Universiteit Leiden

Master Computer Science

Automatic speech recognition for the low-resource language Frisian using Conformer and pre-trained Whisper

Name:	Jialiang Wang
Student ID:	2829746
Date:	[30/05/2023]
Specialisation:	Data Science
1st supervisor:	Dr. E.M. Bakker
2nd supervisor:	Prof.dr. M.S.K. Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 12
2333 CA Leiden

Contents

Abstract

Most modern speech recognition systems are developed for high-resource languages. Low-resource language speech recognition refers to the challenge of developing speech recognition systems for languages that have limited resources, including training data, development tools, and computing resources. Low-resource language speech recognition remains a challenging task, and more research is needed to develop effective solutions for languages with limited resources. The goal of this thesis is to study how languages from the same language family can be used for improving low-resource language automatic speech recognition. This thesis delves into the field of low-resource language speech recognition, and we present three experiments aimed at studying the existing limitations in this area. In the first experiment, we validate the performance of Whisper. In the second experiment, we fine-tune a Whisper model for the Frisian language, using different tokenizers from the same language family. During the last experiment, we build and evaluate a single-language speech recognition model for the Frisian language.
Our proposed approaches involve utilizing both a large pre-trained multi-language model, and a single-language model pretrained on a language from the same language family. We aim to determine which of these proposals is a more effective choice for the training of an ASR for a low-resource language. The primary contribution of this work is in the comparative analysis of performance achieved by utilizing the large pre-trained multi-language model versus utilization of the pretrained single-language model for low-resource language speech recognition.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. E.M. Bakker and Prof.dr. M.S.K. Lew , for guidance and patience through my research project and every weekly meeting.
The past two years have been like a specific adventure about growing up. I would also like to thank Wei, Xiaonan, Yuxuan and Pengxu for their support during the first year. I am really grateful for the help from my friends, Sher, Mingyi, Mingcen, Shengchao, Jianing, Jingbo and Didi. Lastly, my family, who provided unwavering support and encouragement during this journey.

Chapter 1

Introduction

"Hi, Siri!", every time you speak these words to your phone, and give your phone commands to do something, a speech recognition system is converting your speech to text so that it can be shown on your screen and analyzed for your query. Automatic speech recognition(ASR) is a technology that automatically converts human speech content into the corresponding text. It is used in many applications, such as voice-actived commands, voice search, and automated customer services. ASR uses algorithms to analyze audio signals and convert them into text. Due to the development of novel deep learning techniques and the large amounts of training data available, the accuracy of ASR systems has improved a lot during the last decade.

Advances in the performance of ASR systems have especially been made for languages for which huge training datasets are available. However, most of the existing languages are low-resource languages, which means that fewer people speak these languages, and often less high quality datasets exist. This makes it difficult to effectively train ASR systems for such low-resource languages. Currently, companies and research institutions have been investing more in the study of high-resource languages, such as Chinese, English and German. Much progress has been made because huge amounts of labeled data are available. Using supervised learning, ASR systems for these languages could lead to near human or even beyond human performance (Ayvaz, 2022). In addition, unsupervised learning also may lead to great performance for ASR, but it requires even bigger amounts of data. This is a challenge for low-resource languages. In speech recognition, unsupervised learning can generate excellent speech representations by identifying patterns and features in the audio data without labeled data. However, unsupervised learning alone cannot establish the mapping from speech to text, which is crucial for accurate speech recognition (Aldarmaki, 2022). As a result, additional fine-tuning techniques, such as supervised learning, are often necessary to achieve better performance in speech recognition.

Low-resource language speech recognition researchers face two common choices: using large-scale pre-trained models with various pre-training techniques or focusing solely on a specific low-resource language and building a model primarily based on different versions of for example Transformer models (Vaswani, 2017), from a linguistic perspective. Both approaches offer their own benefits. One of the objectives of this thesis is to analyze these two options, choose one exemplary method for each of these approaches, compare the two methods, and provide our conclusions and reasoning for our choices.

Sometimes there may even be many speakers of low resource languages, but often less high quality datasets are available. We suspect that similarities in languages

of the same language family can be exploited, therefore in this thesis we propose to use high-resource languages from the same language family as the low-resource language, to help us improve ASR for low-resource languages. English, German and Dutch all have high quality datasets, Frisian does not have a high quality dataset, which is an example of a low-resource language. In our thesis, we focus on building an ASR system for the Frisian language. The Frisian language belongs to the West Germanic language family, which is close to English, Dutch and German. These are medium to high resource languages. We will study how datasets and pre-trained models can be used to improve ASR systems for the Frisian language.

The research questions and contributions of our thesis are:

Research Question 1: *Can the resources of high-resource languages from the same family as the low-resource language help to improve automatic speech recognition for this language?*

Our main contribution to this question is using different language tokenizers (Choo, 2023) to recognize the Frisian language. The role of a language tokenizer in speech recognition is to convert the speech signal into text data that can be processed. Specifically, the tokenizer segments the speech signal into individual words or phonemes and then converts them into numerical representations for further processing by the model. The tokenizers we used are all for the West Germanic language family, including English, German, and Dutch. We have fine-tuned the speech recognition part of Whisper (Radford, 2022), which is to our knowledge the first time Whisper is used in this way for the Frisian language. Additionally, Frisian has not been used in the training of that specific pre-trained Whisper model.

Research Question 2: *How effective is using a multi-language large pre-trained model and a single-language model for low-resource language speech recognition?*

Our main contribution to this question is by comparing the Conformer-based (Gulati, 2020) model and the Whisper-based model to recognize the Frisian language. Aside from the above contributions, we also tested the Whisper-based model on the performance of transcribing audio. We use Connectionist Temporal Classification (CTC) and a Transducer as the decoder of the Conformer (Gulati, 2020), which is to our knowledge a first for automatic speech recognition of the low-resource language Frisian.

The rest of this paper organized as follows: In Section 2, we introduce the background information and theoretical foundations of automatic speech recognition (ASR). In Section 3, we introduce the methods we used in this thesis, which contains the basic architecture of Whisper, the large pre-trained model Whisper from OpenAI. The datasets are described in Section 4. In Section 5, we give the details of the experiments. In Section 6, the experimental results are given. In the final section, we describe our conclusions and ideas for future research.

Chapter 2

Related Research

Low-resource speech recognition systems often perform poorly in certain speech recognition tasks due to insufficient training data or poor data quality. This situation is usually found for smaller language communities or dialect communities where the amount of speech data is low and the data for training speech recognition models is correspondingly low. In addition, this situation can also occur in non-standard speech scenarios, such as speech in noisy environments, with specific microphones. To solve the low resource problem, researchers need to adopt some special strategies, such as data augmentation, migration learning, transfer learning, semi-supervised learning, etc.

In this section, we give an overview of the research on Automatic Speech Recognition (ASR) and the state-of-the-art on ASR and low-resource ASR.

2.1 Classical methods

From the 1970s, the hidden markov model (HMM) (Jelinek, 1976) and gaussian mixed model (GMM) (Juang, 1985) were the most successful methods for identifying the audio frames of each phoneme. Where GMMs are used to model phoneme audio features and HMMs are used to model the transition between phonemes, which constitute the acoustic model and pronunciation lexicon. Together with a language model, the whole speech recognition architecture is formed. Furthermore, in most traditional speech recognition decoders, weighted finite-state transducers WFST (Mohri, 2008), stores the predicted word sequences in a potentially compact way to avoid taking up too much memory space.

These HMM-based methods ASR architectures constituted the state-of-the-art during the first decade of this century.

