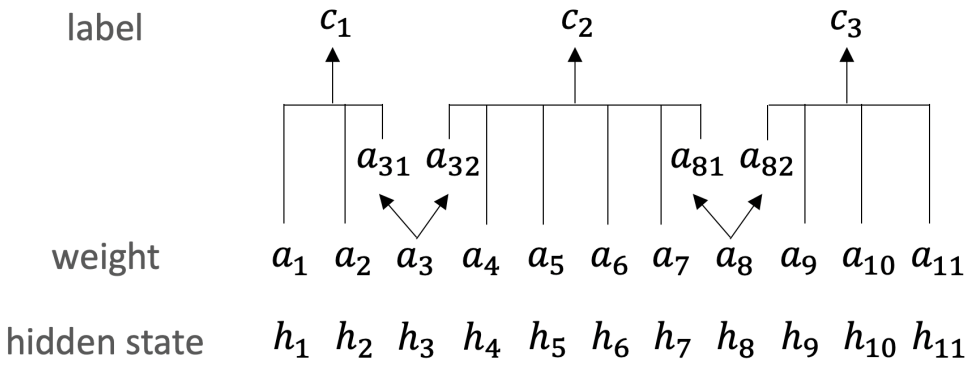



Fig. 4.2: Continuous integrate and fire to locate the label boundary.

The approach taken in this chapter is motivated by the self-attentional acoustic models [155], which applies attention biasing in the encoder self-attention network to account for the context locality. Instead of applying on the acoustic encoder, our idea focuses on the cross attention module to consider the text-speech alignment. We take advantage of cross attention weights as a reference of output-input alignment to be used in current cross attention computation. In particular, we apply a Gaussian mask on attention weights centered at the alignment position. Additionally, we introduce a regularizer which regularizes alignment between output and input to encourage monotonicity. Since lower layers of the Transformer capture more acoustic and local information [158], we apply our cross attention biasing on lower layers of the Transformer model, and leave the cross attention at higher layers to attend to entire speech input to capture global information. Our results on LibriSpeech 100h dataset show that our proposed model yields 14.5%-25.0% relative word error rate (WER) reductions.

## 4.2 Model Architecture

### 4.2.1 Cross Attention with Alignment

Our proposed transformer model with modified cross attention that considers monotonicity between text output and speech input is introduced in this section. As explained earlier, the alignment information is beneficial for the decoder to attend to the corresponding input speech frames for decoding. This is more beneficial for the decoder at

lower layers, where the model captures more local than global information [158]. In this way, the decoder could focus more on aligned speech input.

#### 4.2.1.1 Alignment

Our objective is to find alignment between text output and speech input. Inspired by [159], which used recurrent neural network and proposed location-based attention mechanism computed by previous attention weights, we take advantage of current cross attention weights to locate the alignment position in the input and apply it on the transformer model. Given that a speech input and its corresponding text transcript should be most similar in the embedding space, we propose to take the position with the maximum cross attention weight as the input alignment position for the current decoder input, as the cross attention weight is an indication of the closeness between text output and speech input.

For any text hidden vector $y_i$ in the decoder, the cross attention weight $\alpha_{ij}$ to the encoder final output $x_j$ in $X = (x_1, ..., x_n)$ is:

$$\alpha_{ij} = \text{softmax}(\frac{(y_i W^Q)(x_j W^K)^T}{\sqrt{d_k}}), \tag{4.1}$$

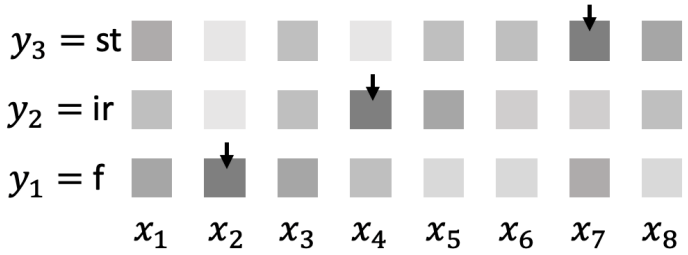
where $W^Q, W^K \in \mathbb{R}^{d_{model} \times d_k}$ are the linear transformation matrices. Since we take the position with the maximum cross attention weight as the alignment position, $x_k$ aligns with $y_i$ if:

$$\alpha_{ik} \geqslant \alpha_{ij} \quad \forall j \in (1, n), j \neq k, \tag{4.2}$$

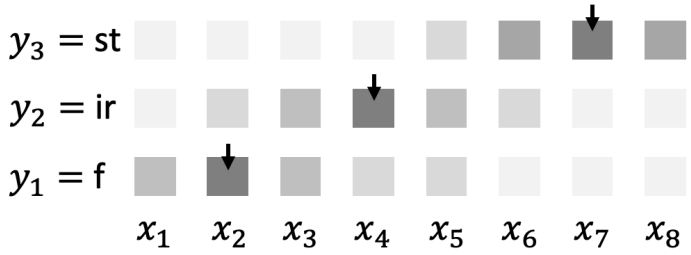
$i$ and $k$ are the aligned output and input positions respectively, which will be used for cross attention computation.

#### 4.2.1.2 Cross Attention Biasing

Attention biasing was proposed in [155] for improving the performance of self-attention acoustic model. Different from it, we propose to use attention biasing on the relevant part of encoder output under the cross attention framework. In particular, we add a Gaussian mask to the attention weights centered at the alignment position of encoder output which matches current decoder input most, which we name as soft attention




(a) Before soft attention biasing




(b) After soft attention biasing

Fig. 4.3: Cross attention weight diagram before and after applying soft attention biasing. $X = (x_1, ..., x_8)$ are encoder final layer output vectors. Text outputs consist of sub-words "f ir st". Grey shade represents the cross attention score between each encoder and decoder vector. Darker shade means higher score value. Squares that arrows point to represent the aligned positions identified using Eq. (4.2).

biasing. The mask keeps the attention weight at the alignment position, and the weight is gradually diminished when moving away from the position. In this way, the cross attention could focus more on the local aligned speech section. The attention weight is defined as:

$$\alpha_{ij} = \text{softmax}(\frac{(y_i W^Q)(x_j W^K)^T}{\sqrt{d_k}} + M_{ij}), \tag{4.3}$$

where $M_{ij}$ is the Gaussian attention mask defined in the following way:

$$M_{ij} = \frac{-(j-k)^2}{2\sigma^2}, \tag{4.4}$$

where $y_i$ aligns with $x_k$, and $\sigma^2$ is a Gaussian variance such that the aligned position has more attention weight value. Figure 4.3 shows the cross attention weights between text sub-words and encoder final layer output vectors before and after applying soft attention biasing.

Besides the soft attention biasing, we also try to mask all the attention weights behind current alignment position while keeping the left hand side attention weights, which we call it as hard attention biasing. The hard attention biasing could maintain the autoregressiveness in the model, and is the typical masking applied in the standard transformer decoder. We will compare these two attention biasing techniques in our experiments.

#### 4.2.1.3 Algorithmic Details

Firstly, we *add* $n$ *look-ahead frames* after the alignment position. This is motivated by the benefit of adding some look-ahead frames after the alignment position for location-aware attention in [15]. Adding some look-ahead frames has the effect to include some future information, which was found to be beneficial. Eq. (4.4) in this case becomes:

$$M_{ij} = \frac{-(j - (k + n))^2}{2\sigma^2}. \tag{4.5}$$

Secondly, we *apply cross attention biasing at lower layers of the decoder* as Eq. (4.3) to capture local information. We define lower half of the decoder layers as *lower layers*, and upper half of the decoder layers as *higher layers*. This is motivated by [158] that the model captures more local than global information at lower layers. With the cross attention biasing, the cross attention could focus more on the local context at lower layers. At higher layers where more global information is learned by the model, we remove the bias term and follow original cross attention model as Eq. (4.1) to attend to larger range of speech frames. In this way, our proposed method could fuse the local and global span of information, where the lower layers capture local context and the higher layers capture global information. This is different from the hybrid global and local attention proposed in [160]. [160] transforms each hidden state into a gating scalar to indicate the ratio between global and local attention. Thereafter, a weighted sum of global and local attention is taken in the self-attention network.

### 4.2.2 Monotonic Alignment Regularization

Alignment positions between the output and input should be strictly monotonic in the input sequence for speech recognition. In order to regularize alignment using the technique

above, we propose a regularizer term to achieve monotonicity. For an encoder output $X = (x_1, ..., x_n)$ and a text transcription embedding from the decoder $Y = (y_1, ..., y_m)$, we define the following misalignment loss as the regularization term:

$$loss_{misalign} = \sum_{l=1}^{m} \sigma(k_l - k_{l+1}), \tag{4.6}$$

where $\sigma$ is the sigmoid function and $k_l$ is the position in $X$ that $y_l$ aligns to. In other words, we want the alignment of $y_l$ to be in front (before) of the alignment of $y_{l+1}$ in $X$. This is a local regularization as we only compare every two consecutive alignment positions and penalize if there is a misalignment.

Under the CTC and attention hybrid multi-task learning framework, the proposed training criterion for improving the monotonic alignment between text output and speech input becomes:

$$loss = \alpha * loss_{ctc} + (1 - \alpha) * loss_{att} + loss_{misalign}, \tag{4.7}$$

$$loss_{ctc} = -\ln P(y|x), \tag{4.8}$$

$$loss_{att} = \sum_{u} \ln P(y_u|x, y_{1:u-1}), \tag{4.9}$$

where $\alpha$ is the ratio of CTC model loss in hybrid model.

## 4.3 Experiments

### 4.3.1 Experimental Setup

We use ESPnet speech recognition toolkit and LibriSpeech corpus for our experiments [39, 45]. The training dataset is 100 hour clean training data uttered by 251 speakers, and the development and test dataset are the default LibriSpeech development and test dataset, where each is around 5 hours and contains 2600 to 3000 utterances. LibriSpeech consists of 16kHz read English speech from audiobooks [39]. Input features are generated by 80-dimensional filterbanks with pitch on each frame, with a window size of 25ms shifted every 10ms. We remove utterances with more than 3000 frames or 400 characters to avoid the excessive use of memory. We adopt the CTC and AED joint decoding [35], where

Table 4.1: WER results on LibriSpeech 100h

| **Model** | **Test_clean** | **Test_other** |
|---|---|---|
| Baseline transformer | 12.0 | 29.7 |
| Encoder-Decoder-Attention [161] | 14.7 | 40.8 |
| Encoder-Decoder-Attention (with data augmentation) [162] | 15.1 | - |
| LAS Model [163] | 12.9 | 35.5 |

output takes 30% of CTC output probability and 70% of attention output probability. The convolutional frontend before transformer encoder is two 2D convolutional neural network layers [13] with filter size (3,2), each followed by a ReLU activation. In the transformer model, the attention dimension $d_{model}$ is 256, and feedforward network hidden state dimension $d_{ff}$ is 2048, there are 4 attention heads, encoder layer number $N_e$ is 12 and number of decoder layers $N_d$ is 6, the initial value of learning rate is 5.0, the attention dropout rate is 0.0, and the encoder and decoder dropout rate is 0.1. We adopt the unigram sub-word algorithm with maximum 5000 vocabularies [154].

