Knowledge Transfer in Accent and Dialect Speech Recognition with Self-Supervised Learning

Bachelor's Thesis of

Boyan Zhong

Artificial Intelligence for Language Technologies (AI4LT) Lab
Institut for Anthropomatics and Robotics (IAR)
KIT Department of Informatics

Reviewer:	Prof. Dr. Jan Niehues
Second reviewer:	Prof. Dr. Alexander Waibel
Advisor:	M.Sc. Zhaolin Li

06. June 2023 – 06. Oct 2023

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . . . .

(Boyan Zhong)

Abstract

This paper presents a detailed analysis of accented-robust automatic speech recognition (ASR) using transfer learning. Accents and dialects often possess limited labeled data resources, while exhibiting significant variety compared to standard languages. For this disparity, we demonstrate the potential of knowledge transfer between the end-to-end models in the downstream learning process instead of further exploring the model architecture.

In our study with regards to knowledge, We aimed to discern the difference between English accents and Chinese dialects as well as to investigate the learning difficulties in various downstream tasks associated with accent/dialect ASR across different language systems. The knowledge transfer in this work refers to a leveraged downstream learning process where two threads of knowledge representations are parallel proceeded and compared with each other in order to evaluate their capacity for handling multiple accents. In both mono- and multi-lingual originated threads, knowledge representations are generated by self-supervised pre-trained models and further optimized. Additionally, hypotheses and further approaches were proposed for fine-tuning adapters.

We conclude that the disparities in knowledge between language systems, accents, and dialects can lead to a significant decline in ASR model performance. A leveraged fine-tuning process, where general knowledge representation is gradually merged with the diversity-leveraged target-related accent, is proposed for a better focus on the target accent as a downstream ASR task.

Zusammenfassung

In diesem Beitrag wird eine detaillierte Analyse der automatischen Spracherkennung mit Akzent-Robustik (ASR) unter Verwendung von Transfer-Lernen. Akzente und Dialekte verfügen oft nur über begrenzte Ressourcen an gelabelten Datenressourcen, während sie im Vergleich zu Standardsprachen eine große Vielfalt aufweisen. Für diese Gleichheit demonstrieren wir das Potenzial des Wissenstransfers zwischen den End-to-End-Modellen im nachgelagerten Lernprozess, anstatt die Modellarchitektur weiter zu erforschen.

In unserer Wissensstudie wollten wir den Unterschied zwischen englischen Akzenten und chinesischen Dialekten erkennen und die Lernschwierigkeiten bei verschiedenen nachgelagerten Aufgaben im Zusammenhang mit der Akzent/Dialekt-ASR in verschiedenen Sprachsystemen untersuchen. Der Wissenstransfer in dieser Arbeit bezieht sich auf einen nachgelagerten Lernprozess, bei dem zwei Threads von Wissensrepräsentationen parallel bearbeitet und miteinander verglichen werden, um ihre Fähigkeit zur Handhabung mehrerer Akzente zu bewerten. Sowohl in den ein- als auch in den mehrsprachigen Threads werden die Wissensrepräsentationen durch selbstüberwachte, vortrainierte Modelle generiert und weiter optimiert. Zusätzlich werden Hypothesen und weitere Ansätze für die Feinabstimmung von Adaptern vorgeschlagen.

Wir kommen zu dem Schluss, dass die Wissensunterschiede zwischen den verschiedenen Sprachsystemen, Akzenten und Dialekten zu einer erheblichen Verschlechterung der ASR-Modellleistung führen können. Es wird ein gehebeltes Feinabstimmungsverfahren vorgeschlagen, bei dem die allgemeine Wissensrepräsentation schrittweise mit dem diversitätsgehebelten zielbezogenen Akzent verschmolzen wird, um eine bessere Konzentration auf den Zielakzent als nachgelagerte ASR-Aufgabe zu erreichen.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

The use of automatic speech recognition technology has become widespread due to its remarkable convenience and user-friendliness. Its accuracy has been refined to an exceptional level. However, natural speech frequently deviates from standard speech due to variations in pronunciation [13, 11, 12], which can be classified into two broad categories: accent and dialect.

The term accents refers to variations in how a language is pronounced across different regions or social groups. It is worth noting that, despite these differences, the fundamental grammar and vocabulary of the language remain constant [18]. While an accent pertains primarily to pronunciation, a dialect [6] encompasses more comprehensive linguistic variations. It can also involve variations in grammar, vocabulary, and syntax. Even individuals who speak the same language may use different terms or phrases based on their native dialects resulting from their unique regional or social backgrounds. Commonly, a dialect is also referred to as a strong accent [12].

Most current research on speech recognition neglects the difference between accent and dialect, and frequently uses the two terms interchangeably because both indicate phonetic variations within a given language [18, 43, 51]. Nonetheless, accent and dialect pose different challenges for ASR [29]. Therefore, exploring the differences between accents and dialects is beneficial in constructing datasets as well as ASR architecture.

Previous techniques have dealt with accent information by mapping accented phones to canonical phones [18]. The recent research focuses on enhancing the robustness of automatic speech recognition (ASR) models in handling variations in accents [18]. Several approaches have been investigated, with some of the earlier techniques have been successful in this regard [18, 1, 35, 28, 42, 2]. Modern accented ASR approaches typically integrate model generalization with accent feature engineering in order to improve data efficiency and match standard ASR performance. They tried to make model architecture better equipped to handle variations in accent and generate more accurate results [18, 35]. Through model generalization, it has been observed that acoustic information is shared across different accent data. This allows for more efficient use of limited accent data resources by applying a multi-task learning strategy. By using accent embedding techniques, accent features can be first extracted with an accent identification model. Subsequently, the input features are augmented with these embedded accent features [28]. Despite the numerous existing methods, the performance still suffers from the scarcity of labeled data resources, which is a common problem in ASR but particularly severe and challenging for accent ASR with significant variety [18].

Instead of focusing on the model architecture. We shift our perspective to the learning process. The use of self-supervised learning has become a popular strategy in speech recognition to overcome the problem of insufficient labeled data. This approach is incorporated with knowledge transfer in the leveraged fine-tuning process for downstream tasks. Research studies [33, 3] have shown that this approach can effectively learn acoustic features from unlabeled speech audio, subsequently excel in downstream speech recognition tasks with limited labeled audio.

In addition, experiments conducted by [10, 27, 31] have demonstrated the effectiveness of self-supervised learning models in accent speech recognition with a focus on different dialects and accents. Self-supervised learning and knowledge transfer not only improve resource-efficiency and -effectiveness, but also enhance robustness and performance.

1.2. Research questions

In this thesis, we aimed to carry out an empirical analysis of the persistent challenges associated with acoustic variation in speech recognition and to investigate potential methods for improving the accuracy and efficiency of the model. To address these challenges, we turned our focus to self-supervised learning. In pursuit of this aim, we carried out ASR training on a constrained set of labeled data. Specifically, our analysis encompassed an English dataset spanning three distinct accents, along with a Chinese dataset featuring four dialects. Through our experiments, we aim to shed light on these pressing issues and offer valuable insights.

Shifting our focus to a new perspective on knowledge transfer, we have identified the following research questions to explore further. Our goal was to delve deeper into these questions and gain a better understanding of the topic at hand. With this in mind, we have compiled a detailed list of our *RESEARCH QUESTIONS* as follows:

RQ1*:* Does the difficulty of building speech recognition systems differ between accents or dialects in different language systems?

This is a point that has not been considered in previous research and is also a potential challenge for accent and dialect ASR. The accent or dialect variations in different language systems may significantly influence the way to achieve ASR tasks. Therefore, exploring the relationship between variants in various language systems and further investigating its influence on ASR is crucial for improving accent and dialect ASR performance.

RQ2*:* What factors of the dataset can impact knowledge representation learning?

The proper selection of a dataset is crucial for optimal training of knowledge representation in ASR. From the current study, it is clear that insufficient dataset resources can significantly hinder the performance of accent knowledge representation learning in ASR. Further research is required to explore other dataset factors that may affect the capacity of knowledge representation. Identifying these factors makes the dataset selection more efficient and effective in order to maximize the degree of match between fine-tuned knowledge representation and the target task.

RQ3*:* How to improve the robustness and correctness of an accent ASR model for speaker changes based on the properties of transfer learning?

The first research question examines the challenges posed by dialect or accent variations within various language systems. To address this issue, it is crucial to explore a model that can improve the accuracy of automatic speech recognition (ASR) for dialects or accents regardless of the language system. By investigating the properties of transfer learning, the training process can be optimized and the compact model architecture can be simplified by ignoring the differences between language systems.

Each sub-question contributes to answering our main research question.

1.3. Outlines

The rest of this thesis is structured as follows:

Background:

This chapter establishes the necessary theoretical foundations for the subsequent chapters, covering the basics of speech recognition, training aspects, and the main techniques used in this thesis.

Approaches:

This chapter presents an overview of the strategies implemented in this thesis to achieve its objectives. This includes the modeling strategy as well as the training strategy. The benefits of the selected strategy and its suitability for the experiments conducted in this study are also analyzed.

Experiments:

In this chapter, the experiment's procedure is outlined in a clear and concise manner. Notably, novel operations that differ from traditional methods are explained in detail, with justification for their use provided and explained.

Result and Analysis:

In this chapter, the experimental findings are summarized and listed separately. The corresponding experimental data confirming the findings are listed and analyzed.

Conclusion:

By synthesizing the various findings and corresponding analyses from Chapter 5. this final chapter summarizes the conclusions of this thesis and answers the research questions.

2. Background

In this chapter, the main techniques used in this thesis will be highlighted. First of all, some basics will be introduced about the basics of speech recognition, as well as the training aspects. Second, the main techniques used in this thesis will be presented, in order, transformer, Wav2vec2, CTC loss, and adapter structure.

2.1. Basic Knowledge

This chapter contains three sections focusing on some of the basics necessary to carry out this research or for ASR.

2.1.1. Automatic Speech Recognition

Automatic speech recognition is a technology to recognize and transcribe spoken speeches into readable text. The ASR system takes continuous audio as input and outputs the text. It is designed to be speaker-independent and further developed as an environment-robust, multilingual, and domain-independent system [18]. One of the major challenges in ASR is the lack of labeled data resources for system training [41]. In this work, the ASR system requires training on accented speeches, which are short on both audio and transcription resources.

2.1.2. Accents/dialects and Multilingual

An accent is the way a person pronounces words in a language. It is shaped by his native language or other languages he speaks frequently and even very personally by where he lives or his daily social relationships. Dialect is a particular variation of a language and is spoken only in a local region. Generally speaking, a dialect only differs from its language in the pronunciation aspect, They still share the same grammar system and language model. So that a dialect can be regarded as a heavily accented version of its language.

There are many factors, that can more or less influence a person's accent, Among all these factors, the native language or the native dialect dominates his accent. Although the accent is only part of the acoustic component of a speech, commonly it is not extracted from a speech. By analyzing the accent information, the complete speech audio is collected [35]. Furthermore, a language or a dialect can represent a type of accent. Instead of learning the language and accent separately, the accent is kept as an implicit sub-feature space in the language feature space.

A system, holding the knowledge of multiple languages, is a multilingual system [49]. Here I propose to treat both languages and dialects as members of a multilingual system.

Corresponding to a multilingual system, a so-called multi-accent system should hold the acoustic knowledge of the same multiple languages.

With the help of this generalization, the task of recognizing the speech in language A from a speaker native to dialect 1 of language B, can be transformed into the task of recognizing the speech (in language A) with native accent a further accented by c.

The accents as well as dialects are no longer arranged in a hierarchical category structure but are flattened. So that an accent in a foreign language and an accent in a local dialect do not sit on different levels. Any of those accents can be mixed, dialects and basic (standard) languages can be mixed, just like data can be mixed in the multilingual system.

One intuition here is that people are learning more and more different kinds of languages and even peculiar dialects. Besides this trend of language internationalization, the languages and dialects originated from fewer root ancient languages where the language category and regions were quite different from the current state. Both the future and history imply that it is reasonable to mix resources of languages from different regions and countries for a more general understanding of language. In ASR research, By mixing speeches in various languages, resources are more efficiently used and performance is further improved [49, 37, 5, 17, 50].

2.1.3. Few-shot and Zero-shot

The lack of resources, as a challenge in ASR research, remains a significant problem in accented ASR [41, 23]. For some specific languages or dialects, only a few transcribed speeches and a limited length of unlabeled speeches are available. The method to train the model on a small number of labeled data in the target type is called few-shot. In the worst case, the model even has no chance to observe data in the target type, which is called zero-shot. The basic strategy to improve few-shot or zero-shot is a combination of the general knowledge from various types and the auxiliary information for the specific target type [18].

2.1.4. Seq2seq

Seq2Seq, short for "Sequence-to-Sequence", is a deep learning architecture used in natural language processing and other sequence-based tasks [34]. Sequence-to-sequence training simply means that both input and output data are in a sequence form, more precisely, in a time sequence. For an ASR system, the model input is a sequence of values sampled from a speech audio and the model output is a sequence of characters representing the transcription of that speech. Both sequences can have different lengths and they are implicitly aligned by the model.

2.1.5. End2end

An end-to-end model is a machine learning or deep learning architecture that aims to solve a problem directly from raw input data to output. The end-to-end model is trained as a whole, from input throughout the model to the output. The training data, as a pair of

input and label, is used on both ends of the model. No intermediate processing or feature engineering in sub-models is required.

2.1.6. Encoder-decoder

The encoder-decoder is a powerful deep-learning architecture with two components [34]. The encoder part transforms the input data into a hidden feature vector, which implies an abstract representation. The encoded hidden features vector is further sent to the decoder part, which produces an output representation.

The encoder usually reduces the dimension of the input data and serves to abstract the information in a hidden feature space, While the decoder is responsible for mapping the information back into another concrete representation similar to the label corresponding to the input data [24, 9]. The state-of-the-art transformer model is a typical encoder-decoder sequence-to-sequence model, which will be introduced in detail in the following section.

2.2. Transfer Learning

Transfer learning is a technique used in machine learning where a model that has already been trained for a certain task or dataset is adapted and improved to work on a different but related task or dataset. In the traditional approach to machine learning, models are usually trained from scratch on specific datasets for specific tasks. However, transfer learning makes use of the knowledge acquired during training for one task to enhance performance for a different but related task. This technique is particularly beneficial when there is a shortage of labeled data for the target task or when training from scratch would be a computationally intensive process.

2.3. Transformer

A transformer architecture is fundamentally attention-based and is a type of neural network that learns context in sequential data and the meaning by tracking relationships in the data context [32, 48]. It was first proposed in a 2017 paper by Google Brain team titled 'Attention Is All You Need' [40].