2.2 Neural networks based methods

The training of acoustic models for speech recognition is done by supervised learning, which requires knowing the corresponding label for each frame in order to train effectively. Two methods Connectionist Temporal Classification (CTC) (Graves, 2006) (Sak, 2015) (Miao, 2015) which may not know what label corresponds to each acoustic feature and creates its own labels using alignment, and Transducers (Graves, 2012) (Battenberg, 2017) that introduce a branch of language model prediction networks, both combined through a joint feedforward neural network, which is able to use both acoustic and linguistic feature information in predicting the final output, lead to powerful ASR systems. When dealing with different languages, the

shareable features can be divided into bottleneck features (Veselỳ, 2012). Essentially, the model processes the raw speech signal through a series of "hidden" layers that progressively abstract away from the raw signal and extract increasingly high-level features. The bottleneck features are used as output of one of the middle layers of the model, before the features become too abstracted to be useful for ASR. Also, ASR for low-resource languages could use the knowledge learned from other languages (Wang, 2015). For low-resource languages, two data enhancement methods are used, semi-supervised learning and channel length perturbations (Ragni, 2014). Data augmentation can further improve the generalization ability of a model by processing the available, often limited amount of training data to generate more training data.

2.3 Transformer based methods

The Transformer model's self-attention mechanism (Vaswani, 2017) is inspired by human's ability to focus only on important information while learning from input sequences. To eliminate label bias (Guo, 2019), researchers have used conditional random fields to define a conditional distribution over output sequences (Xiang, 2019). Additionally, combining acoustic and phonological features for multilingual information sharing and migration (Zhu, 2021) has been successful in training models to recognize multiple languages and transfer phonological features to new languages. Self-supervised learning approaches, such as Wave2vec (Schneider, 2019) (Baevski, 2020), have been used to learn audio representations. OpenAI's Whisper (Radford, 2022) is a state-of-the-art open-source Transformer based speech recognition model that has a performance similar to commercial speech recognition models. The Conformer model (Gulati, 2020) is currently one of the most popular speech recognition models, which uses a convolution-enhanced Transformer. Both Whisper and Conformer models will be used in our studies.

2.4 Low-resource ASR

The low resource problem is an important research direction in speech recognition and requires researchers to employ various strategies to solve it. The goal of these strategies is to improve the generalization ability of the model so that the model can be better adapted to a variety of different speech scenarios. (Wang, 2020a) presents an end-to-end approach based on the Transformer architecture for low-resource speech recognition. By meta-learning the initialization parameters from a variety of pretraining languages (Hsu, 2020), we achieve rapid adaptation to previously unseen target languages. Using high-resource language data to help low-resource language ASR became an option (Zhang, 2021), it can enhance the representation learning of multilingual unannotated data using a small amount of annotated data from high-resource languages. But each input utterance contains numerous sound units, a crucial aspect of this methodology (Hsu, 2021) involves solely applying the prediction loss over the masked regions, which enforces the model to learn an integrated acoustic and language model over the continuous inputs. Our approach is to use pre-trained large models to train low-resource languages in the presence of a shortage of datasets, with the help of connections between languages.

Chapter 3

Fundamentals and Methods

In this chapter, firstly the fundamentals of automatic speech recognition (ASR) are presented, followed by a detailed description of the state-of-the-art models used in this research: Whisper and Conformer.

3.1 Fundamentals of Modern Speech Recognition

More than a decade ago, the state-of-the-art ASR used HMM models, where the time-series speech signal would be used. The ability to effectively represent the dynamic information of a speech signal is crucial for improving the performance of the model. As speech signals are time-series signals, models that can capture their sequence information are particularly effective. The DNN-HMM model combines a DNN with an HMM. While the RNN model is specifically designed for sequence-to-sequence modeling using feedback connections in the hidden layer, allowing it to make use of previous information, making it an ideal choice for processing speech signals.

Currently, the most advanced speech recognition methods utilize deep neural networks (DNNs) and their various network architectures. Among them, end-to-end DNN speech recognition systems have proven to be particularly effective, outperforming traditional methods. Unlike traditional approaches that break down the speech recognition task into multiple subtasks, such as lexical model, acoustic model, and language model, end-to-end systems generate the corresponding natural language text directly, which greatly simplifies the training process. Furthermore, these systems have proven to be capable of achieving state-of-the-art speech recognition performance, and have become an important tool for ASR.

In this section, we will introduce the fundamentals of modern speech recognition, including how to process speech signals, and introduce the Transformer architecture.

3.1.1 Mel spectrogram

Mel spectrograms see Fig 3.3 are a commonly used feature representation in speech processing, particularly in the area of deep learning. The audio signal sample see Fig 3.1 is usually digitally represented by a time series data. Time series data is one-dimensional data. Using a fast Fourier transform (FFT) see Fig 3.2, the signal is mapped from the time domain to the frequency domain. Using a sliding window on the audio signal, the speech data is transformed from a one-dimensional format into a two-dimensional format. The transcribed text of Figures 3.1-3.3 the depicted

20471579.wav is *"de untdekking fan it skaaltsje soarge foar in soad reaksjes"*.

These spectrograms capture both the time-varying nature of speech signals and the energy-based (amplitude) characteristics of the signal. In machine learning, it is crucial to represent data in a way that effectively captures the underlying patterns and information. By using mel spectrograms as a feature representation, information can be extracted at every timestamp, allowing for a detailed and accurate representation of the acoustic properties of the speech signal. This makes the mel spectrogram a powerful feature for speech recognition and other speech processing applications.

3.1.2 Language model

The traditional speech recognition training and recognition process is divided into four steps: feature extraction, training an acoustic model, training a language model, and optimizing a decoding search. A language model is a model that analyzes and scores the "plausibility" of a sentence in a certain language by means of some sets of rules.

$$\widehat{W} = argmax p(W \mid O) = argmax \frac{p(W)p(O \mid W)}{p(O)} \tag{3.1}$$

In Formula 3.1, the goal of ASR is formalizing given the input observation sequence O, the speech recognition model finds the most likely word sequence $\widehat{W}$, $p(W \mid O)$ represents the acoustic model, and p(W) represents the language model. Language models are used to obtain the probability values of certain word groups here by dismissing unreasonable outputs. With the acoustic model and the language model, we can estimate the probability of each phoneme in each frame and find the higher probability phoneme sequence results in the recognition.

The language model helps ASR models to recognize the most likely transcription of a spoken utterance. Language models can be based on statistical methods, such as n-gram models, or more sophisticated neural network-based models, such as recurrent neural networks (RNNs) or transformers. By incorporating language knowledge, the language model assists in selecting the most likely transcription among a set of candidates that are generated by the acoustic model. This results in better accuracy and performance in ASR systems.

3.1.3 Evaluation metric

In order to evaluate the speech recogntion model performance, we use Word Error Rate (WER) as evaluation metric. This is a key evaluation metric in the field of speech recognition, with lower WERs indicating better results. The recognized and transcribed text is compared to the original text. The WER is calculated by determining the number of insertions, deletions, and substitutions, and subsequently obtaining the percentage of text changed from the original reference text. But WER has some limitations. For example, although its value has a lower limit of zero, which indicates a perfect match between the recognized and the reference text, its value has no upper limit, so it may be difficult to evaluate the performance of ASR in an absolute way, and also the consistency between WER and human evaluation is weak (Daniel Licht, 2022). Nevertheless, WER is the most important evaluation

metric used in ASR and the definition is given in Equation 3.2.

$$WER = \frac{S + I + D}{N} \tag{3.2}$$

where **S** is the number of substitutions of a word, **I** is the number of insertions and **D** equals to the number of deletions of a word. **N** is the number of words in the reference sequence.

Furthermore, we use the Character Error Rate (CER), which is also an important metric in the field of speech recognition.

$$CER = \frac{S + I + D}{N} \tag{3.3}$$

where **S, I, D** stands for the number of character substitutions, character insertions and character deletions. **N** is equal to the number of characters in the reference text.