## 4.3.2 Experimental Results

### 4.3.2.1 Baseline System

We use the default settings of ESPnet transformer model as our baseline [45, 66]. From Table 4.1, the baseline model result is comparable with other end-to-end models [161–163] trained using Librispeech 100h dataset.

### 4.3.2.2 The Proposed System

First, we explore the number of look-ahead frames included as mentioned earlier using two attention biasing techniques, i.e., soft attention biasing and hard attention biasing. The standard deviation $\sigma$ in the Gaussian mask (Eq. (4.4)) is initialized to 100 as in [155]. In both cases, we apply attention biasing on all the decoder layers. The results are shown

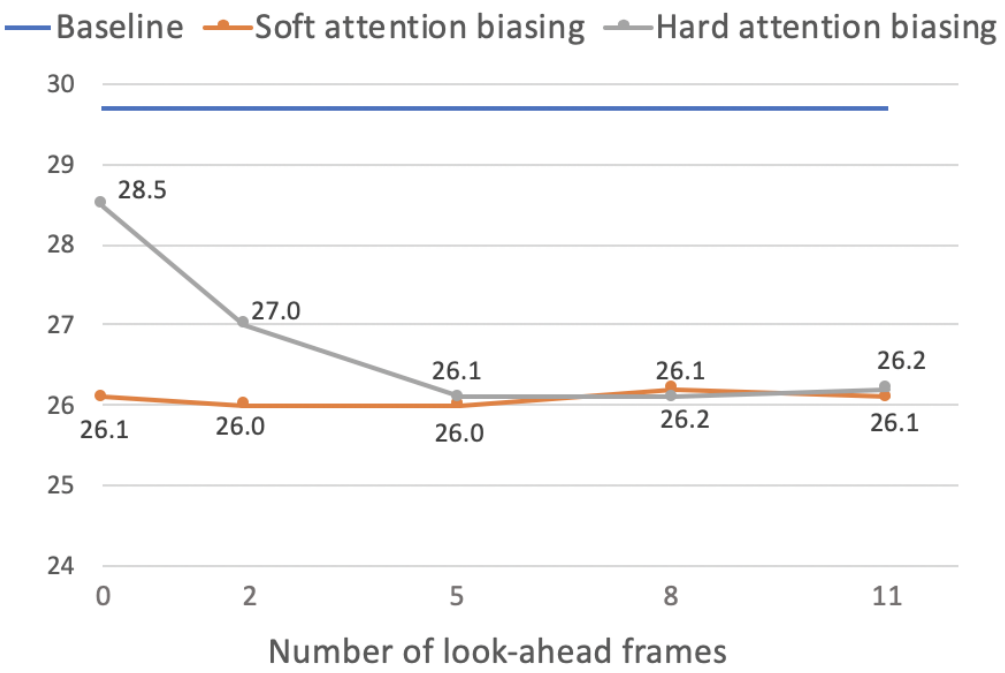



Fig. 4.4: WER results of different numbers of look-ahead frames on LibriSpeech Test_other set. Soft and hard attention biasing are applied on all decoder layers.

in Figure 4.4. We can see that look-ahead frame does not affect much for soft attention biasing. This is because we employ a Gaussian mask with a large learnable variance, so slight movement of Gaussian center position will have less impact. Figure 4.5 shows how the Gaussian mask standard deviation of each head evolves during training. It slowly increases for all four attention heads during training with a small scale. However for hard attention biasing, since it depends entirely on the correctness of alignment position identified, careful tuning of this hyperparameter is required. Only when we add more than 5 look-ahead frames in this case, hard attention biasing has comparable performance with soft attention biasing. To save the effort of tuning parameter, we employ soft attention biasing with 5 look-ahead frames in the following experiments.

Second, the number of decoder layers applying cross attention biasing is explored. As mentioned earlier, model captures more local than global information at lower layers. Table 4.2 lists the experimental results. From the results, using cross attention biasing at the lower layers (layer 1-3) can get slightly better results than at higher layers (layer 4-6). It is consistent with the previous results, that is, lower layers capture more local information, while higher layers capture more global information [158].

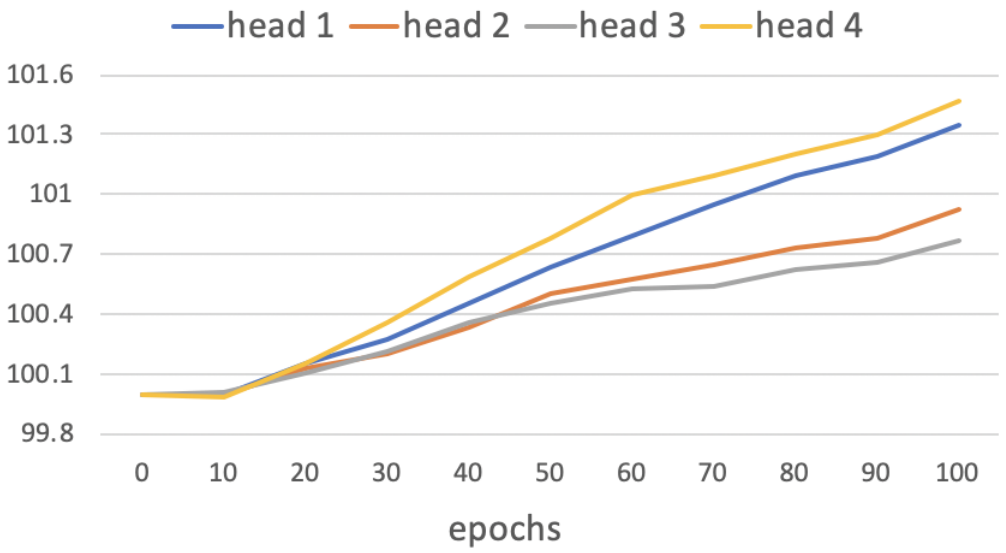



Fig. 4.5: Gaussian mask standard deviation for each of attention head at each training epoch.

We investigate the effect of all three techniques used in our proposed method, that includes adding look-ahead frames, applying on lower layers of the decoder (layer 1-3), and monotonic alignment regularization. Table 4.3 lists the experimental results. We also test any two combination out of the three techniques by specifying missing which technique, e.g. *w/o look-ahead frames* refers to using the techniques of applying on lower layers of the decoder and monotonic alignment regularization, and so on for the other two. It can be seen that these three techniques are equally important, as missing any one of them results in similar performance deterioration (row 3-5). For the last technique to encourage monotonic alignment between text output and speech input, we have tried to equally segment the input speech length based on output text length and then find alignment from corresponding speech frames to ensure monotonicity, but the results cannot surpass our alignment regularization method. Altogether, combination of all three techniques achieves our best result.

The last row of Table 4.3 lists the hybrid global and local attention approach [160]. Our model has 1.9%-4.3% relative WER reductions compared with [160]. Our model applies cross attention biasing on lower three decoder layers, and the rest decoder layers use standard cross attention. While in [160], it mixes two mechanism together with a

Table 4.2: WER results of applying cross attention biasing on different layers on LibriSpeech 100h

| **Decoder layers applied** | **Test_clean** | **Test_other** |
|---|---|---|
| baseline | 12.0 | 29.7 |
| layer 1-3 | 9.4 | 25.9 |
| layer 4-6 | 9.5 | 26.2 |
| layer 1-6 | 9.4 | 26.0 |

Table 4.3: WER results of cross attention biasing and alignment regularization on LibriSpeech 100h

| **Model** | **Test_clean** | **Test_other** |
|---|---|---|
| baseline | 12.0 | 29.7 |
| our proposed method | **9.0** | **25.4** |
| w/o look-ahead frames | 9.3 | 25.9 |
| w/o applying on layer 1-3 | 9.4 | 26.0 |
| w/o alignment regularization | 9.4 | 25.9 |
| hybrid global+local model [160] | 9.4 | 25.9 |

gate derived from encoder hidden states. It is questionable on how much encoder hidden states can tell the importance of global versus local attention.

## 4.4 Chapter Summary

In this chapter, we introduce an effective cross attention biasing technique through making use of cross attention weights. In order to constrain the monotonic alignment attribute between text output and speech input, we propose a regularizer term to achieve monotonicity. Under the cross attention framework, attention biasing limits the speech context range given alignment information. Experiments on LibriSpeech dataset denoted that

our proposed model is effective. Comparing with the baseline system, there are 14.5%-25.0% relative WER reductions. Comparing with the hybrid global and local attention method, there are 1.9%-4.3% relative WER reductions.

# Chapter 5

# Universal Speech Transformer

In this chapter, we turn our attention to the challenging tasks of fixed number of encoder and decoder layers in the transformer model. Predefining the number of encoder and decoder layers is a natural step not only in the transformer model, but also in the standard encoder-decoder model.

The approach taken in this chapter is motivated by the recurrent nature in the recurrent neural network (RNN) model, which could process variable length sequences of inputs. Instead of recurrency in the time dimension, the model in our approach focuses on the depth dimension, i.e., processes inputs with dynamic number of encoder and decoder layers, up to a maximum number of layers defined beforehand. Another notable difference from RNN model is that RNN shares parameters across different time steps of the sequence, whereas the model parameters are not shared across different layers in our approach.

This chapter is organized as follows. Section 5.1 first provides the motivation for introducing dynamic number of encoder and decoder layers to bring inductive bias and avoid hyperparameter tuning. Some previous studies in this direction are also introduced. Section 5.2 then gives a full picture of our proposed method named universal speech transformer. Experiments are carried out in Section 5.3 to evaluate the performance of universal speech transformer. Section 5.4 concludes this chapter.

## 5.1 Introduction

### 5.1.1 Motivation

Chapter 2 discussed the fixed numbers of encoder and decoder layers in the transformer model, which limits its computation capability. Fixed number of encoder and decoder layers is common not only in the transformer model, but also in the standard encoder-decoder model, as seen in Section 2.2.2. However, this predetermined feature is not the optimal solution.

