ANALYZING AND COMPARING DIFFERENT SELF-SUPERVISED LEARNING SPEECH PRE-TRAINED MODELS IN THE VIEW OF PHONETICS

ANALYZING AND COMPARING DIFFERENT SELF-SUPERVISED LEARNING SPEECH PRE-TRAINED MODELS IN THE VIEW OF PHONETICS

Thesis

to obtain the degree of Master of Science at the Delft University of Technology, to be defended publicly on June 28th, 2022.

by

Hang JI (4759745)

Embedded Systems (Software & Networking),
Delft University of Technology, Delft, Netherlands,
born in China.

This thesis project is supervised by

Dr. Odette Scharenborg

Thesis committee:

Dr. Odette Scharenborg	(Chair)
Organisation:	Delft University of Technology
Faculty:	Electrical Engineering, Mathematics and Computer Science
Section:	Multimedia Computing

Dr. Fernando A. Kuipers	
Organisation:	Delft University of Technology
Faculty:	Electrical Engineering, Mathematics and Computer Science
Section:	Embedded and Networked Systems

怕什么真理无穷，进一寸有一寸的欢喜。

pà shén mè zhēn lǐ wú qiǒng，jìn yī cùn yǒu jìn yī cùn dè huān xī。

胡适

hú shì

CONTENTS

PREFACE

This thesis is an original work conducted during my study in the SpeechLab led by Odette Scharenborg. This work implemented a series of experiments, such as articulatory feature probing tasks and phoneme recognition tasks, and quantitatively compared and analyzed self-supervised learning speech pre-trained models. The idea of this thesis work was inspired by the benchmark for comparing self-supervised learning speech pre-trained models proposed by Yang et al. in 2021. My work aimed to bridge the gap between the new technology in the field of speech and the expertise in the field of phonetics. I hope that readers could gain the knowledge of how to utilize these self-supervised learning speech pre-trained models from this thesis work.

During my study in Delft, many things have occurred. The coronavirus and the war have changed the world beyond our expectations. I feel fortunate that I have gone through this dark time with my continuous effort and help from other people. I would like to express my sincere gratitude to my supervisor Dr. Odette Scharenborg who guided me in this thesis work and provided her expertise in phonetics. Meanwhile, I would like to thank Dr. Tanvina Patel for her technical support on my thesis work. Moreover, I would like to thank Odette and Tanvina again for revising my Interspeech paper and my thesis. Besides, It's a pleasure for me that I could participate in the SALT group meeting organized by Bence. These meetings make me jump out of the box and catch up with the latest techniques and topics in speech technology, for example, the work of Yuanyuan Zhang and Yixuan Zhang for mitigating the bias in Dutch ASR. Thanks to all who shared their ideas on these meetings. I also would like to thank Dr. Fernando A. Kuipers for accepting my invitation to be my thesis committee member. Last but not least, I'm grateful for my family. They provide me with their unconditional love and support.

The journey in Delft has come to an end. I believe that these experiences gained in this journey make me a better and braver person. I Wish everyone all the best.

Hang Ji
Delft, June 2022

1

INTRODUCTION

In this chapter, Section 1.1 gives a brief overview of the background knowledge, including ASR and self-supervised speech pre-trained models. It also includes the motivation behind this thesis work. Section 1.2 puts forward the research questions. And Section 1.3 presents an overview of the following chapters.

1

1.1. MOTIVATION

Automatic speech recognition (ASR) is to transform speech audio data into a sequence of modeling units. These modeling units could be phonemes, words, etc. The technology of ASR has a giant leap forward due to the application of deep learning techniques [1]. ASR has been widely used in many aspects of our daily life, for example, voice assistants in banking, healthcare, and marketing, etc.

However, there still exist many challenges unresolved in the field of ASR. For example, traditional ASR systems rely on a large volume of paired transcriptions and speech audio data; however, most languages even lack standard orthography [2]. Therefore, many techniques have been put forward to tackle this challenge. These techniques could be broadly classified into three directions, data augmentation, new architectures, and training strategies [3, 4, 5, 6]. In particular, this thesis work focuses on specific new architectures, which are different self-supervised learning (SSL) speech pre-trained models [7, 5, 8].

Inspired by pre-trained text-based language models which do not require labeled data from the field of natural language processing (NLP)[9], various SSL speech pre-trained models have been put forward recently and shown great potential in the field of ASR [10]. Generally, a speech pre-trained model is a model trained by a large amount of speech audio data before solving a particular ASR task, for example, a phoneme recognition task. Meanwhile, compared to supervised learning, which can only learn information from speech audio data with transcriptions, the self-supervised learning (SSL) technique enables the pre-trained model to learn information from speech audio data without transcriptions. With the low demand for labeled training speech audio data, SSL speech pre-trained models have been regarded as a potential solution for tackling various challenges in ASR, for instance, building ASR systems for low-resource languages which lack standard orthography and transcriptions [11].

SSL speech pre-trained models could be used to extract speech representations from the speech audio data. These speech representations are similar to standard acoustic feature vectors, which are the input of an ASR system. The standard acoustic feature vectors are extracted by feature extraction methods based on signal processing, such as MFCC and PLP, etc [12]. Similar to the standard acoustic feature vectors, speech representations also contain useful linguistic information and discard redundant information for ASR tasks. Thus, speech representations could replace these standard acoustic feature vectors and be utilized as the input of an ASR system.

Moreover, many works have investigated the utilization of these speech representations extracted by SSL speech pre-trained models in ASR and related tasks [5, 13, 14, 15, 16, 17]. These works have shown that these speech representations could improve the performance of an ASR system. For example, Poncelet et al. showed that the speech representations extracted from an SSL speech pre-trained model could improve the performance of a Dutch ASR system [13]. Specifically, their dutch ASR system adopted speech representations instead of standard acoustic feature vector as input and achieved better recognition results.

Despite the good recognition performance achieved by SSL pre-trained models, why these SSL pre-trained models could improve the ASR performance remain obscured to us. Several works have been put forward to answer this question [10, 18, 19, 15, 16]. For

example, Pasad et al. implemented a suite of analysis tools based on non-parametric methods to uncover the information encoded by speech representations extracted by an SSL speech pre-trained model, wav2vec 2.0 [18]. Their experiments showed that these speech representations could encode acoustic and linguistic information. Riviere et al. analyzed the phoneme separability of SSL speech pre-trained models by a distance-based metric, ABX score [19]. The ABX score indicated how phones are separated into different phonemes by speech representations generated by the SSL speech pre-trained model.

Although these works have demonstrated why and how these SSL pre-trained models work well for ASR tasks, the articulatory feature (AF) information captured by different SSL pre-trained models has not been analyzed and compared yet. Articulatory features interpret how different vocal organs are involved in producing different phones. These articulatory features could also be regarded as the building blocks of different phones. In addition, the relationship between the articulatory feature information captured by SSL speech pre-trained models and the performance of phoneme recognition task which utilizes these SSL speech pre-trained models are interesting to investigate. The amount of AF information captured by SSL speech pre-trained models could indicate the performance of phoneme recognition. Meanwhile, the transferability of SSL speech pre-trained models is also interesting to investigate. The transferability means that SSL speech pre-trained models trained on one language could also perform well in capturing AF information and phoneme recognition tasks on other languages. The transferability could improve the ASR performance for low-resource languages by utilizing SSL speech pre-trained models which are trained on high-resource languages.

1.2. RESEARCH QUESTIONS

Research questions driven by the motivation of this work are proposed as follows:

- **RQ1, within-language scenario**
  - ***RQ1.1*** *What articulatory feature information is modeled by different SSL speech pre-trained models?*
  - ***RQ1.2*** *How does the articulatory feature information correlate to phoneme recognition performance in the same language, i.e., English?*
- **RQ2, cross-language scenario**
  - ***RQ2.1*** *To what extent is the articulatory feature information from a different language modelled by different SSL speech pre-trained models*
  - ***RQ2.2*** *How does the articulatory feature information captured above correlate to phoneme recognition in this different language?*

In order to uncover what articulatory feature information is modeled by SSL speech pre-trained models for **RQ1.1** and **RQ2.1**, frame-level articulatory feature (AF) probing tasks based on support vector machines (SVMs) are implemented in this work. Since the work aims to understand what AF information is captured by SSL speech pre-trained models, classifiers adopted in these frame-level AF probing tasks are not required to have

a strong ability of classifying non-linearly separable data. If the classifier have strong nonlinear separability, the results of the probing tasks of different SSL speech pre-trained models would be too close to discriminate.

Moreover, phone-level ASR systems are implemented to demonstrate the performance of SSL speech pre-trained models on phoneme recognition tasks for **RQ1.2** and **RQ2.2**. The input of these phone-level ASR systems are speech representations extracted by SSL speech pre-trained models. They are compared with a baseline system which adopts the standard input, MFCC acoustic feature vectors, for the ASR systems. To understand the relationship between the amount of AF information and the performance of phoneme recognition, Pearson's correlation coefficient between the performances of frame-level AF probing tasks and the performance of phone recognition tasks are computed. If the coefficient is greater than zero and closer to "1.0", it indicates a positive relationship between the amount of AF information and the phoneme recognition performance. More AF information captured by SSL speech pre-trained models could have high performance of phoneme recognition tasks.

Lastly, three state-of-the-art SSL speech pre-trained models, CPC [7], wav2vec 2.0 [5], and HuBert [8] are compared and analyzed by frame-level probing tasks and phoneme recognition tasks for our research objectives. Wav2vec 2.0 and HuBert are derived from CPC. The three SSL speech pre-trained models are different in their architectures and objective functions. These differences might lead to a different performance in capturing AF information and phoneme recognition tasks. While this thesis work is not limited to these three SSL speech pre-trained models. Other SSL speech pre-trained models, such as APC [20], could also be studied by methods proposed in this thesis work. Due to the limitation of time and computation resources, this work firstly focuses on CPC, wav2vec 2.0 and HuBert, and other SSL speech pre-trained models could be added in the future.

1.3. THESIS OUTLINE

In this thesis, Chapter 2 introduces the requisite knowledge of this work. Chapter 3 explains methods and implementations for tackling these research questions. Chapter 4 presents the experiment results. Chapter 5 gives findings from these results and answers to the research questions. Chapter 6 gives conclusions and an outlook of this work.

2

BACKGROUND

In this chapter, the required background knowledge for this thesis work is explained. In Section 2.1, an overview of ASR is briefly introduced. The introduction of MFCC feature extraction method, which is the baseline feature extraction method used in this thesis work, is also explained. The explanation of ASR systems used in phoneme recognition tasks for **RQ1.2** and **RQ2.2** are included as well. In Section 2.2, neural networks used in ASR are introduced. In Section 2.3, SSL speech pre-trained models analyzed in this work are explained. In Section 2.4, articulatory features which are researched in **RQ1.1** and **RQ2.1** are introduced. In Section 2.5, support vector machines which are adopted in frame-level probing tasks for **RQ1.1** and **RQ2.1** are explained. In Section 2.6, related works of this thesis work are introduced.

2.1. ASR

This section briefly introduces feature extraction, conventional ASR systems and end-to-end ASR systems. It also compares the two ASR systems and clarifies why the end-to-end ASR system for phoneme recognition tasks is adopted in this thesis work. The neural networks used in ASR are also introduced in this section.

2.1.1. AN OVERVIEW OF THE PROCESS AND HISTORY OF ASR

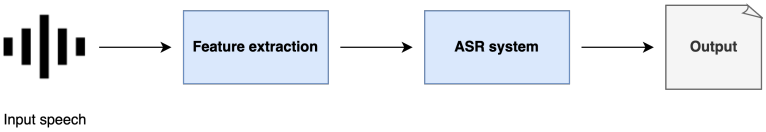

Figure 2.1: The pipeline of ASR

Automatic speech recognition (ASR) is to map a waveform of speech audio data into a sequence of target modeling units. These modeling units could be phonemes, graphemes, or words, etc. Figure 2.1 demonstrates the general pipeline of ASR, and a brief explanation of it is given in the following part.

Firstly, input speech audio data is transformed into a sequence of feature vectors in feature extraction. Feature extraction aims to generate acoustic feature vectors with desirable characteristics for ASR tasks: these acoustic feature vectors should 1. contain sufficient information to distinguish different modelling units, such as different phonemes; 2. robust against speaker variations; 3. robust against noise and channel distortions [21]. Feature extraction methods include traditional methods based on signal processing, such as MFCC and PLP [21], and data-driven methods such as SSL speech pre-trained models. In addition, this thesis work compares different SSL speech pre-trained models as feature extraction methods, with MFCC as the baseline method.

Secondly, the ASR system models the posterior probability $P(W|X)$, where W represents the predicted sequence of modeling units, X represents the sequence of acoustic feature vectors X. The ASR system aims to find the most likely sequence of modeling units W^* given the observed acoustic feature vectors, and W^* is given by

$$W^* = \underset{W}{\operatorname{argmax}} P(W|X) \tag{2.1}$$

With the development of deep learning, conventional ASR systems which model the posterior probability based on Bayes' theorem has been gradually replaced by end-to-end ASR systems, which directly model the posterior probability [22].

FEATURE EXTRACTION

Feature extraction aims to generate acoustic feature vectors from input speech audio data and remove redundant information such as noise. This part introduces the feature extraction method of Mel-frequency cepstral coefficients (MFCC), which is the baseline feature extraction method used in this thesis work.

The feature extraction method of MFCC extracts acoustic feature vectors by exploiting signal processing methods, and it aims to mimic how humans perceive speech audio data in their ears. The process of MFCC is shown in Figure 2.2

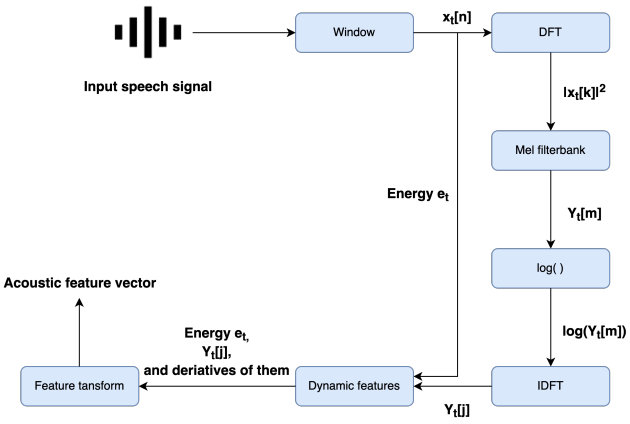

Figure 2.2: The process of MFCC

As Figure 2.2 illustrates, firstly, the windowing operation is applied to the input speech signal. The input speech signal is constantly changing over time. The window operation aims to obtain short segments known as **frames**. These frames are short-time stationary and contain information of a particular phoneme. Typically, the window length is 25ms, and the shift length is 10ms [12]. It is presented in Figure 2.3.