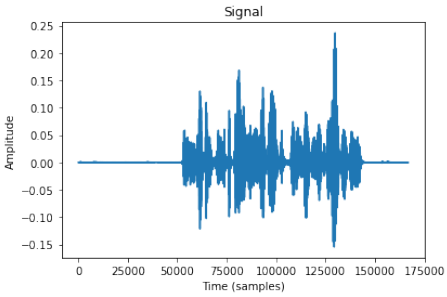

FIGURE 3.1: The utterance raw signal of the Frisian 20471579.wav in CommonVoice, the transcribed text is "de untdekking fan it skaaltsje soarge foar in soad reaksjes"

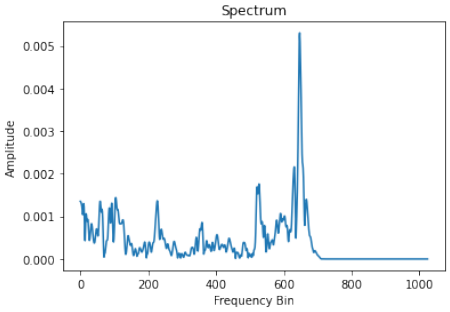

FIGURE 3.2: The FFT result of Frisian 20471579.wav raw signal, the transcribed text is "de untdekking fan it skaaltsje soarge foar in soad reaksjes"

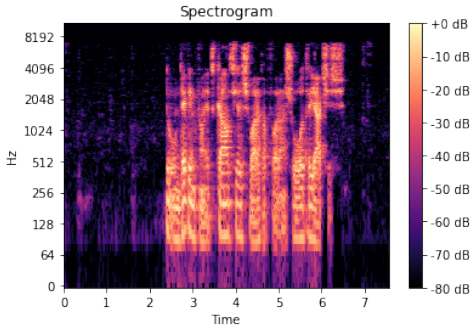

FIGURE 3.3: The mel spectrogram of 20471579.wav

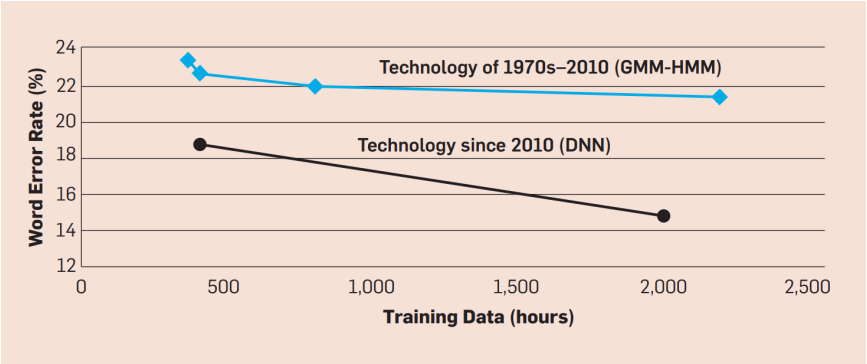

FIGURE 3.4: Historical progress of speech recognition word error rate, which is not precise but cohesive from data scattered over a number of published papers (Huang, 2014)

3.1.4 Convolutional neural network

The state-of-the-art model Conformer combines a CNN with a Transformer. DNNs and CNNs were successfully introduced to the field of ASR a decade ago, where the HMM-based ASR did not show any progress on challenging datasets, such as switchboard (Huang, 2014). Here spectrograms are used to represent audio as a 2D image, which can then be processed using CNNs.

The convolutional neural network is a network structure built by convolutional layers. One of the main reasons why deep learning research exploded in such a short period of time is the introduction of CNNs and their successful applications. Thereby addressing one of the biggest problems of DNN - the explosion of parameters, which makes convergence difficult to obtain. The idea of CNN is that each featuremap shares a common convolutional kernel, and the parameters of the convolutional kernel are independent between different layers.

3.1.5 Connectionist Temporal Classification(CTC)

In our third experiment, CTC acts as the decoder of Conformer. The Connectionist Temporal Classification(CTC) (Graves, 2006) (Sak, 2015) (Miao, 2015) model is the originator of end-to-end speech recognition, predating the Seq2Seq model, where its modeling idea differs from the Seq2Seq model. Early CTC methods computed probability sums by exhaustively enumerating sequences of words (or phonemes) with empty symbols to obtain text (or phoneme) sequences, and later using a forward-backward algorithm similar to the Viterbi method to reduce the computation and obtain the maximum probability path for the predicted sequence. However, since the CTC method cannot adjust the current output by the above output, it often outputs repetitive text, so the general CTC method usually also needs to be followed by a language model.

One of the innovations is to introduce blank symbols to make the far shorter output y to align with input x (see Fig 3.5). Thus there is no need to align and label data individually. The training of acoustic models in speech recognition is mostly done using supervised learning, which needs to know the corresponding label for each frame in order to train effectively, and the speech must be forced to be aligned

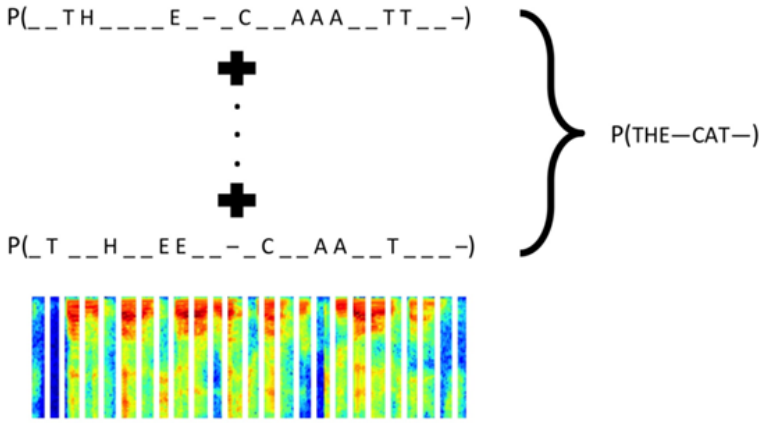

FIGURE 3.5: CTC input audio and text correspondence(Graves, 2006)

in the data preparation stage of training. The introduction of CTC can relax this requirement of one-by-one correspondence, and only an input sequence and an output sequence are needed for training. The blank symbols and the predicted duplicate symbols are finally eliminated.

In training CTC differs from the Seq2Seq model in the following ways:

- CTC can generate a corresponding subword for each frame when decoding, so CTC supports streaming speech recognition better than Seq2Seq.
- The modeling of CTC does not directly model the dependencies between different words in Y, so the generated text is of poor quality from the perspective of the language model. To solve this problem, CTC often has to be decoded together with an external language model to generate better results.

3.1.6 Transducer

In our third experiment, a Transducer acts as the decoder of Conformer. Since the CTC model cannot show the word-to-word dependencies in modeling Y, Transducers (Graves, 2012) (Battenberg, 2017) can be considered as an extension of the CTC model such that it can model the language by a text prediction network, which compensates for the shortcomings of CTC. Originally CTC works with the unreasonable assumption that tags are independent of each other. We know that there are contextual relationships in a language system, and the transducer structure can model and thus compensate for this unreasonable assumption.

Moreover, Transducer supports multiple output units for the same frame, while CTC does not. Since both CTC and Transducer are decoding frame by frame and the decoding process ends at the end of speech, they are also known as frame synchronized models. In contrast, the Seq2Seq model is decoding word-by-word. The output of CTC at the current moment depends only on previous input information; while the output of transducer at the current moment depends not only on previous input information, but also on previous output information. But the training process can be very inconsistent, and relies on many training strategies, further increasing the difficulty of training.

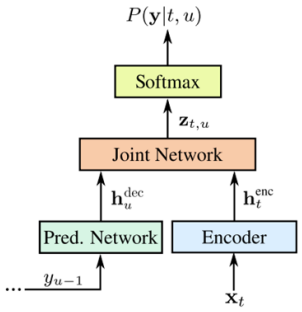

FIGURE 3.6: Transducer architecture(Graves, 2012)

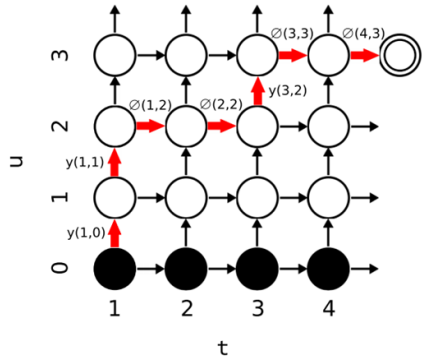

FIGURE 3.7: Transducer transition matrix(Graves, 2012)

3.1.7 Transformer

In our experiments, we use Whisper and Conformer, which are based on Transformers. The basic architecture of Transformer (Vaswani, 2017) (Bahdanau, 2015) (Chorowski, 2015) is depicted in Fig 3.8. It consists of the following key components:

- Encoder: transforms the input speech into a higher-level representation. Plays a role similar to that of an acoustic model.
- Attention: identify the encoded frames.
- Decoder: predict the output. Plays a role similar to that of a language model.