On the one hand, compared with RNN and long short-term memory (LSTM) networks which have iterative or recursive computation, speech transformer model loses the recurrent inductive bias, which is helpful to tackle tasks of varying complexity. Each input speech time step goes through the same and fixed numbers of encoder and decoder layers to compute the final output, regardless of the fact that different speech time steps differ in phoneme obscurity and noise level, thus may require different computation resources. It will be better to provide different number of encoder/decoder layers for different input time steps. On the other hand, determining the numbers of encoder and decoder layers requires careful tuning for each dataset to achieve the optimal performance. Speech transformer model itself was tested for performance in [13] with 5 different depth combinations of encoder and decoder. This step is quite manual and resource consuming.

This finding motivates us to develop dynamic number of encoder and decoder layers to address the issues above. We review some prior works along this direction first and analyzing their drawbacks.

### 5.1.2 Prior Works

There are several studies dealing with the depth of encoder and decoder building blocks. They mostly target at increasing the model depth to enlarge the modeling capability, so as to further improve the model performance.

In [7], it builds a deeper encoder by adding residual blocks and using multiple CNN layers before bidirectional LSTM network. Adding the residual block has the benefit to

pass the input information from the beginning to the model output in a shorter path. In this way, the model could preserve the input information without increasing the training error percentage. Besides, it helps in tackling the vanishing gradient problem using identity mapping. The time-depth separable convolution model [164] trains very deep convolutional encoders via a soft attention window pre-training scheme. It dramatically reduces the number of parameters in the model while keeping the receptive field large. [165] proposes a very deep transformer architecture with up to 48 encoder and decoder layers to enlarge the computation capability. It also applies stochastic residual layers to improve generalizability and to prevent overfitting. However, these methods still require us to tune depth related hyperparameters, e.g. [165] tested 9 different depth combinations with depth ranging from 4 layers to 48 layers.

Different from [165], [166] trains very deep transformer models (up to 40 layers) and then randomly drop layers at training time in order to do efficient layer pruning at inference time. Figure 5.1 presents the idea of layer pruning. During training time, the network layers are randomly dropped. In this way, the network is trained to be robust to pruning, so that during test time, the network can be pruned to any depth on demand. This method could get better model in the sense that during layer pruning, the model will select layers based on their importance and only retain those layers useful for modeling. Again it has the same drawback of determining the layer depth by setting the pruning rate.

### 5.1.3 Inspiration from Universal Transformer

Recently, M. Dehghani et al. [167] proposes the universal transformer model, which achieves the state-of-the-art results on LAMBADA language modeling task. It has the transformer-like architecture and uses a dynamic per-position halting mechanism to choose the required number of layers for each input time step dynamically, which exactly addresses the issues with speech transformer analyzed above.

In this work, we successfully introduce the universal transformer model to ASR task and we term our model as *universal speech transformer*. To the best of our knowledge, this is the first work regarding the dynamic encoder and decoder depth in ASR. The recurrent nature of universal transformer best suits the needs of recognizing phonemes with

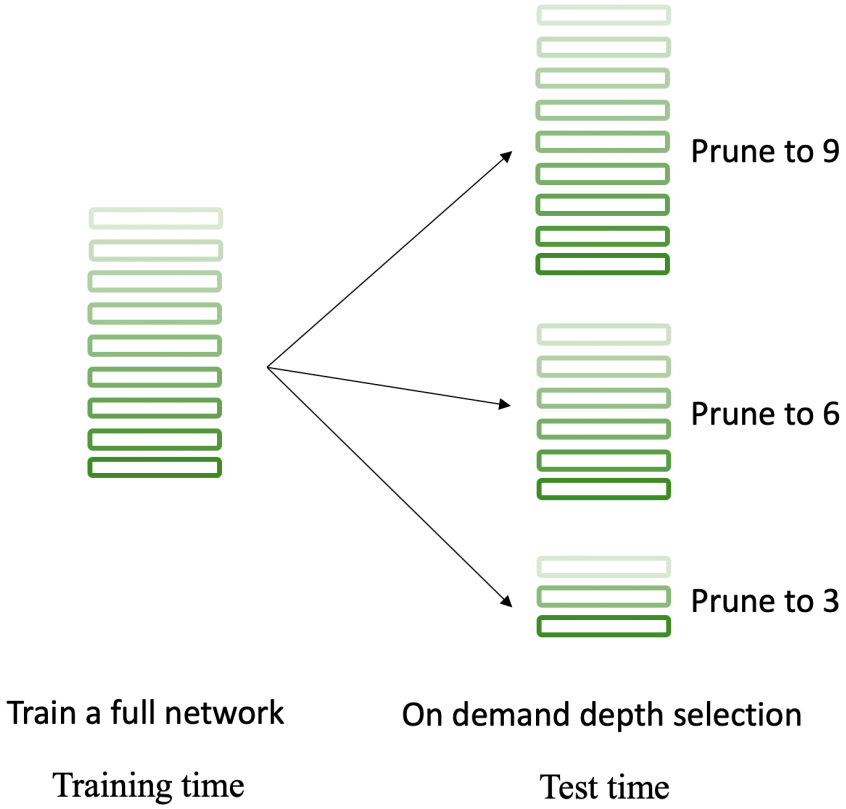



Fig. 5.1: Randomly drop layers at training time.

different complexity and noise level, at the same time dynamically learning the encoder and decoder depth, which relieves the burden of tuning depth related hyperparameters.

However, universal transformer model has two problems when applied on ASR. First, it adds the depth embedding and positional embedding repeatedly for each layer, which dilutes the acoustic information carried by hidden representation. Second, it performs a partial update of hidden vectors between layers, which is less efficient compared to the full update given the same number of update. To tackle these two problems, we remove the depth embedding and only add the positional embedding once at the transformer encoder frontend, and we replace the partial update of hidden representations between layers with a full update.

On the LibriSpeech, Switchboard and AISHELL-1 ASR datasets, our proposed universal speech transformer model outperforms a baseline by 3.88%-13.7%, and achieves better results with much less computation cost compared with the very deep transformer

model using 36 encoder layers and 12 decoder layers in [165]. From the experimental results, it can be seen that the number of encoder layers required varies among different input time steps and different datasets, which further substantiates the value of dynamic depth over fixed depth for datasets with varying complexity.

## 5.2 Model Architecture

### 5.2.1 Universal speech transformer

Universal speech transformer is based on the popular speech transformer model, which we refer the reader to Section 2.2.4 for full details. Same as speech transformer, the core module of universal speech transformer is the multi-head attention network. The main change is on the dynamic encoder and decoder depth. Speech transformer model has fixed encoder and decoder depth. Compared with RNN and LSTM networks which have iterative or recursive computation, speech transformer loses the recurrent inductive bias. Universal speech transformer addresses this issue with dynamic encoder and decoder depth. The overall framework of universal speech transformer is presented in Figure 5.2. The adaptive computation time technique [168] is applied to each input time step to calculate the required computation resources, in this case the numbers of encoder and decoder layers, before emitting the final outputs. Each input speech time step requires different level of computation resources due to obscurity of different phonemes and noise variation along the utterance. So does each text input. In particular, for the hidden state $H^j = (h_1^j, ..., h_t^j) \in \mathbb{R}^{t \times d}$ in $j^{th}$ layer, a probability vector at $i^{th}$ time step is calculated by a sigmoid function:

$$p_i^j = k\sigma(Wh_i^j + b), \tag{5.1}$$

where $W \in \mathbb{R}^{d \times 1}$, $b \in \mathbb{R}^1$, $k$ is a scaling factor, $W$, $b$ and $k$ are shared across all layers, $i \in [1, t]$, $j \in [1, L]$, $L$ is the maximum depth defined beforehand. This probability calculated at each hidden state for each time step is treated as an indicator on whether current hidden state has encoded adequate information.

The halting probability is the summation of the probability calculated in Eq. (5.1), which denotes the probability to emit the final output at $i^{th}$ time step in $j^{th}$ layer:

$$halt_i^j = \sum_{k=1}^{j} p_i^k. \tag{5.2}$$

If the halting probability reaches the threshold, we deem current number of layers is sufficient for encoding the input at current time step. The number of encoder or decoder layers required at each input time step is decided when the halting probability reaches the threshold, or when the encoder or decoder reaches the maximum depth $L$:

$$N_i = \min(L, \max(n' : halt_i^{n'} \leq 1 - \epsilon)), \tag{5.3}$$

where $\epsilon$ is a small constant (0.01 in this paper).

The number of encoder or decoder layers is therefore the maximum number of layers among all input time steps:

$$N = \max(N_i) \quad i \in [1, t]. \tag{5.4}$$

The input at each time step $i$ emits the corresponding output at $N_i^{th}$ layer. For the input time steps where the outputs are emitted earlier than the other time steps, i.e. $N_i < N$, the last hidden states are carried forward until all the time steps emit the outputs or when the encoder or decoder reaches the maximum depth. The hidden state at $i^{th}$ time step in $j^{th}$ layer is thus:

$$h_i^j = \begin{cases} h_i^j & 1 \leq j \leq N_i, \\ h_i^{N_i} & N_i < j \leq N. \end{cases} \tag{5.5}$$

We illustrate the adaptive computation time technique for calculation of the numbers of encoder and decoder layers in Figure 5.3. Each hidden state computes a probability to determine whether it halts at current layer, i.e. reach the required number of layers. Hidden state shaded in grey means the current time step halts already, but because there are other time steps which have not reached the required numbers of layers, this time step hidden state is replicated over to the next layer (by thick arrow). Otherwise, the computation between each layer is through the standard transition function (by

thin arrow), in this case the transformer encoder or decoder multi-head attention and feedforward network.

### 5.2.2 Modifications on universal transformer model

As mentioned earlier, universal transformer model has two problems when applied on ASR. Therefore, we make two modifications to the original universal transformer model here. Firstly, universal transformer model adds both the positional embedding and the depth embedding repeatedly for each encoder and decoder layer as Eq. (5.6) and Eq. (5.7).

$$PE^{dep}_{(pos,2i)} = \sin(pos/10000^{2i/d}) + \sin(dep/10000^{2i/d}), \tag{5.6}$$

$$PE^{dep}_{(pos,2i+1)} = \cos(pos/10000^{2i/d}) + \cos(dep/10000^{2i/d}), \tag{5.7}$$

where $pos$ is the position, $dep$ is the depth, $d$ is the positional and depth embedding dimension, and $i$ is the $\mathrm{i}^{\mathrm{th}}$ dimension.

