university of groningen / faculty of science and engineering

Artificial Intelligence

MASTER'S THESIS

A Comparative Study of State-of-the-Art Speech Recognition Models for English and Dutch

Author
Neda Ahmadi
s3559734

Internal Supervisor
Dr. M.A. (Marco) Wiering
Artificial Intelligence,
University of Groningen

External Supervisors
Coenraad ter Welle
Siamak Hajizadeh

January 27, 2020

Abstract

The advent of deep learning methods has led to significant improvements in speech recognition. As a result, companies are concentrating more and more on taking advantage of these progressive achievements and try to utilize speech recognition in their voice command systems. However, data preprocessing, the size of the dataset and the architecture of the deep learning model may have a huge impact on the accuracy of speech recognition and the best combination of these arrangements is still unknown in different contexts. Therefore, in this study, we aim to figure out whether it is possible and beneficial for companies to put an effort and use these methods on relatively small datasets and in a language other than English (i.e. Dutch).

In order to find the answer, we present a comparative study of two state-of-the-art speech recognition architectures on small datasets to examine the practicality and scalability of these academically well-received architectures in a real-life environment with all its limitations and challenges. We conducted a series of experiments to train different network architectures on different datasets. We realized that with the same data preprocessing and without using any language model, the listen, attend and spell (LAS) model on both English and Dutch datasets outperforms the CNN-BLSTM model. The LAS model achieved 32% character error rate (CER) on LibriSpeech100 test-clean and 47% CER on the Dutch common voice dataset. Comparing the gained results in this research to the previously reported state-of-the-art results of the LAS model, it can be deduced that the size of the dataset is significantly influential on the accuracy of speech recognition systems. Moreover, considering the results of the experiments, it is realized that the LAS model as an encoder-decoder attention-based deep learning approach is also able to automatically recognize different languages. Further analysis also shows that the training process of these models is quite resource-intensive and powerful processors (GPUs or TPUs) are required in order to reach a reasonable accuracy level. This amount of resources may not be available to all companies considering the budget restrictions. Therefore, although the accuracy levels of the LAS-based models are enticing (both in English and Dutch), the practicality of them in different environments can be considered controversial.

Keywords: Speech recognition; deep learning; encoder-decoder networks; end-to-end speech recognition; Dutch speech recognition; sequence-to-sequence; Speech-to-Text.

Acknowledgements

I wish to express my sincere appreciation to my thesis advisor, Dr. M.A. (Marco) Wiering at Groningen university, who has the substance of a genius: he convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. He was always available and responsible even though I was at a distance. Whenever I ran into a trouble spot or had a question about my research or writing he helped me with his insightful guides. He consistently allowed this project to be my own work but steered me in the right direction whenever he thought I needed it. Without his persistent help, the goal of this project would not have been realized.

I would like to pay my special regards to Gijs van Wijk, Siamak Hajizadeh, Coenraad ter Welle and Dr. Masoud Mazloom for their support and help. I am gratefully indebted to them for their very valuable comments and supervision on this thesis.

The physical and technical contribution of the company where I did my internship is truly appreciated.

I wish to express my deepest gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without their unconditional love.

And finally, I want to dedicate this thesis to my source of strength and inspiration, Fouad.

Contents

1 Introduction **6**
- 1.1 Introduction . . . 6
- 1.2 Research Questions . . . 7
- 1.3 Significance of the Study . . . 8
- 1.4 Thesis Layout . . . 8

2 Theoretical Background **10**
- 2.1 Automatic Speech Recognition . . . 10
  - 2.1.1 Dutch Speech Recognition . . . 10
  - 2.1.2 Speech Recognition Applications . . . 11
- 2.2 Recurrent Neural Networks (RNNs) . . . 11
- 2.3 Data Preprocessing . . . 12
  - 2.3.1 Zero Padding and Normalization . . . 13
  - 2.3.2 Mel Frequency Filter Bank . . . 13
  - 2.3.3 One-hot Encoding . . . 14
- 2.4 Components of the Attention-based Encoder-Decoder Models . . . 14
  - 2.4.1 Sequence-to-Sequence Models . . . 14
  - 2.4.2 End-to-End Models . . . 15
  - 2.4.3 Encoder-Decoder Framework . . . 15
  - 2.4.4 Attention Mechanism . . . 16
  - 2.4.5 LAS Model . . . 17
- 2.5 Components of the CNN-BLSTM Model . . . 19
  - 2.5.1 Convolution-based Models . . . 19
  - 2.5.2 Pooling . . . 20
  - 2.5.3 Fully-Connected Layer . . . 21
- 2.6 Literature Review . . . 21
  - 2.6.1 Research Utilizing the LAS Model . . . 21
  - 2.6.2 Research Utilizing the CNN, LSTM or a Combination of Them . . . 22
- 2.7 Problem Statement . . . 23
- 2.8 Justification of Choises . . . 24

3 Method **25**
- 3.1 Research Pipeline . . . 27
- 3.2 Datasets . . . 27
  - 3.2.1 English Dataset . . . 27
  - 3.2.2 Dutch Dataset . . . 29
- 3.3 Implementation Details . . . 29
- 3.4 Evaluation Metrics . . . 31
- 3.5 Hardware Setup . . . 31

4 Experiments and Results **33**
4.1 Experiments 33
4.2 Results 36
4.2.1 Results of the LAS Model 36
4.2.2 Results of the CNN-BLSTM Model 37
4.3 Model Comparisons 39

5 Discussion and Conclusion **40**
5.1 Discussion 40
5.1.1 Resource Limitations 41
5.2 Conclusion 41
5.2.1 Answers to the Supportive Research Questions 41
5.2.2 Answer to the Main Research Question 42
5.3 Future Work 43

List of Figures

2.1 Difference between RNN and LSTM units [48]. 12

2.2 The pipeline of the data preprocessing. The sound waves are first transfered to the log-Mel filter bank. After that, each text file is matched to its corresponding sound file as the label data. In the next step, in order to make the labels understandable for machines, the sentences are cut to characters and then a charachter to index table map is created. Then one-hot encoding is used to create target outputs for the networks. 13

2.3 The details of the data preprocessing 14

2.4 The one-hot encoded characters [1] 14

2.5 Structure of a sequence to sequence model. The input "ABC" is mapped to the output "WXYZ". It can be seen that the length of the input and the output are different [89]. 15

2.6 An encoder-decoder neural network. An example on machine translation: the input sequence is in English and the output of the system would be in Korean [59]. 17

2.7 In figure (a) the LAS model's components are shown. In figure (b) the structure of the encoder-decoder is illustrated. 18

2.8 The detailed structure of the LAS model's pipeline from input to the output. 19

2.9 Architecture of the DS2 system used to train on both English and Dutch speech [4]. 20

3.1 Deployment and management activities in CRISP-DM [91] 26

3.2 The pipeline of steps from starting the research to finding the best results 27

3.3 All the datasets that have been applied on two models in this research. The details of each dataset are described in section 3.2. 29

4.1 Character Error Rate (CER) and Loss of the LAS model on the Dutch dataset. The x axis is the number of iterations and the y axis is CER/Loss rate. Pink is the validation CER, whereas red describes the train CER. In the Loss graph, light blue represents the validation loss and dark blue shows the training loss. 36