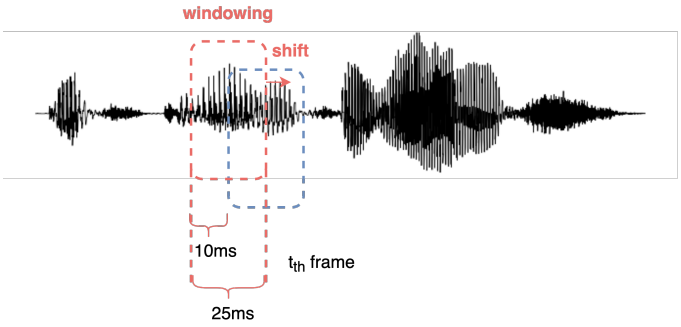

Figure 2.3: The window operation

Secondly, the discrete Fourier transform (DFT) operation is applied to the frame $x_t[n]$. It aims to extract the frame's power spectral information $|x_t[k]^2|$. The power spectral information $|x_t[k]^2|$ presents the frequency components and their amplitude of the frame $x_t[n]$. This process is inspired by human ears, which distinguish different speech signals by their different frequencies. For example, the DFT operation of vowel /iy/ is shown in Figure 2.4.

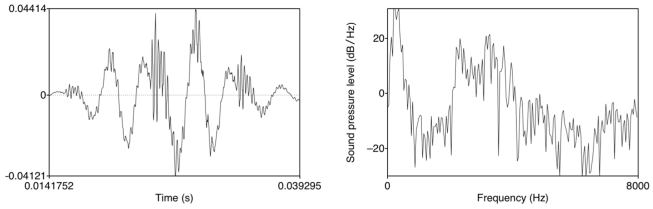

Figure 2.4: The DFT operation of vowel /iy/, the left is /iy/ in time domain, the right is /iy/ in frequency domain, from [21]

Thirdly, the mel filterbank is applied to the power spectrum $|x_t[k]^2|$. The mel filterbank consists of a series of filters, and these filters are applied to the power spectrum to extract different amounts of information given different frequencies. This is also inspired by human ears, which have different sensibilities of speech signals at different frequencies. Human ears are more sensitive to low frequencies and less sensitive to high frequencies.

Afterwards, the log operation is applied to the filtered power spectrum $Y_t[m]$. This aims to mimic how human ears perceive speech signals with different power, which means the loudness. Finally, the operation of inverse discrete Fourier transform (IDFT) is applied on the processed power spectrum $\log(Y_t[m])$ to obtain cepstral coefficients. These cepstral coefficients together with their first order derivatives, second order derivatives, and the energy compose the acoustic feature vector of the input frame.

CONVENTIONAL ASR SYSTEMS

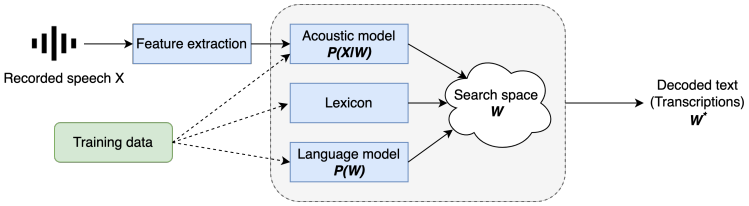

Figure 2.5: The framework of conventional ASR systems

Conventional ASR systems factorize the posterior probability by Bayes' theorem, which is given as follows:

$$P(W|X) = \frac{P(X|W)P(W)}{P(X)} \tag{2.2}$$

where $P(X|W)$ is computed by the acoustic model. The acoustic model maps the input sequence of acoustic feature vectors to their matched phonemes. **Phonemes** are abstract units defined by linguists based on contrastive role in word meanings [21]. $P(W)$ is computed by the language model. The language model is a probability distribution over sequences of words.

In Figure 2.5, the features of recorded speech are extracted at first. Secondly, the acoustic model is trained on the recorded speech with its corresponded phoneme labels,

and the language model is trained on the text data. Typically, the language model utilizes far more training text data. The lexicon, which is usually a handcrafted pronunciation dictionary, maps words to phonemes [22]. The most likely transcriptions are searched by the multiplication of the output probabilities computed from the acoustic model and the language model.

END-TO-END ASR SYSTEMS

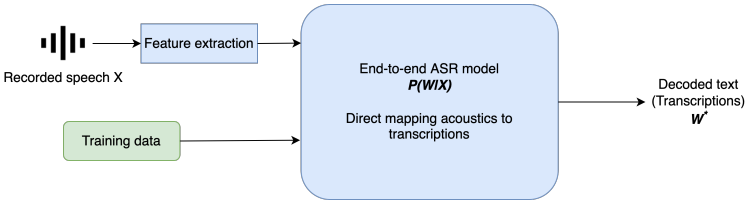

Figure 2.6: The framework of end-to-end ASR systems

Compared with conventional ASR systems, end-to-end ASR systems do not factorize the posterior probability. End-to-end ASR systems directly model the posterior probability by leveraging the deep learning architectures for sequence processing [23, 24].

In Figure 2.6, similar as conventional ASR systems, acoustic feature vectors from recorded speech X are extracted first. Secondly, the end-to-end ASR model is trained with these feature vectors and their paired text data. In general, these end-to-end ASR systems are composed of two modules, an encoder and a decoder. The encoder extracts hidden representations from the input sequence. The decoder tackles the alignment between the hidden representations and the output sequence [22].

COMPARISON BETWEEN CONVENTIONAL ASR SYSTEMS AND END-TO-END ASR SYSTEMS

The key difference between conventional ASR systems and end-to-end ASR systems is whether they factorize the posterior probability $P(W|X)$. Compared with conventional ASR systems, end-to-end ASR systems circumvent the following problems by modeling the acoustic model and the language model together:

- **Complex decoding**: In the decoding process of conventional ASR systems, different models including acoustic model, language model, and lexicon require to be integrated together. Unlike conventional ASR systems, the decoder in the end-to-end systems directly search for the target output sequence, and do not need to integrate the acoustic model, the language model and the lexicon.

- **Incoherence in optimization**: For conventional ASR systems, different models including the acoustic model and the language model are trained separately with different training objective functions. Thus, it could result in incoherence in the training process. While the end-to-end systems eliminate the incoherence due to the utilization for only one model in the system.

Since end-to-end ASR systems mitigate the problems of conventional ASR systems, this thesis work adopts the end-to-end ASR system in phoneme recognition tasks to un-

cover the relationship between articulatory feature information and phoneme recognition performance. Specifically, hybrid CTC/attention end-to-end ASR system [22] is utilized. The following sections give a brief introduction of detailed explanations of CTC-based end-to-end ASR system and attention end-to-end ASR system. They also point out the advantages and disadvantages of these two systems. Lastly, the hybrid CTC/attention end-to-end ASR system is explained.

2.1.2. CTC-BASED END-TO-END ASR

The CTC-based end-to-end ASR systems adopt the connectionist temporal classification (CTC) loss with an encoder. Graves et al. put forward the CTC loss for labelling unsegmented sequence data, such as speech signals [25]. The encoder extracts hidden representations of the sequence of speech signals. It utilizes different deep learning architectures (They are introduced in Section 2.2) for sequence processing. For example, Deep Speech, which is a CTC-based end-to-end ASR systems, adopts multiple layers of recurrent neural networks (RNN, and it's introduced in Section 2.2) in its encoder [23]. The following part presents more detailed explanations of CTC and the ASR systems.

CONNECTIONIST TEMPORAL CLASSIFICATION

Connectionist temporal classification is a decoding method for sequence modeling, such as speech recognition, handwriting recognition, etc. Specifically, it eliminates the issue of lacking forced alignments, which are the time-aligned transcriptions of the speech audio data. CTC solves this issue by introducing a new token, which is a blank token, in the output modeling units.

More detailed, in the decoding process, CTC outputs frame-wise labels for the input sequence. For some frames, they do not represent solid modeling tokens, for example, an actual phoneme. Thus, CTC labels these frames with the blank token ϵ. Afterwards, possible sequences with valid alignments are kept, and the conditional probability of these possible sequences is computed as follows:

$$P(Y|X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} P_t(a_t|X) \tag{2.3}$$

where $A_{X,Y}$ represents the set of all valid alignments, $P_t(a_t|X)$ represents the probability for the single alignment of a frame. The valid alignment means the output sequence with the merged frame-wise labels is in accordance with the ground truth output sequence. CTC complies **the following rules** to merge frame-wise labels:

- If a frame-wise label is followed by the same label, the two labels are merged into one label. For example, "ll" is converted to "l".
- If the blank token ϵ is inserted into two labels which could be different or the same, the blank token is removed. For example, "lϵl" is converted to "ll", "eϵl" is converted to "el".
- If the blank token appears at the beginning or the end of the output sequence, it is removed.

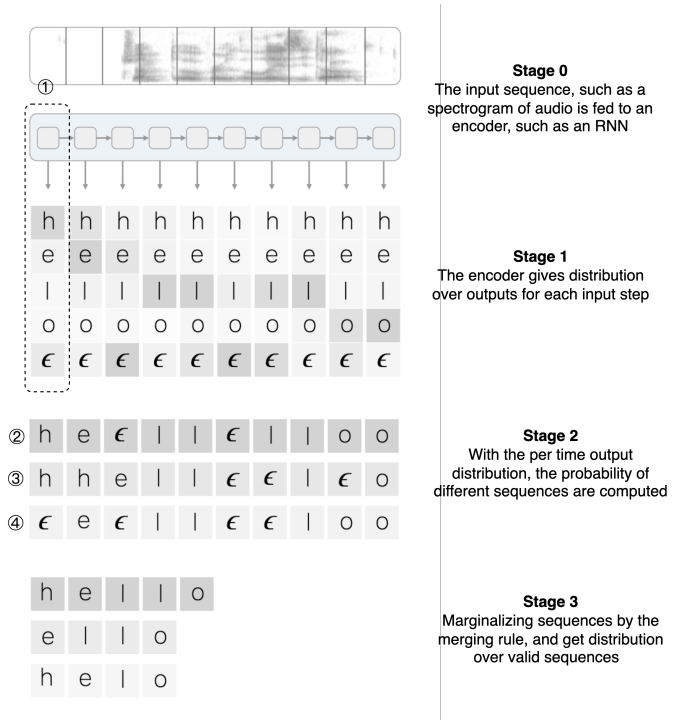

Figure 2.7: The process of CTC alignments

For example, Figure 2.7 presents CTC alignments of the input sequence with the ground truth label “hello”. In Figure 2.7, the input sequence, such as a spectrogram of the audio is fed to an encoder, such as an RNN at Stage 0. At Stage 1, the encoder generates probability distribution $P_t(a_t|X)$ over output tokens {h, e, l, o, ϵ} for each input step. For example, the probability distribution of the first input step over different output tokens are shown in ①, and the darker color represents the higher probability distribution. At Stage 2, the probabilities of different output sequences are computed. These output sequences are combinations of output tokens of each step. At Stage 3, output sequences are merged by the rule mentioned above, and the distributions over valid sequences are added together for further optimization. For example, the output sequences of ② and ③ are valid after merging, and they represents "hello", the ground truth transcription. While, the output sequence of ④ is not valid after merging, and it represents "ello".

In order to improve the computing efficiency of CTC, the technique of dynamic programming is applied in the training process, and the technique of beam search is applied in the inference process [26].

2.1.3. Attention-based end-to-end ASR

The attention-based end-to-end ASR adopts attention mechanism in its decoder. Like CTC, the attention mechanism also eliminates the issue of lacking forced alignments.

The attention mechanism was first put forward in the field of machine translation in 2015 [27]. The idea aims to generate the output sequence from an input sequence, and do not need to provide exact output labels for each frame of the input sequence. Typically, the lengths of the input sequence and the output sequence are different and not fixed. The problem of speech recognition also suits this paradigm. Thus, the attention mechanism has been widely used in the field speech recognition [24]. For example, the end-to-end ASR systems Listen, Attend, and Spell (LAS) adopts the attention mechanism in its structure [24]. The following part presents a more detailed explanation of attention mechanism.

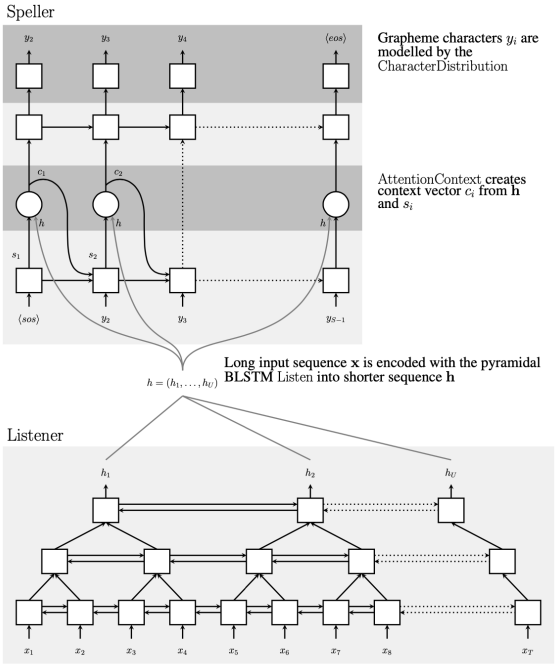

Figure 2.8: The structure of LAS, from [24]

ATTENTION MECHANISM

The attention mechanism performs soft alignments between the input sequence and the output sequence. The soft alignment means that each character output y_i is conditional dependent on the previous output characters $y_{<i}$ and the entire sequence of the input signal. They are achieved by the architectures utilized in the system. For example, in the structure of LAS as shown in Figure 2.8, the encoder (Listener) extracts hidden representations h from the input sequence x, the decoder (Speller) performs further attending and decoding processes on these hidden representations h for soft alignments.

To be specific, the probability distribution of each output character is the function of the decoder state s_i and the context c_i, which is shown as follows:

$$P(y_i|x, y_{<i}) = \text{CharacterDistribution}(s_i, c_i) \tag{2.4}$$

The decoder state s_i depends on the previous decoder state s_{i-1}, the previous output character y_{i-1} and the previous context c_{i-1}, which is shown as follows:

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \tag{2.5}$$

The context c_i is computed by attending hidden representations h by current decoder state s_i, which is shown as follows:

$$c_i = \text{AttentionContext}(s_i, h) \tag{2.6}$$

The context c_i can be regarded as a weighted sum of hidden representations $h_u \in h$, which is shown as follows:

$$e_{i,u} = < \phi(s_i), \psi(h_u) > \tag{2.7}$$

$$\alpha_{i,u} = \frac{exp(e_{i,u})}{\sum_u exp(e_{i,u})} \tag{2.8}$$

$$c_i = \sum_u \alpha_{i,u} h_u \tag{2.9}$$

The weight parameter $\alpha_{i,u}$ is computed by the softmax function of the scalar energy parameters $e_{i,u}$ over the whole input sequence. The scalar energy parameter $e_{i,u}$ is computed by the multiplication of the linear transformed decoder state s_i and the hidden representation h_u. ϕ and ψ are MLP networks.

In the training process, LAS adopts the ground truth label as the previous output character y_{i-1}. In the inference process, the most likely character sequence given the input speech signals $\hat{y} = arg\max_y logP(y|x)$ is to be found.