For the universal speech transformer, we do not add the depth embedding for each encoder and decoder layer. Only the positional embedding is added once before the repeated building blocks as Eq. (5.8) and Eq. (5.9). This is based on the assumption that adding the positional and depth embeddings iteratively for each layer will dilute the acoustic information carried by hidden representation, given that each speech frame under analysis only has a window size of 25ms and does not contain much time-domain information. In contrast, universal transformer encodes the word embedding. A word possibly spans tens of analysis frames in ASR context, so a word embedding contains much richer information, and it will be of less influence to repeatedly add the positional and depth embeddings. Besides, the average depth in our model (around 21) is much higher than the average depth in the universal transformer model (around 8 in LAMBADA language modeling), so adding the depth embedding repeatedly for each layer has more impact in our model. Additionally, adding the depth embedding to each layer provides the depth information to the hidden representation and may be beneficial for calculating the optimal encoder and decoder depth. However, depth information is already implicitly embedded in the hidden representation, which is calculated progressively

from layer to layer. Removing the depth embedding does not affect the dynamic encoder and decoder depth computation.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d}), \tag{5.8}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d}). \tag{5.9}$$

Secondly, in each layer of the universal transformer model, hidden representation is updated as below using $p_i^j$ calculated from Eq. (5.1):

$$H^{j+1} = [p_1^j, ..., p_t^j] * \tilde{H}^{j+1} + (1 - [p_1^j, ..., p_t^j]) * H^j, \tag{5.10}$$

where $\tilde{H}^{j+1}$ is the transformed state after multi-head attention and feedforward network and $H^{j+1}$ is the updated state, i.e. it only performs a partial update of hidden state by a $p_i^j$ probability, with the addition of previous hidden state by a $1 - p_i^j$ probability. The partial update is not very efficient, and calculating hidden state as Eq. (5.10) requires more times of update and deeper layers to reach the optimal hidden state, which in turn may bring the vanishing gradient problem. To avoid this problem, and given that our model is already very deep, we use the transformed state $\tilde{H}^{j+1}$ directly as the next layer hidden representation without the probability factor to perform an efficient full update. We will test these two modifications in experiments.

## 5.3 Experiments

### 5.3.1 Datasets

We conduct experiments with the universal speech transformer model on three publicly available datasets, including LibriSpeech [39], Switchboard [40] and AISHELL-1 [36]. LibriSpeech consists of 16kHz read English speech from audiobooks. Switchboard is a 300 hour corpus of conversational English telephone speech. AISHELL-1 is a 16kHz Chinese Mandarin speech corpus recorded by 400 speakers from different accent areas in China. The characteristics of the datasets are summarized in Table 5.1.

Table 5.1: Details of datasets used for experiments

| | |
|---|---|
| **LibriSpeech** | |
| Training set | 100h |
| Test_clean set | 5.4h |
| Test_other set | 5.1h |
| **Switchboard** | |
| Training set | 300h |
| SWBD set | 2.1h |
| CallHome set | 1.6h |
| **AISHELL-1** | |
| Training set | 150h |
| Test set | 5h |

### 5.3.2 Experimental setup

We use Espnet toolkit [45] for experiments. Input acoustic features are 80-dimensional filterbanks extracted with a window size of 25ms shifted every 10ms, which are mean and variance normalized. Utterances longer than 3000 frames or 400 characters are discarded to keep memory manageable. In the training stage, the input samples are shuffled randomly and trained with batch size 12. We adopt the unigram sub-word algorithm with maximum vocabulary size being 5000. The two CNN layers at the bottom of encoder in Figure 5.2 have filter size (3,2) and stride 2, each followed by a rectified linear unit (ReLU) activation. For multi-head attention network, the attention dimension is 256, the number of attention heads is 4, the dimension of feedforward network is 2048. For universal speech transformer related hyperparameters, the maximum depth $L$ in encoder is 24, and the maximum depth $L$ in decoder is 16, the scaling factor $k$ is set as

Table 5.2: WER results of end-to-end speech recognition models on LibriSpeech 100h

| **Model** | **Test_clean** | **Test_other** |
|---|---|---|
| End-to-end (E2E) [161] | 14.7 | 40.8 |
| E2E with augmented data [162] | 15.1 | - |
| LAS [163] | 12.9 | 35.5 |
| Baseline | 12.0 | 29.7 |

Table 5.3: Two modifications on universal transformer model on LibriSpeech 100h

| **Model** | **Test_clean** | **Test_other** |
|---|---|---|
| Baseline | 12.0 | 29.7 |
| Universal transformer | 24.2 | 43.7 |
| The proposed model | **11.3** | **28.3** |
| w/o remove depth embedding | 19.5 | 37.8 |
| w/o full hidden state update | 15.0 | 32.1 |

0.25. During experiments, the dynamic depth for encoder and decoder obtained tends to be small, so we set a minimum depth for encoder and decoder, i.e. the model only performs dynamic depth computation after the minimum depth is reached. Based on the encoder and decoder depth combinations tested by [165], we set the minimum depth for encoder to 10, and 6 for decoder. The initial value of the learning rate is 5.0, the encoder and decoder dropout rate is 0.1. We have a strong baseline as shown in Table 5.2, which comes from the best results well designed in Espnet toolkit.

### 5.3.3 Results

We first test the two modifications on the universal transformer model analyzed in Section 2.2 using LibriSpeech 100h dataset. From Table 5.3, using universal transformer model

directly for speech recognition deteriorates performance a lot compared to the baseline. The universal speech transformer model using the efficient full hidden state update and removing the depth embedding for each layer achieves the best performance. We believe that using the full hidden state update facilitates the encoder or decoder to obtain the optimal hidden representation. Removing the depth embedding is most efficient, and it confirms our assumption that adding depth embedding repeatedly for each layer dilutes the acoustic information in hidden representation.

Next, we compare our proposed method with [165], which did 9 test runs and set a good benchmark for various encoder and decoder depth combinations on transformer model. The results of three datasets are summarized in Table 5.4. [165] tested 9 encoder and decoder depth combinations ranging from 4 layers to 48 layers, and the best combination is using 36 encoder layers and 12 decoder layers. They further did an ensemble of three models (row 2, 3, 4 of Switchboard experiments) to achieve their optimal performance (row 5 of Switchboard experiments). It can be seen that their approach is very tedious and manual, and the choice of the best model is purely by trial and error. In comparison, our model has the capability to choose the best layer depth for each input time step. Our model surpasses the baseline by 3.88%-13.7%, and outperforms [165] on all three datasets.

Subsequently, we would like to see if depth variation is related to noise level or language. To analyze the influence of noise, we split Librispeech test_other data by different signal to noise ratio (SNR). Number of utterances and average encoder depth for each dataset is listed in Table 5.5. Intuitively, speech mixed with noise is more difficult to recognize and needs more computation resources, i.e. it should have higher average encoder depth for lower SNR. From Librispeech dataset, we can find that average encoder depth for noisy data (test_other dataset) is slightly higher than clean data (test_clean dataset), but within noisy data, there is no clear correlation between average encoder depth and SNR. From language perspective, AISHELL-1 dataset composed of Chinese Mandarin utterances has average encoder depth in between Librispeech and Switchboard dataset, both of which consists of English utterances, so no conclusion can be made from here. Phoneme obscurity may affect the average encoder depth as well.

We randomly sample two utterances from Switchboard dataset and compare the recognition result with the baseline in Figure 5.4. It can be seen that for both cases, wrong

words near the end of the utterances predicted by the baseline are corrected by our model with relatively higher number of layers in those time steps. In addition, we calculate the average encoder depth across all speech time steps for three datasets and list them in Table 5.5. The average numbers of encoder layers are all around 21 in three datasets. Compared with [165] which deploys 36 encoder layers, our model dynamically determines the encoder and decoder depth for each input time step with significantly less training cost overall, which further demonstrates the value and potential of universal speech transformer model.

## 5.4 Chapter Summary

In this work, we introduce the universal speech transformer, which generalizes speech transformer with dynamic encoder and decoder depth for each input time step. Our model is capable of tackling tasks of varying complexity by bringing the recurrent inductive bias to speech transformer model, as well as relieving the burden of tuning depth related hyperparameters. It outperforms the baseline by 3.88%-13.7%, and achieves better performance than 36 layer encoder model with much less computation cost.