The model applies an evolving set of mathematical techniques, called attention or self-attention, to detect subtle ways even distant data elements in a series influence and depend on each other1. Transformers are among the newest and one of the most powerful classes of models invented to date, driving a wave of advances in machine learning some have dubbed transformer AI1. They are notable for requiring less training time than previous recurrent neural architectures, such as long short-term memory (LSTM).

The transformer model has an encoder-decoder structure and is basically made of attention blocks (in Figure 2.1 [40]).

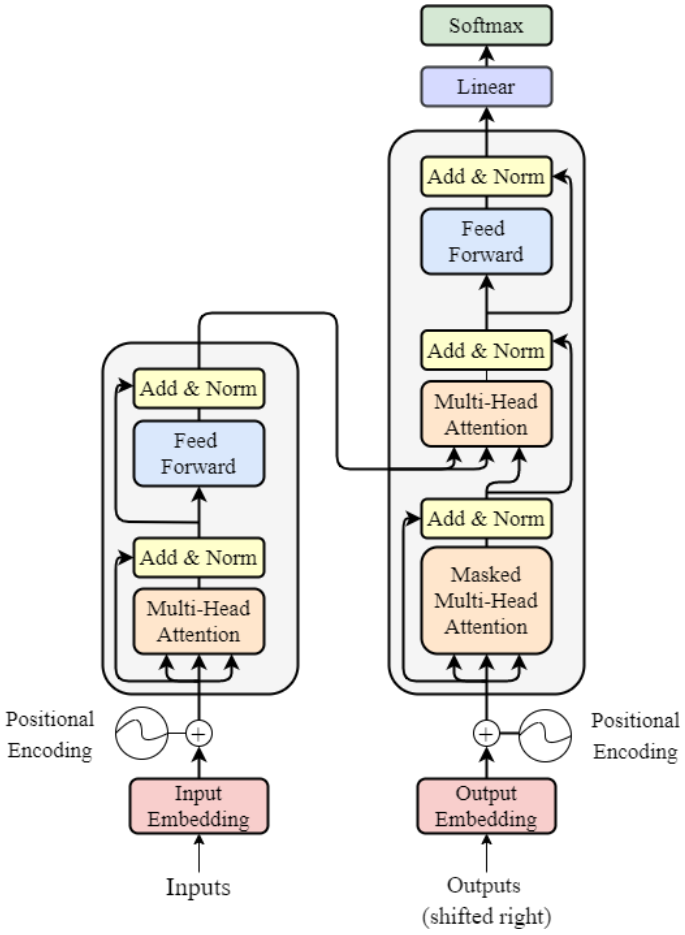

Figure 2.1.: Transformer architecture

2.3.1. Attention Mechanism

The Attention mechanism proposed in the transformer architecture is basically designed to map from a query and a set of key-value pairs to an output [40]. The output vector is the combination of those values in the set of pairs given from the input. In order to compute the combination, the weights for the values need to be calculated by representing the query in the key space corresponding to the values.

Scaled Dot-Product Attention is a basic implementation of the attention mechanism. Another more powerful but also more complex implementation is called multi-head attention, which will be introduced later.

The scaled dot-product attention defines a mapping from three input matrices to an output matrix. The input matrices are Query $\mathbf{Q}$, Key $\mathbf{K}$ and Value $\mathbf{V}$. The Query matrix $\mathbf{Q}$ simply collects queries $\mathbf{q}$. The Key matrix $\mathbf{K}$ consists of keys, which are correspondingly mapped to values $\mathbf{v}$ in matrix $\mathbf{V}$.

The query $\mathbf{q}$ is designed as the keys $\mathbf{k}$. So each query in $\mathbf{Q}$ can be compared with all the keys in $\mathbf{K}$. Those values $\mathbf{v}$, whose corresponding keys are similar to the query $\mathbf{q}$, have more chance to fit the query and contribute more to the answer as the output $\mathbf{o}$ for $\mathbf{q}$. The similarity between the query $\mathbf{q}_j$ and a key $\mathbf{k}_i$ is evaluated by the dot-product, and

it is further used as the weight for the value $\mathbf{v}_i$ for $\mathbf{k}_i$. The output $\mathbf{o}_j$ for $\mathbf{q}_j$ is basically calculated by the combination of values $\mathbf{v}_i \in R^m$, naturally each $\mathbf{o}_j$ is a vector also in R^m. Finally, queries find output vectors [40].

There are different ways that the attention module can collect information to generate the query key and value matrices. Considering a transformer model, the implementation of this scaled dot-product attention builds three parameter blocks sitting in between the input feature vectors and the core attention module. They are responsible for preparing three input matrices for the attention calculation. Three trainable blocks take the same input feature vectors, and generate the query key and value matrices correspondingly. Each of these blocks is typically implemented as a fully connected layer [45].

The layer for the query matrix maps each feature vector in the input sequence $\mathbf{X}$ into a query vector independently and the whole input sequence $\mathbf{X}$ is mapped into the query matrix. In order to represent such a mapping, this fully connected layer requires d inputs and n nodes as outputs.

The layer for the key matrix $\mathbf{K}$ maps each in $\mathbf{X}$ into independently and $\mathbf{X}$ is mapped into $\mathbf{K}$. Same as the layer for query, the fully connected layer $\mathbf{K_M}$ also requires d inputs and n nodes as outputs.

The layer $\mathbf{V}$ for the value matrix maps each x in $\mathbf{X}$ into $\mathbf{v}$ independently and $\mathbf{X}$ is mapped into $\mathbf{V}$. Similar to the other two layers above, the fully connected layer requires d inputs and m nodes as outputs.

After preparing the query key and value matrices, the scaled dot-product attention can then perform the calculation explained in the definition above and output for each input feature vector x a corresponding vector o as a combination of values in $\mathbf{V}$ weighted by the similarity between the $\mathbf{k}$ corresponding to $\mathbf{v}$ and the $\mathbf{q} \in R^n$ for x. Actually, the definition above just tells the brief calculation process, the weights for the combination are usually scaled and operated by the softmax function [45].

$$o = softmax(\frac{x\mathbf{Q_M} \cdot (\mathbf{XK_M})^T}{\sqrt{n}}) \cdot \mathbf{XV_M} \tag{2.1}$$

2.1 helps to pick out the outperforming weights, which are used to find those more important values to contribute to the final output. The softmax function is usually applied for such kind of filtering over a distribution for a backpropagation algorithm, because this function provides good property in derivation. The scaled factor helps to compensate the quick growth in magnitude when the degree n is very large. In such a case, 2.1 takes the risk of handling extremely tiny gradients in the weight distribution [40].

Compared to the traditional recurrent and convolutional neuron-network [40], the latest attention mechanism has its advantages in both practical and theoretical aspects.

In the attention mechanism, the computation of each output for the corresponding element query is independent of the other output. While the computation of the current output requires the previous outputs in the recurrent network. So that all the outputs can be calculated at the same time, and the parallelism is naturally achieved.

Although the convolutional network can also be parallelly calculated, each output only considers the local relationship by using the convolutional kernel. However, in the recurrent network, each output takes into account several previous outputs, which means,

iteratively the current output is related to all the previous elements. Each output in the attention module can even cover all the elements in the input sequence.

In conclusion, the attention mechanism combines both the parallelism and the global dependency [40]. But different from the previous sub-sequence dependency in the recurrent network, where the global relationship is obviously order-dependent, the global relationship in the attention mechanism is order-independent. According to block construction explained above, if the order of elements x in the input sequence $\mathbf{X}$ is changed, the order of key and value rows in the $\mathbf{K}$ and $\mathbf{V}$ will also be changed correspondingly. The order of the weights computed by dot-producing the $\mathbf{q}$ for x and the changed $\mathbf{K}$ will be changed in exactly the same way. As a result, the combination as the output of the attention module will stay the same. In other words, each output can be flexibly related to different elements in the input sequence regardless of their positions.

In order to take into account the position information or sequential relations, additional positional embedding is required in the transformer model [48]. The wav2vec model, popular in ASR tasks and used as the basic architecture in this work, introduces an additional grouped convolution layer for positional embeddings, which will be introduced in the later section.

2.3.2. Multi-Head Attention

Based on the scaled dot-product attention, the multi-head attention (in Figure 2.2 [40]) is developed and actually implemented in the transformer architecture [40, 48]. The multi-head attention applies multiple scaled dot-product attention blocks parallel. For each scaled dot-product attention block, the query key and value matrices are locally mapped into specified spaces, which are independent in between blocks. The implementation of

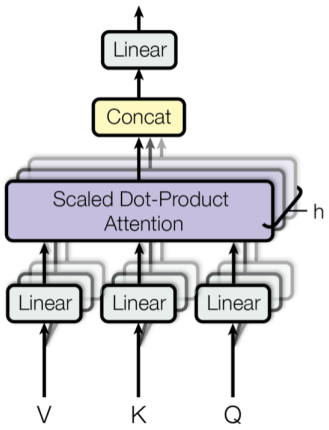

Figure 2.2.: Multi-Head attention

multi-head attention simply takes the implementation of the scaled dot-product attention above as a single block, which includes three layers and the scaled dot-product attention module. So each block works independently in its own representation feature spaces, which are regarded as subspaces of the global query key and value spaces built in the

scaled dot-product attention. In this way, a complex analysis in one scaled dot-product attention is separated into several independent and simpler analysis processes.

All the parallel processes are fed by the same input sequence and their outputs are concatenated into a long vector, which is finally projected into an output space with the same dimension as the value vector space in each block. This multi-head attention mechanism further pushes the parallelism of attention. The attention is not only independent of the position but also of the feature space.

2.3.3. Encoder-decoder Architecture

The critical element of the Transformer architecture is its encoder-decoder structure [32, 48] (in Figure 2.3 [25]), which empowers the model to handle sequence-to-sequence tasks such as machine translation and text summarization. Comprising two primary components, namely the encoder and the decoder, each has its unique roles and attributes. The encoder

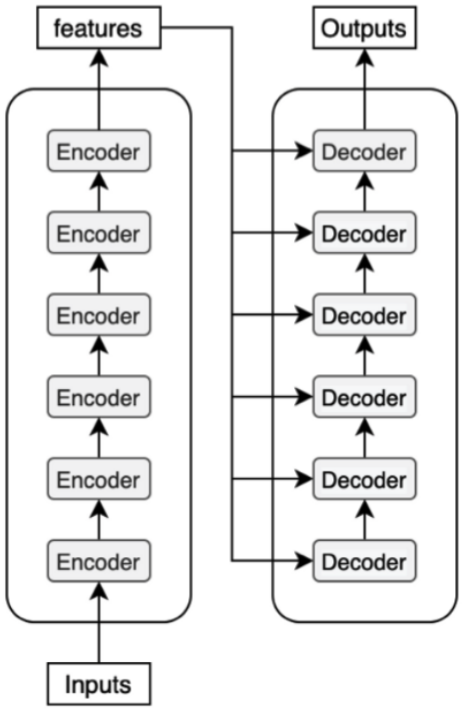

Figure 2.3.: Encoders-Decoders

takes the input sequence (e.g. a sentence in a source language) and processes it step by step. Unlike recurrent neural networks (RNNs) or convolutional neural networks (CNNs), transformers do not process sequences sequentially [40].

First, get the representation vector X for each word of the input sentence, X is obtained by adding the embedding of the word (Embedding is the Feature extracted from the original data) and the embedding of the word position.

In the second step, the vector matrix (each row is a representation of a word x) is passed into the encoder, and after 6 encoder blocks, the encoded information matrix C of the sentence can be obtained.

The matrix C has n rows and d columns, where n is the number of words in the sentence, and d is the dimension of the vector. The dimension of the matrix output from each encoder block is exactly the same as the input. Then the matrix C as the output of the encoder passed to the decoder. The decoder will translate the next $word_{i+1}$ according to the currently translated sequence $word_1, \ldots, word_i$, and need to mask the words after $word_{i+1}$.

This encoder-decoder architecture is basically built by the attention blocks [48]. There are three typical ways how attention mechanism is used here:

- Encoder self-attention Within each attention block in the transformer encoder, the attention layer can access any element in the input sequence. By allowing each element to attend to any position in the same sequence, the block is capable of capturing information about the entire sequence regardless of the length of the input sequence. This kind of attention block is also called auto-encoding. It is suitable for tasks, where the input requires understanding, such as sentence classification and automatic speech recognition. The latter is the topic of this work, so that the transformer encoder and the encoder self-attention will be further discussed in a concrete transformer called as Wav2vec model.

- decoder self-attention Within the first several attention blocks in the transformer decoder, the attention layer can access those elements already generated in the output sequence. In other words, each element in the final output sequence only has chance to attend to any earlier generated elements sitting on the left in the same output sequence. Similarly, this type of block is able to capture the information conditioned on the earlier generated subsequence. This kind of attention block is often called auto-regressive and is best suited for those generative tasks like text generation.

- encoder-decoder attention In the transformer decoder, those later attention blocks taking the output sequence from the transformer encoder follow the first several attention blocks introduced above. This group of attention blocks sits at the end of the transformer and collects information from both input and output sequences. Within this type of block, the attention layer can access not only all the elements in the sequence from the encoder but also those earlier generated elements in the output sequence. So that it can learn from the information in both the hidden representation of the input sequence and the previously generated subsequence. This type of attention block can be called sequence-to-sequence. It is also good at generative tasks which also require a given input, for example, translation and summarization.

2.3.4. Positional Embedding

In addition to word embedding, Transformer also needs to use positional embedding to represent the position of the word in the sentence [40]. Since Transformer does not adopt the structure of RNN, but uses global information, it cannot use the order information of words. Therefore, Transformer uses position embedding to preserve the relative or absolute position of the word in the sentence. Position embedding is represented by PE, which has the same dimensions as word embedding, and can be obtained by training or by using a formula. The latter is used in Transformer, and the formula is as follows:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{2.2}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2.3}$$

where pos denotes the position of the word in the sentence, d denotes the dimension of PE (as in Word Embedding), $2i$ denotes the even dimension, and $2i + 1$ denotes the odd dimension (i.e., $2i \leq d, 2i + 1 \leq d$). Then by adding the word embedding and the positional embedding of the word, we can get the representation vector x of the word, and x is the input to the transformer.

...

2.4. Wav2vec

The Wav2vec 2.0 architecture is built upon the Transformer encoder and features a speech-adapted training objective, comparable to BERT's masked language modeling objective. This new method enables efficient semi-supervised training by pre-training the model on a large quantity of unlabeled speech, followed by fine-tuning on a smaller labeled dataset. In wav2vec 2.0's original paper [3], the authors demonstrated that fine-tuning the model on only one hour of labeled speech data could beat the previous state-of-the-art systems trained on 100 times more labeled data.

The pre-training and fine-tuning of a Wav2vec 2.0 model take slightly different components in its architecture and also different procedures [3, 4, 20], which will be explained in the following sections. The pre-trained model can be used for various speech tasks, such as speech recognition, emotion and speaker detection, and language identification.