2.1.4. Hybrid CTC/attention end-to-end ASR

From previous sections, CTC-based end-to-end systems and Attention-based end-to-end systems solve the lack of forced alignments in the training data by different methods. Thus, these two systems have different characteristics. These characteristics lead to different advantages and shortcomings between the two systems. The following parts present more detailed explanations of the characteristics of CTC and attention mechanism, and demonstrate the structure of hybrid CTC/Attention end-to-end ASR system.

As concluded from section 2.1.2, the CTC-based end-to-end system has the following characteristics:

- The relationship between the input sequence and the output sequence is **monotonic**, which means that the output sequence is generated by the input sequence from left to right.
- The relationship between the input sequence and the output sequence is **many-to-one**, which means that many frames from the input sequence are mapped into one output units in the output sequence.
- It makes several conditional independence assumptions (**Markov assumptions**).

As concluded from section 2.1.3, the Attention-based end-to-end system has the following characteristics:

- The relationship between the input sequence and the output sequence is **not monotonic**.
- It makes **no Markov assumptions**.

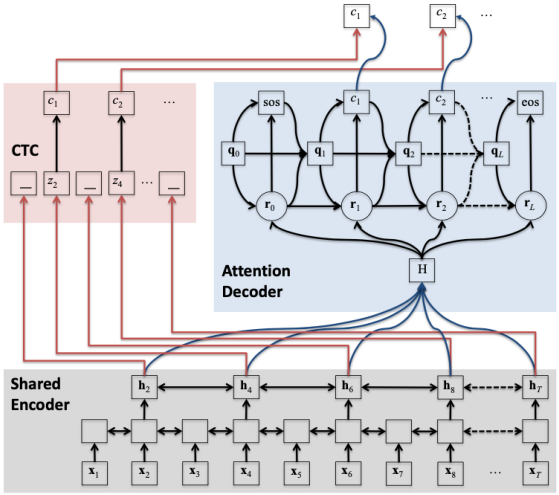

Figure 2.9: The structure of hybrid CTC/Attention end-to-end systems from [22]

By integrating CTC and attention-mechanism, the ASR system could output more rigorous alignments with the monotonic characteristic in CTC, and it could also do not need to rely on the conditional independence assumption, which does not occur the real-life scenario of speech recognition. The structure of the hybrid CTC/Attention ASR system is shown in Figure 2.9. It takes full advantage of CTC and attention mechanism by adopting the multiobjective learning based on the two decoding methods.

In Figure 2.9, an encoder for generating hidden representations h is shared between the CTC decoder and the attention decoder. The system is optimized by minimizing the multiobjective learning loss which is shown as follows:

$$\mathscr{L}_{MOL} = \lambda \log P_{ctc}(y|x) + (1-\lambda)\log P^{*}_{att}(y|x) \tag{2.10}$$

where λ $(0 \leq \lambda \leq 1)$ is a tunable parameter.

2.2. NEURAL NETWORKS USED IN ASR

Neural networks originated from the idea of neurons, which was proposed by McCulloch et al. in 1943 [28]. Neural networks leverage multiple layers of neurons to solve the classification or regression problems. These neurons have high non-linear separability. For example, in ASR, neurons could covert the input acoustic feature vectors to hidden representations that are easier to make classifications.

In general, neural networks contain multiple layers of neurons. Many problems in reality cannot be modeled linearly. For example, the relationship between the input acoustic feature vectors and their related phonemes is not linear. Neurons can convert these input acoustic feature vectors to representations in a subspace. In this subspace, the converted representations and their related phonemes could be modeled linearly. The mechanism behind the neurons is called non-linear separability. Moreover, the neurons have parameters to set for non-linear separability. The optimal parameters are found in the training process of the neural network. In the training process, the neural network generates the predicted values of the input acoustic feature vectors. This process is called forward propagation. Subsequently, the predicted values are compared with the ground truth value. The difference between predicted values and ground truth values is called loss. The parameters of the neurons are found by minimizing the loss. The minimum loss is optimized by gradient descent of the neural networks. The gradient descent updates the parameters and computes the loss in each iteration. This process is called backward propagation. When the loss does not change or the steps of iteration arrives at a specific number, the back propagation stops with the optimal parameters. Then, the neural network with the found parameters could be used on the test data.

Typically, the convolutional neural network (CNN), the recurrent neural network (RNN) are adopted in the end-to-end ASR systems [22, 29, 23]. The following part gives explanations for these neural networks.

CNN

Convolutional neural networks (CNNs) were firstly forward by LeCun et al. in 1995 [30]. Convolutional neural networks are simply neural networks that use convolution instead of general matrix multiplication on its input. In the end-to-end ASR system, 1-D CNNs are usually used for downsampling [29]. Downsampling could convert the sequence of input acoustic feature vectors to a shorter sequence, which could reduce the space complexity and the time complexity in further computation.

RNN

The idea of recurrent neural networks (RNNs) was firstly put forward by Elman in 1990 [31]. In end-to-end ASR systems, RNN could be used in the encoder to generate hidden representations from input speech audio sequence. RNN is unrolled in time to process the input speech audio sequence with different lengths. The output of current time step depends on the input of current time step and the hidden representation of previous time step. The computation of RNN for each time step is shown as follows:

$$h_t = g(Uh_{t-1} + Wx_t) \tag{2.11}$$

$$y_t = f(Vh_t) \tag{2.12}$$

h_t is the hidden representation at time step t, x_t is the input at time step t, and h_{t-1} is the hidden representation at previous time step. $U \in \mathbb{R}^{d_h \times d_{in}}$, $W \in \mathbb{R}^{d_h \times d_{in}}$, and $V \in \mathbb{R}^{d_{out} \times d_h}$ are weight matrices for h_{t-1}, x_t, and h_t respectively. In addition, the weight matrix (U, W, or V) is shared in the unrolling process in different time steps. g is an activation function. f could be a softmax function to provide probability distribution over the possible output classes.

However, there exist several limitations in RNN: **1**. the output of a time step only conditionally depends on a context with very limited history information; **2**. Weight matrices W and U are responsible for capturing useful history information for the current output and providing useful information for future outputs; **3**. During the training process, the gradient descent would occur due to multiplication of W or U following the chain rule in the backpropagation. Thus, the varieties of RNN, the long short-term memory (LSTM) networks and gated recurrent units (GRU) networks have been put forward to solve these limitations [32, 33].

2.3. SSL SPEECH PRE-TRAINED MODELS

Self-supervised learning (SSL) speech pre-trained models are models trained on huge amounts of unlabeled speech audio data [5]. These SSL speech pre-trained models could be used in two ways: **1**. Use these SSL speech pre-trained models to extract feature vectors, which are called speech representations, from speech audio data, and apply these speech representations as the input of ASR systems. In addition, these speech representations contain more linguistic information, and are more robust to unrelated information of ASR tasks, such as noise [17, 34]; **2**. Fine-tune SSL speech pre-trained models with labeled speech audio data for ASR tasks [35]. The first way of utilizing SSL speech pre-trained models is researched in this thesis work. Specifically, speech representations extracted by different SSL speech pre-trained models, CPC[7], wav2vec 2.0 [5] and HuBert [8] are analyzed and compared in this thesis work. Meanwhile, these speech representations are also compared with the traditional speech feature extraction method, MFCC. Thus, this section gives introduction of SSL speech pre-trained models researched in this work, and compares these SSL speech pre-trained models. It also compares SSL speech pre-trained models with MFCC as feature extraction methods.

2.3.1. AN OVERVIEW OF SSL SPEECH PRE-TRAINED MODELS

SSL speech pre-trained models adopt the self-supervised learning strategy. Compared with supervised learning which learns from labeled data, the paradigm of self-supervised learning learns from unlabeled training data, for example, speech audio data without transcriptions. Self-supervised learning is able to leverage the huge amount of unlabeled speech audio data without extra human labour for annotations. Speech pre-trained models based on self-supervised learning enable themselves to learn from the unlabelled training data in two ways: **1**. distinguishing the target data from a set of negative samples, namely the contrastive predictive loss; **2**. or reconstructing the target data, namely, the reconstruction loss [35]. SSL speech pre-trained models which learn by contrastive predictive loss are compared and analyzed in this thesis work.

2.3.2. CONTRASTIVE PREDICTIVE CODING

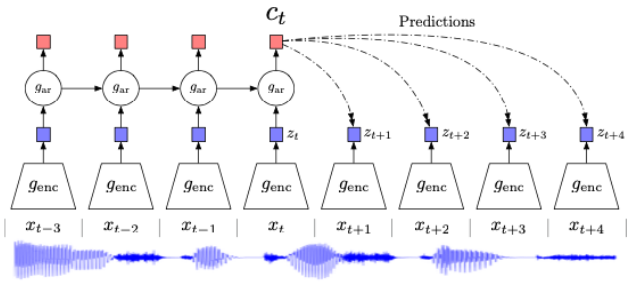

Figure 2.10: The structure of CPC, from [7]

Contrastive Predictive Coding (CPC) [7] includes an encoder module and a context module. In Figure 2.10, the encoder module generates a latent representation $z_t = g_{enc}(x_t)$

from a frame x_t of the raw speech audio data sequence. Afterwards, the context module generates the context representation $c_t = g_{ar}(z_t)$. The context representation c_t is conditionally dependent on the context representations of previous time steps, which means the history context representations also have influences on the context representations of current time step.

Typically, the encoder consists of a multi-layer convolutional neural network, and the context module consists of recurrent neural networks (RNNs) or variants of RNN such as LSTMs and GRUs [12].

The training objective is to minimize the contrastive loss, which aims to maximize the mutual information between the future latent representation z_{t+k} in k time steps ahead and the prediction value $W_k c_t$, and minimize the mutual information between the latent representation z_j from negative samples $Z = \{z_1, ...z_N\}$ and the prediction $W_k c_t$. W_k performs linear transformation of c_t. The training objective is optimized by minimizing the sum of the loss for different time step k, $k \in \{1, ...K\}$, following:

$$\mathcal{L}_{CPC} = -\frac{1}{K}\sum_{k=1}^{K}\left[log\frac{exp(z_{t+k}^T W_k c_t)}{\sum_{z_j \in Z} exp(z_j^T W_k c_t)}\right] \tag{2.13}$$

2.3.3. WAV2VEC 2.0

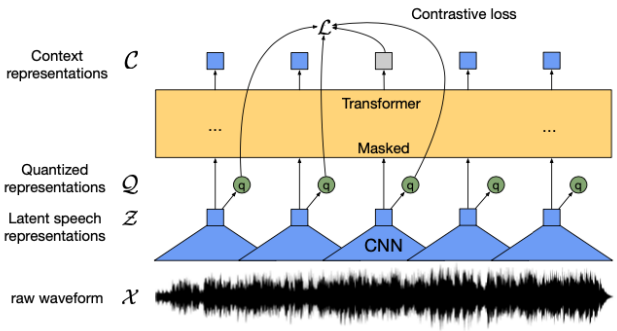

Figure 2.11: The structure of wav2vec 2.0, from [5]

Wav2vec 2.0 [5] has an encoder module, a context module, and a quantization module. In Figure 2.11, the encoder module generates latent representations $z_t = g_{enc}(x_t)$ from a frame x_t of the raw speech audio data sequence. Afterwards, the context module generates the context representation $c_t = g_{con}(z_t)$. Meanwhile, the quantization module converts the continuous latent representations z_t to a discrete representation $q_t = g_{quan}(z_t)$.

Typically, the encoder consists of a multi-layer convolutional neural network. The context module consists of a Transformer network, which is different from the context module of CPC. The quantization module utilizes product quantization [36], which converts latent representations to discrete representations by concatenating entries sampled from different codebooks. For example, the quantization module has G codebooks,

and each codebook has V entries of vectors in them. Then, we choose one entry from each codebook and concatenate the resulting vectors $e_1, ..., e_G$.

Similar to CPC, the training objective of wav2vec2.0 is to minimize the contrastive loss. Specifically, a portion latent representations z_t generated by the encoder module from input time steps are masked before fed to the context module. Afterwards, wav2vec2.0 aims to identify the true quantized latent speech representation q_t instead of the true latent speech representation as in CPC for the input of a masked time step. It is also known as the masked prediction loss, $\mathscr{L}_m$. Moreover, the training objective is augmented by a code diversity loss $\mathscr{L}_d$, which ensures the equal use of codebook entries from different codebooks. The training objective is as follows:

$$\mathscr{L}_{wav2vec2.0} = \mathscr{L}_m + \alpha \mathscr{L}_d \tag{2.14}$$

$$\mathscr{L}_m = -log \frac{exp(sim(c_t, q_t))}{\sum_{\tilde{q} \in Q_t} exp(sim(c_t, \tilde{q}))} \tag{2.15}$$

$$\mathscr{L}_d = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \overline{p}_{g,v} log \overline{p}_{g,v} \tag{2.16}$$

where α is a tuned hyperparameter, Q_t is the set of quantized candidate representations, which consist of the true sample and negative samples from other masked time steps. G is the number of codebooks. V is the number of entries that we want to equally use in each codebooks. $\overline{p}_{g,v}$ is the probability distribution of an entry in a batch of input sequences.

2.3.4. HuBERT

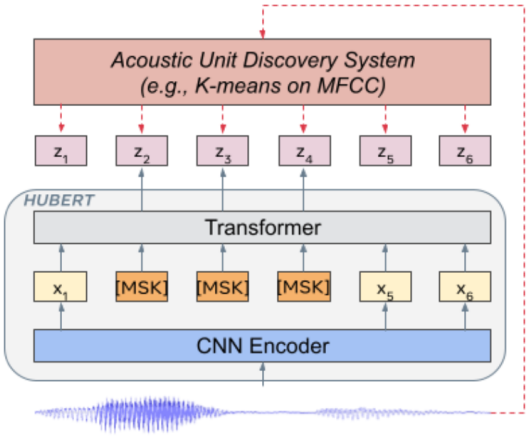

Figure 2.12: The structure of HuBert, from [8]

In Figure 2.12, HuBert [8] also consists of an encoder $z_t = g_{enc}(x_t)$ and a context module $c_t = g_{con}(z_t)$, which is identical to wav2vec 2.0. However, instead of the quantization module used in wav2vec 2.0, HuBert utilizes an offline acoustic unit discovery

(AUD) module which is a clustering module such as k-means. Before training HuBert, the AUD module assigns the related cluster to each frame x_t of the input raw audio data as its pseudo label $u_t \in [C]$. Typically, the encoder consists of a multi-layer convolutional neural network. The context module consists of a Transformer network, which is the same as wav2vec 2.0.

Similar to wav2vec 2.0, HuBert adopts the contrastive loss. Specifically, HuBert aims to identify the true embedding e_c of the pseudo label instead of the quantized latent speech representations in wav2vec 2.0. In addition, the unmasked time steps are included in computing the contrastive loss. The training objective is as follows:

$$\mathscr{L}_{HuBert} = \alpha \mathscr{L}_m + (1-\alpha)\mathscr{L}_u \tag{2.17}$$

where α is a tuned hyperparameter, and $\alpha \in [0,1]$. $\mathscr{L}_m$ or $\mathscr{L}_u$ is similar to Equation 2.15, with q_t replaced by the embedding e_c of the pseudo label $u_t \in [C]$.