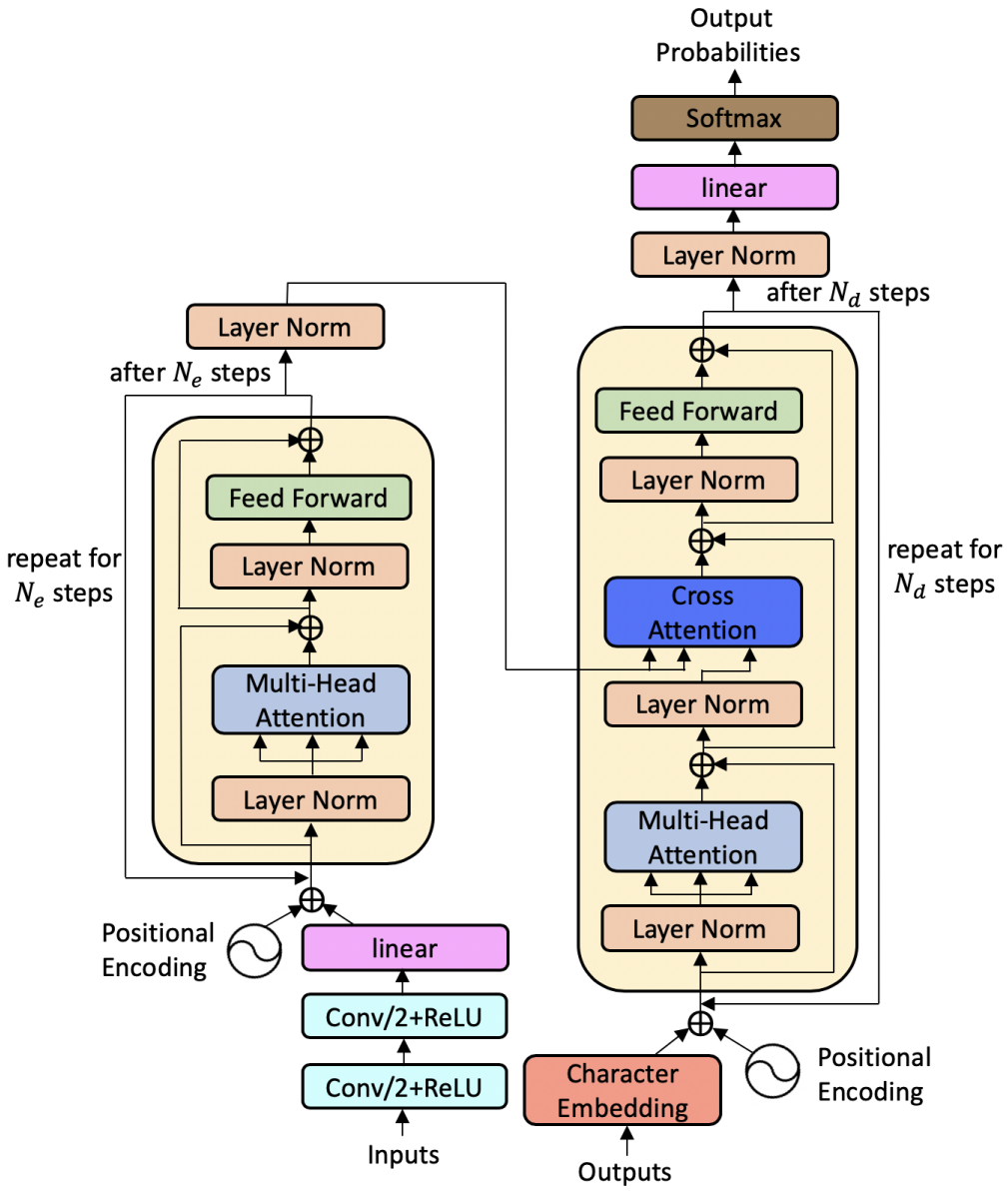



Fig. 5.2: Universal Speech Transformer model architecture.

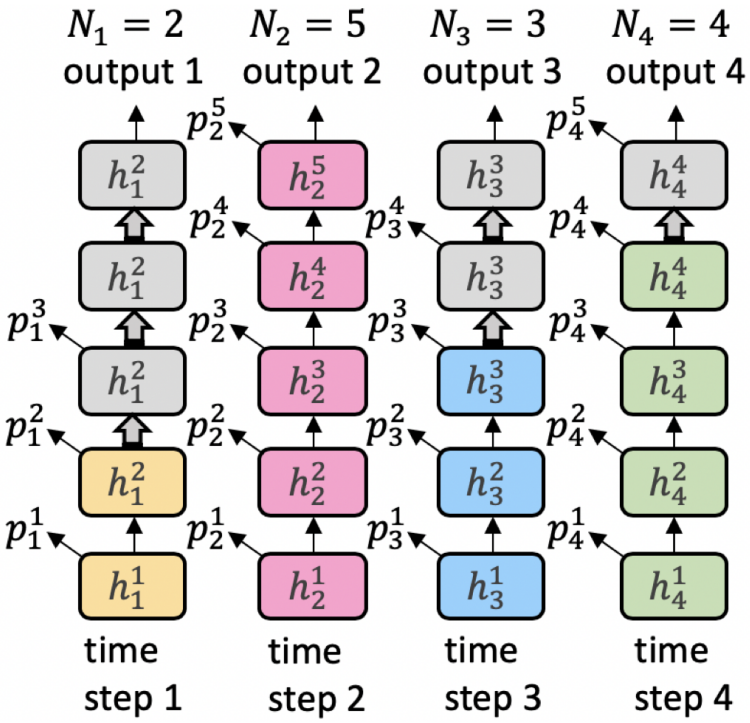



Fig. 5.3: Example of adaptive computation time technique with 4 input time steps. Maximum number of layers $L$ is 5 here. Illustration is applicable to both encoder and decoder.

Table 5.4: Comparison with very deep transformer model on all three datasets

| **Model** | **Switchboard (WER)** | | |
|---|---|---|---|
| | **SWBD** | **CallHome** | **All** |
| Baseline | 8.8 | 18.0 | 13.4 |
| 36Enc-12Dec [165] | 10.4 | 18.6 | - |
| 48Enc-12Dec [165] | 10.7 | 19.4 | - |
| 60Enc-12Dec [165] | 10.6 | 19.0 | - |
| Ensemble of above 3 [165] | 9.9 | 17.7 | - |
| The proposed model | **8.3** | **17.3** | **12.8** |
| **Model** | **AISHELL-1 (CER)** | | |
| | **Dev** | **Test** | |
| Baseline | 6.4 | 7.3 | |
| 36Enc-12Dec [165] | 6.4 | 7.3 | |
| The proposed model | **5.8** | **6.3** | |
| **Model** | **LibriSpeech 100h (WER)** | | |
| | **Test_clean** | **Test_other** | |
| Baseline | 12.0 | 29.7 | |
| 36Enc-12Dec [165] | 12.8 | 30.4 | |
| The proposed model | **11.3** | **28.3** | |

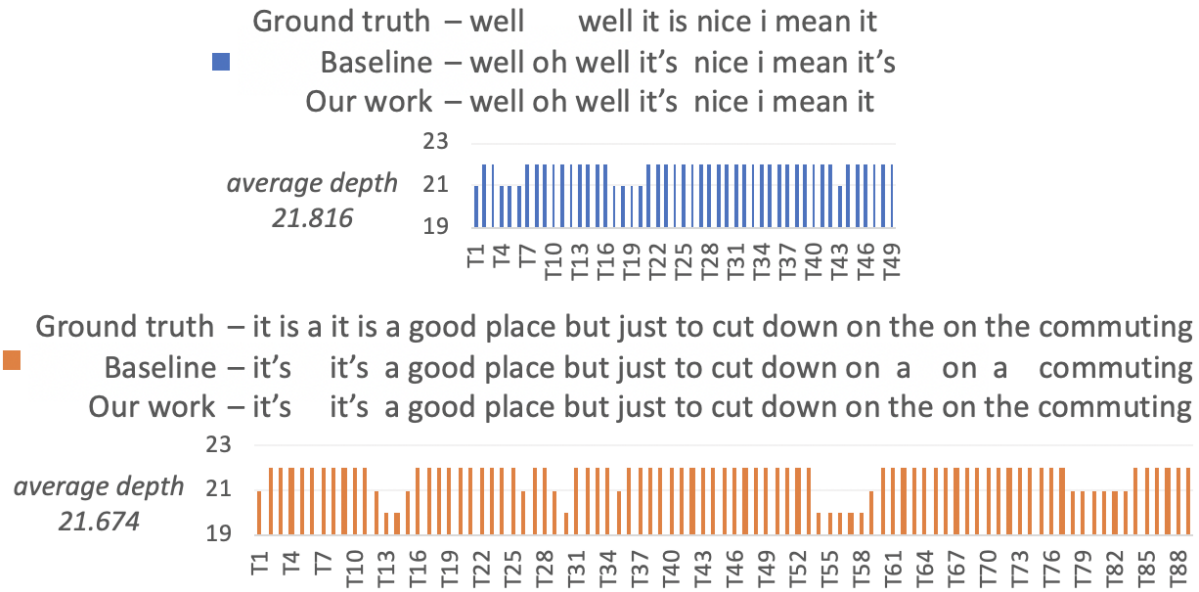



Fig. 5.4: Number of encoder layers required $N_i$ in Eq. (5.3) for each time step $i$ and average encoder depth of two randomly sampled speech utterances from Switchboard test dataset.

Table 5.5: Number of utterances and average encoder depth across all speech time steps for three datasets

| **Dataset** | **No. of utterances** | **Depth** |
|---|---|---|
| **Librispeech** | | |
| Test_clean | 2620 | 21.118 |
| Test_other (All) | 2939 | 21.268 |
| Test_other 30≤SNR | 898 | 21.312 |
| Test_other 25<SNR≤30 | 245 | 21.270 |
| Test_other 20<SNR≤25 | 200 | 21.263 |
| Test_other 15<SNR≤20 | 172 | 21.190 |
| Test_other 10<SNR≤15 | 97 | 21.258 |
| Test_other 5<SNR≤10 | 109 | 21.265 |
| Test_other 0<SNR≤5 | 1218 | 21.248 |
| **Switchboard** | | |
| All | 4458 | 21.738 |
| **AISHELL-1** | | |
| Test | 5741 | 21.428 |

# Chapter 6

# Conclusions and Future Directions

This thesis has focused on improving the transformer model architecture for ASR, which is the mainstream model architecture in NLP and speech fields. The reason behind is ASR has its own characteristics, for example monotonic alignment between text output and speech input. Without taking this into consideration, cross attention will disperse the attention to the entire speech utterance. This means the cross attention may not be able to focus on the corresponding speech input aligned with the current text output, which is both inefficient and imprecise. Besides, speaker adaptation is an exclusive problem in ASR. ASR performance could drop drastically should the test speaker data deviates a lot from training speaker data. This is because the model trained on training speaker data does not adapt well on unseen speakers. Transformer model has its drawback as well, like the static number of encoder and decoder layers, which needs careful parameter tuning to achieve the best performance. Compared with other models such as recurent neural network, transformer model loses the recurrent inductive bias. Moreover, the fixed number of encoder layers is unable to recognize phonemes with different complexity and noise level dynamically.

Because transformer model is developed very recently in 2017, and only applied for ASR in 2019, there is less research work along this direction to improve the transformer model for ASR. This thesis has developed novel approaches accordingly, to boost the ASR performance from three perspective, namely, addressing the speaker adaptation problem in ASR, resolving text output speech input misalignment issue in speech transformer model and improving on the static number of encoder layers in transformer. To conclude

the work, Section 6.1 first summarizes the contributions proposed in this thesis. Finally, Section 6.2 discusses some of the future directions that can be explored, and the challenges that are still to be faced.

## 6.1 Contributions

This thesis highlights three areas of contributions. The first proposed model focuses on speaker adaptation problem in ASR. The second proposed model aims at resolving text output speech input misalignment issue in speech transformer model. The third method improves on the static number of encoder layers in transformer. All of them target at improving the transformer model architecture for ASR.

### 6.1.1 Speaker Adaptation [1, 2]

We propose a unified speaker adaptation approach that can be deployed for general speaker adaptation, specific target speaker adaptation and multi-speaker adaptation. A speaker-aware persistent memory model [1] is presented to capture speaker knowledge through the persistent memory in order to generalize better to unseen test speakers, which belongs to the feature adaptation method. In particular, speaker i-vectors from the training data are sampled and concatenated to speech utterances in each encoder layer, and the speaker knowledge is learnt through attention computation with speaker i-vectors. Furthermore, a model-based gradual pruning approach is deployed to free up partial model encoder parameters for target speaker adaptation without sacrificing the original model performance [2]. Finally, instead of only adapting to general speakers or specific target speaker, we design a multi-speaker adaptation model capable of adapting to multiple speakers simultaneously, which is practical in ubiquitous environment with multiple users. We distribute the pruned free parameters among target speakers equally, randomly, and simultaneously for multi-speaker adaptation.