Transformers were initially successful in the field of machine translation. The Transformer establishes the sequence relationship between speech features and text symbols, which is essentially a seq2seq structure containing multiple Encoders and Decoders. Each layer of the encoder has three operations, namely, Self Attention, Layer Normalization and Feed Forward, while each layer of the decoder has four operations, namely, Self Attention, Layer Normalization, Encoder-Decoder Attention and Feed Forward.

The main problem it solves is to improve the parallelism of the encoder. Self-Attention computes Attention for each separate word and all words, which can capture long-distance dependencies. It captures a kind of spatial semantics, as it can focus on different parts of the encoder output, and is believed to have a high learning capability. It has shown to surpass previous end-to-end approaches using random residual connections, greatly improving model generalization performance and training efficiency. At the same time, after a series of explorations on Attention, a number of optimization techniques have been identified: using large modeling units, such as subwords or words, which are more stable and helpful for language modeling.

3.2 Methods

In this section, we introduce the large language model (LLM), which is used in the first part of the experiments. We study if a large pre-trained model can help us to

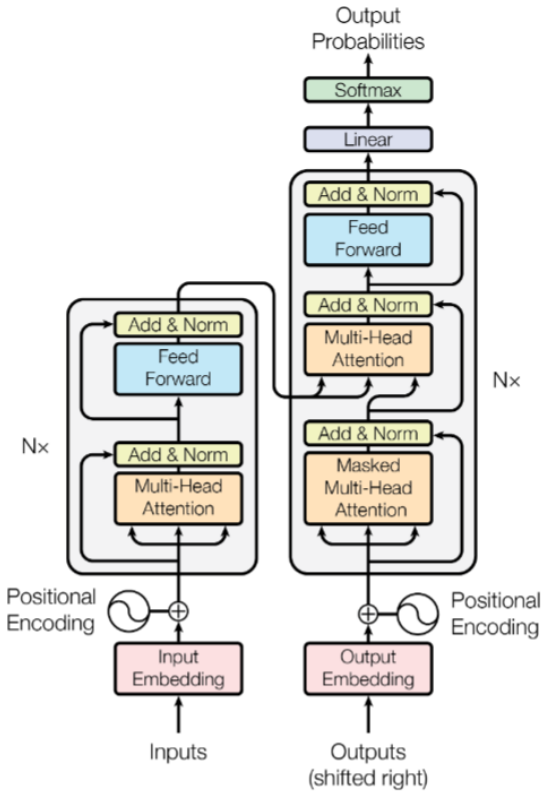

FIGURE 3.8: The basic architecture of Transformer(Vaswani, 2017)

improve the performance of low-resource language speech recognition. Then we introduced the Conformer model used in the second part of our experiments. We trained Conformer as a single low-resource language model.

3.2.1 Fine-tuning large pre-trained model Whisper with a low-resource language

Large Language Model(LLM)
The main reasons why it can be challenging to apply deep learning comes from two aspects: on the one hand, the limited total amount of training data for a specific task. As the model capacity increases, it needs to be supported by a larger amount of training data, otherwise even if the model has enough depth to model the task, the task can not be effectively learned. And before the emergence of pre-trained models like Bert (Devlin, 2019) and GPT (Brown, 2020), it is obvious that this is a serious problem; another aspect is the LSTM/CNN feature extractor, which may not be strong enough to express all the important features for the task.

Many ASR and Natural Language researches have switched to two-phase models: model pre-training phase + application fine-tuning after the introduction of Transformers. Current research shows that for many language-related tasks, the effectiveness improves dramatically as the LM model grows in size. Linguistic knowledge refers to lexical, syntactic, semantic, and other knowledge that helps humans or machines understand natural language. LM can learn linguistic knowledge of various hierarchical types, which is one of the most important reasons why various language-comprehension-like natural language tasks get a substantial improvement after using pre-trained models (Tamborrino, 2020) (Yu, 2010) (Wang, 2020b).

Fine-tuning
Given a pre-trained model, the model can be fine-tuned on a new task. Compared with training a model from scratch, fine-tuning saves a lot of computational resources and computational time, improves computational efficiency, and often can even increases accuracy. The common pre-trained model is characterized by the fact that a large dataset is used for training, and has the ability to extract both shallow basic features as well as powerful deep abstract features.

Often after a few epochs and updating the parameters, it could achieve good performance. However, fine-tuning tends to cause over-fitting to specific data and poor generalization.

Whisper
OpenAI proposed Whisper (Radford, 2022) is a large-scale pre-trained model based on weakly supervised learning. OpenAI built speech datasets consisting of 68,000 hours, but in this paper, it is not mentioned which training dataset is used. According to the results of the subsequent experiments, the data were most likely obtained from video sites or music sites. The data set was poorly labeled, and in the choice between quality and quantity, the authors chose quantity. Based on the results, simple models with large amounts of data can also yield amazing results.

The architecture of Whisper
Since whisper is a multi-task pre-trained model, in this section, we only introduce the speech recognition related part. The architecture of Whisper is a very standard

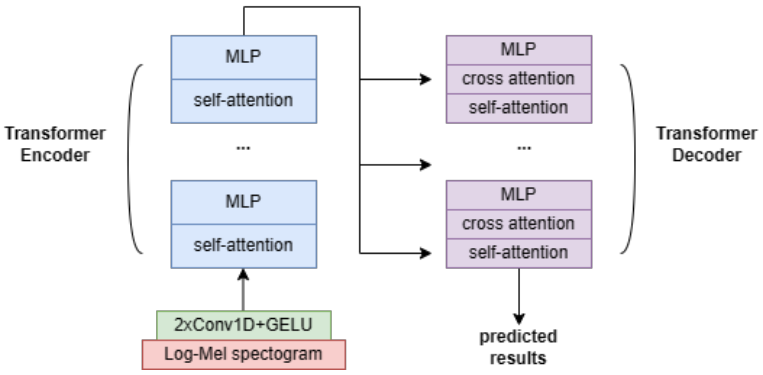

FIGURE 3.9: The basic architecture of Whisper

TABLE 3.1: #parameters of different sizes of Whisper models

	tiny	base	small	medium	large
parameters	39M	74M	244M	769M	1550M

Transformer architecture, which has been explained previously. In this section, we introduce the encoder module of Whisper depicted in Fig 3.9. Shallow linguistic knowledge such as lexical and syntactic knowledge is stored in the lower and middle layers of the Transformer, while abstract linguistic knowledge such as semantic class knowledge is widely distributed in the middle and upper layers of the Transformer structure.

3.2.2 Conformer

Conformer (Gulati, 2020) is the abbreviation of Convolution-augmented Transformer. In fields such as machine vision, convolutional networks have an inherent advantage for local feature acquisition and utilization, while Attention can acquire global dependencies over longer distances, so combining the two together is ideal for speech, because the short-time temporal features of speech determine the sound of speech, while the long-time temporal features determine the meaning of speech. The combination of the two makes it possible to jointly determine the sound and meaning of speech.

Transformer allows for better modeling of content-based global interactions. CNNs can make better use of local features. Combining the two allows for better-unified modeling of local and global features in speech sequences and streamlines the parameter size as much as possible. Although transformers have achieved some good results in the ASR domain, they have difficulties with extracting fine-grained, localized features. The CNN network can model the local area very well based on the size of the window, but for a global perspective (large pictures, or, long speech signals), very deep CNN layers need to be stacked, which to some extent weakens the ability of CNN to express global information.