There are four key components: feature encoder, context network, quantization module, and contrastive loss (used for pre-training) (in Figure 2.4 [4])

2.4.1. Feature Encoder

A feature extractor is in charge of preparing input features for audio or vision models. This includes feature extraction from sequences, e.g., pre-processing audio files to Log-Mel Spectrogram features, feature extraction from images e.g. cropping image image files, but also padding, normalization, and conversion to Numpy, PyTorch, and TensorFlow tensors. The feature extractor block is briefly optimized in the pre-training stage of a Wav2vec 2.0 model, and will usually be frozen in the following fine-tuning stage.

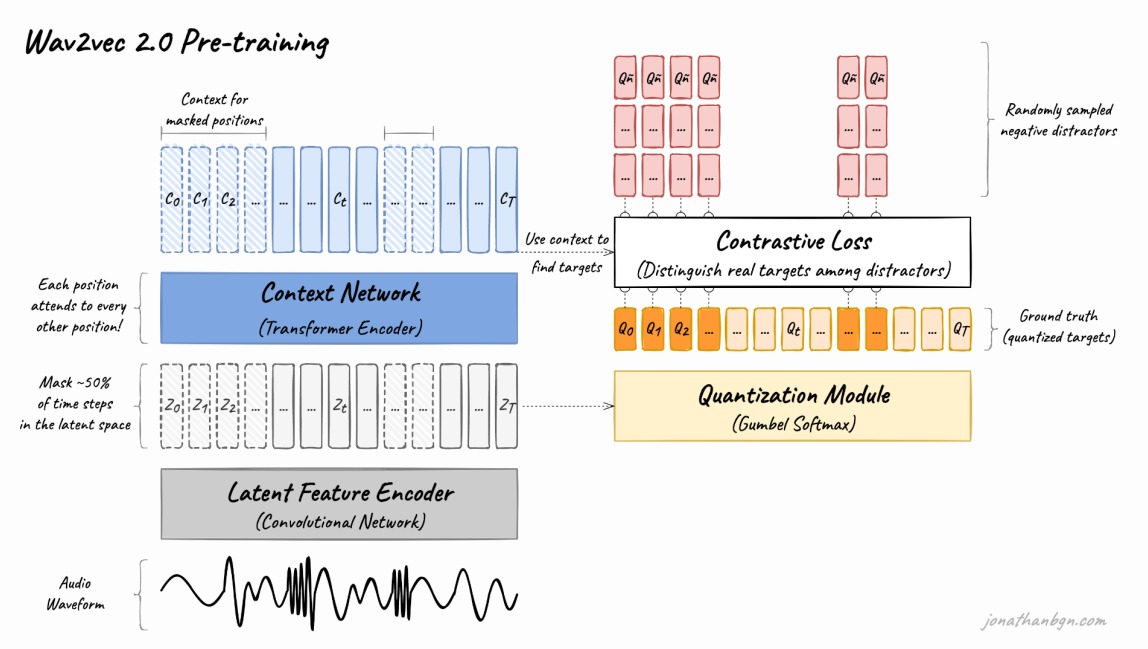

Figure 2.4.: Wav2vec 2.0 Pre-training

2.4.2. Context Network

Wav2vec 2.0 has a Transformer encoder at its core, which is responsible for processing the latent feature vectors from the previous feature encoder. The model comes in two versions, BASE and LARGE, with the BASE version having 12 Transformer blocks and the LARGE version having 24. These Transformer blocks are built based on attention layers and are essential for accurately interpreting the input data and generating high-quality output. The transformer architecture will not be described here. Unlike the original Transformer architecture, the wav2vec model learns relative position embeddings through a newly grouped convolution layer [26].

2.4.3. Quantization Module

Transformers face a significant challenge when it comes to processing speech due to its continuous nature. Written language is made up of words or sub-words, resulting in a limited vocabulary of discrete units. However, speech lacks such natural sub-units. While we could use phones as a discrete system, it would require humans to first label the entire dataset beforehand, which would make it impossible to pre-train on unlabeled data. Wav2vec 2.0 solves this problem by learning distinct speech units by sampling from the Gumbel-Softmax distribution [22]. These units consist of codewords sampled from codebooks, which are groups of possible units. The codewords are then combined to form the final speech unit. Wav2vec 2.0 uses two groups, each with 320 possible words, for a theoretical maximum of 102,400 speech units. As shown in the figure, the input latent features are combined into a matrix where each row is a latent feature, which is multiplied with the codebooks generated above to obtain logits: a score for each of the possible

codewords in each codebook. After subjecting the obtained logits to Gumbel softmax (similar to the operation of *argmax*), a one-hot matrix is obtained, in which each row t represents the position of the words referred to by Z_t in the codebook. After multiplying the result with the quantization projection matrix, we get the final quantization result Q as a sequence of q_t, each corresponding to the input Z_t.

2.4.4. Pre-training and Training Objective

In the pre-training phase, unannotated speech data is utilized to train the model via a contrastive task. Within the latent space, approximately 50% of the projected latent feature vectors undergo a random masking process. These masked positions are subsequently substituted with the same pre-trained vector, denoted as Z'_M , before being input into the Transformer network. The quantization module joins before this masking process and it is still fed by the unmasked feature vectors Z_t.

With the help of a codebook in a parallel quantization module, the training algorithm concludes in a contrastive loss function. In order to match the dimension of the quantized speech units Q_t, the final context vectors are passed through the final projection layer. In the final output sequence of C'_t, each position referring to the masked position in Z' will be compared to a true target as well as a set of distractors.

Obviously, the quantization result Q will be used as the positive true target and marked as Q_p. While the model will uniformly sample one hundred negative distractors from other positions out of those masked ones in the same output sequence and mark them as $Q_{\tilde{n}}$. Then, the comparison is proceeded by calculating the similarity (cosine similarity) [46] between the projected context vector C'_t and the true positive target Q_p, as well as all negative distractors $Q_{\tilde{n}}$.

In the contrastive loss function, high similarity with the true positive target is rewarded but high similarity with negative distractors is penalized, in order to evaluate the quality of the retrieval for those masked positions.

In pre-training, diversity loss is used to ensure that the model uses all codewords equally. This prevents the model from consistently selecting a small subset of codebook entries and encourages a more diverse use of the available options [3].

2.5. Connectionist Temporal Classification

Connectionist Temporal Classification(CTC) is a mathematical framework that is widely used in various sequence-to-sequence tasks, including automatic speech recognition (ASR) [15, 44]. Its primary function is to align input sequences (like speech signals) of different lengths with output sequences of varying lengths. The great thing about CTC is that it doesn't require a one-to-one correspondence between the input and output elements [16]. In the realm of speech recognition, the task of aligning the characters in a transcript to their corresponding audio presents a significant challenge during training due to the natural variation in people's speaking rates. This issue must be taken into consideration when devising effective methodologies for aligning audio and text. Consider mapping input sequences $X = [x_1, \ldots, x_T]$ to the corresponding output sequences $Y = [y_1, \ldots, y_U]$,

such as transcripts. The model should maximize the probability it assigns to the correct answer by computing the conditional probability $p(Y|X)$ and then using gradient descent. And by using the model to try to solve:

$$Y^* = \underset{Y}{argmax} \ \ p(Y|X) \tag{2.4}$$

then ideally Y^* can be found efficiently. It doesn't need to align the input and output in CTC, but it is crucial to understand how the input path maps to the output to calculate the probability. This is because multiple output paths may correspond to a single output result. A naive approach for aligning X and Y is to allocate an output character for every input step and combine the duplicates. However, in speech recognition, silence can occur within the input without corresponding output. Therefore, it is illogical to align every input step to an output. And it is impossible to generate outputs with consecutive characters by collapsing repetitions. To get around these problems, CTC introduces ϵ as a blank token to the set of allowed outputs. With that said, the CTC provides the probability expressed as follows:

$$p(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X) \tag{2.5}$$

If we calculate the score for each alignment directly and add them all up, there could be an enormous number of alignments, which would be too slow. To solve this problem we merge two alignments if they have reached the same output simultaneously. Now that the loss function can be computed efficiently, the subsequent step is to calculate the gradient and commence training the model. The CTC loss function is differentiable concerning the per time-step output probabilities as it only involves the summation and multiplication of these probabilities. For a given training set D, the model's parameters are adjusted to minimize the negative log-likelihood: $sum_{(X,Y)\in D} - \log p(Y|X)$

2.6. Adapter

The adapter module is usually associated with the adapter tuning technique [19]. It is often used in pre-trained language models like the state-of-the-art transformer model. When the language knowledge learned by the attention layers in the pre-training is transferred into a downstream task, the adapter modules stand in and are fine-tuned. In the fine-tuning process, because only these adapter modules with less amount of trainable parameters are optimized and those attention layers with tons of parameters are frozen, this kind of tuning is also known as parameter efficient fine-tuning [19] and the adapters are called attention adapters. For different downstream tasks, only the training on the parameters in the adapter modules makes a difference. Which means that, it is only necessary to switch the adapter modules while switching different tasks.

The two major objectives of the adapter are: 1. to speed up the training for a new language representation. 2. to preserve the memory of those old (or previous) language

representations. Both objectives are achieved by allowing only the training on the parameters in adapter modules. So that every time there comes a dataset with new language representation, instead of the whole large transformer model, only a new set of light-weighted adapter modules needs to be optimized. And for those previous language representations, by switching back to the previous corresponding set of adapter modules, the state of the whole model turns exactly back to the previous one. The memory separated in the frozen attention layers and the corresponding old set of adapter modules can be simply retrieved.

In this work, the attention adapter from the massively multilingual speech project is used [36]. So in the following section, the MMS attention adapter will be briefly explained.

The Massively Multilingual Speech project is developed by a Facebook team [36]. That project dramatically increases the supported languages from hundreds to thousands within some speech-related tasks. They tried to leverage a huge amount of self-supervised learning into smaller pieces and guarantee data- as well as time efficiency at the same time. Among all speech technologies, automatic speech recognition is the exact topic focused on in this work. So the training approaches and the modeling proposed for ASR in that project will be introduced here.

For an ASR system, the basic model architecture used in that project is also a wav2vec model and a CTC head is typically added for the fine-tuning stage. In the pre-training stage, the original wav2vec model shares all attention layers across different languages, but within further fine-tuning, it exposes only the weights within the additional adapter blocks to each language correspondingly.

In that MMS project, the adapter module architecture is inspired by Houlsby's approach [19], where the adapter consists of a feedforward down-project layer, a nonlinearity layer, and a feedforward up-project layer and also contains a residual connection. Similarly, the adapter in the MMS project keeps all those layers as well as the connection, takes a layer norm layer before the feedforward down-project layer, and applies a ReLU activation for the nonlinearity. In both approaches, the adapter modules are set up within each transformer block. The difference between them is how the adapters are added to each block. In Houlsby's approach, after both the attention layer and the feedforward layer in the transformer block, there will be an adapter module added before the layer normalization. Furthermore, an additional feedforward layer is inserted directly before each adapter module, in order to reshape the feature vector before it is fed to the following adapter. In this way, any adapter is guaranteed to take an input feature vector whose size is the same as the input of the whole transformer layer. However, in the approach of the MMS project, the adapter module is simply put at the end of each transformer layer. All the layer normalizations here are preprocessed. Two layernorm layers sit before the attention layer and the feedforward layer respectively in the Wav2vec transformer layer. The layernorm layer in the adapter module also sit before the feedforward down-project layer as well. So the adapter actually sits right after the last feedforward layer in each Wav2vec transformer layer.

3. Knowledge Transfer from Self-Supervised Learning

This chapter presents an overview of the thesis's implemented strategies of knowledge transfer learning to achieve accent and dialect ASR downstream tasks. It analyzes the benefits of the selected strategy and its suitability for the experiments conducted in this study, with a specific emphasis on the selection process and the underlying rationale.

The training process, in this work, started from a pre-training stage followed by fine-tuning stages. The first part provides a detailed explanation of self-supervised learning in the pre-training stage, which was utilized in the upcoming experiments. The thesis discusses the benefits of this technique and how it was applied in this experiment.

In the second part, the training stages were categorized into pre-training- and fine-tuning-stage. The multilingual system is introduced here. Two subsections provide detailed descriptions and benefits of multilingual representation in pre-training and fine-tuning stage respectively, along with reasons for their suitability for use in this research.

The third part describes the innovative knowledge transfer strategies in fine-tuning stages used to improve model performance. It starts by giving an objective overview of the functionality and advantages of fine-tuning as a widely used method for knowledge transfer. This is followed by a detailed analysis of the research question, highlighting the possibilities of using fine-tuning for experimental purposes and the reasons behind this approach.

3.1. Self-Supervised Learning

Before commencing training, it is essential to determine the appropriate baseline model for the main purposes. Thus, in this report, the pre-trained wav2vec2 transformer model was chosen as the baseline. As highlighted in Chapter 2.4, the wav2vec2 technique enables self-supervised learning on an extensive range of unlabelled voices. Self-supervised learning trains the model on the input data without labels. In order to improve the parameters based on the loss- or reward-function, the value predicted on part of the input is evaluated by comparing it to the other part of the input instead of to the corresponding label. Generally speaking, the model is trained on the input with concealed part to learn that concealed part in the same input, which is so-called self-supervised learning. The self-supervised learning actually optimizes the wav2vec2 encoder as an acoustic model. This process is commonly referred to as pre-training.

The advantages of implementing this pre-training process are the following:

- Efficiency: Training deep neural networks from scratch requires significant computing resources, including powerful GPUs and large amounts of data. Pre-trained models save time and resources because they are already trained on large datasets.
- Generalization: Pre-trained models typically require large amounts of unlabeled data to generalize well across tasks and domains. In this thread, it is generally accepted that models need to generalize well to perform better ASR, and thus the use of pre-training enhances the generalization ability of models to maximize performance. At the same time, the overfitting is naturally avoided.
- Reduced Need for Labeled Data: Fine-tuning a pre-trained model requires less labeled data than training from scratch, making it ideal for situations where labeled data is expensive or time-consuming. In this experiment, we have shown in Chapter 1 that the scarcity of speech data is a problem facing ASR today, and that the use of pre-training can effectively alleviate this problem by reducing the need for data.

3.2. Multilingual Representation Learning

The training process, in this work, was a leveraged downstream fine-tuning process, which started from a pre-training stage and followed by fine-tuning stages. We observed on the transfer of the mono- and multi-lingual knowledge representation along the pre-training as well as the fine-tuning stages. In the section, the multilingual representation is focused and discussed within both types of training stages respectively.