Our proposed approach brings relative 2.74-6.52% WER reduction on general speaker adaptation, and outperforms the baseline with up to 20.58% relative WER reduction on target speaker adaptation. The proposed multi-speaker adaptation scheme is effective with a 4.87% relative improvement in the WER over the baseline. Even with extremely

low-resource adaptation data (e.g., 1 utterance), our method could improve the WER by relative 6.53% with only a few epochs of training, which demonstrates the superiority and robustness of our proposed methods.

### 6.1.2 Text Speech Alignment [3]

We propose an effective cross attention biasing method for the transformer model that takes output-input alignments into consideration. Our method does not reference CTC output, which is highly dependent on CTC output accuracy. Furthermore, our method does not add additional parameters on encoder hidden states by utilizing existing attention weights. We take advantage of cross attention weights as a reference of output-input alignment to be used in current cross attention computation. In particular, we apply a Gaussian mask on attention weights centered at the alignment position. Additionally, we introduce a regularizer which regularizes alignment between output and input to encourage monotonicity. Since lower layers of the Transformer capture more acoustic and local information, we apply our cross attention biasing on lower layers of the Transformer model, and leave the cross attention at higher layers to attend to entire speech input to capture global information. Experiments on LibriSpeech dataset find that our proposed model can obtain improved output-input alignment for ASR, and yields 14.5%-25.0% relative WER reductions.

### 6.1.3 Static Number of Encoder Layers [4]

The final work focuses on improving the static number of encoder layers. We propose the universal speech transformer with dynamic number of encoder layers [4]. It suits the needs of recognizing phonemes with different complexity and noise level, and relieves the burden of tuning depth related hyperparameters. This is achieved by generating a halting probability at each hidden state to indicate whether it has enough layers for encoding current input. Universal transformer adds the depth and positional embeddings repeatedly for each layer, which dilutes the acoustic information carried by hidden representation, and it also performs a partial update of hidden vectors between layers, which is less efficient especially on the very deep models. For better use of universal

transformer, we modify its processing framework by removing the depth embedding and only adding the positional embedding once at transformer encoder frontend. This is same as the original transformer model. Furthermore, to update the hidden vectors efficiently, especially on the very deep models, we adopt a full update. This means we drop the component of previous layer hidden state together with the halting probability, and use the latest layer hidden state only as the representation.

On the LibriSpeech, Switchboard and AISHELL-1 ASR datasets, our proposed universal speech transformer model outperforms a baseline by 3.88%-13.7%, and achieves better results with much less computation cost compared with the very deep transformer model using 36 encoder layers and 12 decoder layers in [165]. From the experimental results, it can be seen that the number of encoder layers required varies among different input time steps and different datasets, which further substantiates the value of dynamic depth over fixed depth for datasets with varying complexity.

## 6.2 Future Directions

The objective of this thesis is to improve the transformer model architecture for ASR. We have achieved this from three perspective, i.e., addressing the speaker adaptation problem in ASR, resolving text output speech input misalignment issue in speech transformer model and improving on the static number of encoder layers in transformer. Our experiments have shown that our approaches could improve ASR WER a lot, and we have provided extensive analysis on our methods. In spite of the effectiveness of our approaches in boosting the ASR performance, these are some limitations. We discuss some future directions of each work below.

### 6.2.1 Speaker Adaptation

#### 6.2.1.1 Analysis on multi-speaker adaptation

First, for the speaker adaptation problem, our method is applicable to general speaker adaptation, specific target speaker adaptation and multi-speaker adaptation. In spite of its effectiveness, our research on the multi-speaker adaptation scenario is still at the

preliminary stage. In our experiments, we randomly sampled 10 utterances from four speakers in the test dataset as the additional training data. We did not take the speaker characteristics into consideration, such as gender of speakers and their similarity. Model trained on one speaker may transfer well to similar speakers. Besides, research on how the number of speakers, amount of utterances affect the multi-speaker adaptation are interesting future topics. Generally speaking, the more the number of speakers, the more difficult it is for the model to adapt to all of them at the same time. More utterances of target speakers should help the model adapt to the target speakers.

#### 6.2.1.2 Integrate speaker i-vector extraction and ASR

Another drawback of our feature adaptation method is that we have to do pre-processing on the training data to obtain speaker i-vectors. This is not a truly end-to-end method. One interesting direction is to combine speaker i-vector extraction and ASR into one end-to-end model by multi-task learning. Having meaningful speaker i-vectors provides useful speaker-related information to ASR task.

#### 6.2.1.3 Diversify the speaker i-vectors sampled

Last, in our proposed approach, we randomly sampled $N$ speaker i-vectors from the training data, and we assume their combinations could cover the entire speaker space, i.e. any unknown speaker information could be represented by the weighted sum of speaker i-vectors. During experiments, we need to tune the number $N$ for enough speaker i-vectors. Although our tuning experiment results could provide some insight for the number of speaker i-vectors sampled of new datasets, it is better to tune the number $N$ for each new dataset. Besides, the speaker i-vectors sampled also affect the final model performance. One future work could be analyzing the influence of speaker i-vectors sampled. For example, increasing the diversity of speaker i-vectors may in turn require less speaker i-vectors to reach the same model performance.

### 6.2.2 Text Speech Alignment

#### 6.2.2.1 Improving on speech encoding

For the text speech alignment, our idea is to obtain the text output speech input alignment and perform a local attention based on the alignment position. There are two potential problems with our proposed approach. First, obtaining the text output speech input alignment in our proposed model is determined by taking the maximum cross attention weights. This requires an accurate encoding of speech input, so that the cross attention weights correctly reflect the feature relevance between speech and text. In our proposed method, we have added a monotonicity regularization term in the loss function to refine the alignment positions obtained. Another method could be using the pretrained speech model, so that the speech embedding has rich information beforehand, and our method could further assist in ASR task.

#### 6.2.2.2 Global monotonicity regularization

Second, the monotonic alignment regularization we propose is performed in a local context, not globally over the entire utterances. We compare every two consecutive alignment positions and penalize the misalignment. However, this is only a local optimal solution. For example, if every alignment positions obtained is delayed by some speech frames, the speech encoding applied with attention biasing will be wrong embeddings. However, if the misaligned positions are still monotonically increasing, our alignment regularization cannot capture these wrong alignments, as they are not misaligned locally. Developing an algorithm to identify the misalignment globally over the entire utterances could be one interesting direction.

### 6.2.3 Static Number of Encoder Layers

#### 6.2.3.1 Exploring the computation resource difference

For the static number of encoder layers problem, our universal speech transformer could load encoder/decoder layers dynamically. We are surprised to find out that our approach could at the same time tell the level of difficulty to encode among different inputs through

the number of encoder layers required. The more the number of encoder layers, the more difficult it is to encode the speech input frame. With this, our method is capable of indicating the computation resource requirement, and we are interested in exploring the computation resource differences between different language, between pretrained and non-pretrained models, and between clean and noisy speech.

# References


[1] Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, "Speech transformer with speaker aware persistent memory," in *INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association*, 2020, pp. 1261–1265.

[2] Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, "A unified speaker adaptation approach for asr," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic, 2021, pp. 9339–9349.

[3] Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, "Cross attention with monotonic alignment for speech transformer," in *INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association*, 2020, pp. 5031–5035.

[4] Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, "Universal speech transformer," in *INTERSPEECH 2020 – 21st Annual Conference of the International Speech Communication Association*, 2020, pp. 5021–5025.

[5] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015, pp. 577–585.

[6] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[7] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4845–4849.

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

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[10] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[11] A. Graves, "Sequence transduction with recurrent neural networks," in *International Conference of Machine Learning (ICML)*, 2012, pp. 235–242.

[12] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.

[13] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.

[14] Z. You, D. Su, J. Chen, C. Weng, and D. Yu, "DFSMN-SAN with persistent memory model for automatic speech recognition," 2019. [Online]. Available: https://arxiv.org/abs/1910.13282

[15] N. Moritz, T. Hori, and J. L. Roux, "Triggered attention for end-to-end speech recognition," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5666–5670.

[16] K. H. Davis, R. Biddulph, and S. Balashek, "Automatic recognition of spoken digits," *The Journal of the Acoustical Society of America*, vol. 24, no. 6, pp. 637–642, 1952.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[18] V. Digalakis, D. Rtischev, and L. Neumeyer, "Speaker adaptation using constrained estimation of gaussian mixtures," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 5, pp. 357–366, 1995.

[19] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[20] L. Deng, P. Kenny, M. Lennig, V. Gupta, F. Seitz, and P. Mermelstein, "Phonemic hidden markov models with continuous mixture output densities for large vocabulary word recognition," *IEEE Transactions on Signal Processing*, vol. 39, no. 7, pp. 1677–1681, 1991.

[21] X. Tang, "Hybrid hidden markov model and artificial neural network for automatic speech recognition," in *2009 Pacific-Asia Conference on Circuits, Communications and Systems*. IEEE, 2009.

[22] E. Trentin and M. Gori, "A survey of hybrid ANN/HMM models for automatic speech recognition," *Neurocomputing*, vol. 37, no. 1-4, pp. 91–126, 2001.

[23] H. Bourlard and N. Morgan, "Continuous speech recognition by connectionist statistical methods," *IEEE Trans. Neural Netw.*, vol. 4, no. 6, pp. 893–909, 1993.

[24] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden markov models," in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2002.

[25] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 1, pp. 161–174, 1994.

[26] H. Franco, M. Cohen, N. Morgan, D. Rumelhart, and V. Abrash, "Context-dependent connectionist probability estimation in a hybrid hidden markov model-neural net speech recognition system," *Comput. Speech Lang.*, vol. 8, no. 3, pp. 211–222, 1994.

[27] J. Hennebert, C. Ris, H. Bourlard, S. Renals, and N. Morgan, "Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems," in *5th European Conference on Speech Communication and Technology (Eurospeech 1997)*. ISCA, 1997.

[28] Y. Yen, M. Fanty, and R. Cole, "Speech recognition using neural networks with forward-backward probability generated targets," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE Comput. Soc. Press, 2002.