4.2 CER and Loss of the LAS model on the LibriSpeech100 dataset. Light blue is the validation loss, whereas dark blue describes the train loss. Pink is the validation CER, and red describes the train CER. The x axis is the number of iterations and the y axis is CER/Loss rate. 37

4.3 CER and loss on the LibriSpeech100 dataset with 3CNN and 5BLSTM layers with the batch size of 1. Light blue is the validation CER and dark blue describes the train CER. Red is the validation loss, and orange describes the train loss. The x axis is the number of iterations and the y axis is CER/Loss rate. At the best, it reached to the 0.83% CER. 38

4.4 CER and loss on the Dutch dataset with 3CNN and 5BLSTM layers with the batch size of 1. Light blue is the validation CER and dark blue describes the train CER. Red is the validation loss, and orange describes the train loss. The x axis is the number of iterations and the y axis is CER/Loss rate. At the best, it reached to the 0.76% CER. 38

List of Tables

2.1 Papers which use a variation of the LAS as their base model on the different datasets in speech recognition. The research of the pure LAS model on the LibriSpeech100 dataset has not been done before. 22

2.2 Papers that use models with the CNN, LSTM /BLSTM, or a combination of them on speech recognition. Our study picked the Librispeech100 dataset and the CNN-BLSTM model structure. 23

2.3 The recent state-of-the-art researches on CNN-LSTM and encoder-attention-decoder models on speech recognition on the LibriSpeech dataset since 2015. The numbers in parenthesis in the WER column demonstrate if the results are from Librispeech960 or 100. In [104], the version of the Librispeech data set has not been mentioned. 24

3.1 A list of English datasets for speech recognition. 28

3.2 A list of Dutch datasets on speech recognition task. 29

4.1 Experiments which have been done on the LAS model to find the best hyperparameters on the LibriSpeech100 dataset with the learning rate of 0.0001. Opt denotes optimizer, BS = batch size, and Normalized denotes data normalization. The results are reported in character error rate (CER). 33

4.2 Experiments which have been done on the LAS model to study the effect of the different sizes of datasets on the accuracy. Opt denotes optimizer, BS = batch size, DS means Dataset, and Normalized denotes data normalization. The learning rate is equal to 0.0001. 34

4.3 Experiments which have been done on the BLSTM model on the small-Libri dataset. The reported CER is for the dev set. Max-pooling is used on top of the BLSTM layers. FC denotes the fully connected layer and # BLSTMs shows the number of BLSTM layers. No CNN layers used in these experiments. Init. represent weight initialization which prevents exploding or vanishing of the layer activation outputs. In this thesis random and Kaiming initializaion is used. 34

4.4 Experiments which have been done on the CNN-BLSTM model on the small-Libri dataset. The reported CER is for the dev set. ReLU has been used after the BLSTM layers on top of the Max-pooling and the fully-connected is the final layer. In the BLSTM layers, the hidden dimension of each direction is equal to 256. FC denotes the fully connected layer with the dimension of 30, # CNNs shows the number of CNN layers and # BLSTMs shows the number of BLSTM layers used in the architecture and Init. represent data initialization. 34

4.5 Experiments which have been done on the CNN-BLSTM model on different dataset. According to the results of previous experiments the initialization is Kaiming. ReLU has been used after the BLSTM layers on top of the Max-pooling and the fully-connected is the final layer. FC denotes the fully connected layer, # CNNs shows the number of CNN layers and # BLSTMs shows the number of BLSTM layers used in the architecture. 35

4.6 The summary of the best results of all the experiments which have been done with both models on the Dutch and LibriSpeech100 datasets. The Batch size is set to 1. 35

4.7 The target and the output of the LAS model on the Dutch dataset. 37

4.8 The target and the output of the LAS model on the LibriSpeech100 dataset. 37

Chapter 1

Introduction

1.1 Introduction

Speech is the most natural form of human communication [101]. On average people speak 150 words per minute, while they type on average about 50 words per minute, approximately one-third [3]. This shows that we can communicate faster and convey more information via speech than with text. This motivated humans to crave for building machines that are able to recognize our speech more accurately and to make them better at communicating with us [3]. Such machines are called *spoken dialogue systems* and are developed to interact with humans [60]. Other words such as *voice command systems* and *conversational AI* have also been proposed to describe these machines, but in this report we use the term spoken dialogue systems. Moreover, another common term that emerged when such machines became prevalent was automatic speech recognition (ASR) which refers to the process of converting the speech signal into its corresponding text [56].

Nowadays, companies are aware of the significance of speech in the conversation and want to increase the quality of communication with their costumers. However, in the 30 years history of the spoken dialogue systems, the accuracy of speech recognition was not high enough to be trusted as an alternative to humans. Therefore, using a spoken dialogue system was not in the center of the companies' attention in order to invest in. Only during the past decade some initiatives reached some acceptable accuracy rates and put the speech recognition in the limelight again. The history of the voice command systems can be divided into three generations: symbolic rule or template-based (before the late 90s), statistical learning-based (from 90s till 2014), and deep learning-based (since 2014) [26]. It was just after the advent of the deep learning in speech recognition tasks that companies also got attracted to the use of such a system in their business lines. By utilizing deep learning, speech recognition became accurate enough to be useful in the real-life and business use cases [53]. Recently, speech recognition, as the first step of the human-computer conversation system has attracted increasing attention due to its potentials and enticing commercial values.

With the increased need for having better communication with the customers, many companies feel the urge to at least experiment with the voice command systems based on the staggering adoption rate of such systems in countries like the US and China. According to the Gartner, by 2021, 15% of all customer service interactions will be completely handled by AI [1]. There is another survey which shows that 87% of the participants with the role of retailers believe that conversational AI will lead to higher satisfaction levels in their business lines [2].

Despite all the efforts and improvements in speech recognition, it is still hard to reach a

[1]www.gartner.com/smarterwithgartner/4-trends-gartner-hype-cycle-customer-service-customer-engagement/
[2]www.letslinc.com/wp-content/uploads/2017/07/Lin_Brand-Garage_Customer-Service-and-AI-Report.pdf

100% accuracy due to several factors. Such factors make it difficult for machines to recognize and fully understand human speech. For instance, due to the variability in the speech signal, the analysis of human speech which is considered as continuous speech is a challenging task for machines [26]. Although there has been a huge improvement in English speech recognition, enhancement is still needed for other languages. The Dutch language is not an exception. The success of speech recognition in one language does not necessarily extend to the other languages. Therefore, in this research, we try to apply a speech recognition system in one of the business lines of a Dutch financial company in order to assess the claims of the scholars. The results can be used in order to improve the effects of speech recognition on the Dutch company's customer features. After all, one of the main advantages of acquiring speech recognition in a business is the reduction of human resources in customer service which leads to lower employee costs. Moreover, a personalized voice assistant could increase loyalty and customer satisfaction [95].

Studies show that in the topic of ASR, deep learning has shown to be a successful contribution [44]. Generally, the main focus of research in deep learning is to design better network architectures, such as: deep neural networks (DNNs) [23], convolutional neural networks (CNNs) [55, 79], recurrent neural networks (RNNs) [39], and end-to-end models [8, 14, 37]. Among all of these aforementioned network structures, the focus of this research is on the CNNs, RNNs, and end-to-end encoder-decoder networks and the crucial components in these deep learning systems will be discussed. These components are the model architecture, large labeled training datasets, computational scale, data preprocessing, and hyperparameter settings. The effect of these components on the accuracy of speech recognition will be described as well.