2.3.5. COMPARISON BETWEEN SSL SPEECH PRE-TRAINED MODELS AND MFCC

In this section, we firstly compare different SSL speech pre-trained models including CPC, wav2vec 2.0 and HuBert. Then, we compare SSL speech pre-trained models with the traditional feature extraction method, MFCC.

CPC, WAV2VEC 2.0 AND HUBERT

Table 2.1: Overview of different SSL speech pre-trained methods

Method	Architecture	Objective function	Dimension
CPC	encoder + context module	contrastive prediction loss	256
wav2vec 2.0	encoder + quantization + context module	contrastive prediction + diversity loss	768
HuBert	encoder + context + AUD modules	contrastive prediction loss	768

As shown above, these SSL speech pre-trained models are different in architectures and their actual computation of contrastive loss. Table 2.1 gives an overview of the three SSL speech pre-trained models. The following part demonstrates their differences.

Compared with CPC, wav2vec 2.0 and HuBert utilize the Transformer network in their context modules. While, CPC utilize the RNN network or its variations including GRU and LSTM in its context module. In a nutshell, Transformer, RNN and its variations are all deep learning architectures for sequence processing. They process input sequences with variable lengths and capture information at different time steps in the input sequence.

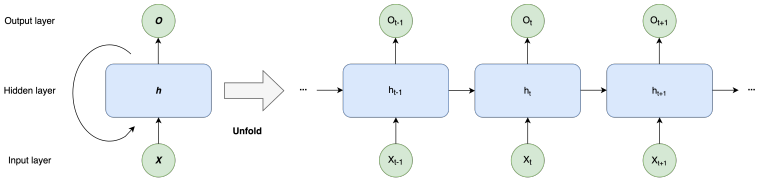

Figure 2.13: The example of RNN for sequence processing

In Figure 2.13, the input of the RNN in CPC is the sequence of hidden representations X generated by the encoder module. The RNN moves along the input sequence X to generate output representations O sequentially. It generates the output representation o_t of the input x_t frame by frame. The computation of the output representation O_t is shown as follows:

$$o_t = \text{RNN}(h_{t-1}, x_t) \tag{2.18}$$

where h_{t-1} is the hidden state of the previous frame x_{t-1}.

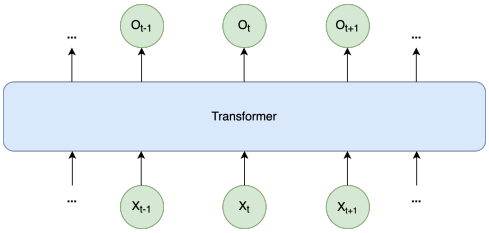

Figure 2.14: The example of Transformer for sequence processing

In Figure 2.14, the input of the Transformer in wav2vec 2.0 or HuBert is also the sequence of hidden representations X generated by the encoder module. Transformer generates output representations o_0, ...o_{t-1}, o_t, o_{t+1}... o_T of all frames of the input sequence X at once. The output representation O_t is the weighted sum of all frames from the input sequence X, which is shown as follows:

$$o_t = \text{Transformer}(x_0, ...x_{t-1}, x_t, x_{t+1}, ...x_T) \tag{2.19}$$

As a conclusion, the RNN adopted in CPC and the Transformer adopted in wav2vec2.0 and HuBert resulted in the following differences:

- **Sequential or parallel**: The RNN implemented by CPC could only generate output speech representations of frames from the input sequence sequentially. While, the Transformer utilized in wav2vec 2.0 and HuBert could generate output speech representations of frames from the input sequence in parallel. This characteristic of Transformer enable itself to fully utilize the modern computer architecture, GPU, which has the inherent attribute of parallel computing.

- **Dependencies**: The output speech representation of the RNN utilized in CPC is only dependent on the history frames and the current frame. While, the output speech representation of the Transformer utilized in wav2vec 2.0 and HuBert is dependent on the whole input sequence, including the history frames, the current frame, and the future frames. Although, a variation of RNN, the bidirectional RNN [37] could make the output representation depended on future frames, CPC in this thesis work doesn't adopt this architecture.

Compared with CPC, apart from the difference in architectures of the context module, wav2vec 2.0 and HuBert also have additional modules (quantization module for

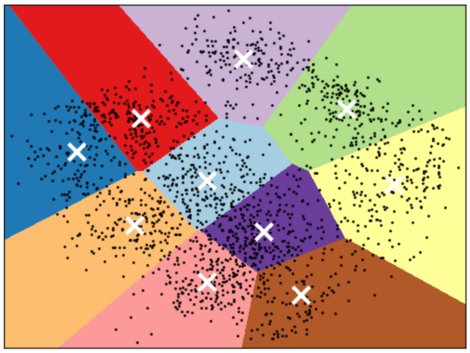

Figure 2.15: The example of K-means clustering algorithm, centroids are marked with white cross, from [38]

wav2vec 2.0, AUD moduel for HuBert) to transform the continuous hidden representations to a fixed set of different discrete hidden representations. For example, in Figure 2.15, the K-means algorithm used in the AUD module of HuBert maps continuous hidden representations (black dots) to their nearest centroids (white crosses). Previous study [39] showed that this implementation could lead to good results for generating context representations . By utilizing these context representations in ASR systems, performances could be improved [39].

Moreover, compared with wav2vec 2.0, HuBert also takes unmasked frames into account for computing contrastive preditive loss. Thus, HuBert are more likely to learn both acoustic and language models by intuition [8].

SSL SPEECH PRE-TRAINED MODELS AND MFCC

As presented above, SSL speech pre-trained models and MFCC introduced in Section 2.1.1 have the following differences for feature extraction:

- **Data driven or human inspired**: Before extracting speech representations from input speech audio data, SSL speech pre-trained models are trained with huge amount of speech audio data without annotations, and learn how to extract speech representations from these unlabeled speech audio data. On the contrary, MFCC are inspired by human ears, the design of MFCC aims to mimic the structure of human ears as a series of filters.

- **Dependencies**: Speech representations generated by SSL speech pre-trained models depend on speech representations of history frames, or even speech representations from future frames. While, acoustic feature vectors generated by MFCC are independent of acoustic feature vectors of other frames.

2.4. ARTICULATORY FEATURES

Phones are speech sounds which could represent the basic sound unit in a pronunciation of a word [12]. For example, the word "tea" is composed of two phones, [t] and [iy]. When producing different phones, organs in our mouth, throat, and nose modify the airflow from the lungs differently. **Articulatory features** describe how different vocal organs are involved in for producing different phones. Seven articulatory features (AFs) and their quantized classes are shown in Table 2.2.

Table 2.2: Articulatory features and their quantized classes [40]

AF	Values
'manner'	approximate, retroflex, fricative, nasal, stop, vowel, nil
'place'	bilabial, labiodental, dental, alveolar, velar, nil
'voice'	+voice, -voice
'high-low'	high, mid, low, nil
'fr-back'	front, central, back, nil
'round'	+round, -round, nil
'static'	static, dynamic

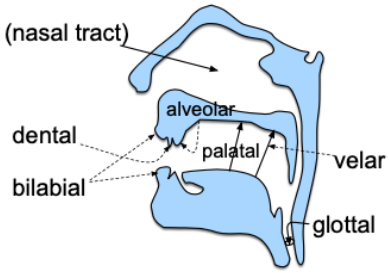

Figure 2.16: Major English places of articulation, from [12]

2.5. SUPPORT VECTOR MACHINE

Support vector machine (SVM) is a supervised machine learning model that solve the problems of classification or regression. In this thesis work, SVM is utilized in the frame-level probing task for classify the right class of an articulatory feature. A brief introduction of SVM is given in the following section.

In order to solve the classification problem, SVMs utilize decision boundaries to separate samples from different classes. The decision boundaries are hyperplanes which separate samples from different classes into different subspaces. For example, Figure 2.17 shows the decision boundary $y = wx+b$ of a binary classification problem. The blue points and the red points belong to two classes. And the decision boundary $y = wx + b$ separates samples from two classes into two subspaces.

The decision boundary of the SVM is obtained by separating samples of different classes from the training dataset. As for the binary classification problem, given a train-

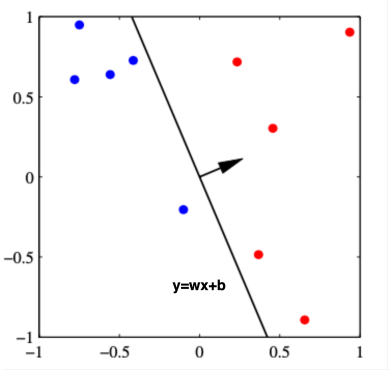

Figure 2.17: The decision boundary of SVM

ing dataset of n points of the form $(x_1, y_1), ..., (x_n, y_n)$, where y_i are either 1 or -1, each indicating the class to which the sample x_i belongs. The decision boundary is the hyperplane which divides these samples into different groups by their classes, and also ensures that the distance between the nearest point x_i from either group and itself is maximized [41]. Thus, the decision boundary $y = wx + b$ is found by solving

$$\arg\max_{w,b} \left\{ \frac{1}{||w||} \min_n [t_n(w^T x_n + b)] \right\} \tag{2.20}$$

where t_n is the ground truth class of the point x_n. After obtaining the decision boundary by the training dataset, the class of a sample from the test dataset could be recognized by its belonging subspace.

2.5.1. LINEAR SVM WITH SOFT-MARGIN

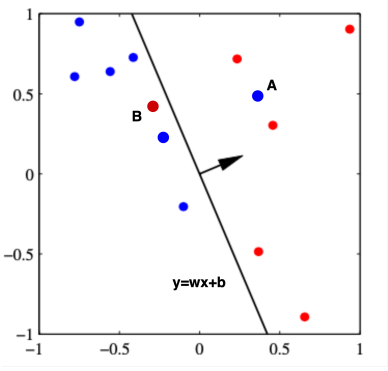

Figure 2.18: The decision boundary which could not separate samples from different classes

In practice, samples from different classes are not always linearly separable. In figure 2.18, the decision boundary could not separate samples from different classes into

different subspaces. For example, the sample A is assigned to the wrong subspace by the decision boundary. Thus, the linear SVM with soft-margin aims to solve this problem. The decision boundary is found by minimizing

$$\lambda||w||^2 + \left[\frac{1}{n}\sum_{i}^{n}\max(0, 1 - y_i(w^T x_i + b))\right] \tag{2.21}$$

where $\max(0, 1 - y_i(w^T x_i - b)$ is the hinge loss function, y_i is the ground truth class of sample x_i, and $w^T x_i + b$ is the output of sample x_i. The hinge loss function equals zero when the decision boundary could determine the right class of the sample x_i, in other words, x_i lies on the correct side of the decision boundary. While, for the sample on the wrong side of the decision boundary, the value of the hinge loss function is the distance between the sample and the decision boundary. The parameter $\lambda > 0$ determines the trade-off between the margin size and ensures that samples lie on the correct side of the decision boundary. The margin size is the distance between the nearest point x_i from either group and decision boundary.

2.5.2. MULTICLASS SVM

As mentioned above, the binary classification solved by SVM is explained. In this thesis work, the SVM classifier is aim to classify different class from an articulatory feature. The articulatory feature could have two classes and more. For example, the articulatory feature 'fr-back' has four classes. Thus, the SVM classifier for multiclass classification problem is required.

The one-versus-the-rest multiclass strategy [41] is one of the methods for solving multiclass classification problem. In this strategy, K separate SVMs are constructed for solving the multiclass classification problem for K classes. The K_{th} SVM is trained using the samples from class C_k as the positive examples and the samples from the remaining $K - 1$ as the negative examples.

2.6. RELATED WORKS

In this section, related works of comparing and analyzing different SSL speech pre-trained models are given. Besides, related works how to analyzing phonetics information in speech representations are also briefly introduced.

With the development of SSL speech pre-trained models, many works have been proposed to compare and analyze different SSL speech pre-trained models. For example, Yang et al. proposed a benchmark for comparing different SSL speech pre-trained models [10]. They aimed to explore whether speech representations extracted by SSL speech pre-trained models could be adopted in different speech downstream tasks and lead to good performance. These speech downstream downstream tasks were not only limited to tasks of automatic speech recognition, but also covered tasks from the following four aspects, content, speaker, semantics and paralinguistics. They adopted a series of light weighted linear classifiers for these downstream tasks. A classifier was applied on top of the speech representations extracted from a SSL speech pre-trained model for its related downstream task. Afterwards, performances of the downstream task which utilized different SSL speech pre-trained models were evaluated and compared. They also found that SSL speech pre-trained models showed great results on different downstream tasks compared with standard representations, such as MFCC. Riviere et analyzed and compared the phoneme discriminability of CPC and its variety [19]. Their SSL speech pre-trained model were trained on English data. They applied their SSL speech pre-trained model to discriminate phonemes in another language other than English. The phoneme discriminability was evaluated by ABX score. ABX score measures the discriminability between phonemes by probability of speech segments, and lower ABX score indicates a high discriminability. Their results showed that their modified CPC had higher discriminability.

Moreover, there also have been some works which analyzed phonetics information in speech representations. For example, Scharenborg et al. adopted a series of automatic feature classifiers base on support vector machines to analyze how articulatory feature information encode in MFCC acoustic feature vectors [40]. Ma et al. analyzed and compared phonetic properties of speech representations extracted by different SSL speech pre-trained models [16]. In their work, different classifiers based on different methods for analyze phonetic properties were compared. These methods included support vector machines, linear regression models and neural networks models.

3

METHODOLOGY

In this chapter, the methods and implementations for answering research questions are explained. In Section 3.1, a brief overview of methods is presented. In Section 3.2, datasets used in this thesis work including LibriSpeech, TIMIT, and Mboshi are introduced. In Section 3.3, frame-level probing tasks for answering **RQ1.1** and **RQ2.1** are explained. In Section 3.4, phoneme recognition tasks for answering **RQ1.2** and **RQ2.2** are explained. In Section 3.6, the evaluation metrics of previous tasks are introduced and explained.

3.1. AN OVERVIEW OF METHODS IMPLEMENTED IN THIS WORK

Figure 3.1 gives an overview of our experimental set-up[1]. In step 1, the three SSL speech pre-trained models studied in this work, CPC, wav2vec2.0 and HuBert, are firstly trained on a large amount of unlabeled English speech audio data. In step 2, these SSL speech pre-trained models are used to extract speech representations for English or another language, Mboshi. Afterwards, frame-level probing tasks based on SVMs investigate what articulatory feature information is captured in these speech representations. In step 3, speech representations extracted by these SSL speech pre-trained models are used as input for an English phone-level ASR system and an Mboshi phone-level ASR system for phoneme recognition tasks. For the baseline, the same tasks in Step 2 and Step 3 also utilize MFCC acoustic feature vectors as input. Finally, the Pearson's correlation coefficients between the results of frame-level probing tasks in Step 2 and the results of phone recognition tasks in Step 3 of each language are computed respectively. The Pearson's correlation coefficient could investigate whether the amount of articulatory feature information is strongly correlated with the performance of phoneme recognition tasks.