[29] A. J. Robinson, G. D. Cook, D. P. W. Ellis, E. Fosler-Lussier, S. J. Renals, and D. A. G. Williams, "Connectionist speech recognition of broadcast news," *Speech Commun.*, vol. 37, no. 1-2, pp. 27–45, 2002.

[30] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[31] A.-R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[32] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[33] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8599–8603.

[34] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8604–8608.

[35] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

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

[37] K. Maekawa, "Corpus of spontaneous japanese: Its design and evaluation," in *Proceedings of SSPR*, 2003, pp. 7–12.

[38] Y. Liu, P. Fung, Y. Yang, C. Cieri, S. Huang, and D. Graff, "HKUST/MTS: A very large scale mandarin telephone speech corpus," in *Chinese Spoken Language Processing*. Springer Berlin Heidelberg, 2006, pp. 724–735.

[39] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[40] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: telephone speech corpus for research and development," in *1992 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1992.

[41] R. A., D. P., and E. Y., "TED-LIUM: an automatic speech recognition dedicated corpus," in *LREC 2012*, 2012, pp. 125–129.

[42] A. Rousseau, P. Deleglise, and Y. Esteve, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *LREC 2014*, 2014.

[43] P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young, "Large vocabulary continuous speech recognition using HTK," in *1994 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1994.

[44] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *2011 IEEE ASRU*, 2011.

[45] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "Espnet: End-to-end speech processing toolkit," in *INTERSPEECH 2018 – 19$^{th}$ Annual Conference of the International Speech Communication Association*, 2018, pp. 2207–2211.

[46] Y. Wang, T. Chen, H. Xu, S. Ding, H. Lv, Y. Shao, N. Peng, L. Xie, S. Watanabe, and S. Khudanpur, "Espresso: A fast end-to-end neural speech recognition toolkit," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.

[47] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "Speechbrain: A general-purpose speech toolkit," 2021.

[48] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 48–53.

[49] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised Pre-Training for Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 3465–3469.

[50] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2020.

[51] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[52] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022. [Online]. Available: https://cdn.openai.com/papers/whisper.pdf

[53] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: end-to-end speech recognition using deep rnn models and wfst-based decoding," in *2015 IEEE ASRU*, 2015.

[54] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, "End-to-end speech recognition using lattice-free MMI," in *INTERSPEECH 2018 – 19$^{th}$ Annual Conference of the International Speech Communication Association*, 2018.

[55] J. Shen, P. Nguyen, Y. Wu, Z. Chen *et al.*, "Lingvo: a modular and scalable framework for sequence-to-sequence modeling," *arXiv preprint arXiv:1902.08295*, 2019.

[56] O. Kuchaiev, B. Ginsburg, I. Gitman, V. Lavrukhin, C. Case, and P. Micikevicius, "Openseq2seq: Extensible toolkit for distributed and mixed precision training of sequence-to-sequence models," in *Workshop for NLP-OSS*, 2018.

[57] A. Zeyer, T. Alkhouli, and H. Ney, "RETURNN as a generic flexible neural toolkit with application to translation and speech recognition," in *Association for Computational Linguistics (ACL)*, 2018.

[58] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert, "Wav2letter++: A fast open-source speech recognition system," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[59] F. Filippidou and L. Moussiades, "*A* benchmarking of IBM, google and wit automatic speech recognition systems," in *IFIP Advances in Information and Communication Technology*, ser. IFIP advances in information and communication technology. Cham: Springer International Publishing, 2020, pp. 73–82.

[60] S. wen Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K. tik Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee, "SUPERB: Speech processing universal performance benchmark," in *INTERSPEECH 2021 – 22nd Annual Conference of the International Speech Communication Association*, Brno, Czech Republic, 2021.

[61] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of ICML*. ACM Press, 2006, pp. 369–376.

[62] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6645–6649.

[63] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, 2014, pp. 1764–1772.

[64] J. Li, "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022.

[65] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[66] S. Karita, N. E. Y. Soplin, S. Watanabe, M. Delcroix, A. Ogawa, and T. Nakatani, "Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," in *INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association*, 2019, pp. 1408–1412.

[67] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, Waikoloa, HI, USA, 2011, pp. 24–29.

[68] S. P. Rath, D. Povey, K. Vesely, and J. Cernocky, "Improved feature processing for deep neural networks," in *INTERSPEECH 2013 – 14$^{th}$ Annual Conference of the International Speech Communication Association*, Lyon, France, 2013, pp. 109–113.

[69] D. Povey, G. Zweig, and A. Acero, "Speaker adaptation with an exponential transform," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, Waikoloa, HI, USA, 2011, pp. 158–163.

[70] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015.

[71] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015, pp. 3586–3589.

[72] Y. Huang and Y. Gong, "Acoustic model adaptation for presentation transcription and intelligent meeting assistant systems," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, Barcelona, Spain, 2020, pp. 7394–7398.

[73] N. Jaitly and G. E. Hinton, "Vocal tract length perturbation (vtlp) improves speech recognition," in *Proceedings of the 30$^{th}$ International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013.

[74] X. Cui, V. Goel, and B. Kingsbury, "Data augmentation for deep neural network acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1469–1477, 2015.

[75] J. Fainberg, P. Bell, M. Lincoln, and S. Renals, "Improving children's speech recognition through out-of-domain data augmentation," in *INTERSPEECH 2016 – 17$^{th}$ Annual Conference of the International Speech Communication Association*, San Francisco, USA, 2016, pp. 1598–1602.

[76] X. Cui, V. Goel, and B. Kingsbury, "Data augmentation for deep convolutional neural network acoustic modeling," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, South Brisbane, QLD, Australia, 2015, pp. 4545–4549.

[77] Y. Stylianou, O. Cappe, and E. Moulines, "Continuous probabilistic transform for voice conversion," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, pp. 131–142, 1998.

[78] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. L. Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Advances in Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 4480–4490.

[79] Y. Huang, L. He, W. Wei, W. Gale, J. Li, and Y. Gong, "Using personalized speech synthesis and neural language generator for rapid speaker adaptation," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, Barcelona, Spain, 2020, pp. 7399–7403.

[80] Y. Huang, J. Li, L. He, W. Wei, W. Gale, and Y. Gong, "Rapid rnn-t adaptation using personalized speech synthesis and neural language generator," in *INTERSPEECH 2020 – 21$^{st}$ Annual Conference of the International Speech Communication Association*, Shanghai, China, 2020, pp. 1256–1260.

[81] S. H. K. Parthasarathi, A. M. N. S. B. Hoffmeister, S. Matsoukas, and S. Garimella, "fmllr based feature-space speaker adaptation of dnn acoustic models," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015.

[82] J. Fainberg, O. Klejch, E. Loweimi, P. Bell, and S. Renals, "Acoustic model adaptation from raw waveforms with sincnet," in *2019 IEEE Workshop on Automatic Speech Recognition & Understanding*, Singapore, 2019.

[83] T. Kim, I. Song, and Y. Bengio, "Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition," in *INTERSPEECH 2017 – 18$^{th}$ Annual Conference of the International Speech Communication Association*, Stockholm, Sweden, 2017.

[84] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *2013 IEEE Workshop on Automatic Speech Recognition & Understanding*, Olomouc, Czech Republic, 2013, pp. 55–59.

[85] M. Karafiát, L. Burget, P. Matějka, O. Glembek, and J. Černocký, "ivector-based discriminative adaptation for automatic speech recognition," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, Waikoloa, HI, USA, 2011, pp. 152–157.

[86] S. Garimella, A. Mandal, N. Strom, B. Hoffmeister, S. Matsoukas, and S. H. K. Parthasarathi, "Robust i-vector based adaptation of dnn acoustic model for speech recognition," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015.

[87] Y. Miao, H. Zhang, and F. Metze, "Towards speaker adaptive training of deep neural network acoustic models," in *INTERSPEECH 2014 – 15$^{th}$ Annual Conference of the International Speech Communication Association*, Singapore, 2014, pp. 2189–2193.

[88] Y. Miao, L. Jiang, H. Zhang, and F. Metze, "Improvements to speaker adaptive training of deep neural networks," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, South Lake Tahoe, NV, USA, 2014, pp. 165–170.

[89] A. Senior and I. Lopez-Moreno, "Improving DNN speaker independence with i-vector inputs," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, 2014, pp. 225–229.

[90] L. Samarakoon and K. C. Sim, "Learning factorized feature transforms for speaker normalization," in *2015 IEEE Workshop on Automatic Speech Recognition & Understanding*, Scottsdale, AZ, USA, 2015, pp. 145–152.

[91] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[92] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, 2014, pp. 1695–1699.

[93] N. Dehak, P. A.Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via ivectors and dimensionality reduction," in *INTERSPEECH 2011 – 12$^{th}$ Annual Conference of the International Speech Communication Association*, Florence, Italy, 2011, pp. 867–860.

[94] Z. Fan, J. Li, S. Zhou, and B. Xu, "Speaker-aware speech-transformer," in *2019 IEEE Workshop on Automatic Speech Recognition & Understanding*, Singapore, 2019, pp. 222–229.

[95] L. Sarı, N. Moritz, T. Hori, and J. L. Roux, "Unsupervised speaker adaptation using attention-based speaker memory for end-to-end asr," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, Barcelona, Spain, 2020, pp. 7384–7388.

[96] X. Cui, V. Goel, and G. Saon, "Embedding-based speaker adaptive training of deep neural networks," in *INTERSPEECH 2017 – 18$^{th}$ Annual Conference of the International Speech Communication Association*, Stockholm, Sweden, 2017.

[97] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, Calgary, AB, Canada, 2018, pp. 5329–5333.

[98] J. Rownicka, P. Bell, and S. Renals, "Embeddings for dnn speaker adaptive training," in *2019 IEEE Workshop on Automatic Speech Recognition & Understanding*, Singapore, 2019, pp. 479–486.

[99] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, 2014, pp. 4052–4056.

[100] X. Li and X. Wu, "Modeling speaker variability using long short-term memory networks for speech recognition," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015, pp. 1086–1090.

[101] Y. Shi, Q. Huang, and T. Hain, "H-vectors: utterance-level speaker embedding using a hierarchical attention model," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, Barcelona, Spain, 2020, pp. 7579–7583.

[102] K. Veselý, S. Watanabe, K. Žmolíková, M. Karafiát, L. Burget, and J. H. Černocký, "Sequence summarizing neural network for speaker adaptation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, Shanghai, China, 2016, pp. 5315–5319.