In particular, this work will be supported by the experimental investigation of the functional accuracy of neural networks on speech recognition tasks. We also conduct numerous experiments with two different neural network models to predict speech transcriptions from audio.

From a high-level view, the final goal of this research is to pave the way for designing a scalable and practical speech recognition model which considers the limitations in the real-life. A model from which both users and companies can benefit.

1.2 Research Questions

The main objective of this thesis is to first find the two most acclaimed and accurate deep learning architectures for speech recognition tasks on the English dataset, and then, to investigate the effect of the size of the training datasets on the accuracy and the performance of the models. Finally, the models will be applied to the Dutch dataset. In the end, by scrutinizing the results of the experiments, some of the significant factors which may influence reaching higher accuracy can be characterized.

The conversational agent at the first step should correctly understand the customers' questions and then answers them in a reliable and convenient way. To achieve such a system the first step is having an accurate speech recognition system. In the literature, different deep learning methods have been applied to understand and recognize the user's utterances and they achieved high accuracy in speech recognition tasks [67, 15, 42].

Research Objective: The main objective of this research is to help Dutch organizations to choose the best deep learning architecture based on the available resources and datasets at hand.

To support the main objective, this thesis attempts to address the main research question (MRQ) followed by three supportive research questions (SRQ):

1. MRQ: *"Which deep learning architecture for speech recognition tasks performs best on relatively small Dutch datasets"*
   (a) SRQ1: What are the two most used, state-of-the-art deep learning architectures for speech recognition tasks?
   To answer this question, the literature will be investigated to find two state-of-the-art network structures for speech recognition tasks. Among all the networks, we will pick those two architectures which have been used the most in order to have more literature available to compare the results.
   (b) SRQ2: What are freely available datasets in English and Dutch languages for speech recognition tasks?
   This question will be answered by searching for the datasets for speech recognition tasks in English and Dutch languages.
   (c) SRQ3: What are the effects of the size of the dataset on the accuracy of the state-of-the-art models?
   Analyzing the results of the experiments and comparing them with the literature will specify if the models are sensitive to the size of the datasets or not.

By answering the three supportive questions separately, we will be able to answer the main research question. Some experiments will be conducted to apply both of the models on the selected English and Dutch datasets. By comparing the results of the experiments on both models the more accurate model will be found. Moreover, in order to fulfil the objective of this thesis, we will find out if the more accurate model on the English dataset can achieve high accuracy on the Dutch dataset as well.

1.3 Significance of the Study

The main contribution of this thesis is to examine the practicality of the two most commonly used state-of-the-art deep learning architectures in the real-life environment with a limitation on the processing power and also a limited accessible dataset. These resources have been provided to us for this research by a Dutch financial company with near 20,000 employees. In the literature, state-of-the-art models are trained on big datasets such as LibriSpeech960h, Google voice search with 2000 hours of speech, Baidu dataset, etc. However, the obstacles of having limited resources in the real world are usually not considered by scholars. Through this research, we want to investigate if reaching such accuracy is achievable using the resources available to the companies. Otherwise, although these models are reported to be able to give us a very high accuracy but using them in a company can not be possible and there will be a constant need for big companies like Google or Baidu to provide the speech recognition service to other companies because of their huge dataset and very high processing power.

1.4 Thesis Layout

The rest of this thesis will be structured as follows. The next chapter will describe the theoretical background of the relevant methods and models in the speech recognition task. Here some explanations will be provided on the details of the models used throughout this thesis. In chapter 3 the methodologies are discussed, explaining the selection of datasets, together with implementation details and experiments. In chapter 4, the experimental setup to address the research questions will be described. Section 4.2 contains the results of these experiments and the elaboration on the possible reasons behind them. Finally, in chapter 5 the research questions are addressed and the conclusions drawn from the experiments together with potential future work are presented.

Chapter 2

Theoretical Background

2.1 Automatic Speech Recognition

Speech recognition is the first ability which an artificial intelligence (AI) system needs in order to have a smooth human-like conversation with a human. Speech recognition can detect the human voice and convert speech into text which is some form of computer-readable data [86]. This research is in continuation of the previous works in the field of deep learning for speech recognition. In the field of ASR with more than 60 years of research history [57, 25, 9], speech recognition has always been a difficult task. The complication in speech recognition leads to the complexity of the language understanding components and may also affect the efficiency of spoken dialogue systems [13]. Errors in speech recognition limited the capabilities of spoken language understanding and made the text-based language understanding more in the center of attention [94] rather than speech. Therefore, in order to give ASR systems the value that it actually has in human communication, it is important to increase the accuracy of speech recognition. To have accurate speech recognition, applying some deep learning techniques helped to achieve huge successes in the field [57]. It has been more than 20 years that feed-forward neural network acoustic models were explored [73, 30, 11]. Consequently, the number of applications in speech recognition has increased sharply. Speech recognition consists of three main sections; signal level, acoustic level, and language level. In the signal level, the aim is to extract features from the signal by doing pre-processing and denoising (if needed). This step is common in most of the machine learning tasks. The acoustic level is responsible for the classification of the features into different acoustic states which provide more precise criteria. In the language level, we try to create meaningful sentences based on words which are the combination of different sounds [12].

2.1.1 Dutch Speech Recognition

There are a couple of services and open source projects which work on Dutch speech recognition. There are services such as Google cloud speech-to-text, Speechmatics, Nixidia, and open-source Kaldi project. However, at the time of writing this report, Google's specialized model for phone conversations is not available in Dutch. The accuracy of the Nixidia is not satisfying enough and the Speechmatics is costly [96]. Therefore, for the companies, especially those who work with confidential data, it is better to create their own Dutch speech recognition system. One reason is that due to the higher availability of English datasets, most of the available systems are not as accurate as they are in English. Moreover, by having their own speech recognition system, companies can have full control over the data. Although there are some rules which say Google is not allowed to use the audio data for its own projects unless this permission has been given to Google, the data still has to be sent over. Furthermore, in order to use their professional phone model, Google asks for the audio data as well. Therefore, it sounds more secure and peaceful to keep everything in-house especially with the new GDPR laws in place [96]. Therefore there is a need for building a more accurate Dutch speech recognition system which is also applicable for the companies to use and customers will benefit from it as well.

2.1.2 Speech Recognition Applications

The technology of speech recognition and using digital assistants have rapidly moved from cellphones to the homes and offices, and its application in different industries such as banking, marketing, travel agency, and healthcare is becoming apparent. Here are some of the applications in which speech recognition is used as one of the main components [10]:

- At the office
    1. Search for documents on the computer
    2. Request for printing
    3. Start video conferences
    4. Schedule meetings
- Banking
    1. Request information about the account balance and transactions history
    2. Payments
    3. Ask mortgage related questions
    4. Customer intention analysis
- Medical
    1. Immediate, hands-free information retrieval
    2. Payments
    3. Fast and targeted access to the required information
    4. Repeating processes and the steps of specific instructions
    5. Effective way for entering the data and information

By utilizing speech recognition in these areas, we can see the prominent potential in speech recognition to increase the efficiency of all these areas and it can be very effective in many other industries as well.