For accented speech recognition, a multilingual system can be extended to a multi-accent or multi-dialect system, which is previously explained in Chapter 2 2.1.2. Multi-lingual or -accent representation learning requires the multi-lingual or -accent datasets to share the same block of parameters during the training process so that the parameters in this specific block can learn knowledge from all those different kinds of data resources in order to form a so-called multi-lingual or -accent representation. By using the wav2vec 2.0 model, the set of all the attention blocks is responsible for learning the acoustic representation. As the core of the acoustic model, attention blocks hold the multi-lingual or -accent representation.

The wav2vec 2.0 model has the chance to learn multi-lingual or -accent representation in the pre-training stage as well as in the fine-tuning stage. In order to have a better view of multilingual representation learning, the model also learned some comparable mono-lingual or -accent representations in our approaches.

3.2.1. Multilingual Representation in Per-training

After determining the baseline structure, various pre-trained models were acquired based on the given pre-training data, which can be found on Huggingface [21]. For the purpose of comparison, I had opted for two diverse models as the baseline based on the variety of pre-trained languages. One of the models was referred to as the multilingual model. This involved utilizing a speech dataset sourced from various languages in the pre-training phase without labels. In contrast, the alternative baseline was a monolingual model, which

implies that the model only received a single unlabelled dataset from one language during the pre-training phase.

Because the input datasets provided acoustic knowledge in different degrees of variety during the pre-training stage, the pronunciation representations learned by the model may also have variation. A multilingual model learning from a greater number of pronunciations had the potential to reach a more general representation compared to a monolingual model. The purpose of selecting these two pre-trained model variants was to determine the effect of training on multiple languages in the pre-training phase on speech recognition performance.

3.2.2. Multi-accent Representation in Fine-tuning

The pre-trained models were further fine-tuned on several languages separately, each consisting of transcribed speech datasets containing various accents or dialects. For the purpose of comparison, the baseline was trained on two kinds of datasets. One kind was referred to as the multi-accent dataset. It collected as many types of accents in the same target language system as possible into one dataset. Another kind, in contrast, was the mono-accent dataset, which took only a single type of accent in the target language system.

Although the training datasets were labelled in fine-tuning stage instead of unlabelled in the previous pre-training stage, the learning of accent knowledge happened only within the wav2vec 2.0 encoder as the acoustic model. Similar to the situation in the pre-training stage, a model could learn different accent representations from these two kinds of datasets. A model learning from a variety of accents may be able to form a more general accent representation than that learning only from a mono-accent dataset. We built these two kinds of datasets in order to show the effect of training on multiple accents or dialects of the same language in the fine-tuning phase on the performance of accented speech recognition.

Although the effectiveness of a multi-lingual or -accent representation system depended on the model architecture of our baseline, the quality of our training datasets, and our leveraging fine-tuning strategies introduced in the next section for improving the response accuracy to the multiple accents or dialects. We still expected a better performance of the multi-lingual or -accent representation learned from a variety of languages or accents [49], based on the following advantages of a multilingual system in ASR:

- Shared Semantic Space: Multilingual systems create hidden word representations in a shared semantic space across multiple languages. This framework enables positive parameter transfer, which is particularly useful for languages with limited resources.

- Handling Low-resource Languages: Multilingual systems can benefit from transfer learning by utilizing the knowledge obtained from languages with abundant resources to learn languages with fewer resources. This technique proves to be particularly effective when there exists a substantial amount of parallel data that can aid in establishing useful mappings between languages. By facilitating cross-lingual transfer, this process can ultimately enhance the performance of other languages

that typically lack resources. Additionally, the effectiveness of the multilingual model usually rises along with the quantity of incorporated languages.

- Unsupervised Learning: Some multilingual systems utilize unsupervised learning methods to improve low-resource language representation across different languages. This involves creating word translation pairs from monolingual text to enhance pre-trained language model alignment.
- Zero-Shot Learning: Multilingual systems can address the zero-shot problem by using multilingual data to acquire knowledge of translation directions not included in the available training material. This is accomplished through an iterative self-training process that gradually improves the system's ability to handle zero-shot directions by relying solely on monolingual data.
- Increased Training Data: Expanding the amount of training data can greatly benefit multilingual models, especially since they are structured to integrate acoustic data from diverse source languages. Such an approach allows for a more comprehensive contextual range, which is crucial for achieving precise and significant results.
- Cross-lingual Task Adaptation: Multilingual models can benefit from optimization through fine-tuning on a high-resource language for a particular task before application to that same task in a low-resource language. This multilingual system approach produces superior outcomes compared to training directly on the low-resource language.
- Robustness: During training, exposure to a variety of languages can enhance the model's robustness by requiring it to generalize across different linguistic structures and nuances.

3.3. Knowledge Transfer

In order to observe how the knowledge representation behaved along the leveraged transfer learning on downstream accent ASR tasks, various approaches are proposed and explained in the section.

The above two sections have shown that the amount as well as the variety of the training data resources can improve the performance of accented speech recognition by learning a more general language acoustic representation. The more general knowledge we have, the more various phenomena we can understand. General knowledge supports our analysis and understanding in a specific task, which was to transcribe the accented speeches in this work. For this purpose, we proposed several leveraging fine-tuning strategies to transfer the general language acoustic knowledge gradually into more specific accented speech recognition tasks.

fine-tuning, as a kind of transfer learning, further optimizes the weights in a pre-trained or fine-tuned model based on the new dataset [47]. It can be performed on the entire neural network, or on only a subset of its layers, in which case those layers that are not being fine-tuned require to be "frozen", so that they will not be updated during the

backpropagation phase. Since the selected baseline model was pre-trained, subsequent approaches were grounded on these baselines for fine-tuning. Additionally, incorporating an adapter into the architecture involved fine-tuning certain parameters while freezing the rest, just like in the preceding tip when the adapter was introduced.

The following approaches endeavored to employ various fine-tuning strategies to enhance the model's performance. Of course, different kinds of datasets mentioned in the next section were also applied within each approach for comparison.

3.3.1. One-Step Fine-tuning

The first fine-tuning approach was the one-step fine-tuning(1-step-FT), which is the most popular and straightforward approach. According to the wav2vec 2.0 approach [3], due to semi-supervised learning, with only a limited number of labeled voices, the model can be pre-trained and fine-tuned (in Figure 3.1 [14]), resulting in far superior outcomes than if the model was directly fine-tuned using a hundred times more labeled voices.

Figure 3.1.: Pre-training and Fine-tuning

In light of the limited accent data currently available, as previously mentioned in the initial section, pre-training with wav2vec might significantly enhance training. We expected that, multilingual acoustic knowledge transferred from the pre-trained model could help the learning of the accent representation in this one-step fine-tuning approach.

In this approach, datasets for different languages were selected to fine-tune the baseline model as the pre-trained model. By doing this, we could observe the relationship between the model's performance and the dataset used. This approach also enabled us to compare the performance of different models on a common test target.

3.3.2. Two-Step Fine-tuning

After completing the one-step fine-tuning process, several fine-tuned models were generated, which could be used as the basis for the next step of fine-tuning. This procedure was the two-step fine-tuning(2-step-FT). To begin with, an appropriate basis needed to be chosen within those result models from the one-step fine-tuning, because when running two-step fine-tuning, one wants to maximize the level of the generality of knowledge migrated through transfer learning from the basis and thus improve model performance

in the specific task. We expected that, the more general the transferred multi-accent knowledge was, the more helpful that knowledge could be for further learning the specific accent representation. Therefore, such kind of model was preferred as a baseline, that had been one-step fine-tuning on the dataset containing more various accents. This implies that those models that had achieved the ability to recognize the speech in the language of the accent in the one-step fine-tuning, were more capable of improving their accent recognition by further refining or specifying the accent knowledge, especially on the pronunciations of some typical accented words in the two-step fine-tuning.

During the second fine-tuning stage, the objective was to enhance the model's efficiency and performance solely for a specific accent. Therefore, only these mono-accent datasets instead of multi-accent datasets were employed for two-step fine-tuning.

3.3.3. Adapter Fine-tuning

Furthermore, we tried to change the model at the structure level by incorporating the adapter structure, with the expectation of enhancing the model's performance. As outlined in Chapter 2.6, adapters have several advantages when used in conjunction with a pre-trained model. Typically, the parameters of the pre-trained model are frozen during fine-tuning, with only the parameters of the Adapter, LayerNorm, and task-related layers being updated during the fine-tuning stage. This approach reduces the number of parameters during fine-tuning, thus maintaining the efficiency of the model. The paper shows that parameters can be reduced without compromising model performance [39, 38].

Based on the aforementioned article, we can discern the adapter structure. An adapter added following the attention block is commonly designated as an attention adapter. Correspondingly, an encoder adapter is incorporated directly after the whole encoder block.

The difference between this adapter fine-tuning approach and the one-step fine-tuning approach is that the accent knowledge learned from the new data used in the adapter approach was actually only stored in the adapter component, but the knowledge from the new data used in the one-step fine-tuning could actually influence the entire model including the pre-trained language knowledge kept in the attention layers. Because the pre-trained knowledge is general enough to directly support the further learning of accent knowledge, we expected that the further optimization of the general knowledge could be saved for higher efficiency with little loss on performance.

3.3.4. Two-Step Fine-tuning with Adapter

The same as the 2-step-FT introduced above, this approach with adapter also tried to transfer the multi-accent knowledge from 1-step-FT to ease the learning of specific dialect representation. Considering that, the dialect learned in the 2-step-FT is a kind of variation on the phonetics learned in the 1-step-FT. By retaining as much of the memory on the multi-accent knowledge transferred from the first step as possible, the 2-step-FT on the relevant dialect knowledge could maintain efficiency without compromising on the performance. With a similar motivation in the previous adapter fine-tuning approach, which was designed based on the characteristics of the adapter, I decided to use the

attention adapter again for the training during the second stage of fine-tuning and kept the rest of the process the same as a normal 2-step-FT.

Knowledge transfer has similar properties to transfer learning, which shares the generalization and efficiency advantages with pre-training and offers more.

- Generalization: The technique of transfer learning is useful for allowing a model to recognize and apply general patterns within data. Fluctuations in the data can occasionally cause complications for a model; transfer learning mitigates the impact of these fluctuations by accounting for real-world data's dynamic nature.
- Efficiency: Transfer learning can save time and energy when training a new model in a related problem domain with limited training data. The transfer learning approach allows knowledge to be transferred from a previously trained model to a new model, resulting in enhanced performance.
- Overcoming Data Deficit: In real-world scenarios, it is common to encounter tasks with data deficits and models that lack generalization. Transfer learning mitigates these challenges by enabling us to use pre-trained models for other tasks.
- Domain Adaptation: It enables domain adaptation, where a model trained in one domain can be adjusted for use in another domain. This study transfers knowledge from a multi-lingual domain to a multi-accent domain.

In conclusion, the utilization of knowledge transfer may enhance the efficiency of the experiment when data is unavailable.

4. Experiments

In this chapter, the experiment's procedure is outlined in a clear and concise manner. Notably, novel operations that differ from traditional methods are explained in detail, with justification for their use provided.

The first section primarily describes the data processing procedure preceding the experiment. Then the used model is described. Finally, this section presents the model evaluation criteria and the process for using them.

4.1. English and Chinese Data Setup

This section first provides a detailed description of two selected resources to give more information about English and Chinese datasets including the reasons for choosing. Then the operations performed on the experimental datasets are described.

4.1.1. Dataset Resource

Two resources of datasets were chosen for training: Common Voice [7] and Magic Hub [30].

Firstly, the Common Voice database is publicly accessible, easily downloadable, and can be used for training purposes. The dataset offers a wide variety of languages, thereby providing an extensive range of language options for the main thesis. In addition, Common Voice allows for the recording of audio with subtitles by anyone, facilitating the inclusion of multiple accents, including non-native ones, within the same language. Additionally, all datasets are labeled to indicate the type of accent, the speaker's gender, and age. The audio is sent to various speakers for confirmation of subtitle-audio alignment. The goal is to have strong accents where appropriate for clarity and comprehension. The accent markers enable easier manipulation of the dataset to obtain the desired parameters for the main thesis. However, Common Voice has limitations; in certain Asian languages, the audio prioritizes standard pronunciation and it is not feasible to ascertain the presence of accents.

Another selected resource is Magic Hub, an open-source data platform developed by Beijing Magic Data Technology Company Limited to assist AI developers with model training. Magic Hub contains datasets for different tasks, and the dataset used for this thesis can be easily found. This resource was chosen because it features numerous Asian audios and differentiates between various accents and dialects, which can compensate for the lack of Asian accents in Common Voice.

Once the data source to be used had been identified, the exact language(s) to be selected from the data source to be used for the dataset could be determined.

Here I chose English and Chinese datasets for the experiment. There were several reasons for choosing English: in Common Voice, the English dataset is the largest, which can prevent the dataset from being too small. English is a very widely used language, spoken in different parts of the world, even as a second language in many countries, so English accents are relatively rich, the accents available in the English dataset are also diversified. On the other hand, there are now more available models trained in English, and when using these languages for training, there is a greater variety of models to choose from, and it is also easier to compare with existing studies.

Since this thesis aimed to study the interaction between languages and their corresponding accents, it is necessary to select languages with large differences in pronunciation styles. In summary, Chinese was chosen as the second language for training.

4.1.2. Audio Process

After determining the language for experiments and the data resource, the datasets from these resources could be loaded. By perspective the details in each dataset, the datasets from Common Voice have a mixture of accented with the non-accented dataset for English and a non-accented dataset for Chinese. For the mixture English dataset we call it EN-dataset following, and call the non-accent Chinese dataset CN-dataset. So the first step was to derive the accent audio from the EN-dataset.

For this purpose, the dataset was loaded from huggingface. Then the accent of each audio was shown under the label 'accent'. According to this label, a table of EN-dataset shows information about the accents and the corresponding audio durations (see EN-dataset time length in the table in A.7). The speakers are either native or non-native. They can be from any area of the world and carry their own accents from their dialect or their native language. That means one speech can contain one or more accents. There are various types of accents, including non-native speaker accented speech, English-as-a-Second-Language (ESL) accents, and English-as-a-Native-Language (ENL) accents from different regions. The regions could refer to different parts of a country or various countries. The time length of each audio was calculated by dividing the length of the array, which contained the resampled signal in float type, by the corresponding sampling rate.

The other different accents dataset was extracted from this origin dataset. The most widely used US accent English(accents from the United States) was chosen as the first accent dataset to be extracted, whose accent label contained 'United States English'.

With a similar method, we selected audio that contained 'India' as an IN-dataset, because IN-accent English is also widely used and has pronunciation differences from US-accent English. These two accents have their own characteristics and the audio duration is not too short compared to other accents. These two accents are relatively well-defined in the table and do not create too much ambiguity.

In addition, I selected another English data from Magic Hub with a more pronounced accent: Pakistani English (PAK), which, unlike the other two accent datasets, is a dataset used by non-native speakers.