[103] M. Delcroix, S. Watanabe, A. Ogawa, S. Karita, and T. Nakatani, "Auxiliary feature based adaptation of end-to-end asr systems," in *INTERSPEECH 2018 – 19$^{th}$ Annual Conference of the International Speech Communication Association*, Hyderabad, 2018, pp. 2444–2448.

[104] L. Sarı, S. Thomas, M. Hasegawa-Johnson, and M. Picheny, "Speaker adaptation of neural networks with learning speaker aware offsets," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, Graz, Austria, 2019.

[105] X. Xie, X. Liu, T. Lee, and L. Wang, "Fast dnn acoustic model speaker adaptation by learning hidden unit contribution features," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, Graz, Austria, 2019, pp. 759–763.

[106] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, 2013, pp. 7947–7951.

[107] K. C. Sim, Y. Qian, G. Mantena, L. Samarakoon, S. Kundu, and T. Tan, "Adaptation of deep neural network acoustic models for robust automatic speech recognition," in *New Era for Robust Speech Recognition: Exploiting Deep Learning*, S. Watanabe, M. Delcroix, F. Metze, and J. R. Hershey, Eds. Springer, 2017, ch. 9, pp. 219–243.

[108] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. de Mori, "Linear hidden transformations for adaptation of hybrid ann/hmm models," *Speech Communication*, vol. 49, no. 10-11, pp. 827–835, 2007.

[109] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid nn/hmm systems," in *INTERSPEECH 2010 – 11$^{th}$ Annual Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, 2010, pp. 526–529.

[110] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[111] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[112] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *2014 IEEE Spoken Language Technology Workshop*, South Lake Tahoe, NV, USA, 2014, pp. 171–176.

[113] P. Swietojanski, J. Li, and S. Renals, "Learning hidden unit contributions for unsupervised acoustic model adaptation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 8, pp. 1450–1463, 2016.

[114] L. Samarakoon and K. C. Sim, "Subspace lhuc for fast adaptation of deep neural network acoustic models," in *INTERSPEECH 2016 – 17$^{th}$ Annual Conference of the International Speech Communication Association*, San Francisco, USA, 2016, pp. 1593–1597.

[115] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid nn/hmm model for speech recognition based on discriminative learning of speaker code," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, 2013, pp. 7942–7946.

[116] S. Xue, O. Abdel-Hamid, H. Jiang, and L. Dai, "Direct adaptation of hybrid dnn/hmm model for fast speaker adaptation in lvcsr based on speaker code," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, 2014, pp. 6339–6343.

[117] Z.-Q. Wang and D. Wang, "Unsupervised speaker adaptation of batch normalized acoustic models for robust asr," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, New Orleans, LA, USA, 2017, pp. 4890–4894.

[118] F. Mana, F. Weninger, R. Gemello, and P. Zhan, "Online batch normalization adaptation for automatic speech recognition," in *2019 IEEE Workshop on Automatic Speech Recognition & Understanding*, Singapore, 2019, pp. 875–880.

[119] C. Zhang and P. C. Woodland, "Parameterised sigmoid and relu hidden activation functions for dnn acoustic modelling," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015, pp. 3224–3228.

[120] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *2012 IEEE Spoken Language Technology Workshop*, Miami, FL, USA, 2012, pp. 366–369.

[121] Y. Zhao, J. Li, and Y. Gong, "Low-rank plus diagonal adaptation for deep neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, Shanghai, China, 2016, pp. 5005–5009.

[122] Y. Zhao, J. Li, K. Kumar, and Y. Gong, "Extended low-rank plus diagonal adaptation for deep and recurrent neural networks," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, New Orleans, LA, USA, 2017, pp. 5040–5044.

[123] L. Samarakoon and K. C. Sim, "Factorized hidden layer adaptation for deep neural network based acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2241–2250, 2016.

[124] T. Tan, Y. Qian, and K. Yu, "Cluster adaptive training for deep neural network based acoustic model," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 459–468, 2016.

[125] M. Delcroix, K. Kinoshita, A. Ogawa, C. Huemmer, and T. Nakatani, "Context adaptive neural network based acoustic models for rapid adaptation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 5, pp. 895–908, 2018.

[126] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *INTERSPEECH 2013 – $14^{th}$ Annual Conference of the International Speech Communication Association*, Lyon, France, 2013, pp. 2365–2369.

[127] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, 2014, pp. 6359–6363.

[128] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, 2013, pp. 7893–7897.

[129] K. Li, J. Li, Y. Zhao, K. Kumar, and Y. Gong, "Speaker adaptation for end-to-end ctc models," in *2018 IEEE Spoken Language Technology Workshop*, Athens, Greece, 2018, pp. 542–549.

[130] Z. Meng, Y. Gaur, J. Li, and Y. Gong, "Speaker adaptation for attention-based end-to-end speech recognition," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, Graz, Austria, 2019.

[131] F. Weninger, J. Andrés-Ferrer, X. Li, and P. Zhan, "Listen, attend, spell and adapt: Speaker adapted sequence-to-sequence asr," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, Graz, Austria, 2019.

[132] Z. Meng, J. Li, and Y. Gong, "Adversarial speaker adaptation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, UK, 2019, pp. 5721–5725.

[133] O. Klejch, J. Fainberg, and P. Bell, "Learning to adapt: a meta-learning approach for speaker adaptation," in *INTERSPEECH 2018 – 19$^{th}$ Annual Conference of the International Speech Communication Association*, Hyderabad, 2018.

[134] O. Klejch, J. Fainberg, P. Bell, and S. Renals, "Speaker adaptive training using model agnostic meta-learning," in *2019 IEEE Workshop on Automatic Speech Recognition & Understanding*, Singapore, 2019, pp. 881–888.

[135] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.

[136] Z. Huang, S. M. Siniscalchi, I.-F. Chen, J. Wu, and C.-H. Lee, "Maximum a posteriori adaptation of network parameters in deep models," 2015. [Online]. Available: https://arxiv.org/abs/1503.02108

[137] X. Xie, X. Liu, T. Lee, S. Hu, and L. Wang, "Blhuc: Bayesian learning of hidden unit contributions for deep neural network speaker adaptation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, UK, 2019, pp. 5711–5715.

[138] R. Caruana, "Multitask learning: A knowledge-based source of inductive bias," in *Proceedings of the Tenth International Conference on Machine Learning*, 1993, pp. 41–48.

[139] ——, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[140] Z. Huang, J. Li, S. M. Siniscalchi, I.-F. Chen, J. Wu, and C.-H. Lee, "Rapid adaptation for deep neural networks through multi-task learning," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015.

[141] R. Price, K. ichi Iso, and K. Shinoda, "Speaker adaptation of deep neural networks using a hierarchy of output layers," in *2014 IEEE Spoken Language Technology Workshop*, South Lake Tahoe, NV, USA, 2014, pp. 153–158.

[142] P. Swietojanski, P. Bell, and S. Renals, "Structured output layer with auxiliary targets for context-dependent acoustic modelling," in *INTERSPEECH 2015 – 16$^{th}$ Annual Conference of the International Speech Communication Association*, Dresden, Germany, 2015, pp. 3605–3609.

[143] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. D. Mori, "Adaptation of hybrid ann/hmm models using linear hidden transformations and conservative training," in *2006 IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, 2006, pp. 1189–1192.

[144] S. Sukhbaatar, E. Grave, G. Lample, H. Jegou, and A. Joulin, "Augmenting self-attention with persistent memory," 2019. [Online]. Available: https://arxiv.org/abs/1907.01470

[145] N. Dehak, P. A.Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *INTERSPEECH 2011 – 12$^{th}$ Annual Conference of the International Speech Communication Association*, Florence, Italy, 2011, pp. 857–860.

[146] J. Frankle and M. Carbin, "The lottery ticket hypothesis: finding sparse, trainable neural networks," in *2019 International Conference on Learning Representations*, New Orleans, Louisiana, United States, 2019.

[147] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," in *2018 International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.

[148] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *2019 International Conference on Learning Representations*, New Orleans, Louisiana, United States, 2019.

[149] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *2017 International Conference on Learning Representations*, Toulon, France, 2017.

[150] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *29$^{th}$ International Conference on Neural Information Processing Systems*, Montréal, Canada, 2015, pp. 1135–1143.

[151] J. Liang, C. Zhao, M. Wang, X. Qiu, and L. Li, "Finding sparse structures for domain specific neural machine translation," in *35$^{th}$ AAAI Conference on Artificial Intelligence*, 2021, pp. 13 333–13 342.

[152] P.-Y. Shih, J.-F. Wang, Y.-N. Lin, and Z.-H. Fu, "Multi-speaker adaptation for robust speech recognition under ubiquitous environment," in *2009 Oriental COCOSDA International Conference on Speech Database and Assessments*, Urumqi, China, 2009, pp. 126–131.

[153] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic, 2021, pp. 3045–3059.

[154] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56$^{th}$ ACL*. Association for Computational Linguistics, 2018, pp. 66–75.

[155] M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel, "Self-attentional acoustic models," in *INTERSPEECH 2018 – 19$^{th}$ Annual Conference of the International Speech Communication Association*, 2018.

[156] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping," in *INTERSPEECH 2017 – 18$^{th}$ Annual Conference of the International Speech Communication Association*, 2017, pp. 1298–1302.

[157] L. Dong and B. Xu, "CIF: Continuous integrate-and-fire for end-to-end speech recognition," *arXiv preprint arXiv:1905.11235*, 2020.

[158] A. Raganato and J. Tiedemann, "An analysis of encoder representations in transformer-based machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018, pp. 287–297.

[159] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[160] M. Xu, D. F. Wong, B. Yang, Y. Zhang, and L. S. Chao, "Leveraging local and global patterns for self-attention networks," in *Proceedings of the 57$^{th}$ Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019, pp. 3059–3075.

[161] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Rwth asr systems for librispeech: Hybrid vs attention," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, 2019, pp. 231–235.

[162] A. Bérard, L. Besacier, A. C. Kocabiyikoglu, and O. Pietquin, "End-to-end automatic speech translation of audiobooks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[163] K. Irie, R. Prabhavalkar, A. Kannan, A. Bruguier, D. Rybach, and P. Nguyen, "On the choice of modeling unit for sequence-to-sequence speech recognition," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, 2019.

[164] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, 2019.

[165] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," in *INTERSPEECH 2019 – 20$^{th}$ Annual Conference of the International Speech Communication Association*, 2019.

[166] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," in *2020 International Conference on Learning Representations (ICLR)*, 2020.

[167] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Łukasz Kaiser, "Universal transformers," in *2019 International Conference on Learning Representations (ICLR)*, 2019.

[168] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016.