2.2 Recurrent Neural Networks (RNNs)

Recurrent neural networks or RNNs [76] are a class of neural networks where connections between nodes form a directed graph along with sequential data. As a result, RNNs are able to exhibit temporal dynamic behavior. RNNs use their internal state (memory) to process sequences of inputs. They can be considered as neural networks with loops. Therefore, RNNs are a natural choice for processing sequential data of variable length. Due to their internal memory, RNNs are able to remember previous inputs while taking in new input. This memory structure makes the RNNs a perfect predictor of what will come next. By utilizing memory, RNNs have a deeper understanding of sequences compared to other algorithms [28]. As a result, RNNs are the preferred algorithm for those Machine Learning (ML) problems like time series, speech/text recognition, financial data, audio, etc. The short-term memory of RNNs comes from the recurrency which helps the network to remember the past [2]. Deep neural networks (DNNs) are dominant in many ML tasks because of their high-performance [85]. However, for sequential data like Natural Language Processing (NLP), machine translation [89] or speech recognition, RNNs and especially Long-Short-Term Memory (LSTM) networks perform better [45]. In the naive RNN implementation, the input will not remain in the memory for a long time [66] and the gradients may either increase or decrease exponentially over time depending on the largest eigenvalue of the state-update matrix. Thus, it is hard to train such a network. To overcome this problem, LSTMs are used [33]. The difference between a simple RNN unit and an LSTM unit is shown in figure 2.1. The LSTM also shows better results compared to fully-connected feed-forward DNNs [92]. So far, LSTMs have been effective for learning sequence mappings in speech recognition [40]. They are commonly built as a pipeline of the ASR component to transcribe the user's speech to text. One of the conventional RNNs downsides is that they can only use the previous context. However, in speech recognition, we need the whole utterance to be transcribed at once which increases the need of exploiting future context. To do this, bidirectional RNNs (BRNNs) are used to process the data with two hidden layers in both directions. The data then feeds forwards to the same output layer. BLSTM is the combination of BRNNs with LSTM which make it possible to access long-range context in both input directions [40]. Apart from all the advantages of BLSTM networks, the disadvantage of BLSTM acoustic models is that they need more tuning to their training compared to feed-forward neural networks.

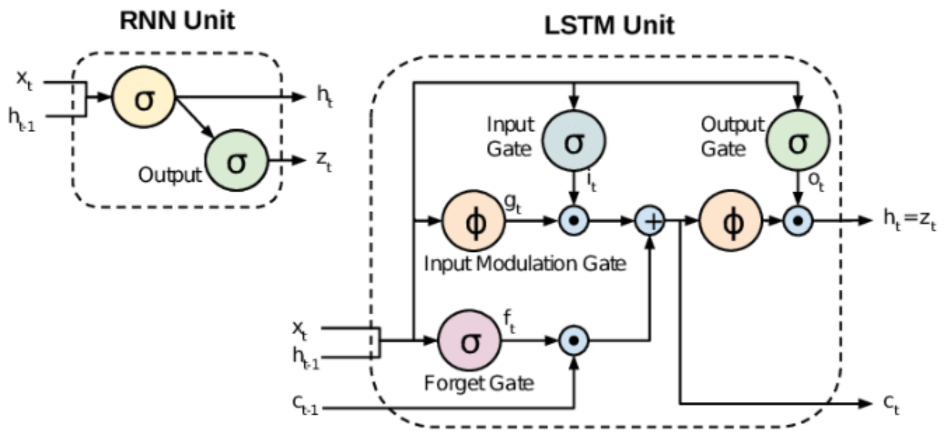


Figure 2.1: Difference between RNN and LSTM units [48].

2.3 Data Preprocessing

After attaining the data, data preprocessing is the first step in speech recognition. The general work of speech recognition is to take the sound signals as inputs and create the characters as the output. Sound waves are one-dimensional. At every moment in time, based on the height of the wave, the sound waves have a single value. However, these waves should be changed into numbers to be understandable for computers. Therefore, in a process named as sampling, only the height of the wave will be recorded at equally-spaced points. Based on the Nyquist theorem, which suggests to sample a sine wave at least twice the sample frequency, it is now possible to reconstruct the original sound wave from the spaced-out samples perfectly without losing any data [21]. In this research, the format of the voice signals has been converted from .flac into .wav files. Then the log-Mel filter bank extracts features from the signals and creates the spectrogram of each sound file. In this research, a 40-dimensional log Mel-frequency filter bank feature with a window size of 0.025 has been extracted from the raw speech data with delta and delta-delta acceleration. As a result, from the spectrograms made from the Mel-frequency filter bank, tensors of the sound data will be created. For labeling the sounds, a text file corresponding to each sound file is provided in the dataset. Each text file has been separated character by character. Then a character map is created which maps each character to a number so that we can easily map them to the One-Hot encoding system. Figure 2.2 demonstrates the whole process for the data preprocessing in this research.

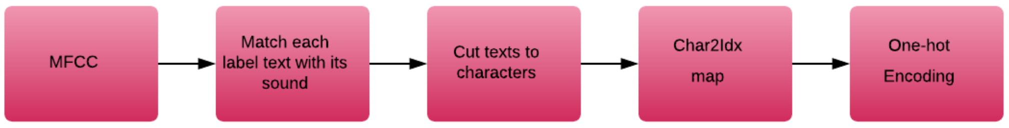


Figure 2.2: The pipeline of the data preprocessing. The sound waves are first transfered to the log-Mel filter bank. After that, each text file is matched to its corresponding sound file as the label data. In the next step, in order to make the labels understandable for machines, the sentences are cut to characters and then a charachter to index table map is created. Then one-hot encoding is used to create target outputs for the networks.

2.3.1 Zero Padding and Normalization

Zero padding is a crucial feature to increase the width of the input and make it wider. In general, zero padding is used to control the input width and the size of the output consequently [34]. In this research, all the sound sequences are padded to a max length so that the width of all the input data becomes unified. For the normalization, the input feature sequence H is normalized as follow:

$$H' = \frac{H - \mu}{\sigma} \tag{2.1}$$

where μ and σ are vectors of per feature channel means and standard deviation over all of these feature channels. To explain more, the mean is the sum of all input data divided by the total number of them as shown in equation 2.2.

$$\mu = \frac{1}{m}\sum_i H_i \tag{2.2}$$

and the standard deviation as shown in equation 2.3, demonstrates to what extent a set of numbers are different from each other [34].

$$\sigma = \sqrt{\frac{\sum_i (H_i - \mu)^2}{m}} \tag{2.3}$$

2.3.2 Mel Frequency Filter Bank

The Mel (Melody) scale was first suggested by Stevens and Volkman in 1937 [87]. A Mel frequency filter bank is suitable for speech recognition tasks because it considers the sensitivity of humans to a specific frequency. It is computed by converting the conventional frequency to Mel Scale. This manipulation helps to understand the human voice better. Based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency, the Mel frequency cepstrum (MFC) can represent the short-term power spectrum of a sound [78]. On top of the fast Fourier transform (FFT) on the magnitude spectrum, a filter bank will be used to derive the MFC feature.

In figure 2.3 the process from the sound wave to the tensor data has been illustrated.