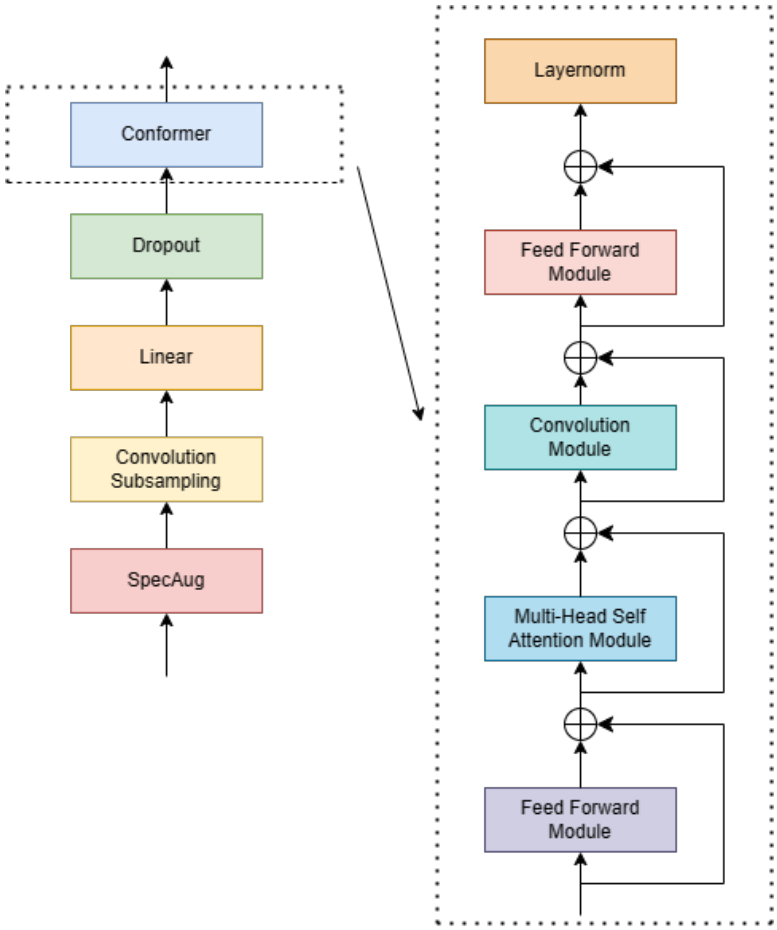

FIGURE 3.10: The basic architecture of Conformer encoder

The architecture of Conformer

The structure of the Conformer is based on Transformer, with the Encoder part improved as shown in Fig 3.10, retaining the multi-headed attention module and effectively interspersing the residual connection and feedforward layer in it, with the difference that it adds downsampling in the embedding layer and the convolution module in the core block. The input vector first passes through spectral enhancement, downsampling of the embedding layer, etc., and then enters the Conformer module, which is a layered structure with a convolutional and multi-headed attention module in the middle, sandwiched by two feedforward modules at the top and bottom, and finally a Layer Norm normalization for convergence. The Decoder part of the original Conformer uses a single LSTM layer.

The original Conformer model is divided into the following main parts:

- **Spectrum Enhancement:** The original Conformer model first does a spectral enhancement of the input vector, which is not internal to the model and belongs to the pre-processing of the data. The spectral enhancement is performed in three ways: time warping, frequency masking and time masking.
- **Embedding Layer:** In the Conformer model, the input vector embedding layer no longer contains position encoding, the position information is implemented

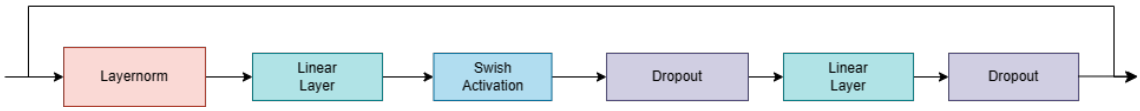

FIGURE 3.11: The architecture of Feedforward module

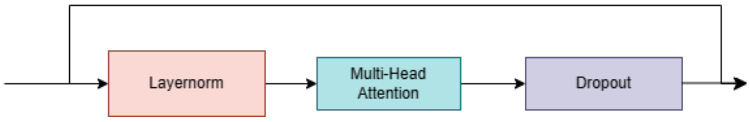

FIGURE 3.12: The architecture of Multi-head self-attention module

in the encoder using relative position encoding. The embedding layer convolutes and downsamples the data to reduce the dimensionality, and then passes it through a linear layer. The linear layer is to further linearly transform the downsampled feature dimension to ensure that it is equal to the specific dimension encoders, which can be chosen as 1024 or 2048, etc. The model will be selected later after experimentation, and the dropout mechanism to obtain robustness improvements.

- **Encoder module:** The encoder is a Conformer module, which is divided into five submodules: the feedforward module (FFN), the multi-headed self-attentive module (MHSA), convolution module (Conv), feedforward module (FFN), and layer normalization (Layernorm).

To be specific, in the encoder module, in the feedforward module, the data is first normalized by layer normalization to accelerate convergence, which means that the data of a sample is normalized to have mean 0 and variance 1. This operation is applied to each sample. Then it is dimensioned by a linear layer with an expansion factor of 4, and then Swish activation is performed.

Swish function is:

$$f(x) = x \cdot sigmoid(\beta x) \tag{3.4}$$

β is a trainable parameter or a constant determined by grid search. Experiments show that it performs better than the Relu function on deep models, as well as being smooth, and non-monotonic (**ramachandran2017swish**).

After activation, the dimensions are then projected back to the original dimensions of the model through a linear layer, and another dropout. Finally it is multiplied by $1/2$ and the previous residuals are summed.

The multi-head self-attention module is similar to the feedforward module, it first imposes layer normalization on each sample, then computes multi-headed q, k, v for the in put multiple sample features, and then calculates their correlation scores with each other after getting them, then transforms them into weights by softmax, sums and superimposes the result vectors of multi-headed.

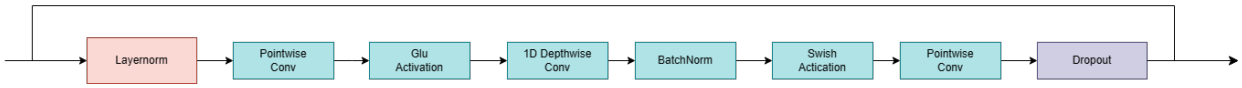

FIGURE 3.13: The architecture of Convolution module

The convolution module is depicted in Fig 3.13, it consists of a one-dimensional convolution by depth, batch normalization by BatchNorm, activation by Swish function, another convolution by point before dropout, and then summed with the previous input residuals for final output.

Chapter 4

Datasets

In total, there are nearly 7,000 language varieties spoken in the world, but most of them, due to the labor intensity of corpus acquisition and formation, lack sufficient annotated data, which makes it very difficult to develop good performing speech recognition systems using a general end-to-end training approach to these languages. Especially in the era of the deep learning, the DNN models are very data-demanding. This lack of data is also the case for Frisian language. In our experiments, we use Frisian language datasets from CommonVoice Corpus 11.0 version 2022-9-21. It contains 130 hours of Frisian speech, of which 50 hours are annotated. After pre-processing, only close to 15 hours of useful annotated data remain. For each audio file, which has an average of 6 secs, it contains text, up_votes, down_votes, age, gender, duration, accents and client_id.