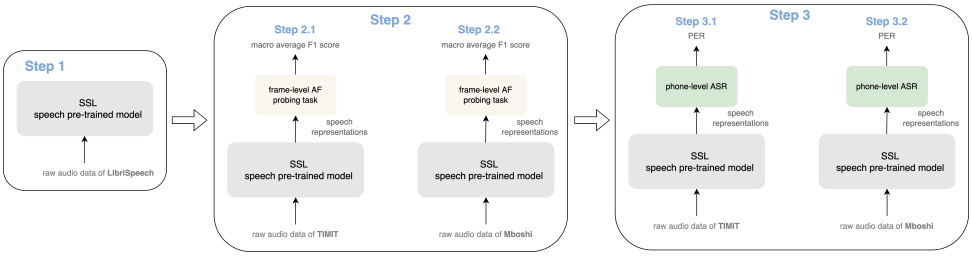

Figure 3.1: Overview of the experimental set-up of this work.

3.2. DATASETS

In this section, all datasets used in this thesis work are introduced. Firstly, LibriSpeech [42] is used to train all SSL speech pre-trained models. Secondly, TIMIT [43] are used to answer **RQ1**, and Mboshi [44] are used to answer **RQ2**.

3.2.1. LIBRISPEECH

LibriSpeech is a read English speech corpus derived from audiobooks [42]. This thesis work only utilizes the training subsets in LibriSpeech to train SSL speech pre-trained models. Table 3.1 presents these subsets used in this work.

In Table 3.1, the three training subsets with approximate size of 960 hours are used to train SSL speech pre-trained models. Moreover, the subsets of *train-clean-100* and *train-clean-360* are collected from lower-WER speakers, the subset of *train-other-500* are collected from higher-WER speakers. There are 2338 speakers involved in these subsets. For each speaker in these subsets the amount of speech was limited to 25 minutes or 30 minutes, it aims to avoid imbalances in per-speaker audio duration [42].

[1] Implementation: https://github.com/KarenMars/IS22Code

Table 3.1: Subsets of LibriSpeech used in this thesis work

subsets	hours	per-spk minutes	female spkrs	male spkrs	total spkrs
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-other-500	496.7	30	564	602	1166

3.2.2. TIMIT

TIMIT is used in the within-language scenario. By using TIMIT, we aim to investigate what articulatory feature (AF) information is captured by SSL speech pre-trained models in the within-language scenario. We also aim to reveal the relationship between the amount of AF information and the performance of the phoneme recognition task. TIMIT is a read English speech corpus which contains a total of 6300 utterances, 10 sentences spoken by each of 630 speakers [43]. The approximate size of TIMIT is 5.4 hours. In addition, TIMIT contains the time-aligned phonetic transcription for each utterance, and it uses a set of 64 phone symbols for transcription. Table 3.2 presents part of the time-aligned phonetic transcription for the utterance "Don't ask me to carry an oily rag like that." For example, the phone symbol "d" starts at 0.2260s and ends at 0.2730s. In general, the set of 64 phone symbols is usually mapped to a set of 39 phonemes in research [45]. Thus, this thesis work also use the mapped set of 39 phone symbols.

Table 3.2: part the phonetic transcription for the utterance "Don't ask me to carry an oily rag like that."

start	end	phone symbol
0	2260	h#
2260	2730	d
2730	4120	uh
4120	4600	n

Moreover, TIMIT [43] contains three kinds of sentences, the dialect sentences (1260 sentences), the phonemically-compact sentences (3150 sentences), and the phonemically-diverse sentences (1890 sentences). Phones in the dialect sentences may have pronunciations that differ from the standard pronunciation, which would result in a difference in articulatory features. Thus, these dialect sentences are not used in this thesis work. The train-test split of follows the suggested split provided by TIMIT[2], with removing dialect sentences. Table 3.3 presents the detailed information for the training dataset and the test dataset. There are 3696 sentences and 462 speakers in the training dataset, and 1344 sentences and 168 speakers in the test dataset.

Table 3.3: The detailed information of the TIMIT training dataset and test dataset used in this work

	#sentences	#male speakers	#female speakers	#total speakers
training dataset	3696	326	136	462
test dataset	1344	112	56	168

[2]The suggested split provided by TIMIT: https://catalog.ldc.upenn.edu/docs/LDC93S1/TESTSET.TXT

3.2.3. MBOSHI DATASET

Mboshi dataset [44] is used in the cross-language scenario. By using the Mboshi dataset, we aim to investigate what AF information is captured by SSL speech pre-trained models in the cross-language scenario. We also aim to reveal the relationship between the amount of AF information and the performance of the phoneme recognition task. In addition, the transferability of SSL speech pre-trained models could be investigated by using Mboshi. Transferability means that SSL speech pre-trained models trained on one language could also have good performances on frame-level probing tasks and phoneme recognition tasks of other languages. Mboshi is a Bantu language of Congo Brazzaville, Africa [44]. Mboshi database is a read Mboshi speech corpus which contains a total of 5130 sentences, and read by 3 speakers. The approximate size of Mboshi database is 4.9 hours. In addition, this thesis work utilizes the time-aligned phonetic transcriptions[3] of Mboshi from the work by Ondel et al [46]. The number of phonemes in Mboshi is 68. Phonemes in Mboshi and phonemes in English have differences in articulatory features, Mboshi is suitable for the cross-language scenario, in order to reveal the transferability of SSL speech pre-trained models.

Moreover, Table 3.4 presents the detailed information of Mboshi dataset. It only contains three speakers, 'abiayi', 'kouarata', and 'martial'. 3681 sentences spoken by 'abiayi' are used as training data. 1449 sentences spoken by 'kouarata' and 'martial' are used as test data.

Table 3.4: The detailed information of Mboshi dataset

	abiayi	kouarata	martial
#sentences	3681	1234	215

3.3. FRAME-LEVEL AF PROBING TASKS

A frame-level articulatory feature (AF) probing task for each SSL speech pre-trained models or MFCC consists of seven SVM classifiers. There are four frame-level AF probing tasks for different feature extraction methods in the within-language scenario or the cross-language scenario. These feature extraction methods include three SSL speech pre-trained models compared in this work and the baseline MFCC. The seven SVM classifiers correspond to the seven articulatory features, which are 'voice', 'place', 'manner', 'high-low', 'fr-back', 'round', and 'static' mentioned in Section 2.4. The input of the SVM classifier is the speech representation or the MFCC acoustic feature vector of a frame. Each SVM classifier aims to classify the class of the articulatory feature from the input. For example, the SVM classifier of 'voice' determine whether an input frame belongs to class '+voice' or class '-voice'. In the frame-level AF probing task of an SSL speech pre-trained model, if a SVM classifier could distinguish classes of an articulatory feature more correctly, the SSL speech pre-trained model is able to capture more AF information of the articulatory feature than other SSL speech pre-trained models.

Specifically, the SVM classifier of articulatory features with more than two classes is a multi-class SVM classifier. The SVM classifier of the articulatory feature 'voice' which

[3] https://github.com/beer-asr/beer/blob/master/recipes/aud/local/mboshi/mboshi.ali

has only two classes is a binary SVM classifier. In addition, the one-versus-the-rest multi-classification strategy [41] explained in Section 2.5.2 is adopted in the multi-class SVM classifiers. Moreover, the linear SVMs with soft-margin are adopted in these SVM classifiers. These SVM classifiers are not required to have high non-linear separability. The classification accuracy of a SVM classifier is determined by the non-linear separability of the SVM classifier itself and the AF information captured by SSL speech pre-trained models or the baseline MFCC. If these SVM classifiers have high non-linear separability, the AF information would have a minor influence on the classification accuracy. Thus, the different between AF information captured by different methods would be difficult to observed. Therefore, these SVM classifiers do not utilize non-linear kernels.

3.4. PHONEME RECOGNITION TASKS

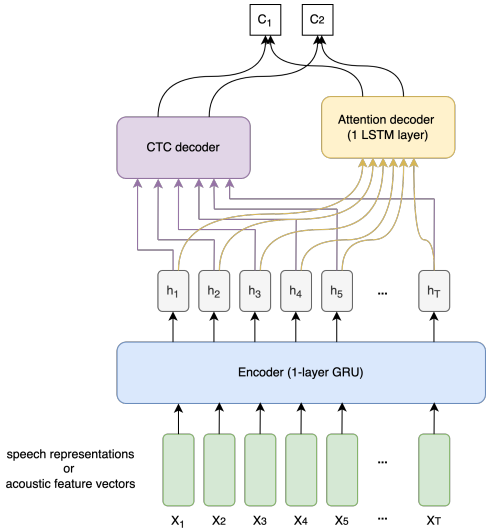

Figure 3.2: Phone-level hybrid CTC/attention end-to-end ASR system implemented in this work

The phoneme recognition task aims to reveal the performances of phone-level ASR systems which utilize speech representations extracted by SSL speech pre-trained models as the systems' input. The performances of phone-level ASR systems could reveal the relationship between the amount of AF information and the performance of phoneme recognition tasks. Like frame-level AF probing tasks, four phoneme recognition tasks are carried out for different feature extraction methods in each scenario. The phoneme recognition task utilizes the phone-level hybrid CTC/attention end-to-end ASR system given in Section 2.1.4. The phone-level hybrid CTC/attention end-to-end ASR system is implemented by toolkit *ESPnet*[4] [47]. To be detailed, it has an encoder of one GRU layer and the hybrid CTC/attention decoder with one LSTM layer shown in Figure 3.2.

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

3.5. IMPLEMENTATIONS

3.5.1. SSL SPEECH PRE-TRAINED MODELS

Before the frame-level probing tasks and the phoneme recognition tasks in the within-language scenario and the cross-language scenario, three SSL speech pre-trained models, which include CPC, wav2vec 2.0 and HuBert, are firstly trained on an English dataset. CPC is trained on 960 hours LibriSpeech by ourselves in this thesis work. Due to the limitation of computing resources, the checkpoints of wav2vec 2.0[5] and HuBert[6] trained on 960 hours LibriSpeech are used in this work.

To be detailed, the setup of CPC follows the implementation[7] in [19]. The encoder of CPC is a 5-layer convolutional network with kernel sizes (10,8,4,4,4) and stride sizes (5,4,2,2,2). The context module of CPC is a 1-layer GRU. It is trained on the 960 hours LibriSpeech for 15 epochs by the Adam [48] optimizer, with an initial learning rate of 0.0002 and a batch size of 8.

Moreover, setups of wav2vec 2.0 and HuBert follow implementations in [5] and [8]. The encoder of wav2vec 2.0 contains 7 blocks of convolutional networks. The convolutional network has 512 channels with kernel sizes (10,3,3,3,3,2,2) and stride sizes (5,2,2, 2,2,2,2). The context module of wav2vec 2.0 contains 12 transformer blocks with a model dimension of 768, inner dimension of 3072 and 8 attention heads. For HuBert, the settings of its encoder and context module are the same as those for wav2vec 2.0.

3.5.2. IMPLEMENTATIONS IN WITHIN-LANGUAGE SCENARIO

Within-language scenario means that SSL speech pre-trained models trained on an English speech corpus are used as the feature extractor for English speech audio data. The following section presents the implementations of frame-level probing tasks and phoneme recognition tasks in the within-language scenario.

STEP 1: MAPPING ARTICULATORY FEATURES

The first step is to map phone symbols to articulatory features for the training dataset and test dataset of TIMIT. As mentioned in Section 3.2.2, TIMIT has the time-aligned phonetic transcription for each utterance. Thus, the time-aligned articulatory features' transcription could be obtained by mapping phone symbols to their articulatory features[8]. Table 3.5 presents mappings between phone symbols and their articulatory features in TIMIT. For example, /*aa*/ is mapped to the articulatory features of *{manner:vowel, place:nil, voice:voiced, high-low:low, fr-back:back, round: round, static:static}*.

[5] wav2vec 2.0 checkpoint: https://huggingface.co/facebook/wav2vec2-base
[6] HuBert checkpoint: https://huggingface.co/facebook/hubert-base-ls960
[7] CPC implementation: https://github.com/tuanh208/CPC_audio
[8] Thank Odette for providing her expertise on mapping phoneme to their related articulatory features.

Table 3.5: mappings between articulatory features and phonemes in TIMIT

phoneme	'manner'	'place'	'voice'	'high-low'	'fr-brack'	'round'	'static'
aa	vowel	nil	voiced	low	back	round	static
ae	vowel	nil	voiced	low	front	unround	static
ah	vowel	nil	voiced	mid	central	unround	static
ao	vowel	nil	voiced	low	back	round	static
aw	vowel	nil	voiced	low	front	unround	dynamic
ax	vowel	nil	voiced	mid	central	unround	static
ax-h	vowel	nil	voiced	low	back	unround	static
axr	retroflex	alveolar	voiced	nil	nil	nil	dynamic
ay	vowel	nil	voiced	low	front	unround	dynamic
eh	vowel	nil	voiced	mid	front	unround	static
el	approximant	alveolar	voiced	nil	nil	nil	dynamic
em	nasal	bilabial	voiced	nil	nil	nil	dynamic
en	nasal	alveolar	voiced	nil	nil	nil	dynamic
eng	nasal	velar	voiced	nil	nil	nil	dynamic
er	retroflex	nil	voiced	nil	nil	nil	dynamic
ey	vowel	nil	voiced	mid	front	unround	dynamic
ih	vowel	nil	voiced	high	front	unround	static
ix	vowel	nil	voiced	high	front	unround	static
iy	vowel	nil	voiced	high	front	unround	dynamic
ow	vowel	nil	voiced	mid	back	unround	dynamic
oy	vowel	nil	voiced	low	back	round	dynamic
uh	vowel	nil	voiced	high	back	round	static
uw	vowel	nil	voiced	high	back	round	dynamic
ux	vowel	nil	voiced	high	back	round	dynamic
p	stop	bilabial	voiceless	nil	nil	nil	dynamic
t	stop	alveolar	voiceless	nil	nil	nil	dynamic
k	stop	velar	voiceless	nil	nil	nil	dynamic
b	stop	bilabial	voiced	nil	nil	nil	dynamic
d	stop	alveolar	voiced	nil	nil	nil	dynamic
g	stop	velar	voiced	nil	nil	nil	dynamic
pcl	closure	bilabial	voiceless	nil	nil	nil	dynamic
tcl	closure	alveolar	voiceless	nil	nil	nil	dynamic
kcl	closure	velar	voiceless	nil	nil	nil	dynamic
bcl	closure	bilabial	voiced	nil	nil	nil	dynamic
dcl	closure	alveolar	voiced	nil	nil	nil	dynamic
gcl	closure	velar	voiced	nil	nil	nil	dynamic
ch	fricative	alveolar	voiceless	nil	nil	nil	dynamic
th	fricative	dental	voiceless	nil	nil	nil	static
f	fricative	labiodental	voiceless	nil	nil	nil	static
s	fricative	alveolar	voiceless	nil	nil	nil	static
sh	fricative	alveolar	voiceless	nil	nil	nil	static
jh	fricative	alveolar	voiced	nil	nil	nil	dynamic
dh	fricative	dental	voiced	nil	nil	nil	dynamic
v	fricative	labiodental	voiced	nil	nil	nil	static
z	fricative	alveolar	voiced	nil	nil	nil	static
zh	fricative	alveolar	voiced	nil	nil	nil	static
hh	fricative	velar	voiceless	nil	nil	nil	static
w	approximant	velar	voiced	nil	nil	nil	dynamic
y	approximant	velar	voiced	nil	nil	nil	dynamic
l	approximant	alveolar	voiced	nil	nil	nil	dynamic
r	retroflex	alveolar	voiced	nil	nil	nil	dynamic
m	nasal	bilabial	voiced	nil	nil	nil	static
n	nasal	alveolar	voiced	nil	nil	nil	static
ng	nasal	velar	voiced	nil	nil	nil	static
nx	nasal	alveolar	voiced	nil	nil	nil	static
dx	stop	alveolar	voiced	nil	nil	nil	dynamic
hv	fricative	velar	voiceless	nil	nil	nil	static
q	stop	alveolar	voiceless	nil	nil	nil	dynamic
sil	silence	silence	slience	silence	silence	silence	silence

STEP 2: FEATURE EXTRACTION

The second step is to extract frame-level speech representations by SSL speech pre-trained models from TIMIT. Frame-level MFCC acoustic feature vectors are extracted as baseline. To be detailed, frame-level MFCC acoustic vectors are generated by the toolkit *librosa*[9], with the context size of 5 windows, window size of 25ms, step size of 10ms. Frame-level speech representations are generated by SSL speech pre-trained models given in Section 3.5.1.