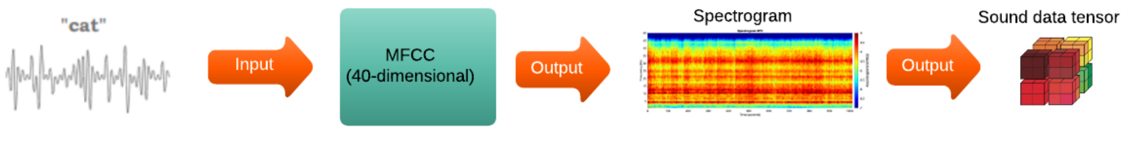


Figure 2.3: The details of the data preprocessing

2.3.3 One-hot Encoding

When there is a classification problem, the one-hot encoding can be used. By using one-hot encoding, categorical variables are converted into an ML friendly form so that the ML techniques have easy access to the target output. In figure 2.4 the conversion from characters to one-hot encoded numbers is shown. It can be seen that the character "A" has the same value (1000) in different positions.

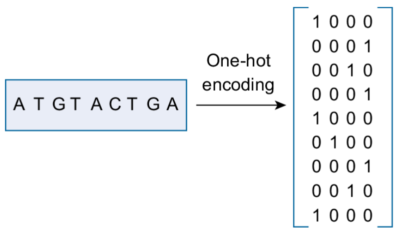


Figure 2.4: The one-hot encoded characters [1]

2.4 Components of the Attention-based Encoder-Decoder Models

Here, the essential components of the end-to-end *encoder-attention-decoder* models will be described.

2.4.1 Sequence-to-Sequence Models

Despite all the advantages of DNNs, they have a significant limitation. They can only be applied when the inputs and desirable output can be sensibly encoded with fixed-dimension vectors. However, there are many problems like speech recognition and machine translation in which the length of the sequences is not known a-priori. In speech recognition, for instance, the network should learn to transcribe an audio sequence signal to a text word sequence [15]. In such problems, a sequence of input should match to a sequence of outputs. Therefore, a domain-independent method that learns to map sequences to sequences is needed [89]. A sequence to sequence model is a framework that tries to address the problem of learning variable-length input and output sequences. Such a model could be an LSTM sequence-to-sequence model which is a type of neural generative model [89]. It uses an encoder-decoder structure to first map the sequential variable-length input into a fixed-length vector by an RNN encoder and then in the RNN decoder, this fixed-size vector will be used to produce the variable-length output sequence, one token at a time [15]. In figure 2.5 the structure of a sequence to a sequence model has been illustrated.

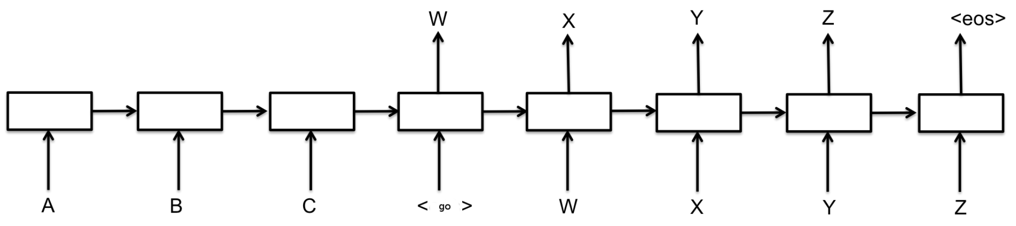


Figure 2.5: Structure of a sequence to sequence model. The input "ABC" is mapped to the output "WXYZ". It can be seen that the length of the input and the output are different [89].

2.4.2 End-to-End Models

An end-to-end model in speech recognition, simply means *"from speech directly to transcripts"* [15]. In general, it denotes a simple single model that can be trained from scratch, and usually directly operates on words, sub-words or characters/graphemes. Accordingly, it simplifies the decoding part by removing the need for a pronunciation lexicon and the whole explicit modeling of phonemes [105]. The neural network then provides a distribution over sequences directly. All speech recognition systems with an end-to-end neural network, utilize RNNs in at least some part of the processing pipeline [106]. Since the most successful RNN architecture used in speech recognition is the LSTM [39, 35, 45, 97], LSTMs have become a default method for end-to-end models.

2.4.3 Encoder-Decoder Framework

In general, an encoder-decoder is a neural network that learns a lower-dimensional representation of the original data. It can be considered as a multi-layer PCA (a dimensionality reduction technique) with non-linearities. The main idea of the framework is to divide the process of generating output into two subprocesses; encoding phase and decoding phase. The encoding part is responsible for providing a good representation of the input. To do so, a projection function also known as "encoder", should project the input into another space like in a feature extraction process. It extracts a feature vector from the input. Now that the encoder generated a "latent vector", it will pass this vector to the decoder. The decoder then processes the extracted feature vector and creates the output based on this vector which although it is smaller than the input, it contains the most important features and information of the input. Both encoder and decoder have a neural structure. However, the structure of the encoder could vary according to the type of input. For instance, for text or other sequential input data, an RNN is used as the encoder, but CNNs can also be used to encode the images [5]. The neural encoder-decoder framework has recently been applied as a solution to a wide variety of challenges in natural language processing, computer vision and speech recognition. This framework can enhance the accuracy of speech recognition task significantly [105, 15, 8]. More specifically, the encoder-decoder structure is usually used when the goal is to generate sequential outputs such as text.

One of the main problems of decoders in the encoder-decoder based models is extracting long-term dependencies and generating different sentences. A simple solution to this is using LSTM cells. Another approach is to make deeper decoders by using stacked structures of RNNs. As a result, the model will be able to generate more detailed sentences. However, using stacked decoder structures may cause the problem of vanishing gradients. To avoid this, an attention mechanism can be used. The attention mechanism is a technique to focus on different parts of the input, at each step of generating outputs.

The encoder-decoder framework with attention has become the standard approach for machine translation [7, 102], object recognition in images [6] and speech recognition [15, 93]. In this work, we also investigate the LAS model which is an *encoder-attention decoder* based model for speech recognition on the Dutch dataset.

2.4.4 Attention Mechanism

Generally, the attention mechanism was developed to improve the performance of RNNs in the encoder-decoder framework. It is an increasingly popular mechanism used in many neural architectures [32]. In speech recognition, it is important to process elements of the sound wave correctly. These elements are being characterized by having a different relevance to the task at hand. For instance, in a visual question-answering task, objects in the foreground are usually more relevant in answering a question but background pixels are more relevant to questions regarding the scenery. An effective solution to such problems is to find the relevant part so that the system could provide the computational resources efficiently on a restricted set of important elements. This solution is viable, especially if the input is very information-rich, like in speech recognition, where the input is possibly a long audio sequence. There is a machine learning-based approach which tries to find the relevance of input elements by automatically weighing the relevance of any region of the input, and then taking such a weight into account while performing the main task. The attention mechanism is the most common and well-known solution to this problem [32]. Bahdanau, Cho, and Bengio were the first researchers to introduce the attention mechanism in their research on natural language processing (NLP) for machine translation tasks [7].

In the encoder-decoder approach, the encoder should compress all the important information of the source sentence into a fixed-size vector. This may cause difficulty for the neural network to model long sentences, especially those that are longer than the sentences in the training corpus [7]. The advantage of applying attention from the baseline encoder-decoder is that it does not attempt to encode the input sequence into a single fixed-size context vector. Instead, it encodes the input into a sequence of annotated context vectors that are filtered for each output time step specificaly. Then, it selects a combination of these vectors while decoding and generating the output in each step.