The Common Voice datasets are a collection of publicly contributed recordings of text passages from Wikipedia in various languages. For this thesis, we will be utilizing Common Voice version 11.0, which includes roughly 15 hours of labeled Frisian data after pre-processing. Most ASR datasets typically offer audio samples and their transcribed text, CommonVoice provides additional metadata, including accent and locale, that is not related to our ASR task. We will only utilize the audio samples and corresponding text transcriptions for fine-tuning, omitting any extraneous metadata information. After pre-processing, we have for example as annotation for 20760163.wav, the text is *"ik ha al de hiele dei ofgryslike pineholle"*, up_votes is 2, down_votes is 0, and the duration is 9.024. In the test set, we have for example 28364002.wav, the text with *"hoe let komt de trein oan"*, up_votes is 2, down_votes is 0, and the duration is 4.464. An example from the validation set is 23770516.wav, where the text is *"de feestlikheden geane tebek op in tradysje fan iuwen"*, up_votes is 2, down_votes is 1, and the duration is 5.232.

The Whisper pre-trained model probably used 680000 multitask hours datasets (Radford, 2022), note that this is a speculation as in this paper, the training data is never specified. To our knowledge, in the CommonVoice, the English dataset contains 3209 hours. The German dataset contains 1340 hours. Whereas the Dutch dataset contains 115 hours. All these three languages are members of the West Germanic language family are represented by datasets that are much larger than the

TABLE 4.1: The basic information of CommonVoice 11.0-Frisian

	train	test	validation
#audio files	3702	3027	3027
total duration	5h	5h	5h

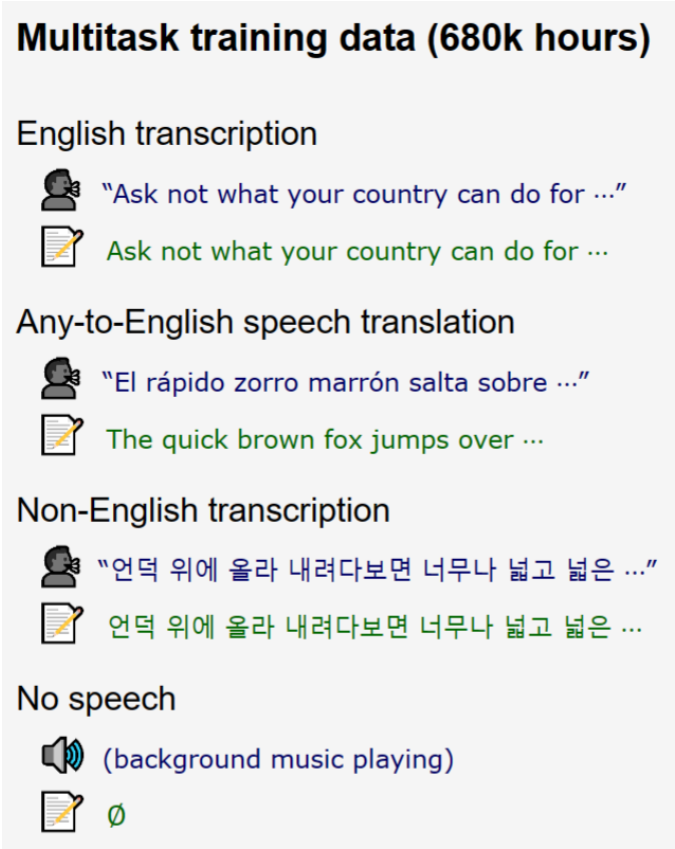

FIGURE 4.1: Some examples of the multi-task training data of Whisper (Radford, 2022)

TABLE 4.2: The CommonVoice datasets for English, German and Dutch compared to the Frisian datasets we used

	English	German	Dutch	Frisian
training set (hours)	1440	720	60	5
test set (hours)	480	240	20	5
validation set (hours)	480	240	20	5

Frisian datasets we used.

For our experiments, we used the mel-specrogram of the audio signal as input. We used the transformer-based pre-trained Whipser model for our experiments. We then also trained a transformer-based Conformer model for our experiments with CTC and a Transducer as the decoder of the Conformer model.

Chapter 5

Experimental Setup

In our thesis, we did three experiments, first as a reference, we test the Whisper performance on a selected sample of 10 minutes. In the second experiments, we fine-tune the Whisper model with the Frisian language dataset from CommonVoice. During the final experiment a Conformer speech recognition model is trained with different loss function, CTC and Transducer.

5.1 Experiment I: Performance test of Whisper

The goal of this experiment is to conduct a small test with different sized pre-trained Whisper models with respect to WER and computational requirements. Following the instructions on the Whisper website, we installed different sized model from the repository, we used PYTHON 3.9.9 and PYTORCH 1.10.1 to run the models. We used an audio file of 10 minutes in length containing a conversation, and used the different sized Whisper models to transcribe these audio files.

Usage of the Python package WHISPER for speech recognition:
(1) First load a pre-trained model named "base" using the WHISPER.LOAD_MODEL function. Use AUTOCUT to automatically generate subtitles for 10 minutes audio files.

(2) Load an audio file using the WHISPER.LOAD_AUDIO function and pads or trims it to fit 30 seconds slides using the WHISPER.PAD_OR_TRIM function.

(3) A log-Mel spectrogram is created from the audio using the WHISPER.LOG_MEL_SPECTROGRAM function, and it is moved to the same device as the model using .TO(MODEL.DEVICE).

(4) The MODEL.DETECT_LANGUAGE function is then used to detect the spoken language in the audio, and the result is printed using a print statement.

(5) The WHISPER.DECODINGOPTIONS() function is used to set decoding options, and the WHISPER.DECODE function is applied on the model, the mel spectrogram and the decoding options to recognize the text in the audio. Finally, the recognized text is printed using print statement.

TABLE 5.1: The config of Conformer-CTC and Conformer-Transducer

model	d_model	n_heads	n_layers
Small(14M)	176	4	16
Medium(32M)	256	4	16
Large(120M)	512	8	17

5.2 Experiment II: Fine-tuning Whisper model for the Frisian language

In this experiment the Whisper model is fine-tuned for the Frisian language, using different tokenizers from the same language family, for example, English, German and Dutch tokenizers. The goal of this experiment is to study if languages from the same language family can help the performance of a low-resource language speech recognition model. We also hope to get new insights in the relation between languages for future directions in low-resource language ASR research.

5.2.1 Environment

We used DATASETS functionality and TRANSFORMERS functionality from HUGGINGFACE to prepare the data and model. Also, we used JIWER functionality and EVALUATE functionality to evaluate the performance after fine-tuning.

5.2.2 Load data

First, we load Frisian dataset from CommonVoice(CV), where the language code is *fy-NL*. As mentioned before, CommonVoice also has other information besides text, such as gender and accent. We removed these columns not related to the fine-tuning. Then, we process the raw audio input to generate a series of audio samples with 30s input length. If the audio samples are shorter than 30s, we will make it 30s by adding more zeros. Next, Whisper feature extractor converts the audio samples to log-Mel spectrograms, which is used as input by Whisper model.

5.2.3 Build feature extractor and tokenizer

In the second experiment, we tested different tokenizers for languages related to Frisian, specifically languages in the West Germanic language family. For using the Whisper model, we also had to downsample the audio file from 48kHz to 16kHz. We used the feature extractor from TRANSFORMERS, from HUGGINGFACE to produce the log-mel spectrograms. A word list containing all basic characters can also be very large, for example, if all unicode characters are considered as basic characters.

5.2.4 Training and evaluation

For the training part, we use AdamW as the optimizer, the hyperparameter β_1 is set to 0.9, the hyperparameter β_2 is set to 0.999, the hyperparameter ϵ is set to 1e-8, the learning rate is 1e-5. In our experiment, AdamW converges faster when dealing with large amounts of data. We use WER as the evaluation metric.

5.3 Experiment III: Frisian speech recognition with Conformer

The goal of this experiment was to build a single-language speech recognition model, and compare it with the results of Experiment II, in which Whisper represented the multi-languages speech recognition model.

5.3.1 Data preprocessing

In this experiment, we make use of the NEMO TOOLKIT from NVIDIA. During the preprocessing part, we convert the *.tsv* file into *.json* manifests. Similar to the downsample operation in Whisper, the audio samples were downsampled sample rate 16kHz. Furthermore, text preprocessing is important, which helps us reduce ambiguity. In this step, we have chosen very conventional methods that should not greatly affect the results of performance, such as replacing all punctuations with spaces, delete apostrophes, make all characters lowercase ,merge multiple spaces. Furthermore some characters, like â, ê, ô and û, which appear often in the Frisian language are replaced with a, e, o and u, respectively.