STEP 3: FRAME-LEVEL AF PROBING TASKS

The third step is to carry out frame-level AF probing tasks on different frame-level speech representations and the baseline MFCC acoustic feature vectors. As we know, the ground truth of time-aligned articulatory features' transcription is created in **Step 1**. The frame-level AF probing tasks are carried out on each type of speech representations or MFCC acoustic feature vectors respectively. For one frame-level AF probing task, firstly, the SVM classifiers of different articulatory features are trained on speech representations (or MFCC acoustic feature vectors) of the TIMIT training dataset with the ground truth. These SVM classifiers are implemented by *sklearn's SGDClassifier* [38]. After training, these SVM classifers are applied on speech representations (or MFCC acoustic feature vectors) of the TIMIT test dataset, and classify classes of articulatory features from an input frame's speech representation. Thirdly, the classification results are compared with the ground truth articulatory features' transcriptions. Finally, the results of different probing tasks including CPC,wav2vec 2.0, HuBert and MFCC are compared.

STEP 4: PHONEME RECOGNITION TASKS

The phone-level ASR systems use speech representations extracted by different SSL speech pre-trained models or MFCC acoustic feature vectors as input respectively. These speech representations or MFCC acoustic feature vectors are generated in **Step 2**. Each phone-level ASR system is trained with the input of speech representations or acoustic feature vectors and the target of phoneme transcriptions. In addition, each phone-level ASR system is trained on the TIMIT training dataset and tested on the TIMIT test dataset. Finally, the performances of the phoneme recognition tasks are compared.

3.5.3. IMPLEMENTATIONS IN CROSS-LANGUAGE SCENARIO

Cross-language scenario means that SSL speech pre-trained models trained on an English speech corpus are used as the feature extractor for Mboshi speech audio data. The following section presents the implementations of frame-level probing tasks and phoneme recognition tasks in the cross-language scenario.

STEP 1: MAPPING ARTICULATORY FEATURES

The same as **Step 1** in the within-language scenario, phone symbols are mapped to articulatory features for the training dataset and the test dataset of Mboshi. Table 3.6 presents mappings between phone symbols and their articulatory features. For example, *a* is mapped to the articulatory features of *manner:vowel, place:nil, voice:voiced, high-low:low, fr-back:front, round: unround, static:static.*

[9] https://librosa.org/doc/latest/index.html

Table 3.6: mappings between articulatory features and phonemes in Mboshi

phoneme	'manner'	'place'	'voice'	'high-low'	'fr-back'	'round'	'static'
á	vowel	nil	voiced	low	front	unround	static
a	vowel	nil	voiced	low	front	unround	static
l	approximant	alveolar	voiced	nil	nil	nil	dynamic
i	vowel	nil	voiced	high	front	unround	dynamic
í	vowel	nil	voiced	high	front	unround	dynamic
o	vowel	nil	voiced	mid	back	round	static
m	nasal	bilabial	voiced	nil	nil	nil	static
s	fricative	alveolar	voiceless	nil	nil	nil	static
j	approximant	velar	voiced	nil	nil	nil	dynamic
b	stop	bilabial	voiced	nil	nil	nil	dynamic
k	stop	velar	voiceless	nil	nil	nil	dynamic
^{n}g	stop	velar	voiced	nil	nil	nil	dynamic
ó	vowel	nil	voiced	mid	back	round	static
w	approximant	labiodental	voiced	nil	nil	nil	static
e	vowel	nil	voiced	mid	front	unround	static
é	vowel	nil	voiced	mid	front	unround	static
n	nasal	alveolar	voiced	nil	nil	nil	static
ɔ	vowel	nil	voiced	low	back	round	static
d	stop	alveolar	voiced	nil	nil	nil	dynamic
r	retroflex	alveolar	voiced	nil	nil	nil	dynamic
ú	vowel	nil	voiced	high	back	round	dynamic
áa	vowel	nil	voiced	low	front	unround	static
β	fricative	bilabial	voiced	nil	nil	nil	static
u	vowel	nil	voiced	high	back	round	dynamic
^{m}b	stop	bilabial	voiced	nil	nil	nil	dynamic
^{n}d	stop	alveolar	voiced	nil	nil	nil	dynamic
p	stop	bilabial	voiceless	nil	nil	nil	dynamic
t	stop	alveolar	voiceless	nil	nil	nil	dynamic
ɛ́	vowel	nil	voiced	mid	front	unround	static
ɛ	vowel	nil	voiced	mid	front	unround	static
áá	vowel	nil	voiced	low	front	unround	static
ʤ_1	stop	alveolar	voiced	nil	nil	nil	dynamic
ʤ_2	fricative	alveolar	voiced	nil	nil	nil	dynamic
ɔ́	vowel	nil	voiced	low	back	round	static
b^{v}	stop	bilabial	voiced	nil	nil	nil	dynamic
^{n}ʤ	stop	alveolar	voiced	nil	nil	nil	dynamic
ʦ_1	stop	alveolar	voiceless	nil	nil	nil	dynamic
ʦ_2	fricative	alveolar	voiceless	nil	nil	nil	dynamic
aá	vowel	nil	voiced	low	front	unround	static
^{m}w	approximant	labiodental	voiced	nil	nil	nil	static
aa	vowel	nil	voiced	low	front	unround	static
ɲ	nasal	alveolar	voiced	nil	nil	nil	static
ée	vowel	nil	voiced	mid	front	unround	static
íi	vowel	nil	voiced	high	front	unround	dynamic
óo	vowel	nil	voiced	mid	back	round	static
íí	vowel	nil	voiced	high	front	unround	dynamic
oo	vowel	nil	voiced	mid	back	round	static
éé	vowel	nil	voiced	mid	front	unround	static
f	fricative	labiodental	voiceless	nil	nil	nil	static
^{m}b^{v}	stop	bilabial	voiced	nil	nil	nil	dynamic
p^{f}	stop	bilabial	voiceless	nil	nil	nil	dynamic
ii	vowel	nil	voiced	high	front	unround	dynamic
ɔ́ɔ́	vowel	nil	voiced	low	back	round	static
ɔɔ	vowel	nil	voiced	low	back	round	static
ɔ́ɔ	vowel	nil	voiced	low	back	round	static
ee	vowel	nil	voiced	mid	front	unround	static
ɛ́ɛ́	vowel	nil	voiced	mid	front	unround	static
oó	vowel	nil	voiced	mid	back	round	static
úu	vowel	nil	voiced	high	back	round	dynamic
óó	vowel	nil	voiced	mid	back	round	static
úú	vowel	nil	voiced	high	back	round	dynamic
ɛ́ɛ	vowel	nil	voiced	mid	front	unround	static
uu	vowel	nil	voiced	high	back	round	dynamic
ɔɔ́	vowel	nil	voiced	low	back	round	static
íí	vowel	nil	voiced	high	front	unround	dynamic
ɣ	fricative	velar	voiced	nil	nil	nil	static
ɛɛ	vowel	nil	voiced	mid	front	unround	static
ɛɛ́	vowel	nil	voiced	mid	front	unround	static
eé	vowel	nil	voiced	mid	front	unround	static
uú	vowel	nil	voiced	high	back	round	dynamic

STEP 2: FEATURE EXTRACTION

The same as **Step 2** in the within-language scenario, frame-level speech representations of Mboshi are extracted by three SSL speech pre-trained models, CPC, wav2vec 2.0, and HuBert. In addition, MFCC acoustic features vectors are also extracted as baseline.

STEP 3: FRAME-LEVEL AF PROBING TASKS

The third step is to carry out frame-level AF probing tasks on different frame-level speech representations and the baseline acoustic feature vectors from Mboshi. The ground truth of time-aligned articulatory features' transcription on Mboshi is created in **Step 1** of the cross-language scenario. The frame-level AF probing tasks are carried out on each type of speech representations of MFCC acoustic feature vectors respectively. For one frame-level AF probing task, the SVM classifiers of different articulatory features are trained with the input of speech representations (or MFCC acoustic feature vectors) in the Mboshi training dataset and the target of the their ground truth. The implementation of these SVM classifiers are also implemented by *sklearn's SDGClassifier* [38], which is the same with the implementation in the within-language scenario. After training, these SVM classifiers are applied on speech representations (or MFCC acoustic feature vectors) of the Mboshi test dataset, and determine classes of articulatory features from an input speech representation (or MFCC acoustic feature vector).

PHONEME RECOGNITION TASKS

The phone-level ASR systems use speech representations or MFCC acoustic feature vectors generated in **Step 2** in the cross-language scenario as input respectively. Each phone-level ASR system is trained with the speech representations (or MFCC acoustic feature vectors) and their paired ground truth phoneme transcriptions from the Mboshi training dataset. Afterwards, the phone-level ASR system is tested with the speech representations (or MFCC acoustic feature vectors) from the Mboshi test dataset to predict their phoneme transcriptions.

3.6. EVALUATION METRICS

In this section, the evaluation metrics for frame-level AF probing tasks and phoneme recognition tasks are introduced.

3.6.1. FRAME-LEVEL AF PROBING TASK: MACRO-AVERAGED F1 SCORE

The frame-level AF probing task of a specific articulatory feature is evaluated by Macro-averaged F1 score, which is the average of F1 score of all classes in an articulatory feature. The explanations and computations of F1 score and Macro-averaged F1 score are presented as follows:

F1 SCORE

F1 score is the harmonic mean of of Precision and Recall. The computation of Precision is given as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.1}$$

Table 3.7: Precision and recall confusion matrix

		Predicted condition	
		Positive	Negative
Actual condition	Positive	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

Precision is the value of True positive (TP) divided by the sum of True positive (TP) and False positive (FP) shown in Table 3.7. For a classification task of class x, TP represents the number of samples from class x which are correctly classified into class x. FP represents the number of samples from other classes which are misclassified into class x. For example, for the classification task of class *+voice* from articulatory feature 'voice', TP represents the number of inputs from *+voice* which are correctly classified into *+voice*, FP represents the number of inputs from *-voice* which are misclassified into *+voice*.

The computation of Recall is given as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.2}$$

Recall is the value of TP divided the sum of TP and False negative (FN) shown in Table 3.7. FN represents the number of samples from class x which are misclassified into other classes. For example, for the classification task of class *+voice* from articulatory feature 'voice', FN represents the number of samples from *+voice* which are misclassified into *-voice*.

The computation of F1 score is given as follows:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

F1 score is the harmonic mean of Precision and Recall. Compared with Accuracy, F1 score could demonstrate how well a classifier could recognize a specific class from other classes. The computation of Accuracy is given as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.4}$$

Accuracy is the value of the sum of TP and True negative (TN) divided by the sum of all tested samples. For example, for the accuracy of the articulatory feature 'voice', it is the result of the number of right classified samples divided by the number of all samples. The right classified samples include samples which belong to class *+voice* and samples which belong to class *-voice*. The Accuracy couldn't present how well samples from class *+voice* or *-voice* are correctly classified. While F1 score of class *+voice* or *-voice* could present how well samples from different classes are correctly classified. Thus, F1 score is used to evaluation each class of an articulatory feature.

MACRO-AVERAGED F1 SCORE

Macro-averaged F1 score is the average of F1 score of all classes in an articulatory feature, which is shown as follows:

$$\text{macro-averaged F1 score} = \frac{1}{|C|} \sum_{i \in C} F1_i \tag{3.5}$$

where C is the set of classes of an articulatory feature. For example, the macro-averaged F1 score of the frame-level probing task for the articulatory feature 'voice' is computed as follows:

$$\text{macro-averaged F1 score for 'voice'} = \frac{1}{2}(F1_{+voice} + F1_{-voice}) \tag{3.6}$$

3.6.2. PHONEME RECOGNITION TASK: PHONE ERROR RATE

The phoneme recognition task is evaluated by phone error rate (PER). It is computed as follows:

$$\text{PER} = \frac{S + D + I}{N} \tag{3.7}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the number of phones in the ground truth transcription.

3.6.3. PEARSON'S CORRELATION COEFFICIENT

Pearson's correlation coefficient is used to reveal the relationship between the articulatory feature information and the performance of the phoneme recognition task. The computation of Pearson's correlation coefficient is given as follows:

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{3.8}$$

Pearson's correlation coefficient is a measure of linear correlation between two sets of data X, and Y. It is the covariance of the two variables ($\text{cov}(X, Y)$) divided by the product of their standard deviations ($\sigma_X \sigma_Y$). For example, Pearson's correlation coefficient in the within-language scenario is computed between the set of performances of frame-level probing tasks by different SSL speech pre-trained models and the set of performances of phoneme recognition tasks on TIMIT. Pearson's correlation coefficient in the cross-language scenario is computed between the set of performances of frame-level probing tasks by different SSL speech pre-trained models and the set of performances of phoneme recognition tasks on Mboshi.

4

RESULTS

In this Chapter, Section 4.1 presents the results of **RQ1**, and Section 4.2 presents the results of **RQ2**.