The attention mechanism addresses both alignment and translation. Alignment is about matching which parts of the input sequence are relevant to each word in the output and by using translation the network tries to highlight the relevant information to select the appropriate output. To find the appropriate output, a distribution function is used which takes into account locality in different dimensions, such as space, time, or even semantics. It is also possible to model attention in such a way that it will be able to compare the input data with a given pattern, based on similarity or significance. However, an attention mechanism can learn the concept of relevant information by itself, consequently, it can create a representation to which the important data should be similar [32]. As a result, by injecting knowledge into the neural model, attention can also play a role in addressing fundamental AI challenges.

In figure 2.6 the structure of an encoder-decoder framework with attention mechanism is demonstrated.

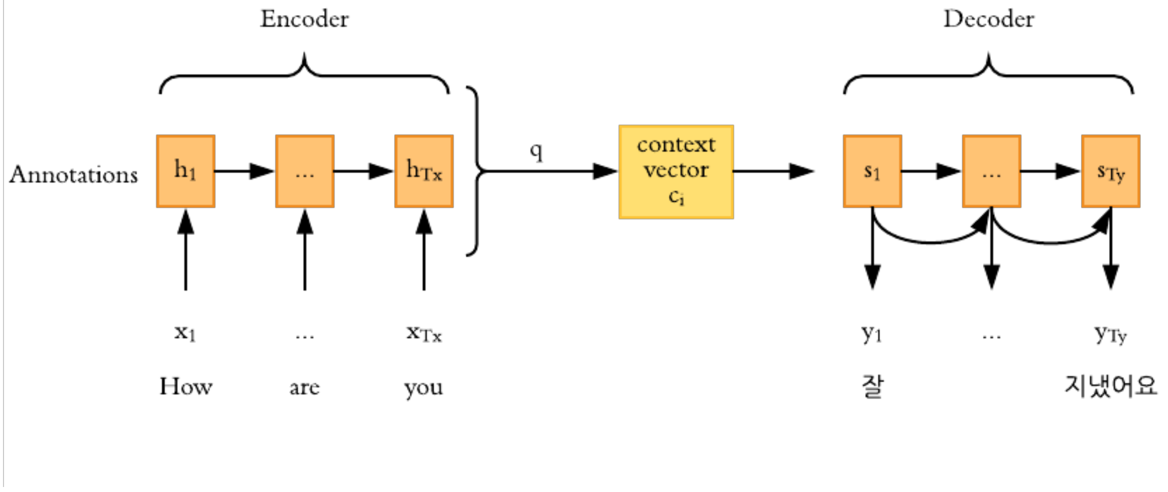


Figure 2.6: An encoder-decoder neural network. An example on machine translation: the input sequence is in English and the output of the system would be in Korean [59].

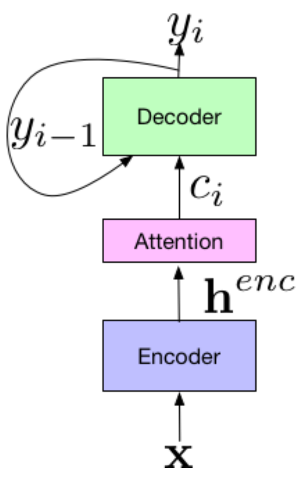
Interestingly, attention has two positive effects: it improves the performance of the network, and interprets the behavior of neural networks which are hard to understand [41]. Understanding the behavior of neural networks is difficult because the outputs are numeric elements that do not provide any means of interpretation by themselves. Attention helps this challenge by computing the weights and providing relevant information about the results of the neural network. Therefore, it is able to explain the ambiguity in the output of the neural network [32]. In general, the RNN encoder-decoder model with attention provides good results in speech recognition.

2.4.5 LAS Model

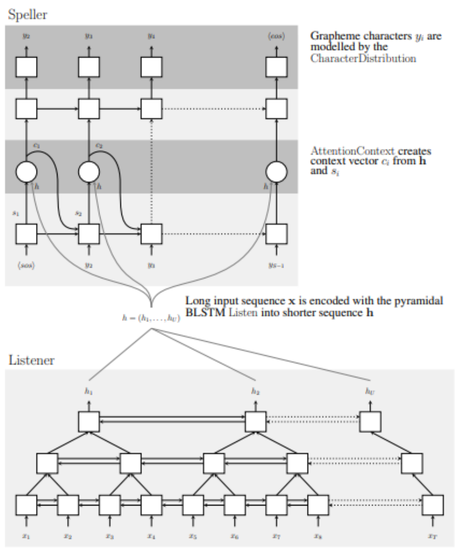
The LAS model is an attention-based encoder-decoder neural network which tries to learn how to create the transcript of an audio signal, one character at a time. There are also some other approaches such as connectionist temporal classification (CTC) [36] and sequence-to-sequence models with attention [7]. However, there are some limitations to using them. CTC is a neural network output and scoring algorithm, for training sequence-to-sequence models to tackle sequence problems where the timing is variable and the length of the input and output sequences are not equal [29]. Label outputs in CTC are assumed to be conditionally independent of each other. Moreover, in the sequence-to-sequence approach, only the phoneme sequences have been taken into account [18, 19].

Unlike previous state-of-the-art approaches that separate acoustic, pronunciation, and language modeling components and train them independently [83], the LAS model attempts to rectify this disjoint training issue by using the sequence-to-sequence attention-based encoder-decoder learning framework [18, 19, 89].

The LAS model includes an encoder as a listener and a decoder as an attender and speller. The encoder consists of two pyramidal BLSTM (pBLSTM) layers that convert low-level speech signals into higher-level features. The speller is an attention-based character decoder that utilizes these higher-level features and produces the output as a probability distribution over sequences of characters using the attention mechanism [18, 19]. The listener and the speller are trained jointly. Figure 2.7a demonstrates the three modules in the LAS model. Moreover, Figure 2.7b illustrates the pyramidal model in the LAS model with its two components. As it can be seen in the picture, in the decoder, the teacher forcing is also used. It means that the output of the previous layer is given to the current layer of the decoder as the input [100].



(a) Components of the LAS end-to-end model [16]


Speller
Grapheme characters y_i are modelled by the CharacterDistribution
AttentionContext creates context vector c_i from h and s_i
Long input sequence x is encoded with the pyramidal BLSTM Listen into shorter sequence h
$h = (h_1, \ldots, h_U)$
Listener

(b) The listener in the LAS model is a pyramidal BLSTM encoding the input *x* into high level features *h*, the speller is an attention-based decoder producing the *y* characters from *h* [15]

Figure 2.7: In figure (a) the LAS model's components are shown. In figure (b) the structure of the encoder-decoder is illustrated.

Structure of the LAS Encoder

The encoder uses a BLSTM [38] with a pyramid structure. This pyramidal structure reduces the length of *h* (a high-level representation of the input *x*). Due to the difficult process of the decoder extracting the relevant information from a large number of input time steps, a simple BLSTM converged very slowly. Therefore, using a pyramid BLSTM (pBLSTM) is helpful, since it reduces the temporal dimension of the decoder, which results in less computational complexity [15]. As a result, the attention model will be able to extract the relevant information in a smaller number of time steps. Moreover, based on the deep architecture of both the listener and speller, the model is able to learn nonlinear feature representations of the data.