5.3.2 Build tokenizer

We use Byte-pair encoding as our tokenizer algorithm. Byte-Pair Encoding (BPE) is proposed in (Sennrich, 2016). BPE first needs to rely on a tokenizer that can pre-slice the training data into words, they can be some simple space-based tokenizer. After using these tokenizers, we can obtain a collection of words that have appeared in the training data and their corresponding frequencies. Next, BPE uses all the symbols in this set (splitting words into letters) to create a base word list, and then learns the merge rule to update the base word list by forming a new symbol from two symbols in the base word list. It will continue this operation until the size of the word list reaches a preset size.

5.3.3 Training and evaluation

For the training part, we again use AdamW as the optimizer. Furthermore, we choose CTC and Transducer as decoder of Conformer, which is to our knowledge the first time that using two conventional decoders are combined with Conformer for low-resource speech recognition.

Chapter 6

Experimental Results

Experiment I Performance test of Whisper, we recorded the running time for a 10 minutes audio file contains a Chinese conversation based on different sizes of Whisper. Whisper was used on a machine with a GPU (one 3080ti). In Table 6.1, as for the differences in computation time are small. Computation time is listed for the different sized models of Whisper. We omitted the tiny Whisper model, the WER of tiny Whisper model over 30% on a 10 minutes Chinese conversation audio file. In the second experiment and use the other four sizes of Whisper.

Experiment II Fine-tuning the Whisper model for the Frisian language, First, we fine-tune different sizes of the Whisper model using the cross-entropy objective function, after that we test the performance of different sizes of the Whisper model on the speech recognition task for Frisian language. In Table 6.4, we test 4 different sizes of the Whisper model, with the results as listed. We fine-tune these four models in 3 epochs. As we can see, the WER of the base model is 34.086, the WER of the small model is 33.303, as for the medium model, the WER is 32.309, and for the large model, the WER is 32.819. We can see that the difference in performance is not very large. According to the size of different model, a bigger difference in performance would be expected, but the results from the medium to large model, we observe only a 0.5 percent difference as listed in WER.

The other experiment is to verify how different tokenizers will affect the fine-tuning process and the performance of the fine-tuned model on recognizing the Frisian language. In this experiment, we choose the base Whisper model and 4 different tokenizers, English, Dutch, German, and Hindi. The first three languages are in the same family as the Frisian language, i,e., the West Germanic language family. The final language Hindi, which is from the Indo-European language family, which is not closely related to Frisian. The tokenizer could help us construct a sentence that can deeply reflect the characteristics of the language. In Table 6.3 we see that when using the Dutch tokenizer, and trained for 3, 5 and 10 epochs, the WER results

TABLE 6.1: The results of Experiment I, including the number of parameters, the running time and the WER of transcribing a selected 10 minutes audio

Whisper model	#parameters	time(sec)	WER
large	1550M	78	3
medium	769M	60	7
small	244M	30	13
base	74M	30	22
tiny	39M	30	36

TABLE 6.2: The language proximity between Frisian and West Germanic language family, Hindi

	Dutch	English	German	Hindi
proximity	22.0	25.2	27.2	67.6

TABLE 6.3: The Frisian language WER results of fine-tuning base Whisper model, using different language tokenizers after 3, 5 and 10 epochs

	Dutch	English	German	Hindi
3 epochs	33.471	34.086	34.601	38.468
5 epochs	31.898	31.725	32.617	36.243
10 epochs	30.263	31.606	32.529	35.009

are 33.471, 31.898, and 30.263. When using the German tokenizer, the WER results are 34.601, 32.617, and 32.529, respectively. When using the English tokenizer, the WER results are 34.086, 31.725, and 31.606, respectively. As for the Hindi tokenizer, we found a very large gap with the other three language tokenizers. The WER results are 36.468, 36.243, and 35.009, when training for 10 epochs, the performance becomes even worse than training for only three epochs with the English tokenizer. In Table 6.2, the proximity (Beaufils, 2020) between Frisian and other languages are listed, lower means more related to Frisian. The table 6.3 clearly reflects the fact that Hindi is from a different language family and performs worse than the other three languages from the same language family.

Experiment III Frisian speech recognition with Conformer, we test the performance of Conformer with different loss functions, CTC and Transducer as decoders. The different configuration of these two models are listed in Table 5.1. And we run these two models in 10 epochs. The results are listed in Table 6.5. In this experiment, we built a real Frisian tokenizer, without using other similar languages. In contrast, when using Transducer as loss function, the performance is better than CTC, but using CTC is computationally less demanding than using a Transducer. The WER of Conformer using CTC as decoder, are 38.42 for Small, 33.76 for Medium, and 29.32 for Large. The CER of Conformer using CTC as decoder, are 11.50 for Small, 10.10 for Medium, and 8.78 for Large. The WER of Conformer using Transducer as decoder, are 35.60 for Small, 29.19 for Medium, and 23.48 for Large. The CER of Conformer using Transducer as decoder, are 10.53 for Small, 10.07 for Medium, and 8.26 for Large.

Training and fine-tuning a large Whisper model for 10 epochs needs around 2 hours. Training Conformer for 10 epochs from scratch requires 13 hours, the training is often not very stable. The more attention heads are available, the more they will divide and distract the truly meaningful attention head, making attention less

TABLE 6.4: The Frisian language WER results of fine-tuning different sized Whisper models, using the English tokenizer after 3 epochs

	base	small	medium	large
WER	34.086	33.303	32.309	32.819

TABLE 6.5: The Frisian language WER and CER results of Conformer with different loss function, CTC and Transducer, respectively

Model		WER	CER
Conformer+CTC	Small	38.42	11.50
	Medium	33.76	10.10
	Large	29.32	8.78
Conformer+Transducer	Small	35.60	10.53
	Medium	29.19	10.07
	Large	**23.48**	**8.26**

effective. It is probably not better to have a deeper or wider network, probably because the depth of error backpropagation is more costly. Training a single-language Conformer model costs more time than training and fine-tuning a large Whisper model.

Chapter 7

Conclusion and Future Research

7.1 Conclusion

In this thesis, we have explored the area of low-resource language speech recognition through three experiments. To address the challenges of low-resource language speech recognition, we propose two related methods, using a large pre-trained multi-language model and using single-language models, and try to answer which is the better choice. The main contribution of this thesis is the performance comparison of using a large pre-trained multi-language model to using the single-language model for low-resource language speech recognition, thereby, addressing two research questions.

Research Question 1: *Can high-resource languages from the same family as the low-resource language help to improve automatic speech recognition for low-resource language?*

We evaluated the results of using different tokenizers, and find that, if two languages are closer to each other, the tokenizer could help speech recognition more, obtaining a lower WER. Our experimental results support the idea that tokenizers indeed reflect the inherent language information and when using tokenizers from related or even less related languages, this could help low-resource language speech recognition. When building a large pre-trained model, this was only observed when using languages in the same family.

Research Question 2: *How effective is using a multi-language large pre-trained model and a single-language model for low-resource language speech recognition?*

By comparing the WER results in our experiments, using Conformer with a Transducer decoder showed the best results, but for low-resource language speech recognition. A fine-tuned Whisper model for a low-resource language is more convenient to train in practice. In a large speech recognition model, usually the training set is very large, and at the same time the number of parameters is very large. The potential ability to extract language features inside large models, especially when there are similar languages appearing in the training set is a very powerful mechanism. For low-resource languages, using a pre-trained model is a convenient choice when the WER differences are expected to be small.

7.2 Future direction

After exploring the area of low-resource speech recognition, we point out some future directions in low-resource speech recognition.

(1) Speech recognition technology relies excessively on supervised learning and large amounts of manually labeled data. How to make use of available data more efficiently is very critical. We observed this need when collecting the Frisian datasets. From more than 100 hours of available data, only a small part was useful for our study.