Using the same method of calculating audio duration, the total length of the CN-dataset was also calculated. But unlike the EN dataset, there are no accents in the CN-dataset, so there was no need to categorize the dataset. The accented dataset was available by

downloading from Magic Hub. We selected the 'ASR' dataset for training. Because in Magic Hub, there exist three kinds of datasets for ASR: mandarin, accents, and dialect. And the usable resources of accent data are less. On the other hand, dialect is also known as a heavy accent, so we first chose a dialect dataset for training. Another thing that we needed to consider was that, in China there exist many different dialects, even in one region there are different dialects. So we only selected 4 representative dialects Guangzhou(GZ), Shanghai(SH), Zhengzhou(ZZ), and Sichuan(SC) from the four regions of China: south, east, north, and southwest respectively. And the total length of each dataset was listed in the table 4.2.

In addition to analogizing the EN-dataset (which contains various accents), we also mixed all our selected Chinese datasets together, called CN-MIX-dataset, which contained the dialects of various places in Chinese and also non-accented-dataset.

4.1.3. Dataset Splitting

For training, datasets were divided into three parts: the training dataset used to allow the model to learn, the evaluation/validation dataset used during the training process to determine if training is sufficient, and the testing dataset used to test the model's performance after the training has been finished. These three datasets must not overlap with each other. Thus, in order to ensure that the experiment was not biased, the complete dataset was partitioned into three splits, and the samples in each split were randomly disrupted first.

	all accented (EN)	**US accented (US)**	**indian accented (IN)**	**pakistani accented (PAK)**
Resource	CommonVoice	CommonVoice	CommonVoice	MagicData
Training length(h)	1503.2	378.53	123.61	2.79
Evaluation length(h)	27.31	1.6	0.62	0.8
Test length(h)	26.89	1.24	0.58	0.41

Table 4.1.: EN dataset split time length.

	Chinese (CN)	**Zhengzhou (ZZ)**	**Sichuan (SC)**	**Shanghai (SH)**	**Guangzhou (GZ)**
Resource	CommonVoice	MagicData	MagicData	MagicData	MagicData
accent	no(mandarin)	dialect	dialect	dialect	dialect
Training length(h)	41.05	3.52	4.95	3.08	2.88
Evaluation length(h)	15.68	1.02	1.4	0.88	0.82
Test length(h)	17.26	0.51	0.69	0.44	0.41

Table 4.2.: CN dataset split time length.

Huggingface [7] offers methods to automatically split the dataset for Common Voice. Whereas the dataset in Magic Hub [30] must be manually split. The dataset was segmented into train, validation, and test splits with a ratio of approximately $7 : 2 : 1$.

Time length after splitting the dataset was shown in the table 4.1 for English datasets and the table 4.2 for Chinese datasets.

4.1.4. Vocabulary

After getting the dataset, there was one more thing that needed to be done before training, which was to create a vocabulary. Vocabulary, as the name suggests, is a dictionary of characters and their corresponding numerical numbers, which allows the transformer to encode the reference words one by one according to the generated vocabulary, and also to decode the predicted words by the decoder when using the model. When using the model, the predicted words are decoded by the decoder, so how the vocabulary was constructed is also very important.

The first step in building a vocabulary was to extract the transcription from the dataset and separate them into a character hierarchy. The dataset was loaded into a dataset object as a table form, where all the transcriptions were sitting under the transcription column. After the transcriptions were extracted, some special characters(in Figure A.1) that were not helpful for speech understanding were removed, and then all the characters were transferred in lowercase.

These transcriptions were later used as labels for supervised learning. For English labels, the filtered transcriptions needed an additional step to merge consequent white spaces, while for Chinese labels, every Chinese character in the filtered transcriptions needed to be separated from both neighboring characters by a single white space respectively. These filtered transcriptions needed further operations for building the vocabulary. Those duplicated characters were removed by merging the list of characters obtained from a filtered transcription into a set. Finally, a vocabulary was completed by adding pad tokens [pad] and unknown tokens [unk], replacing the white spaces with the delimiters | in the set, and numbering each element in the set.

By default, each dataset has its own vocabulary. Different vocabularies for English datasets don't differ much, because all of them contain those 26 characters. But for Chinese, the number of characters is huge, and there is still no clear indication of how many characters there are in total. Then the coverage of characters in different Chinese datasets is different. In order to understand how many characters are the same in different Chinese dialect datasets, I made a vocabulary for each Chinese dataset and compared how many characters were the same in different vocabularies as well as how many characters were different, and summarized the information in table A.6. In this table, the numbers in each grid represent the number of identical characters in the dataset of the row and the dataset of the column. The diagonal lines are then the total number of characters in each dataset.

The point to note here is that, in Chinese, the same pronunciation can correspond to many different characters. In the dialect dataset, the same piece of audio corresponds to two different transcripts, one that is semantically the same as the audio, and the other that is phonetically the same as the audio. The second kind of transcript was chosen, because the first kind of character does not correspond to the audio in terms of pronunciation.

It can be seen that the vocabulary of different datasets can be very different, which may cause the model to be trained to perform poorly on another test target and to predict a large number of unknown [unk] tokens. Therefore, for the Chinese dataset, I extracted all the characters of the Chinese dataset (CN and the four dialect datasets) and fused them together, eliminated the recurring characters, and made the final vocabulary.

4.2. Mono- and Multi-lingual Pre-trained Models

After processing the data, according to the model-building strategy in Chapter 3, the model to be used for training needed to be specified. According to the approaches in Chapter 3.2.1, two pre-trained models needed to be selected.

For the one pre-trained in a monolingual language, I chose the 'facebook/wav2vec2-large' model, which used the English dataset in librispeech for pre-training. In subsequent experiments, this model was called mono-model. Accordingly, I chose 'facebook/wav2vec2-large-xlsr-53' as multi-model, which is also a large model pre-trained in 53 languages [8].

In this model, the feature encoder contains seven blocks and the temporal convolutions in each block have 512 channels with strides (5,2,2,2,2,2,2) and kernel widths (10,3,3,3,3,2,2). This results in an encoder output frequency of 49 Hz with a stride of about 20ms between each sample, and a receptive field of 400 input samples or 25ms of audio. The convolutional layer modeling relative positional embeddings has kernel sizes of 128 and 16 groups.

The large model contains 24 transformer blocks with model dimension 1,024, inner dimension 4,096 and 16 attention heads. We used dropout 0.1 in the Transformer, at the output of the feature encoder and the input to the quantization module. Layers were dropped at a rate of 0.2 for large [3].

4.2.1. Model Configuration

Both pre-trained models introduced above were loaded as checkpoints from the remote server. Furthermore, a fine-tuned model was also loaded in the same way but locally, which was quite convenient in implementing the two-step fine-tuning approaches.

Together with the model, a processor needed to be loaded or initialized for preprocessing the audio pieces and transcriptions in the dataset. At the same time, a feature extractor and a tokenizer were required. In order to preprocess the audio pieces and transcriptions in the dataset, the processor object needed to be called properly. To resample a piece of audio and transform the sampled signal into an array in float type, the feature extractor set in the processor was applied in the background. The tokenizer helped to encode a transcription by token ids kept in the vocabulary file and then add paddings in the encoded array of ids.

After loading a Wav2vec2 model, some additional modifications were performed to change or freeze the parameters.

4.2.2. Training Details

A trainer object is responsible for managing the whole training or evaluation process. Before running this automatic trainer, the trainer needed to be configured. In all the following experiments, 'Adafactor' was set as the optimizer. For the one-step-fine-tuning approach on accented English datasets, the learning rates were set to $1e-4$. For the rest of the experiments in this work, the learning rates were always $5e-5$. The number of warm-up steps was set as 1000, and the decay ratio was 0.005 static in all the experiments.

4.3. Evaluation Metrics

This section sets out the criteria for judging the quality of a model.

4.3.1. Word Error Rate

Word error rate (WER) is a common metric of the performance of an automatic speech recognition system. WER assesses recognition by measuring the dissimilarity between the recognized word sequence and the reference word sequence. In normal, the difficulty of performance measurement is due to the different lengths of the recognized and reference sequences (e.g. distance measurements). This problem is solved by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment. And the Word error rate can then be computed as following:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}, \tag{4.1}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, C is the number of correct words, and N is the number of words in the reference ($N = S + D + C$).

This value indicates the average number of errors per reference word. ASR performance is best when WER is 0.

4.3.2. Character Error Rate

Another evaluation method is the character error rate(CER). CER is similar to Word Error Rate (WER). The formula is the same as WER formula (4.1). Unlike WER, CER is computed at the character level. CERs are typically used to evaluate character-based language. T

5. Results and Analysis

In this chapter, the findings based on the results of the experiment were summarized and listed separately in the following sections. The corresponding experimental data confirming the findings were collected and analyzed. The complete experimental data tables were located in the appendix.A

5.1. Difficulty of Accent and Dialect for ASR

Dialects and accents are able to bring a completely different level of difficulty to speech recognition.

It is easy to see from the experimental results 5.1that the worst model in the English experiment was fine-tuned with the PAK dataset, and the best results were obtained when tested with the PAK accent. All other accents were tested with worse results, averaging around 55%. In the Chinese test results 5.2, except for the test results on the same dialect as fine-tuning, which are very good, the test results on other dialects are around 90%, and such an error rate is not meaningful for ASR. (Model fine-tuned by CN-mix performed well on each test dataset, it explained in section 5.3.

The huge gap between the worst two outcomes demonstrates that Chinese dialects and English accents have different levels of difficulty in speech recognition, with accents being significantly easier than dialects.

PT-model	FT-dataset	Test-dataset	WER(%)
Mono	PAK	EN	59.0
		US	57.4
		IN	59.7
		PAK	28.9
Multi		EN	49.0
		US	48.0
		IN	46.2
		PAK	**17.6**

Table 5.1.: Experimental results after one-step fine-tuning on mix accented English. This table is the bottom-right block in the table A.1 The row in each local part indicates the test dataset and the last column indicates the corresponding result in WER(%).

This is due to the fact that an accent is merely a small modification in pronunciation, while a dialect may have grammatical distinctions. On the other hand, the vast differences

		SH	ZZ	SC	GZ	CN	CN-mix
Mono	SH	**10.8**	88.7	91.6	95.7	98.3	95.6
	ZZ	83.5	13.7	67.0	94.9	82.4	79.4
	SC	88.6	59.2	**18.8**	92.7	80.2	77.3
	GZ	95.0	97.2	96.4	6.2	102.0	99.1
	CN	88.2	68.8	75.4	95.8	28.5	34.9
Multi	SH	24.5	87.5	90.0	94.5	98.5	95.7
	ZZ	80.7	**11.6**	62.1	92.5	78.1	75.5
	SC	86.0	57.1	25.3	91.9	76.1	73.9
	GZ	94.3	96.8	97.2	**5.4**	103.6	100.6
	CN	85.6	65.6	72.6	94.0	**22.3**	**29.3**

Table 5.2.: Experimental results after 1-step-FT on SH,ZZ,SC,GZ,CN. The row in each local part indicates the fine-tuning dataset and the column indicates the test dataset. The bold values are optimal in the corresponding column. The results are presented in CER(%).

between the various Chinese dialects make it impossible for the model to generalize common features across all dialects, each dialect is more akin to an individual language. This highlights the need to distinguish between dialects and accents in speech recognition. Sometimes it is necessary to consider different dialects within the same language system separately as the variation between them can be significant, requiring a reliance on established linguistic knowledge.

5.2. Multilingual vs Monolingual Model

The multilingual- and the monolingual-model show different generalites of the knowledge representation learned in the pret-raining stage. In this section, only the 1-step-FT approach3.3.1 was used for experimental comparisons.

The superiority of the multilingual model over the monolingual models was apparent by comparing the multilingual and monolingual models fine-tuned on the same datasets. This outcome was predicted before the experiment, and result table 5.3 depicted the outcomes after utilizing the EN-dataset as the fine-tuning dataset and conducting tests on each dataset. This observation was consistently reflected in the other test results when other English datasets were used for fine-tuning. In addition, similar trends were observed in the Chinese results. Table 5.4 showcases the results of the model fine-tuned on CN-mix tested on different Chinese datasets. An identical conclusion can be drawn from these results.

PT-model	FT-dataset	Test-dataset	WER(%)
Mono	EN	EN	32.5
		US	29.2
		IN	34.1
		PAK	26.3
Multi		EN	23.4
		US	20.8
		IN	23.5
		PAK	20.6

Table 5.3.: Experimental results after 1-step-FT on mix accented English. This table is the upper-left block in table A.1 for the experimental results after 1-step-FT on English accents. Mono indicates monolingual pre-trained model and Multi indicates the multilingual pre-trained model. The upper part is pre-trained by monolingual dataset and the bottom part is pre-trained by multilingual dataset. The row in each local part indicates the test dataset and the last column indicates the corresponding result in WER(%).

There was still the exception of certain cases. In the Chinese experiment, the monolingual model outperformed the multilingual model when it was fine-tuned with the SH dialect. The same results occurred with ZZ and SC dialects. The table 5.5 below illustrated these results.

PT-model	FT-dataset	Test-dataset	CER(%)
Mono	CN-mix	SH	18.1
		ZZ	16.2
		SC	15.3
		GZ	9.3
		CN	28.2
		CN-mix	27.9
Multi		SH	12.1
		ZZ	10.7
		SC	10.0
		GZ	4.8
		CN	21.4
		CN-mix	21.3

Table 5.4.: Experimental results after 1-step-FT on mix Chinese dialects together with accented mandarin. This table is the bottom-right block in the table A.1 for the experimental results after 1-step-FT on English accents. The row in each local part indicates the test dataset and the last column indicates the corresponding result in CER(%).

		SH	SC
Mono	SH	**10.8**	91.6
	SC	88.6	**18.8**
Multi	SH	24.5	90.0
	SC	86.0	25.3

Table 5.5.: Experimental results after 1-step-FT on Chinese dialects: SH, SC. SH, ZZ, The row in each local part indicates the fine-tuning dataset and the column indicates the test dataset. The results are presented in CER(%).

Utilizing a multilingual model that has been pre-trained on a wide range of linguistic data, encompassing various phonetic and phonological properties from diverse languages, can significantly improve the model's ability to generalize. These linguistic variations can substantially enhance the model's performance. Additionally, multilingual models possess greater adaptability to languages that were not included in the training set, as they can transfer across languages by sharing certain linguistic features from the pre-training stage.

For fine-grained tasks requiring in-depth understanding or recognition of very specific linguistic nuances, a monolingual model, fine-tuned with substantial data for that language, might be preferable. In this experiment, SH and SC dialects are more difficult than the other dialects from a linguistic point of view, especially since the gap in pronunciation is more subtle compared to the other, so the monolingual model performed better.