Structure of the LAS Decoder

The second part of the model is the decoder. In the decoder, an attention-based LSTM transducer is used. At each step of the output, the transducer generates a probability distribution over the next characters conditioned on all the characters seen previously and the given encoder vector.

In this research, the log-Mel spectrogram is passed as the input through an encoder with three layers of pBLSTMs with a cell size of 256 on top of a BLSTM layer to yield a series of attention vectors. The attention vectors are fed into two layers of LSTMs with the cell dimension of 512, which yields the tokens for the transcript. At the end, the output text is evaluated using character error rate (CER) metrics. Figure 2.8 demonstrates the LAS model which has been implemented in this project. We used the exact structure as explained in [15].

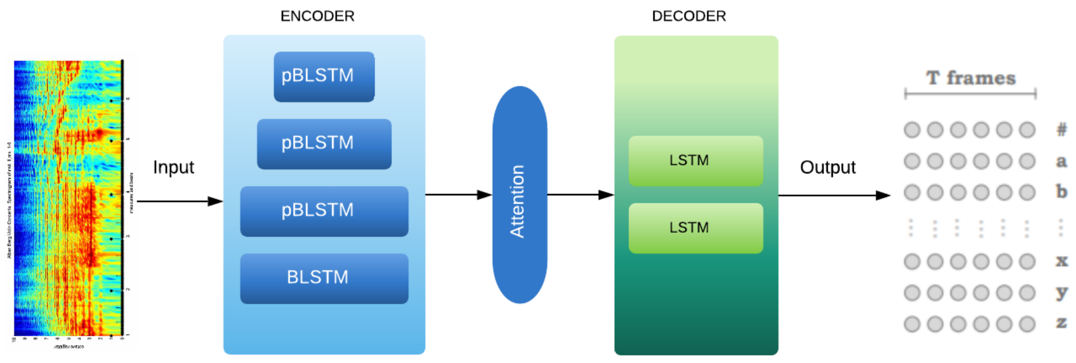


Figure 2.8: The detailed structure of the LAS model's pipeline from input to the output.

2.5 Components of the CNN-BLSTM Model

2.5.1 Convolution-based Models

Due to the promising reported results of DNNs, almost all state-of-the-art ASR models in the literature use a variation of DNNs [23, 24, 44, 62, 50, 103]. For instance, RNNs and more specifically LSTMs, are more recently being used in the state-of-the-art speech recognition systems [39, 83, 84]. Moreover, deep LSTM networks have shown better performance compared to the fully-connected feed-forward DNNs in speech recognition tasks [4, 14, 16, 70]. Apart from RNNs, using CNNs is also an option for sequence learning tasks, such as speech recognition [58]. The combination of RNNs and CNNs has also shown accurate results in speech recognition [74, 98].

Based on the study in [106], CNN-based end-to-end speech recognition frameworks have shown promising results on the TIMIT dataset. Based on this capacity of the model, in this research, we decide to use CNNs and see how the results will be in the speech recognition task on the Librispeech100 and the Dutch dataset. To efficiently scale our model, in this project different numbers of convolutional layers are added before the BLSTM layers to see how deeper CNNs will affect the accuracy of the speech recognition task [4, 14, 16, 70]. In [43], they also use a deeper CNN in a visual recognition task and benefit from it. Moreover, following the results of the previous research [39], to increase the accuracy we build a hierarchical architecture and added more hidden layers to the networks and made it deeper, rather than making each layer larger.

It has been found that for the feature extraction, the combination of LSTMs together with convolutional layers give a very good result. In [82] and [42], researchers could reach state-of-the-art results by using this combination of CNNs and LSTMs. Using CNNs in addition to RNNs resulted in better network architectures and proved to be efficient, especially to facilitate efficient training. Consequently, this architecture improves the key components of sequence-to-sequence models [77].

In this research, we followed the architecture which has been used in [4]. The model starts with one or more convolutional layers, after that one or more recurrent layers are added, and one fully connected layer will be the last layer of the network. Figure 2.9 demonstrates the architecture of this model.

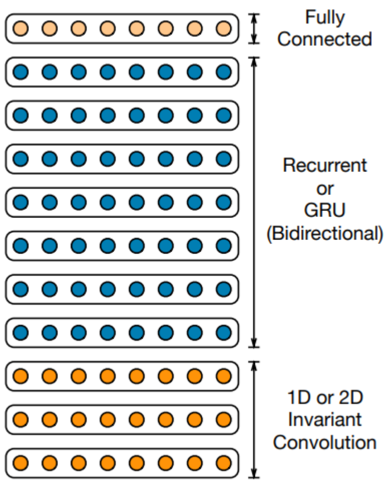

Fully Connected
Recurrent or GRU (Bidirectional)
1D or 2D Invariant Convolution

Figure 2.9: Architecture of the DS2 system used to train on both English and Dutch speech [4].

Studies reveal that using a stack of BLSTM and max-pooling layers in natural language inference yields state-of-the-art results [90, 22]. Therefore, we use a deep BLSTM structure and after each layer, we optionally add the max-pooling layer in the time dimension to reduce the input's feature-length.

Given a sequence of T words $(w_1...,w_T)$ as the input, the BLSTM layer gives a set of vectors $(h_1,...,h_T)$ as the output. Each $h_t \in (h_1,...,h_T)$ is the concatenation of a forward and backward LSTM layer in the BLSTM. The output dimension of the max-pooling layer is the same as h_t. For each dimension, max-pooling returns the maximum value over the hidden units $(h_1,...,h_T)$ [90].

2.5.2 Pooling

Pooling uses local summary statistics to summerize the output feature maps over a region. The summarization is done within a local neighborhood like in convolution [34]. Here are the common pooling operations:

- Max pooling: returns the maximum activation among rectangular neighbors.
- Average pooling: returns the average activation among rectangular neighbors.

Due to its regional summarization, usually fewer pooling operations are used, compared to convolution. Pooling is a technique for downsampling the input data. For instance, max-pooling with a window size of three with a stride of two can summarize the three activations of an input and its neighbors into one summary statistic. In fact, with a pooling with a stride of s, the feature map is downsampled with a factor of s.

Moreover, some advantages of pooling are that it is invariant to small translations and it does not need any parameters to be trained [34]. The only thing that is important in pooling are the values within its pooling window. This means that the location of the neighbors in the window has no effect on this process. Although small translations of the input may cause shifting in the activation in the feature maps, they may still fall within the same pooling window. This is very useful in speech recognition where matching the label to the sound wave could be tricky. Pooling also uses a parameter offset. Different input sizes can be managed by varying the offset of pooling windows, such that the resulting output will always be the same size before the classification layer.

In this research max-pooling in the time-dimension has been applied. In the structures with two layers of BLSTM, max-pooling has been applied between the two BLSTM layers as it was done in [15]. But when the number of BLSTM layers is more than two, max-pooling is positioned at the end of all the BLSTM layers and before the ReLU to prevent more reductions in the time-dimension.