(2) Design new evaluation metrics, WER or CER hardly reflects the accuracy of speech recognition in real life, since mispronounced and even semantically wrong words do not always affect people's understanding of a spoken text.

(3) Find the relation between languages, and make better use of related languages in the same family. For instance, train a multilingual speech recognition model, only using languages that have close relations. And try to find and exploit the intrinsic connections between them.

Bibliography

Aldarmaki Hanan, Ullah Asad Ram Sreepratha Zaki Nazar (2022). "Unsupervised automatic speech recognition: A review". In: *Speech Communication*.

Ayvaz Uğur, Gürüler Hüseyin Khan Faheem Ahmed Naveed Whangbo Taegkeun Bobomirzaevich Abdusalomov (2022). "Automatic speaker recognition using mel-frequency cepstral coefficients through machine learning". In: *CMC-Computers Materials & Continua* 71.3.

Baevski Alexei, Zhou Yuhao Mohamed Abdelrahman Auli Michael (2020). "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in Neural Information Processing Systems* 33, pp. 12449–12460.

Bahdanau Dzmitry, Cho Kyung Hyun Bengio Yoshua (2015). "Neural machine translation by jointly learning to align , translate". In: *3rd International Conference on Learning Representations, ICLR 2015*.

Battenberg Eric, Chen Jitong Child Rewon Coates Adam Li Yashesh Gaur Yi Liu Hairong Satheesh Sanjeev Sriram Anuroop Zhu Zhenyao (2017). "Exploring neural transducers for end-to-end speech recognition". In: *2017 IEEE Automatic Speech Recognition , Underst,ing Workshop (ASRU)*. IEEE, pp. 206–213.

Beaufils Vincent, Tomin Johannes (2020). "Stochastic approach to worldwide language classification: the signals , the noise towards long-range exploration". In.

Brown Tom, Mann Benjamin Ryder Nick Subbiah Melanie Kaplan Jared D Dhariwal Prafulla Neelakantan Arvind Shyam Pranav Sastry Girish Askell Am a others (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Choo Sanghyun, Kim Wonjoon (2023). "A study on the evaluation of tokenizer performance in natural language processing". In: *Applied Artificial Intelligence* 37.1, p. 2175112.

Chorowski Jan K, Bahdanau Dzmitry Serdyuk Dmitriy Cho Kyunghyun Bengio Yoshua (2015). "Attention-based models for speech recognition". In: *Advances in neural information processing systems* 28.

Daniel Licht Cynthia Gao, Janice Lam Francisco Guzmán Diab Mona Koehn Philipp (2022). "Consistent Human Evaluation of Machine Translation across Language Pairs". In: *Volume 1: MT Research Track*.

Devlin Jacob, Chang Ming-Wei Lee Kenton Toutanova Kristina (2019). "Bert: Pre-training of deep bidirectional transformers for language underst,ing". In: *Proceedings of naacL-HLT*, pp. 4171–4186.

Graves, Alex (2012). "Sequence transduction with recurrent neural networks". In: *arXiv preprint arXiv:1211.3711*.

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

Gulati Anmol, Qin James Chiu Chung-Cheng Parmar Niki Zhang Yu Yu Jiahui Han Wei Wang Shibo Zhang Zhengdong Wu Yonghui (2020). "Conformer: Convolution-augmented transformer for speech recognition". In: *arXiv preprint arXiv:2005.08100*.

Guo Haohan, Soong Frank K He-Lei Xie Lei (2019). "A New GAN-based End-to-End TTS Training Algorithm". In.

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

Hsu Jui-Yang, Chen Yuan-Jui Lee-Hung-yi (2020). "Meta learning for end-to-end low-resource speech recognition". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech , Signal Processing (ICASSP)*. IEEE, pp. 7844–7848.

Huang Xuedong, Baker James Reddy-Raj (2014). "A historical perspective of speech recognition". In: *Communications of the ACM* 57.1, pp. 94–103.

Jelinek, Frederick (1976). "Continuous speech recognition by statistical methods". In: *Proceedings of the IEEE* 64.4, pp. 532–556.

Juang, B-H (1985). "Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains". In: *AT&T technical journal* 64.6, pp. 1235–1249.

Miao Yajie, Mohammad Florian (2015). "EESEN: End-to-end speech recognition using deep RNN models , WFST-based decoding". In: *2015 IEEE Workshop on Automatic Speech Recognition , Underst,ing (ASRU)*. IEEE, pp. 167–174.

Mohri Mehryar, Fern o Michael (2008). "Speech recognition with weighted finite-state transducers". In: *Springer H,book of Speech Processing*. Springer, pp. 559–584.

Radford Alec, Kim Jong Wook Xu Tao Brockman-Greg McLeavey Christine Sutskever Ilya (2022). "Robust speech recognition via large-scale weak supervision". In: *OpenAI Blog*.

Ragni Anton, Knill Kate M Rath Shakti P Gales Mark JF (2014). "Data augmentation for low resource languages". In: *INTERSPEECH 2014: 15th Annual Conference of the International Speech Communication Association*. International Speech Communication Association (ISCA), pp. 810–814.

Sak Haşim, Senior rew Rao Kanishka Irsoy Ozan Graves Alex Beaufays Françoise Schalkwyk Johan (2015). "Learning acoustic frame labeling for speech recognition with recurrent neural networks". In: *2015 IEEE international conference on acoustics, speech , signal processing (ICASSP)*. IEEE, pp. 4280–4284.

Schneider Steffen, Baevski Alexei Collobert Ronan Auli Michael (2019). "wav2vec: Unsupervised pre-training for speech recognition". In.

Sennrich Rico, Haddow Barry Birch Alex ra (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL), pp. 1715–1725.

Tamborrino Alex, re Pellicanò Nicola Pannier Baptiste Voitot Pascal Naudin Louise (July 2020). "Pre-training Is (Almost) All You Need: An Application to Commonsense Reasoning". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Vaswani Ashish, Shazeer Noam Parmar Niki Uszkoreit Jakob Jones Llion Gomez Aidan N Kaiser Łukasz Polosukhin Illia (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Veselỳ Karel, Karafiát Martin Grézl František J a Miloš Egorova Ekaterina (2012). "The language-independent bottleneck features". In: *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 336–341.

Wang Changhan, Pino Juan Gu Jiatao (2020a). "Improving cross-lingual transfer learning for end-to-end speech recognition with speech translation". In: *INTERSPEECH 2020*.

Wang Chengyi, Wu Yu Liu Shujie Yang Zhenglu Zhou Ming (2020b). "Bridging the gap between pre-training , fine-tuning for end-to-end speech translation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05, pp. 9161–9168.

Wang Dong, Zheng Thomas Fang (2015). "Transfer learning for speech , language processing". In: *2015 Asia-Pacific Signal , Information Processing Association Annual Summit , Conference (APSIPA)*. IEEE, pp. 1225–1237.

Xiang Hongyu, Ou Zhijian (2019). "CRF-based single-stage acoustic modeling with CTC topology". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech , Signal Processing (ICASSP)*. IEEE, pp. 5676–5680.

Yu Dong, Deng Li Dahl George (2010). "Roles of pre-training , fine-tuning in context-dependent DBN-HMMs for real-world speech recognition". In: *Proc. NIPS Workshop on Deep Learning , Unsupervised Feature Learning*. sn.

Zhang Zi-Qiang, Song Yan Wu Ming-Hui Fang Xin Dai Li-Rong (2021). "Xlst: Cross-lingual self-training to learn multilingual representation for low resource speech recognition". In: *Circuits Systems , Signal Processing 41(12)*.

Zhu Chengrui, An Keyu Zheng Huahuan Ou-Zhijian (2021). "Multilingual , crosslingual speech recognition using phonological-vector based phone embeddings". In: *2021 IEEE Automatic Speech Recognition , Underst,ing Workshop (ASRU)*. IEEE, pp. 1034–1041.