5.3. Similarity and Variety of Accents/dialects

The similarity and the variety describe the relation between the knowledge contained in accent- or dialect-datasets. A training dataset similar to the target dataset contains accents of the same kind as the target accent. In other words, similar datasets can be regarded as sampled from the same common dataset. The variety, however, measures how different the accents can differ from each other. A dataset as a various mixture of the target dataset collects different accents that vary from but are still related to the target accent. For the test-split of the US-dataset, the train-split of the US-dataset was a similar one, and the train-split of EN-dataset was a varied version.

Considering the target accent, a similar training dataset performs better than a general one, and a various mixture training dataset works even better than a similar one.

Typically, the top-performing model was fine-tuned with the same accent as the test target. Figure 5.6 demonstrated the performance of various models using EN- and PAK-datasets as test datasets.

The same results were found in the experiment on the Chinese. The table 5.7 displayed the results of different models tested on SH-dataset and CN-mix-dataset, and the optimal results were highlighted.

		EN	PAK
Mono	EN	32.5	26.3
	US	30.9	28.4
	IN	35.2	24.7
	PAK	59.0	28.9
Multi	EN	**23.4**	20.6
	US	28.8	27.1
	IN	33.1	20.1
	PAK	49.0	**17.6**

Table 5.6.: Experimental results after 1-step-FT on English accents. This table is tested on EN- and PAK- datasets. The row in each local part indicates the fine-tuning dataset and the last column indicates the corresponding result in WER(%).

There were still some test results that did not follow the pattern above. Regarding the test targets of US-accent and IN-dataset, the best model performance results were obtained using the EN-dataset for fine-tuning. In the Chinese experiments, the model utilized CN-mix as the fine-tuning dataset yielded superior results when the test dataset comprised ZZ, SC, GZ, and CN datasets. Table 5.8 presented the test results of the model on the US-dataset and IN-dataset. And table 5.9 presented the test results on Chinese dialects.

		SH	CN-mix
Mono	SH	**10.8**	95.6
	ZZ	83.5	79.4
	SC	88.6	77.3
	GZ	95.0	99.1
	CN	88.2	34.9
	CN-mix	18.1	27.9
Multi	SH	24.5	95.7
	ZZ	80.7	75.5
	SC	86.0	73.9
	GZ	94.3	100.6
	CN	85.6	29.3
	CN-mix	12.1	**21.3**

Table 5.7.: Experimental results after 1-step-FT on Chinese dialects. The row in each local part indicates the fine-tuning dataset and the column indicates the test dataset. The bold values are optimal in the corresponding column. The results are presented in CER(%).

		US	IN
Mono	EN	29.2	34.1
	US	27.3	37.9
	IN	33.3	28.8
	PAK	57.4	59.7
Multi	EN	**20.8**	**23.5**
	US	24.4	33.5
	IN	28.9	24.7
	PAK	48.0	46.2

Table 5.8.: Experimental results after 1-step-FT on English accents. This table is tested on US- and IN- datasets. The row in each local part indicates the fine-tuning dataset and the last column indicates the corresponding result in WER(%).

The model performs best when the fine-tuning accent and the test target accent are the same. This is very reasonable because each accent has its own set of phonemes, morphemes, syntax, and semantics. Training a model on the specific accent characteristics of one language ensures that it becomes highly familiar with the nuances of that accent, leading to superior performance on test data in the same accent. In addition, the same accent has consistent acoustic properties across training and testing datasets. This consistency helps the model make better predictions.

By observing the same tables 5.1 and 5.2 as already analyzed in the section 5.1, these results may be due to the low similarity of accents or dialects. Understanding through English and Chinese knowledge illuminates that PAK-accented English differs significantly from the commonly heard US or UK English. Similarly, various Chinese dialects vary greatly, giving rise to this phenomenon.

		ZZ	SC	GZ	CN
Mono	SH	88.7	91.6	95.7	98.3
	ZZ	13.7	67.0	94.9	82.4
	SC	59.2	18.8	92.7	80.2
	GZ	97.2	96.4	6.2	102.0
	CN	68.8	75.4	95.8	28.5
	CN-mix	16.2	15.3	9.3	28.2
Multi	SH	87.5	90.0	94.5	98.5
	ZZ	11.6	62.1	92.5	78.1
	SC	57.1	25.3	91.9	76.1
	GZ	96.8	97.2	5.4	103.6
	CN	65.6	72.6	94.0	22.3
	CN-mix	**10.7**	**10.0**	**4.8**	**21.4**

Table 5.9.: Experimental results after 1-step-FT on Chinese dialects. The row in each local part indicates the fine-tuning dataset and the column indicates the test dataset. The bold values are optimal in the corresponding column. The results are presented in CER(%).

However, PAK-accent and IN-accent English are similar in terms of pronunciation in theory, because both regions use the same language family: Hindi. Theoretically speaking, the test result of IN-dataset should be very good, but in reality, it is very bad, because the PAK-dataset is from Magic Hub, while the other English The PAK-dataset is from Magic Hub, while the other English datasets are from Common Voice, although they are labeled as daily language, the recording environment and application scenarios are not guaranteed. Therefore, the domain difference is also a reason. Moreover, a limited dialect/accent dataset could lead to overfitting of the model.

However, when training with mixed accent data, sometimes the model performance exceeded that of training with the corresponding accent. The mixed accent dataset, as a various mixture can exhibit varied but related acoustic properties. By training on various mixture accent datasets, the ASR system was exposed to a wide range of acoustic patterns, leading to better learning of acoustic.

General knowledge transfer learning helps the model to learn specific accents. The General knowledge transferred from that model is eventually learned from some general data resources. By learning on a dataset containing not only the specific accent but also various accents closely related to this target accent, the general knowledge and the specific knowledge are learned at the same time, and the general one can already help to learn the specific one even without extra transferring.

Compared to learning on the specific accent only, learning on a variety of related accents is not only processed on that specific accent but also enhanced by a local general knowledge transfer. So that the learned representation in the model can hold a deeper and more precise understanding and perform better in recognizing the target accent.

5.4. Two-step Fine-tuning vs One-step Fine-tuning

We used the 2-step-FT approach in chapter 3 3.3.2 and compared it with 1-step-FT. The general knowledge in 1-step-FT helps the model to learn the specific knowledge presentation in 2-step-FT. Compared to the 1-step-FT, the general knowledge transferred from pre-training has a chance to be specified on the information in an additional fine-tuning stage. This attended specified general knowledge is still enough general for but more related to the target task than that knowledge directly transferred from pre-training. We first used both monolingual and multilingual models for 2-step-FT and compared it with 1-step-FT, the data presented in the table 5.10 leads to the conclusion that the multilingual model performs better than the monolingual model in 2-step-FT. This finding aligns with the results obtained in the section5.2. The general knowledge representation formed in the pre-training stage benefits the downstream task all along the fine-tuning process. In the Chinese experiments, only the multilingual model was used for the 2-step-FT. 5.11

			EN	US	IN	PAK
Mono	1-Step-FT	US	30.9	27.3	37.9	28.4
		IN	35.2	33.3	28.8	24.7
		PAK	59.0	57.4	59.7	28.9
	2-Step-FT	US	27.8	24.5	29.6	24.0
		IN	28.3	26.5	26.5	23.9
		PAK	33.6	30.5	33.5	21.1
Multi	1-Step-FT	US	28.8	24.4	33.5	27.1
		IN	33.1	28.9	24.7	20.1
		PAK	49.0	48.0	46.2	17.6
	2-Step-FT	US	21.6	19.2	22.2	19.6
		IN	22.7	21.5	20.1	18.6
		PAK	24.3	22.1	24.3	16.1

Table 5.10.: Experimental results after 2-step-FT on English accents vs experimental results after 1-step-FT. 1-Step-FT indicates the model directly based on the mono- or multi-lingual pre-trained model and 2-Step-FT indicates the model already fine-tuned by EN-dataset based on the mono- or multi-lingual pre-trained model. The upper part is a 1-Step-FT model and the bottom part is a 2-Step-FT model. The row in each local sub-part indicates the fine-tuning dataset and the column indicates the test dataset. The results are presented in WER(%).

Notably, the 2-step-FT approach outperformed the 1-step-FT approach in all aspects. The previous findings from 1-step-FT demonstrate that transferring multilingual representations from a pre-trained model could improve the fine-tuning process on accented datasets. The current finding not surprisingly confirmed that, also the multi-accent representation transferred from 1-step-FT could support further fine-tuning on the specific accent or dialect.

It is also observed in the 2-step-FT for English, that starting from a pre-trained model holding a more general multilingual representation outperformed the training starting from

		SH	ZZ	SC	GZ	CN
1-Step-FT	SH	24.5	87.5	90.0	94.5	98.5
	ZZ	80.7	11.6	62.1	92.5	78.1
	SC	86.0	57.1	25.3	91.9	76.1
	GZ	94.3	96.8	97.2	5.4	103.6
2-Step-FT	SH	12.3	77.5	80.9	94.6	69.2
	ZZ	81.2	9.7	52.5	93.0	60.6
	SC	85.4	43.4	9.5	91.7	57.3
	GZ	92.7	92.0	89.8	4.7	79.8

Table 5.11.: Experimental results after 2-step-FT on Chinese dialects vs experimental results after 1-step-FT. The row in each local part indicates the fine-tuning dataset and the column indicates the test dataset. The results are presented in CER(%).

a less general monolingual representation. The current fine-tuning is not only influenced by the knowledge learned in the last fine-tuning, The general knowledge transferred along the leveraged down-streaming tasks will actually impact all the following fine-tunings.

Considering the target as US accent, such kind of 2-step-FT with the first fine-tuning on the EN-dataset exclusive US accent and the second fine-tuning on the US accent dataset should be competitive to the 1-step-FT on the EN-dataset, which also outperformed the 1-step-FT on the US-dataset (in table **??**).

The improvements until now observed in our findings are basically brought by the generality of the knowledge either as the raw information contained in the current training dataset or as the knowledge representation directly transferred from a previously trained model.

In the first case, the composition of the training dataset decides the generality of the knowledge inside. In this work, the multilingual and monolingual datasets for pre-training typically represented a more general and a less general example respectively.

In the second case, by transfer learning, the training process is leveraged into successive steps, for example, the 1-step or 2-step fine-tuning in this work. Instead of improving the generality of the knowledge transferred through the leveraged fine-tuning process, more fine-tuning stages offer more flexibility to deploy the leveraged datasets mentioned in the first case, so that we have more chances to intentionally orient or impact the changing of the knowledge representation. In this work, we mainly tried to deploy a mixed dataset like EN-mix or CN dataset in an additional fine-tuning stage to offer more related accent information. By leveraging the training, general and focusing datasets are used successively. General datasets can go first to save energy for later focusing datasets, which can be under-resourced.

5.5. Adapter

In this section, the two kinds of adapters previously introduced in chapter 3.3.3 were integrated into the model according to the approaches 3.3.4 and were compared. It can be

inferred that the multilingual model was more effective than the monolingual model5.2. Therefore, only the multilingual model was utilized in this section for experiments.

Considering the knowledge gap caused by the dialect information in the downstream task, it is hard to keep the performance by saving the further optimization on the general knowledge transferred from upstream. The transferred general knowledge can surely enhance further fine-tuning, but the learning ability of the model needs to at least fit the downstream task.

		SH	ZZ	SC	GZ	CN	CN-mix
Enc-Adapter	SH	43.3	89.9	91.7	92.4	96.3	93.7
	ZZ	89.9	51.0	75.4	92.8	85.1	83.6
	SC	91.2	69.7	44.9	93.9	83.2	81.3
	GZ	94.2	95.1	96.1	33.7	98.2	95.7
	CN	89.6	75.0	77.9	95.3	35.5	43.3
Attn-Adapter	SH	41.6	88.3	90.7	92.8	96.2	93.4
	ZZ	82.5	32.8	62.8	92.1	80.8	78.3
	SC	86.8	59.2	33.6	91.8	78.5	76.1
	GZ	92.4	94.5	94.1	18.7	98.8	95.6
	CN	87.7	72.6	76.0	95.9	33.5	41.4
No-Adapter	SH	24.5	87.5	90.0	94.5	98.5	95.7
	ZZ	80.7	11.6	62.1	92.5	78.1	75.5
	SC	86.0	57.1	25.3	91.9	76.1	73.9
	GZ	94.3	96.8	97.2	5.4	103.6	100.6
	CN	85.6	65.6	72.6	94.0	22.3	29.3

Table 5.12.: Experimental results using 1-step-FT on SH, ZZ, SC, GZ, CN based on the multilingual pre-trained model. Enc-Adapter indicates the model with encoder adapter and Attn-Adapter indicates the model with attention adapter. The upper part is fine-tuned on encoder adapter and the bottom part is fine-tuned on attention adapter. The row in each local part indicates the fine-tuning dataset and the column indicates the test dataset. The bold values are optimal in the corresponding column. The results are presented in CER(%).

The model was equipped with the encoder adapter and attention adapter, and then 1-step-FT was performed to compare the obtained results with those from the original model, resulting in the table 5.12 being obtained. The results showed that the model with the attention adapter experienced a performance reduction of about 10%, while the performance of the model with the encoder adapter experienced a reduction of 12%. It was difficult to maintain performance while saving further optimization on the general knowledge transferred in the attention blocks.

Then the results of 2-step-FT with adapter and 2-step-FT without adapter were in table 5.13, it was found that the unfrozen attention block with additional adapter layers improved results when the target dialect differed from the fine-tuning dialect. However, the results decreased when the target dialect was the same as the fine-tuning dialect.

		SH	ZZ	SC	GZ	CN	CN-mix
No-Adapter	SH	12.3	77.5	80.9	94.6	69.2	68.8
	ZZ	81.2	9.7	52.5	93.0	60.6	59.9
	SC	85.4	43.4	9.5	91.7	57.3	56.4
	GZ	92.7	92.0	89.8	4.7	79.8	78.9
Attn-Adapter	SH	23.2	78.0	81.1	90.0	62.5	63.5
	ZZ	81.2	18.8	50.8	89.2	47.7	49.3
	SC	84.3	43.3	22.5	91.2	46.8	48.3
	GZ	90.7	87.3	89.2	15.4	74.6	74.6

Table 5.13.: Experimental results using 2-step-FT without adapter on Chinese dialects vs experimental results after 2-step-FT with adapter. The model in this table is always a multilingual pre-trained model further 1-step-FT by accented Chinese mandarin dataset. No-Adapter indicates the model without adapter and Attn-Adapter indicates the model with attention adapter. The upper part is fine-tuned without adapter and the bottom part is fine-tuned on the attention adapter. The row in each local part indicates the fine-tuning dataset and the column indicates the test dataset. The bold values are optimal in the corresponding column. The results are presented in CER(%).