4.1. RQ 1: WITHIN-LANGUAGE SCENARIO

Table 4.1 presents the macro-averaged F1 scores of frame-level AF probing tasks on TIMIT. Meanwhile, it also shows the average and the standard deviation of these macro-averaged F1 scores for each speech pre-trained model. The columns of *#classes* present the number of classes in each articulatory feature. Figure 4.1 presents the relative change of macro-averaged F1 score for each articulatory feature. The relative change shows how each speech pre-trained model improves the performances of frame-level AF probing tasks, and the baseline is MFCC. For example, the relative change of the macro-averaged F1 score by CPC is computed by the difference between the macro-averaged F1 score of CPC and the macro-averaged F1 score of MFCC divided by the macro-averaged F1 score of MFCC.

Table 4.1: Macro-averaged F1 scores of frame-level AF probing tasks carried out on MFCC, CPC, wav2vec 2.0 and HuBert respectively

TIMIT	MFCC	CPC	wav2vec 2.0	HuBert	#classes
voice	0.870	0.866	0.891	0.921	2
static	0.669	0.773	0.786	0.887	2
manner	0.666	0.733	0.782	0.842	7
round	0.661	0.722	0.763	0.866	3
high-low	0.633	0.685	0.747	0.850	4
fr-back	0.581	0.635	0.699	0.789	4
place	0.376	0.621	0.715	0.840	6
Avg	0.637	0.719	0.769	**0.856**	
Std	0.146	0.084	0.063	0.041	

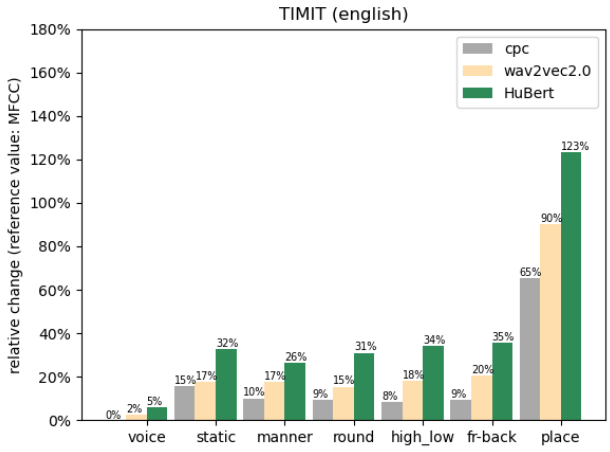

Figure 4.1: Relative change of the results of frame-level AF probing tasks carried out on CPC, wav2vec 2.0 and HuBert, with the reference value of the results of frame-level AF probing tasks on MFCC

From Table 4.1, we have the following findings:

- **For frame-level AF probing tasks carried out on baseline MFCC**, the average for the results of frame-level probing tasks is **0.637**, with the standard deviation of **0.146**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.870**. The frame-level AF probing task of the articulatory feature 'place' has the worst performance, with the result of **0.376**.

- **For frame-level AF probing tasks carried out on CPC**, the average for the results of frame-level probing tasks is **0.719**, with the standard deviation of **0.084**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.866**. The frame-level AF probing task of the articulatory feature 'place' has the worst performance, with the result of **0.621**.

- **For frame-level AF probing tasks carried out on wav2vec 2.0**, the average for the results of frame-level probing tasks is **0.769**, with the standard deviation of **0.063**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.891**. The frame-level AF probing task of the articulatory feature 'fr-back' has the worst performance, with the result of **0.669**. The frame-level AF probing task of the articulatory feature 'place' has the second worst performance, with the result of **0.715**.

- **For frame-level AF probing tasks carried out on HuBert**, the average for the results of frame-level probing tasks is **0.856**, with the standard deviation of **0.041**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.921**. The frame-level AF probing task of the articulatory feature 'fr-back' has the worst performance, the the result of **0.789**. The frame-level AF probing task of the articulatory feature 'place' has the second worst performance, with the result of **0.789**.

- **Compared with the average performance of baseline MFCC**, the performance of CPC increases **12.9%** relatively, that of wav2vec 2.0 increases **20.7%** relatively, and that of HuBert increases **34.4%** relatively.

In Figure 4.1, we have the following finding:

- **For all articulatory features**, the improvement of the articulatory feature 'place' is the largest. While improvement of the articulatory feature 'voice' is the smallest.

Table 4.2: PER on the phoneme recognition tasks on TIMIT

TIMIT	MFCC	CPC	wav2vec 2.0	HuBert
%PER	24.9	22.3	13.4	**10.2**
%substitution	14.8	13.3	7.8	5.5
%deletion	6.3	5.6	3.6	2.6
%insertion	3.8	3.5	2	2

Table 4.2 shows the results of phoneme recognition tasks on TIMIT. The performance of the phoneme recognition task is evaluated by PER. In Table 4.2, we have the following findings:

- The performance of MFCC on phoneme recognition is the worst, with the PER of **24.9%**. The performance of Hubert on phoneme recognition is the best, with the PER of **10.2%**. The performance of wav2vec 2.0 on phoneme recognition is the second-best, with the PER of **13.4%**. The performance of CPC on phoneme recognition is the third-best, with the PER of **22.3%**.
- Compared with the baseline MFCC, the PER of CPC decreases **10.4%** relatively, that of wav2vec 2.0 decreases **46.1%** relatively, and that of HuBert decreases **59.0%** relatively.
- Compared with the %substitution of the baseline MFCC, the %substitution of CPC decreases **10%** relatively, the %substitution of wav2vec 2.0 decreases **47%** relatively, and the %substitution of HuBert decreases **62%** relatively.
- Compared with the %deletion of the baseline MFCC, the %deletion of CPC decreases **11%** relatively, the %deletion of wav2vec 2.0 decreases **42%** relatively, and the %substitution of HuBert decreases **58%** relatively.
- Compared with the %insertion of the baseline MFCC, the %substitution of CPC decreases **7%** relatively, the %insertion of wav2vec 2.0 decreases **47%** relatively, and the %insertion of HuBert decreases **47%** relatively.

Table 4.3: Performances of phoneme recognition tasks and performances of frame-level probing tasks on TIMIT

TIMIT	MFCC	CPC	wav2vec 2.0	HuBert
averaged Macro-averaged F1 score	0.637	0.719	0.769	0.856
Accuracy	0.751	0.777	0.866	0.898
Pearson's correlation coefficient			0.949	

Table 4.3 shows the performances of frame-level AF probing tasks and the performances of phoneme recognition tasks. The performance of frame-level AF probing task is evaluated by the averaged Macro-averaged F1 score, which is the average of macro-averaged F1 scores on different articulatory features. The performance of phoneme recognition tasks is evaluated by the accuracy, which is 1 – PER. The correlation of the performances on the two tasks is evaluated by the Pearson's correlation coefficient. In Table 4.3, we have the following findings:

- The order of the averaged Macro-averaged F1 score of different feature extraction methods is consistent with the accuracy. For example, HuBert has the highest averaged Macro-averaged F1 score and the highest accuracy. MFCC has the lowest averaged Macro-averaged F1 score and the lowest accuracy.
- The Pearson's correlation coefficient between the results of frame-level AF probing tasks and the results of phoneme recognition tasks is **0.949**.

4.2. RQ 2: CROSS-LANGUAGE SCENARIO

Table 4.4 presents the macro-averaged F1 scores of frame-level AF probing tasks on Mboshi. Figure 4.2 presents relative change of macro-averaged F1 score for each articulatory feature. The baseline is MFCC.

Table 4.4: Macro-averaged F1 scores of frame-level AF probing tasks carried out on MFCC, CPC, wav2vec 2.0 and HuBert respectively

Mboshi	MFCC	CPC	wav2vec 2.0	HuBert	#classes
voice	0.736	0.791	0.887	0.923	2
fr-back	0.741	0.761	0.806	0.861	3
round	0.738	0.766	0.806	0.861	3
static	0.732	0.769	0.814	0.858	2
high-low	0.682	0.697	0.741	0.812	4
place	0.496	0.545	0.682	0.786	5
manner	0.466	0.517	0.598	0.713	6
Avg	0.656	0.692	0.762	**0.831**	
Std	0.121	0.114	0.097	0.067	

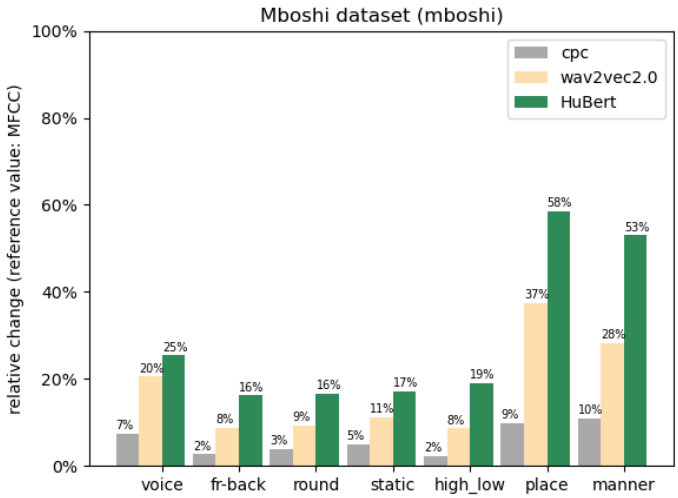

Figure 4.2: Relative change of the results of frame-level AF probing tasks carried out CPC, wav2vec 2.0 and HuBert, with the reference value of the results of frame-level AF probing tasks on MFCC

In Table 4.2, we have the following findings:

- **For frame-level AF probing tasks carried out on baseline MFCC**, the average value for the results of frame-level probing tasks is **0.656**, with the standard deviation of **0.121**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.736**. The frame-level AF probing task of the articulatory feature 'manner' has the worst performance, with the result of **0.466**.

- **For frame-level AF probing tasks carried out on CPC**, the average value for the results of frame-level probing tasks is **0.692**, with the standard deviation of **0.114**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.791**. The frame-level AF probing task of the articulatory feature 'manner' has the worst performance, with the result of **0.517**.

- **For frame-level AF probing tasks carried out on wav2vec 2.0**, the average value for the results of frame-level probing tasks is **0.762**, with the standard deviation of **0.598**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.887**. The frame-level AF probing task of the articulatory feature 'manner' has the worst performance, with the result of **0.598**.

- **For frame-level AF probing tasks carried out on HuBert**, the average value for the results of frame-level probing tasks is **0.831**, with the standard deviation of **0.067**. The frame-level AF probing task of the articulatory feature 'voice' has the best performance, with the result of **0.923**. The frame-level AF probing task of the articulatory feature 'fr-back' has the worst performance, the the result of **0.713**.

- **Compared with the average performance of baseline MFCC**, the performance of CPC increases **5%** relatively, that of wav2vec 2.0 increases **16.1%** relatively, and that of HuBert increases **26.7%** relatively.

- **Comparing Table 4.4 and Table 4.1**, the performances of SSL speech pre-trained models and the baseline MFCC are similar. For example, HuBert performs the best in both scenarios. MFCC performs the worst in both scenario. In addition, the performances in cross-language scenario are degraded a little. Some articulatory features are easy to discriminate in within-language scenario, while they are not easy to discriminate in the cross-language scenario, for example, the articulatory feature 'manner'.

In Figure 4.2, we have the following finding:

- **For all articulatory features**, the performances of frame-level AF probing tasks for the articulatory features 'place' and 'manner' are largely improved. Improvements for other articulatory features, which include 'fr-back', 'round', 'static' and 'high_low', do not have significant differences.

Table 4.5 shows the results of phoneme recognition tasks on Mboshi. The performance of the phoneme recognition is evaluated by PER. In Table 4.5, we have the following findings:

Table 4.5: PER on the phoneme recognition tasks on Mboshi

Mboshi	MFCC	CPC	wav2vec 2.0	HuBert
%PER	56.3	45.9	32.6	**23.0**
%substitution	34.0	33.7	23.3	16.9
%deletion	19.6	9.1	5.0	3.3
%insertion	2.7	3.2	4.3	2.9

- The performance of MFCC on phoneme recognition is the worst, with the PER of **56.3%**. The performance of HuBert on phoneme recognition is the best, with the PER of **23.0%**. The performance of wav2vec 2.0 on phoneme recognition is the second-best, with the PER of **32.6%**. The performance of CPC on phoneme recognition is the third-best, with the PER of **45.9%**.
- Compared with the baseline MFCC, the PER of CPC decreases **18.4%** relatively, that of wav2vec 2.0 decreases **42.1%** relatively, and that of HuBert decreases **59.1%** relatively.
- Compared with the %substitution of the baseline MFCC, the %substitution of CPC decreases **0.01%** relatively, the %substitution of wav2vec 2.0 decreases **31%** relatively, and the %substitution of HuBert decreases **50%** relatively.
- Compared with the %deletion of the baseline MFCC, the %substitution of CPC decreases **53%** relatively, the %substitution of wav2vec 2.0 decreases **74%** relatively, and the %substitution of HuBert decreases **83%** relatively.
- Compared with the %insertion of the baseline MFCC, the %substitution of CPC increases **18%** relatively, the %substitution of wav2vec 2.0 increases **59%** relatively, and the %substitution of HuBert increases **7%** relatively.

Table 4.6: Performances of phoneme recognition tasks and performances of frame-level probing tasks on Mboshi

Mboshi	MFCC	CPC	wav2vec 2.0	HuBert
averaged Macro-averaged F1 score	0.656	0.692	0.762	0.831
Accuracy	0.437	0.541	0.674	0.770
Pearson's correlation coefficient			0.990	

Table 4.6 shows the performances of frame-level probing tasks and the performances of phoneme recognition tasks on Mboshi. In Table 4.6, we have the following findings:

- The order of the averaged Marco-averaged F1 score of different feature extraction methods is consistent with the accuracy.
- The Pearson's correlation coefficient between the results of frame-level AF probing tasks by different speech representations in Table 4.4 and the four phoneme error rates is **0.990**.

5

Discussion

In this chapter, research questions are solved by the results given in the previous chapter. Other findings and explanations of these results are also included. In Section 5.1, answers to **RQ1** are given. Moreover, explanations of results related to **RQ1** are included as well. In Section 5.2, answers to **RQ2** are given. In Section 5.3, the comparison between our results and other works is presented.

5.1. RQ 1: WITHIN-LANGUAGE SCENARIO

We could derive these answers to **RQ1** as shown follows:

- ***RQ1.1*** *What articulatory feature (AF) information is modeled by different SSL speech pre-trained models?*

  Answer:

  Compared with the baseline MFCC, all SSL speech pre-trained models are able to capture more AF information. In capturing the AF information, HuBert ranks first, wav2vec 2.0 ranks second place, and CPC ranks third.

  The order of AF information of different articulatory features captured in CPC is the same as MFCC. For the baseline MFCC and CPC, the AF information of 'voice' is captured in the first place. The AF information of 'static' is captured in the second place. The AF information of 'manner' is captured in the third place. The AF information of 'round' is captured in the fourth place. The AF information of 'high-low' is captured in the fifth place. The AF information of 'fr-back' is captured in the sixth place. The AF information of 'place' is captured in the seventh place. Compared with MFCC, CPC captures more AF information for all articulatory features except 'voice'. The AF information of 'voice' captured by CPC is close to it captured by MFCC with a minor degradation.