We hoped that by fine-tuning the adapter instead of all the parameters of a pre-trained ASR model, adapters allowed for task-specific adaptation with a much smaller number of parameters. This could lead to efficient transfer learning with fewer resources. By freezing the attention layers (in table 5.12), the learning simply relies on the adapter lays, which are light-weighted but also weaker than the attention layers in learning. The reduction in performance is probably because the knowledge required to specify the representation from a multi-lingual to an accented representation is significantly larger than the knowledge gap between the multi-lingual and the standard language representation according to the accent. However, an adapter is simply constructed by fully connected down- and up-projection layers, and is not competitive enough to fill the gap formed by accent or dialect in speech.

Even by releasing the attention layers to join the learning together with adapter layers (in table 5.13), the performance was only enhanced by less than 5%. The results implied again the weak learning ability of the adapter. Nevertheless, the learning of accent information is concentrated within the first several layers of the attention block [35]. However, the adapter was added at the end of each attention block, in this work, referring to the MMS project.

In addition, adapters can be particularly useful in scenarios where the training accent varies from but still similar to the target task (like a specific dialect or a domain-specific vocabulary). We are aware of significant differences between its various dialects5.1, such as disparities in vocabulary4.1.4 and pronounced phonetics. This reality implies that an adapter cannot simply solve the problem. So the success of the attention adapter is hard to replicate here.

6. Conclusion

By synthesizing the various findings and corresponding analysis from Chapter 5, I summarize the conclusions for this thesis and answer the research questions, in this chapter.

***RQ1**:* Does the difficulty of building speech recognition systems differ between accents or dialects in different language systems?

First, the experiments5.1 were conducted using languages that are in different language systems. In 5.1 models performed barely badly when using PAK- and IN- accents as test targets, which were heavily accented in cognition. However, the error rate would be more than 90% in the Chinese experiment if the test target dialect and the fine-tuning dialect were not the same. Such a result doesn't even make sense in ASR. It indicates that English accents and Chinese dialects in speech present a large difference in levels of difficulty, which causes a huge gap in the performance of ASR and brings varying levels of difficulty in training. Moreover, because the operation of vocab creation was also different, wrong vocab operation could also lead to worse test results, so it can be concluded that the difficulty of ASR caused by accent and dialect is obviously different in different language systems.

***RQ2**:* What factors of the dataset can impact the knowledge representation learning?

Secondly, based on 5.1 it can be understood that, in addition to the low resources of the dataset, the significant difference between the accent/dialect of the test target and the fine-tuning dataset also affectes the model's performance. In addition, 5.3 shows the domain match between the test dataset and the training dataset had an impact on the test. It is crucial to use similar datasets as possible for training.

The dataset doesn't only have its focused area or domain. In 5.3, although the EN-mix dataset didn't totally match to test target accent, it achieved better performance, because a various mixture dataset indicated more generality or variety of pronunciation knowledge. A perfect dataset is supposed to vary as much as possible but remain related to the target. Another option is to apply leveraged datasets within a leveraged learning process, in order to gradually specify the knowledge representation targeting a refined downstream task, where the knowledge gap is also refined and the adapter is capable of maintaining the performance.

***RQ3**:* How to improve the robustness and correctness of an accent ASR model for speaker changes based on the properties of transfer learning?

Further on the knowledge transfer, according to 5.4, using a 2-step fine-tuning approach could enhance the performance of automated speech recognition (ASR) for various language systems, without a direct focus on the language system. This indicates that teaching an ASR system basic pronunciation information in advance and then providing

target-focused training can improve its ability to recognize dialects and accents. Based on the 5.5, the adapter is weak in learning complex language knowledge, and it is challenging to optimize the model by adding adapters. Further experiments are necessary to verify the trade-off between its learning ability for performance and retaining previously learned knowledge for efficiency.

In conclusion, it is currently best to consider utilizing separate approaches to handle accents and dialects due to their differing levels of difficulty.

A model's ability to understand audio is enhanced by the generality of its knowledge and broad understanding of various language systems. For constructing a dataset, it's crucial to consider a specific accent in mind, while also ensuring that it has enough variety. This concentration helps the model's knowledge to converge towards the target accent, rather than deviate from it. The diversity, on the other hand, offers a higher level of abstraction, reducing the risk of overfitting. Subsequent experiments are required to confirm the degree of disparity between a specific accent and the target accent, as well as the level of diversity of various accents.

The model's performance can be improved by increasing the diversity of dialects and the amount of training data. Therefore, collecting datasets labeled with accents and conducting effective qualitative and quantitative analyses of the datasets remains a challenge.

Secondly, the model can learn accents effectively in conjunction with existing general knowledge during transfer learning and demonstrates high performance. Thus, combining general knowledge with target-related accent pronunciation information is critical for the model's success in transfer learning.

Furthermore, exploring the feasibility of transferring general knowledge over multi-accents or -dialects dataset or even over a multilingual and multi-accent dataset helps to reduce the knowledge gap in the further fine-tuning. Although adapters do not currently significantly improve model performance, it is still worthwhile to explore their potential role. Possible approaches can adjust the deployment of the adapter to the beginning of each attention block [35], to fit the accented ASR task by considering the distribution of the model parameters which are typically optimized for learning accent.

For feature research, in leveraged knowledge transfer, the generality of knowledge contained in the datasets can be adapted for different approaches according to the composition of mixed training dataset. How variances in accent similarity within a mixed training dataset can affect the model's performance still needs to be explored.

Bibliography

[1] Alëna Aksënova et al. "Accented speech recognition: Benchmarking, pre-training, and diverse data". In: *arXiv preprint arXiv:2205.08014* (2022).

[2] Dario Amodei et al. "Deep speech 2: End-to-end speech recognition in english and mandarin". In: *International conference on machine learning*. PMLR. 2016, pp. 173–182.

[3] Alexei Baevski et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in neural information processing systems* 33 (2020), pp. 12449–12460.

[4] Jonathan Bgn. *An Illustrated Tour of Wav2vec 2.0*. `https://jonathanbgn.com/2021/09/30/illustrated-wav2vec-2.html`. [Online; accessed 23-September-2023]. 2021.

[5] Rafal Černiavski. *Cross-lingual and Multilingual Automatic Speech Recognition for Scandinavian Languages*. 2022.

[6] Jack K Chambers and Peter Trudgill. *Dialectology*. Cambridge University Press, 1998.

[7] Commonvoice. *Datasets*. `https://commonvoice.mozilla.org/en/datasets`. [Online; accessed 23-September-2023].

[8] Alexis Conneau et al. "Unsupervised cross-lingual representation learning for speech recognition". In: *arXiv preprint arXiv:2006.13979* (2020).

[9] d2l. *modern-recurrent-NN the-encoder-decoder-architecture*. `https://d2l.ai/chapter_recurrent-modern/encoder-decoder.html`. [Online; accessed 26-September-2023]. 2020.

[10] Keqi Deng, Songjun Cao, and Long Ma. "Improving accent identification and accented speech recognition under a framework of self-supervised learning". In: *arXiv preprint arXiv:2109.07349* (2021).

[11] Ministry of Education of the People's Republic of China. *Overview of Chinese language and characters (2021 Edition)*. `http://www.moe.gov.cn/jyb_sjzl/wenzi/202108/t20210827_554992.html`. [Online; accessed 23-September-2023]. 2021.

[12] Arlo Faria. "Accent classification for speech recognition". In: *International Workshop on Machine Learning for Multimodal Interaction*. Springer. 2005, pp. 285–293.

[13] Github. *Commonvoice demographics*. `https://github.com/common-voice/common-voice/blob/main/web/src/stores/demographics.ts`. [Online; accessed 23-September-2023].

[14] graphcore. *Pre-Training and Fine-Tuning BERT for the IPU:Introduction*. `https://docs.graphcore.ai/projects/bert-training/en/latest/bert.html`. [Online; accessed 29-September-2023]. 2023.

[15] Alex Graves et al. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 369–376.

[16] Awni Hannun. “Sequence Modeling with CTC”. In: *Distill* (2017). https://distill.pub/2017/ctc. DOI: 10.23915/distill.00008.

[17] Georg Heigold et al. “Multilingual acoustic models using distributed deep neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 8619–8623.

[18] Arthur Hinsvark et al. “Accented speech recognition: A survey”. In: *arXiv preprint arXiv:2104.10747* (2021).

[19] Neil Houlsby et al. “Parameter-efficient transfer learning for NLP”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2790–2799.

[20] Huggingface. *Boosting Wav2Vec2 with n-grams in Transformers*. https://huggingface.co/blog/wav2vec2-with-ngram. [Online; accessed 23-September-2023]. 2022.

[21] Huggingface. *Facebook Meta AI*. https://huggingface.co/facebook?sort_models=alphabetical#models. [Online; accessed 23-September-2023].

[22] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical reparameterization with gumbel-softmax”. In: *arXiv preprint arXiv:1611.01144* (2016).

[23] Sarah Samson Juan et al. “Merging of native and non-native speech for low-resource accented ASR”. In: *Statistical Language and Speech Processing: Third International Conference, SLSP 2015, Budapest, Hungary, November 24-26, 2015, Proceedings 3*. Springer. 2015, pp. 255–266.

[24] Tom Keldenich. *Encoder Decoder What and Why – Simple Explanation*. https://inside-machinelearning.com/en/encoder-decoder-what-and-why-simple-explanation/. [Online; accessed 26-September-2023]. 2021.

[25] KiKaBeN. *Transformer’s Encoder-Decoder*. https://kikaben.com/transformers-encoder-decoder/. [Online; accessed 29-September-2023]. 2021.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[27] Mehul Kumar et al. “Self-Supervised Accent Learning for Under-Resourced Accents Using Native Language Data”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[28] Song Li et al. “End-to-end multi-accent speech recognition with unsupervised accent modelling”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6418–6422.

[29] Zhaolin Li, Boyan Zhong, and Jan Niehues. “Knowledge Transfer in Accent and Dialect Speech Recognition with Self-Supervised Learning”. 2023.

[30] Magichub. *Magicdata datasets*. https://magichub.com/datasets/. [Online; accessed 23-September-2023].

[31] Lucas Maison and Yannick Esteve. “Improving Accented Speech Recognition with Multi-Domain Training”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[32] Rick Merritt. *What Is a Transformer Model*. `https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/`. [Online; accessed 26-September-2023]. 2022.

[33] Abdelrahman Mohamed et al. “Self-supervised speech representation learning: A review”. In: *IEEE Journal of Selected Topics in Signal Processing* (2022).

[34] Kriz Moses. *Encoder-Decoder Seq2Seq Models, Clearly Explained*. `https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c34186fbf49b`. [Online; accessed 23-September-2023]. 2021.

[35] Archiki Prasad and Preethi Jyothi. “How accents confound: Probing for accent information in end-to-end speech recognition systems”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 3739–3753.

[36] Vineel Pratap et al. “Scaling speech technology to 1,000+ languages”. In: *arXiv preprint arXiv:2305.13516* (2023).

[37] Alberto Pérez Robledo. “Exploiting multi-lingual data in end to end automatic speech recognition and spoken language translation”. In: (2021).

[38] Bethan Thomas, Samuel Kessler, and Salah Karout. “Efficient adapter transfer of self-supervised speech models for automatic speech recognition”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 7102–7106.

[39] Katrin Tomanek et al. “Residual adapters for parameter-efficient ASR adaptation to atypical and accented speech”. In: *arXiv preprint arXiv:2109.06952* (2021).

[40] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[41] Ngoc Thang Vu. “Automatic speech recognition for low-resource languages and accents using multilingual and crosslingual information”. PhD thesis. Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2014, 2014.

[42] Ding Wang et al. “An End-to-End Dialect Identification System with Transfer Learning from a Multilingual Automatic Speech Recognition Model.” In: *Interspeech*. 2021, pp. 3266–3270.

[43] Xuefei Wang et al. “Multi-pass Training and Cross-information Fusion for Low-resource End-to-end Accented Speech Recognition”. In: *arXiv preprint arXiv:2306.11309* (2023).

[44] Shinji Watanabe et al. “Hybrid CTC/attention architecture for end-to-end speech recognition”. In: *IEEE Journal of Selected Topics in Signal Processing* 11.8 (2017), pp. 1240–1253.

[45] Wikipedia. *Attention (machine learning) — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Attention%20(machine%20learning)&oldid=1170261956`. [Online; accessed 23-September-2023]. 2023.

[46] Wikipedia. *Cosine similarity — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Cosine%20similarity&oldid=1169535260. [Online; accessed 27-September-2023]. 2023.

[47] Wikipedia. *Fine-tuning (deep learning) — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Fine-tuning%20(deep%20learning)&oldid=1176821756. [Online; accessed 30-September-2023]. 2023.

[48] Wikipedia. *Transformer (machine learning model) — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Transformer%20(machine%20learning%20model)&oldid=1176084185. [Online; accessed 24-September-2023]. 2023.

[49] Hemant Yadav and Sunayana Sitaram. "A survey of multilingual models for automatic speech recognition". In: *arXiv preprint arXiv:2202.12576* (2022).

[50] Mu Yang et al. "Learning asr pathways: A sparse multilingual asr model". In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[51] Fengrun Zhang, Xiang Xie, and Xinyue Quan. "Chinese Dialect Speech Recognition Based on End-to-end Machine Learning". In: *2022 International Conference on Machine Learning, Control, and Robotics (MLCR)*. IEEE. 2022, pp. 14–18.

A. Appendix

A.1. Tables

PT-model	FT-dataset	Test-dataset	WER %	PT-model	FT-dataset	Test-dataset	WER %
Mono	EN	EN	32.5	Mono	US	EN	30.9
		US	29.2			US	27.3
		IN	34.1			IN	37.9
		PAK	26.3			PAK	28.4
Multi		EN	23.4	Multi		EN	28.8
		US	20.8			US	24.4
		IN	23.5			IN	33.5
		PAK	20.6			PAK	27.1
Mono	IN	EN	35.2	Mono	PAK	EN	59.0
		US	33.3			US	57.4
		IN	28.8			IN	59.7
		PAK	24.7			PAK	28.9
Multi		EN	33.1	Multi		EN	49.0
		US	28.9			US	48.0
		IN	24.7			IN	46.2
		PAK	20.1			PAK	17.6

Table A.1.: Experimental results after 1-step fine-tuning on English accents. EN, US, IN and PAK represent mix accented English, American accented English, Indian accented English and Pakistan accented English, respectively. The upper-left block is fine-tuned by mix accented English dataset. The upper-right block is fine-tuned by American accented English dataset. The bottom-left block is fine-tuned by Indian accented English dataset. The bottom-right block is fine-tuned by Pakistan accented English dataset. Mono indicates monolingual pretrained model and Multi indicates the multilingual pretrained model. Within each block, the upper part is pretrained by monolingual dataset and the bottom part is pretrained by multilingual dataset. The row in each local part indicates the test dataset and the last column indicates the corresponding result in WER(%).