  In wav2vec 2.0, the order of AF information of all articulatory features except 'fr-back' and 'place' is the same as MFCC and CPC. The AF information of 'place' is captured in the sixth place. The AF information of 'fr-back' is captured in the seventh place. Compared with MFCC and CPC, the AF information of all articulatory features captured by wav2vec 2.0 is improved.

  In HuBert, the order of AF information of 'voice' and 'static' is the same as MFCC, CPC and wav2vec 2.0. The AF information of 'round' is captured in the third place. The AF information of 'high-low' is captured in the fourth place. The AF information of 'manner' is captured in the fifth place. The AF information of 'place' is captured in the sixth place. The AF information of 'fr-back' is captured in the seventh place. Compared with MFCC, CPC, and wav2vec 2.0, the AF information of all articulatory features captured by HuBert is significantly improved.

- ***RQ1.2*** *How does articulatory feature information modeled by SSL speech pre-trained models correlate to phoneme recognition performance in the same language, i.e., English?*

  Answer:

  The Pearson's correlation coefficient between the results of frame-level probing tasks and phoneme recognition tasks is **0.949**. Frame-level AF probing tasks indicate the amount of AF information modelled in speech representations extracted by different SSL speech pre-trained models. Thus, the amount of AF information is strongly correlated with the performance of the phoneme recognition tasks in the same languages.

As shown above, SSL speech pre-trained models capture more AF information than MFCC. The excellent performance of SSL speech pre-trained models in capturing AF information could be attributed to two reasons. The first reason is that SSL speech pre-trained models are data-driven models. As mentioned in Section 3, these SSL speech pre-trained models are trained on 960 hours LibirSpeech. On the contrary, MFCC is a feature extraction method based on signal processing. The computation and the settings of parameters in MFCC are inspired by humans' ears. Thus, MFCC has limitations in capturing AF information due to the limitation of humans' cognition. While the data-driven SSL speech pre-trained models learn AF information from a vast amount of data. The second reason is that SSL speech pre-trained models are built with deep learning architectures. SSL speech pre-trained models leverage the non-linear separability of deep learning architectures to uncover the AF information.

Compared with CPC, HuBert and wav2vec 2.0 capture more AF information. The good performance of HuBert and wav2vec 2.0 is attributed to the utilization of different architectures for sequence processing. As mentioned in Section 2.3.5, CPC utilizes RNN and its varieties as the architectures for sequence processing. Wav2vec 2.0 and HuBert utilize the Transformer as the architecture for sequence processing. The dependencies of the RNN and the Transformer are different. For CPC, which utilizes the RNN, the captured AF information of the current frame is dependent on this current frame and history frames. For wav2vec 2.0 and HuBert, the captured AF information of the current frame is also dependent on future frames except for this current frame and history frames. Therefore, wav2vec 2.0 and HuBert could capture more AF information with extra knowledge introduced by future frames.

In addition, HuBert performs better than wav2vec 2.0 in capturing AF information. The better performance of HuBert could be attributed to the utilization of the acoustic unit discovery (AUD) module of HuBert. As mentioned in Section 2.3.3 and Section 2.3.4, HuBert adopts the AUD module in its training stage. The AUD module assigns pseudo labels to input frames. Input frames with similar linguistic information are likely to have the same pseudo label. Afterward, the HuBert is trained with the targets of these pseudo labels. Therefore, the AUD module performs as a teacher to HuBert. Thus, Hubert could capture more AF information.

5.2. RQ 2: CROSS-LANGUAGE SCENARIO

From results given in Section 4.2, we could derive these answers to **RQ2** as shown follows:

- ***RQ2.1*** *To what extent is the articulatory feature information from a different language modeled by different SSL speech pre-trained models*

  Answer:

  Compared with the baseline MFCC, all SSL speech pre-trained models are able to capture more AF information. The performances of these SSL speech pre-trained models in the cross-language scenario are similar to the performances in the within-language scenario. Besides, the performances in the cross-language scenario have minor degradation. In capturing the AF information, HuBert ranks first, wav2vec 2.0 ranks second, and CPC ranks third. This order in the cross-language scenario is the same as the within-language scenario.

Compared with MFCC, the AF information of all articulatory features captured by CPC is improved. The order of AF information of articulatory features except 'high-low', 'place', and 'manner' is also changed. In MFCC, the AF information of 'fr-back' ranks first. The AF information of 'round' ranks second. The AF information of 'voice' ranks third. The AF information of 'static' ranks fourth. In CPC, the AF information of 'voice' ranks first. The AF information of 'static' ranks second. The AF information of 'round' ranks third. The AF information of 'fr-back' ranks fourth. Other articulatory features have the same order in both CPC and MFCC. The AF information of 'high-low' ranks fifth. The AF information of 'place' ranks sixth. The AF information of 'manner' ranks seventh.

In wav2vec 2.0, the AF information of all articulatory features is improved compared with the baseline MFCC and CPC. The AF information of 'voice' is captured the most. The AF information of 'manner' is captured the least. The AF information of 'place' is captured the second-least. The AF information of these articulatory features is captured the same as CPC. While the captured AF information of other articulatory features is slightly different from CPC. The AF information of 'fr-back' and 'round' is captured in the third place. The AF information of 'high-low' is captured in the fourth place.

In HuBert, the AF information of all articulatory features is significantly improved compared with the baseline MFCC and other SSL speech pre-trained models. Like CPC and wav2vec 2.0, the AF information of 'voice' is captured the most. The AF information of 'manner' is captured the least. The order of AF information captured by HuBert is almost the same as wav2vec 2.0.

In both scenarios, the AF information of 'voice' is captured the most by different SSL speech pre-trained models. The AF information of other articulatory features is captured differently by different SSL speech pre-trained models.

- ***RQ2.2*** *How does articulatory information modeled by SSL speech pre-trained models correlate to phoneme recognition performance in other languages, i.e., Mboshi?*

  Answer:

  The Pearson's correlation coefficient between the results of frame-level probing tasks and phoneme recognition tasks is **0.990**. It answers that the amount of articulatory information is strongly correlated with the performance on the phoneme recognition tasks even in different languages, i.e., Mboshi.

As shown above, SSL speech pre-trained models could still capture more AF information in the cross-language scenario. As articulatory features are language-independent, SSL speech pre-trained models could perform well in both scenarios. Moreover, the PERs on Mboshi are improved by SSL speech pre-trained models but not as good as the PERs on TIMIT. The performance gap can partially be explained by the fact that phonemes are language-dependent. Another explanation is that Mboshi has more phonemes than TIMIT, which makes the phoneme recognition task more difficult in Mboshi.

5.3. COMPARISONS WITH OTHER WORKS

As given in Section 4.1, the best performance of TIMIT phoneme recognition tasks in this work is achieved by hybrid CTC/attention end-to-end ASR system, which utilizes speech representations extracted by HuBert, with the PER of **10.2%**. Compared with the state of art results of TIMIT shown in Table 5.1, this result performs worse than the result achieved by wav2vec 2.0 in [5], but performs better than the results achieved by wav2vec in [49] and vq-wav2vec in [39]. Specifically, the SSL speech pre-trained model wav2vec 2.0 in [5] was firstly trained on 960 hours LibriSpeech and then fine-tuned on 10 hours subset of Libri-light. The SSL speech pre-trained model wav2vec in [49] was firstly trained on LirbiSpeech and WSJ (another English dataset). While, HuBert in this work is only trained on 960 hours LibriSpeech, and achieves a relative low PER. The result indicates that the implementation of the TIMIT phoneme recognition task in this thesis work has the potential to achieve lower PER.

Table 5.1: TIMIT phoneme recognition accuracy in terms of phoneme error rate (PER)

	trainning data	fine-tuning data	PER
wav2vec [49]	LibriSpeech + WSJ	No	14.7
vq-wav2vec [39]	960 hours LibriSpeech	No	11.6
wav2vec 2.0 [5]	960 hours LibriSpeech	10 hours subset of Libri-light	**8.3**
This work HuBert (base) with a phone-level ASR system	960 hours LibriSpeech	No	10.2

6

CONCLUSIONS AND FUTURE WORKS

6.1. CONCLUSIONS

In this thesis work, we could draw the following conclusions. First, the work investigates what articulatory feature information is captured by different SSL speech pre-trained models. These SSL speech pre-trained models include CPC, wav2vec 2.0 and HuBert. The results of frame-level AF probing tasks show that all SSL speech pre-trained models capture more articulatory feature information than the baseline MFCC. Hubert ranks first in capturing the articulatory feature information, wav2vec 2.0 ranks second, and CPC ranks third. The work also investigates whether the above-mentioned articulatory feature information could influence the performance of the phoneme recognition tasks, which adopt these SSL speech pre-trained models as feature extraction methods. This thesis work shows that the performance of the phoneme recognition task is strongly correlated with the amount of articulatory feature information captured by the SSL speech pre-trained models. A more significant amount of articulatory feature information could achieve better performance on the phoneme recognition task. These conclusions are drawn in both scenarios, which are the within-language scenario and the cross-language scenario. In addition, SSL speech pre-trained models which are trained on an English dataset could also perform better than the baseline MFCC in capturing articulatory feature information and phoneme recognition of an African language. This finding shows that SSL speech pre-trained models could transfer to other languages in capturing articulatory feature information and phoneme recognition.

6.2. FUTURE WORKS

The future works of this thesis could follow the directions given below. In this thesis work, the context representations of SSL speech pre-trained models are used as the speech representations in the frame-level AF probing tasks and phoneme recognition tasks. The future work could adopt hidden representations of SSL speech pre-trained models as the speech representations, and investigate their performance on frame-level probing tasks and phoneme recognition tasks.

Besides, this thesis work only carried out experiments on SSL speech pre-trained models without fine-tuning. Fine-tuning is to make small adjustments of SSL speech pre-trained models for better performance of desired tasks. A general fine-tuning process is: Firstly, an SSL speech pre-trained model is trained with a large amount of unlabeled data. Secondly, this SSL speech pre-trained model is tuned with a small amount of labeled data for a desired task. For example, for English word recognition, the SSL speech pre-traind model is tuned with a small amount of English speech audio data and paired transcriptions. Other works have shown that fine-tuning could significantly improve the performance of desired tasks [5, 13]. Therefore, how the fine-tuning technique would change the articulatory feature information modeled by different SSL speech pre-trained models could be researched in the future.

Although SSL speech pre-trained models perform well in modeling articulatory feature information and phoneme recognition tasks, they have a huge amount of parameters. For example, Hubert studied in this thesis work contains 95 million parameters. Compared with MFCC, these SSL speech pre-trained models have very high space complexity. Thus, how to reduce the number of parameters and retain the performance of

the SSL speech pre-trained models could be researched in the future.

BIBLIOGRAPHY

[1] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[2] Ewan Dunbar, Xuan Nga Cao, Juan Benjumea, Julien Karadayi, Mathieu Bernard, Laurent Besacier, Xavier Anguera, and Emmanuel Dupoux. The zero resource speech challenge 2017. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 323–330. IEEE, 2017.

[3] S Shahnawazuddin, Nagaraj Adiga, Kunal Kumar, Aayushi Poddar, and Waquar Ahmad. Voice conversion based data augmentation to improve children's speech recognition in limited data scenario. In *Interspeech*, pages 4382–4386, 2020.

[4] Qiang Gao, Haiwei Wu, Yanqing Sun, and Yitao Duan. An end-to-end speech accent recognition method based on hybrid ctc/attention transformer asr. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7253–7257. IEEE, 2021.

[5] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[6] Sining Sun, Ching-Feng Yeh, Mei-Yuh Hwang, Mari Ostendorf, and Lei Xie. Domain adversarial training for accented speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4854–4858. IEEE, 2018.

[7] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[8] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, et al. Superb: Speech processing universal performance benchmark. *arXiv preprint arXiv:2105.01051*, 2021.

[11] Siyuan Feng and Odette Scharenborg. The effectiveness of self-supervised representation learning in zero-resource subword modeling. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pages 1414–1418. IEEE, 2021.

[12] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[13] Jakob Poncelet et al. Comparison of self-supervised speech pre-training methods on flemish dutch. *arXiv preprint arXiv:2109.14357*, 2021.

[14] Jialu Li, Vimal Manohar, Pooja Chitkara, Andros Tjandra, Michael Picheny, Frank Zhang, Xiaohui Zhang, and Yatharth Saraf. Accent-robust automatic speech recognition using supervised and unsupervised wav2vec embeddings. *arXiv preprint arXiv:2110.03520*, 2021.

[15] Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, et al. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint arXiv:2111.09296*, 2021.

[16] Danni Ma, Neville Ryant, and Mark Liberman. Probing acoustic representations for phonetic properties. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 311–315. IEEE, 2021.

[17] Tu Anh Nguyen, Maureen de Seyssel, Patricia Rozé, Morgane Rivière, Evgeny Kharitonov, Alexei Baevski, Ewan Dunbar, and Emmanuel Dupoux. The zero resource speech benchmark 2021: Metrics and baselines for unsupervised spoken language modeling. *arXiv preprint arXiv:2011.11588*, 2020.

[18] Ankita Pasad, Ju-Chieh Chou, and Karen Livescu. Layer-wise analysis of a self-supervised speech representation model. *arXiv preprint arXiv:2107.04734*, 2021.

[19] Morgane Riviere, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. Unsupervised pretraining transfers well across languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7414–7418. IEEE, 2020.

[20] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James R. Glass. An unsupervised autoregressive model for speech representation learning. *CoRR*, abs/1904.03240, 2019.

[21] Steve Renals. Decoding, alignment, and wfsts, Feb 2019.

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

[23] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[24] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

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

[26] Awni Hannun. Sequence modeling with ctc. *Distill*, 2017. https://distill.pub/2017/ctc.

[27] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[28] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[29] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

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

[31] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[34] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass. An unsupervised autoregressive model for speech representation learning. *arXiv preprint arXiv:1904.03240*, 2019.

[35] Liu Andy T. Self-supervised learning, June 2020.

[36] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[37] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[39] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*, 2019.

[40] Odette Scharenborg, Vincent Wan, and Roger K Moore. Towards capturing fine phonetic variation in speech using articulatory features. *Speech Communication*, 49(10-11):811–826, 2007.

[41] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[42] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[43] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93:27403, 1993.

[44] Pierre Godard, Gilles Adda, Martine Adda-Decker, Juan Benjumea, Laurent Besacier, Jamison Cooper-Leavitt, Guy-No"el Kouarata, Lori Lamel, H'el'ene Maynard, Markus M"uller, Annie Rialland, Sebastian St"uker, François Yvon, and Marcely Zanon Boito. A very low resource language speech corpus for computational language documentation experiments. *CoRR*, abs/1710.03501, 2017.

[45] K-F Lee and H-W Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

[46] Lucas Ondel, Bolaji Yusuf, Lukas Burget, and Murat Saraçlar. Non-parametric bayesian subspace models for acoustic unit discovery. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.

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

[48] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[49] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.