PT & 1-step-FT	2-step-FT	Test-dataset	WER %	PT & 1-step-FT	2-step-FT	Test-dataset	WER %
Mono & EN	US	EN	27.8	Mono & EN	IN	EN	28.3
		US	24.5			US	26.5
		IN	29.6			IN	26.5
		PAK	24.0			PAK	23.9
Multi & EN		EN	21.6	Multi & EN		EN	22.7
		US	19.2			US	21.5
		IN	22.2			IN	20.1
		PAK	19.6			PAK	18.6
Mono & EN	PA	EN	33.6				
		US	30.5				
		IN	33.5				
		PAK	21.1				
Multi & EN		EN	24.3				
		US	22.1				
		IN	24.3				
		PAK	16.1				

Table A.2.: Experimental results after 2-step fine-tuning on English accents. This table is constructed in a similar way how the table for the experimental results after 1-step fine-tuning is constructed. But in this table, there are only three datasets used in the 2-step fine-tuning. In each block, the first column combines the type of pretrained model and the dataset used in the 1-step fine-tuning, which is always the mix accented English dataset in this table. All the four accented English datasets are still tested like in the table for the results after 1-step fine-tuning. The results are also represented in WER(%).

PT-model	FT-dataset	Test-dataset	CER %	PT-model	FT-dataset	Test-dataset	CER %
Mono	SH	SH	10.8	Mono	ZZ	SH	83.5
		ZZ	88.7			ZZ	13.7
		SC	91.6			SC	67.0
		GZ	95.7			GZ	94.9
		CN	98.3			CN	82.4
		CN-mix	95.6			CN-mix	79.4
Multi		SH	24.5	Multi		SH	80.7
		ZZ	87.5			ZZ	11.6
		SC	90.0			SC	62.1
		GZ	94.5			GZ	92.5
		CN	98.5			CN	78.1
		CN-mix	95.7			CN-mix	75.5
Mono	SC	SH	88.6	Mono	GZ	SH	95.0
		ZZ	59.2			ZZ	97.2
		SC	18.8			SC	96.4
		GZ	92.7			GZ	6.2
		CN	80.2			CN	102.0
		CN-mix	77.3			CN-mix	99.1
Multi		SH	86.0	Multi		SH	94.3
		ZZ	57.1			ZZ	96.8
		SC	25.3			SC	97.2
		GZ	91.9			GZ	5.4
		CN	76.1			CN	103.6
		CN-mix	73.9			CN-mix	100.6
Mono	CN	SH	88.2	Mono	CN-mix	SH	18.1
		ZZ	68.8			ZZ	16.2
		SC	75.4			SC	15.3
		GZ	95.8			GZ	9.3
		CN	28.5			CN	28.2
		CN-mix	34.9			CN-mix	27.9
Multi		SH	85.6	Multi		SH	12.1
		ZZ	65.6			ZZ	10.7
		SC	72.6			SC	10.0
		GZ	94.0			GZ	4.8
		CN	22.3			CN	21.4
		CN-mix	29.3			CN-mix	21.3

Table A.3.: Experimental results after 1-step fine-tuning on Chinese dialects. SH, ZZ, SC, GZ, CN and CN-mix represent Shanghai dialect, Zhengzhou dialect, Sichuan dialect, Guangzhou dialect, accented Chinese mandarin and mix Chinese dialects together with accented mandarin, respectively. The upper-left block is fine-tuned by Shanghai dialect dataset. The upper-right block is fine-tuned by Zhengzhou dialect dataset. The middle-left block is fine-tuned by Sichuan dialect dataset. The middle-right block is fine-tuned by Guangzhou dialect dataset. The bottom-left block is fine-tuned by accented Chinese mandarin dataset. The bottom-right block is fine-tuned by mix Chinese dialects together with accented mandarin dataset. Mono indicates monolingual pretrained model and Multi indicates the multilingual pretrained model. Within each block, the upper part is pretrained by a monolingual dataset and the bottom part is pretrained by a multilingual dataset. The row in each local part indicates the test dataset and the last column indicates the corresponding result in CER(%).

PT or FT model	FT-dataset	Test-dataset	CER %	PT or FT model	FT-dataset	Test-dataset	CER %
Multi & CN	SH	SH	12.3	Multi & CN	ZZ	SH	81.2
		ZZ	77.5			ZZ	9.7
		SC	80.9			SC	52.5
		GZ	94.6			GZ	93.0
		CN	69.2			CN	60.6
Multi		SH	24.5	Multi		SH	80.7
		ZZ	87.5			ZZ	11.6
		SC	90.0			SC	62.1
		GZ	94.5			GZ	92.5
		CN	98.5			CN	78.1
Multi & CN	SC	SH	85.4	Multi & CN	GZ	SH	92.7
		ZZ	43.4			ZZ	92.0
		SC	9.5			SC	89.8
		GZ	91.7			GZ	4.7
		CN	57.3			CN	79.8
Multi		SH	86.0	Multi		SH	94.3
		ZZ	57.1			ZZ	96.8
		SC	25.3			SC	97.2
		GZ	91.9			GZ	5.4
		CN	76.1			CN	103.6

Table A.4.: Experimental results after 2-step fine-tuning on Chinese dialects vs experimental results after 1-step fine-tuning. This table is constructed in a similar way how the table for the experimental results after 1-step fine-tuning is constructed. But in this table, there are only four datasets used in the fine-tuning. Within each block, the upper part is a multilingual pretrained model further fine-tuned by accented Chinese mandarin dataset and the bottom part is just pretrained by a multilingual dataset. All four Chinese dialects and accented Chinese mandarin dataset are still tested like in the table for the results after 1-step fine-tuning. The results are also represented in CER(%).

2-step-FT model with	2-step-FT dataset	Test-dataset	CER %	2-step-FT model with	2-step-FT dataset	Test-dataset	CER %
no adapter	SH	SH	12.3	no adapter	ZZ	SH	81.2
		ZZ	77.5			ZZ	9.7
		SC	80.9			SC	52.5
		GZ	94.6			GZ	93.0
		CN	69.2			CN	60.6
adapter		SH	23.2	adapter		SH	81.2
		ZZ	78.0			ZZ	18.8
		SC	81.1			SC	50.8
		GZ	90.0			GZ	89.2
		CN	62.5			CN	47.4
no adapter	SC	SH	85.4	no adapter	GZ	SH	92.7
		ZZ	43.4			ZZ	92.0
		SC	9.5			SC	89.8
		GZ	91.7			GZ	4.7
		CN	57.3			CN	79.8
adapter		SH	84.3	adapter		SH	90.7
		ZZ	43.3			ZZ	87.3
		SC	22.5			SC	89.2
		GZ	91.2			GZ	15.4
		CN	46.8			CN	74.6

Table A.5.: Experimental results after 2-step fine-tuning without adapter on Chinese dialects vs experimental results after 2-step fine-tuning with adapter. This table is constructed in a similar way how the table for the experimental results after 1-step fine-tuning is constructed. But in this table, there are only four datasets used in the 2-step fine-tuning. The model in this table is always a multilingual pretrained model further 1-step fine-tuned by accented Chinese mandarin dataset. Within each block, the upper part is 2-step fine-tuned without adapter and the bottom part is 2-step fine-tuned with adapter. All four Chinese dialects and accented Chinese mandarin dataset are still tested like in the table for the results after 1-step fine-tuning. The results are also represented in CER(%).

overlap char number	Mandarin	Shanghai	Zhengzhou	Sichuan	Guangzhou
Mandarin	4850	1567	2268	2586	1357
Shanghai	\	1645	1432	1418	1099
Zhengzhou	\	\	2358	2022	1237
Sichuan	\	\	\	2636	1232
Guangzhou	\	\	\	\	1450

Table A.6.: The coverage in between the vocabularies of CN datasets. The table displays the number of characters that intersect between the horizontal and vertical dataset vocabularies. The number on the diagonal then indicates the amount of words per vocab.

continent	country	length (in s)	length (in H)
	’ ’	2167949	602.208
	2nd Language	99	
Africa	East Africa	485	
	Kanya (East Africa)	96	
	South Africa	23901	6.639
	Afrikaan (West Germanic language of South Africa)	18	
	West Africa	61	
	Nigeria (West Africa)	83	
Pacific	Australia	185435	51.510
	New Zealand	29438	8.177
United Kingdom	England	439502	122.084
	Ireland	67624	18.784
	Scotland	69258	19.238
	Wales	3506	0.974
Europe (continent)	fluent, ESL,European	51	
	Russia	241	
	Poland	179	
	Ukraine	214	
	Slavs (North east Europe)	792	
	East Europe	510	
	West Europe	427	
	German	286527	79.591
	Austria	482	
	France	221	
	Spain	329	
	Dutch	950	0.264
	Sweden	44	
	Norway	30	
	Italy	188	
	Greece	18	
	Serbia (east south europe)	82	
	Midwest Europe	32	

Asia	India and South Asia	446936	124.149
	India	35	
	Bangladesh (south Asia)	444	
	Singapore	14437	4.010
	Philippines	21859	6.072
	Malaysia	6381	1.773
	Thailand	403	
	Japan	61	
	Hong Kong	16206	4.502
	Guangdong (China)	99	
South America	Latin America	203	
	Colombia	23	
Atlantic	North-Atlantic	1792	0.498
	South-Atlantic	8	
North America	Canada	255363	70.934
	United States	1368594	380.165
Global	international english	23	

Table A.7.: Commen voice: EN time length of accent.

L3_Dialect	**L2_Dialect**	**L1_Dialect**	**L0_Dialect** (Region: province)	**L0_Length** (in s)	**L2_Length** (in s)
Northeastern Mandarin	Northeastern Mandarin	Northeastern Mandarin（Jiaoliao Mandarin）	210000-辽宁省	4300 2.85%	16304 10.79%
		Northeastern Mandarin	220000-吉林省	928 0.61%	
		Northeastern Mandarin	230000-黑龙江省	11076 7.33%	
Beijing Mandarin	Beijing Mandarin	Beijing Mandarin	110000-北京市	13082 8.66%	13082 8.66%

Jilu Mandarin	Jilu Mandarin	Jilu Mandarin（Central Plains Mandarin）	120000-天津市	2380 1.58%	17271 11.44%
		Jilu Mandarin Beijing Mandarin	130000-河北省	5720 3.79%	
		Jilu Mandarin（Jiaoliao Mandarin，Central Plains Mandarin）	370000-山东省	9171 6.07%	
Central Plains Mandarin（Jin Chinese，Lanyin Mandarin）	Jin Chinese	Jin Chinese（Central Plains Mandarin）	140000-山西省	3556 2.35%	3556 2.35%
	Central Plains Mandarin	Central Plains Mandarin	410000-河南省	8766 5.80%	10506 6.95%
		Central Plains Mandarin	610000-陕西省	1740 1.15%	
		Central Plains Mandarin（Tibetic languages）	630000-青海省	0 0.00%	
	Lanyin Mandarin	Lanyin Mandarin（Northeastern Mandarin，Jilu Mandarin）	150000-内蒙古自治区	1023 0.68%	2793 1.86%

		Lanyin Mandarin（Central Plains Mandarin, Tibetic languages）	620000-甘肃省	452 0.30%	
		Lanyin Mandarin	640000-宁夏回族自治区	144 0.10%	
		Lanyin Mandarin（Central Plains Mandarin）	650000-新疆维吾尔自治区	1174 0.78%	
Tibetic languages	Tibetic languages	Tibetic languages	540000-西藏自治区	0 0.00%	0 0.00%
Lower Yangtze Mandarin	Lower Yangtze Mandarin	Lower Yangtze Mandarin（Central Plains Mandarin）	340000-安徽省	5319 3.52%	17749 11.75%
		Lower Yangtze Mandarin Wu Chinese	320000-江苏省	12430 8.23%	
Wu Chinese	Wu Chinese	Wu Chinese	330000-浙江省	10081 6.67%	20774 13.75%
		Wu Chinese	310000-上海市	10693 7.08%	
Southwestern Mandarin	Xiang Chinese	Xiang Chinese（Southwestern Mandarin）	430000-湖南省	2995 1.98%	2995 1.98%
	Southwestern Mandarin	Southwestern Mandarin	420000-湖北省	5845 3.87%	20028 13.26%
		Southwestern Mandarin	500000-重庆市	4503 2.98%	
		Southwestern Mandarin	510000-四川省	8374 5.54%	
		Southwestern Mandarin	520000-贵州省	600 0.40%	

		Southwestern Mandarin	530000-云南省	706 0.47%	
Yue-Min-Hakka Chinese（Gan Chinese）	Gan Chinese	Gan Chinese（Hakka Chinese）	360000-江西省	6255 4.14%	6255 4.14%
	Min Chinese	Min Chinese	350000-福建省	3801 2.52%	3801 2.52%
	Yue Chinese	Yue Chinese（Min Chinese，Hakka Chinese）	440000-广东省	9135 6.05%	15513 10.27%
		Yue Chinese Southwestern Mandarin (Hakka Chinese）	450000-广西壮族自治区	5030 3.33%	
		Yue Chinese（Min Chinese）	460000-海南省	1244 0.82%	
		Yue Chinese	810000-香港特别行政区	104 0.07%	
		Yue Chinese	820000-澳门特别行政区	0 0.00%	
	Hakka Chinese	Hakka Chinese（Min Chinese）	710000-台湾省	442 0.29%	442 0.29%
				151069 s	41.96 h

Table A.8.: Commen voice: CN time length of accent.

A.2. Figures

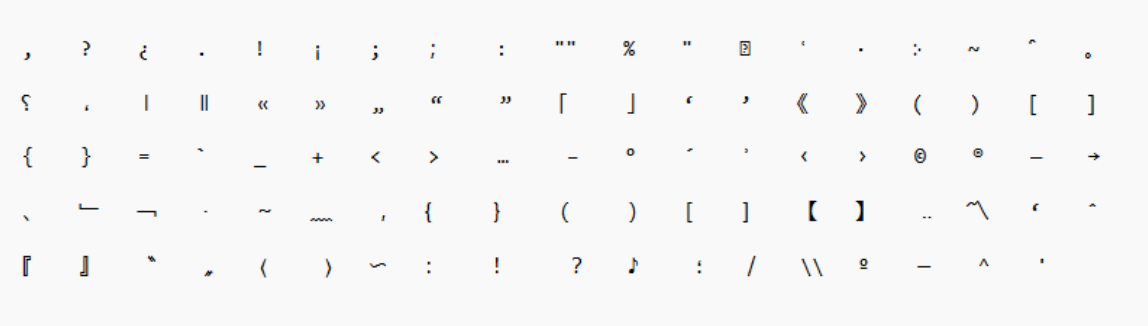

Figure A.1.: Special characters