2.5.3 Fully-Connected Layer

As traditionally seen in standard feed-forward Neural Networks, the final features from multiple convolutions, BLSTMS and pooling are flattened and fed to the fully-connected layers [34]. Convolutional layers work as feature extractors and the fully-connected layers map the features to classes. Both are trained at the same time with the same training procedure. The output of the fully-connected layer as the final layer is 30 scores (one for each letter in the English dictionary) plus the apostrophe, silence (#), and two special characters for the encoding of duplications (once or twice) of the previous letter known as special “repetition” graphemes.

2.6 Literature Review

Generally, deep learning led to an enhancement of speech recognition. Among all the various deep learning architectures, CNN-LSTM based and end-to-end models are dominant in the literature due to their promising results. End-to-end models combine the acoustic model (AM), language model (LM) and an acoustic-to-text alignment mechanism which leads to a simpler ASR system [15, 49]. In addition, the combination of CNN and LSTM layers also work well for the feature extraction [82].

Here, more details about the CNN-LSTM model and the LAS model as a well-known, state-of-the-art architecture among all the *encoder-attention-decoder* models will be discussed.

2.6.1 Research Utilizing the LAS Model

The authors of [61] applied the LAS model with sub-word units. They use an extended pre-training procedure where besides increasing the encoder depth, the hidden-layer dimension of the LSTMs is grown as well. The model starts with two layers of 512 dimensions in the encoder and increases to six layers with a dimension of size 1024. Moreover, the first pre-training step is done without dropout. Then they tune the curriculum learning schedule and improve upon that model. The researchers in [61] work on LibriSpeech100 and get the word error rate (WER) of 14.7% on the test-clean variant. However, on LibriSpeech 960, the performance is 4.8% WER in test-clean. In [105], an end-to-end, encoder-attention-decoder model is implemented in which a deep BLSTM network is the encoder and LSTM layers form the decoder. In the encoder, they use max-pooling after every layer in the time dimension in order to decrease the length of the encoder. Therefore, we also decided to use max-pooling in our CNN-BLSTM model structure. Moreover, the RWTH extensible training framework for universal recurrent neural networks (RETURNN) is used in [105] as the pre-training process. RETURNN is an optimized implementation for fast and reliable training of RNNs on multiple GPUs based on Theano/TensorFlow [27].

The model is tested on LibriSpeech 1000h and the reported performance is 4.87% WER.

In [15], the researchers use the LAS model on the clean Google voice search dataset and get 16.2% WER. The best result of 10.3% WER has been reached when both LM and sampling is used.

Paper	Dataset	Model	WER %
Luscher [61]	LibriSpeech100	Dynamic pre-training on LAS encoder	14.7
Chan [15]	Google voice search	LAS	16.2
Irie [47]	LibriSpeech100	CNN + LAS	12
D. Park [64]	LibriSpeech960	CNN + LAS	4.7
Zeyer [105]	LibriSpeech960	BLSTM+LSTM	4.8
Our Study	LibriSpeech100	LAS	

Table 2.1: Papers which use a variation of the LAS as their base model on the different datasets in speech recognition. The research of the pure LAS model on the LibriSpeech100 dataset has not been done before.

In this thesis research, the exact same architecture is used for the LAS model. Since the final goal of this project is to research Dutch speech recognition, the LAS model has been selected due to its very good reputation and promising results on different datasets. However, first, we need to find out how the model reacts to smaller datasets. By the time of writing this thesis project, the pure LAS model without any manipulation has not been applied on the LibriSpeech100 dataset.

Although very recently and by the time of running our experiments, we realized that researchers in [47], applied their model on the LibriSpeech100, they changed the LAS model slightly and it is no longer the pure LAS model. They add convolutional layers before the BLSTM layers in the encoder layer of the LAS model and get the result of 12% WER on test-clean. They also changed the configurations of the LAS component. Instead of the BLSTM in the encoder, they put LSTM layers with a various number of LSTM cells in each layer. Moreover, 16 GPUs were used during the training of their models. In another study in [64] researchers want to improve the accuracy of the LAS model. Therefore, they apply their model to different datasets. Yet again, neither the model nor the dataset is exactly the same as the one which is selected in this thesis project. They use the model of [47] and add a learning schedule to it. They reach 4.7% WER on LibriSpeech 960. The peak learning rate in the experiments is set to 0.001 and the batch size is 512. They utilize 32 Google Cloud TPU chips for running their experiments. Table 2.1 shows the list of papers which worked on the LAS model or a modification of the LAS model on different datasets. From this table, the originality of the work of this research can be found.

2.6.2 Research Utilizing the CNN, LSTM or a Combination of Them

The idea of using CNNs and the LSTM for speech recognition in this project is inspired by the following research which either uses the CNN or LSTM networks for speech recognition tasks separately or utilizes a combination of them.

A fully convolutional model has been introduced in [104]. All the processes from data preprocessing to AM and the LM are based on CNN. On the LibriSpeech 1000h they achieve 3.26% WER.

Researchers in [65] aim for having fast and accurate speech recognition on smartphones and embedded systems. They work with RNNs in the AM and LM. However, the LSTM appears not to be a viable option for them because it does not permit multi-time step parallelization as a key component of their model. They use the same network architecture as our CNN-BLSTM model which is basically inspired by the one in [4] but instead of LSTM, a simple recurrent unit (SRU) is used [65]. The accuracy of their model with LM on LibriSpeech100 is 26.10% WER.

According to the aforementioned research, it can be concluded that using the LSTM and CNN separately, have shown promising results in the speech recognition task. The followings

Paper	Dataset	Model	WER %
Zeghidour [104]	LibriSpeech960	CNN	3.26
J. Park [65]	**LibriSpeech100**	CNN+SRU+LM	26.1
Amodei [4]	Libri960 + Fisher+ WSJ + Baidu + Switchboard	**CNN + BLSTM**	5.33
The CAPIO [42]	LibriSpeech960	CNN + Dense LSTM	3.51
Our Study	LibriSpeech100	CNN+BLSTM	

Table 2.2: Papers that use models with the CNN, LSTM /BLSTM, or a combination of them on speech recognition. Our study picked the Librispeech100 dataset and the CNN-BLSTM model structure.

are research that work on the combination of CNN and LSTM and report high accuracy as the result of their study.

The combination of CNN and LSTM in [4] is applied on a training set consisting of 11,940 hours of labeled speech data containing eight million utterances from LibriSpeech 960, WSJ, Switchboard, Fisher and Baidu datasets. They test their model on LibriSpeech 960 and reported the WER of 5.33% on test-clean.

Finally, the CAPIO team [42] uses the concept of [4] but instead of a simple connection between LSTM layers, they use dense networks inspired by [46]. Their main focus is on the Switchboard which is a telephony conversational dataset. However, they also test their model on LibriSpeech 960 and they get the accuracy of 3.51% WER with LM on the test-clean.

In table 2.2 the summary of papers that use models with the CNN, unidirectional or bidirectional LSTM or a combination of these two has been listed. In this research, the LibriSpeech100 dataset and CNN-BLSTM model is selected.

2.7 Problem Statement

It can be seen in the aforementioned studies in the previous section that both the LAS model and the CNN-LSTM model have been reported to give state-of-the-art results in speech recognition on large datasets. In addition to utilizing large datasets, in order to get higher accuracy, scholars mostly either use some specific pre-training strategy or different model structures on the LAS and CNN-LSTM model components. Moreover, data augmentation or a particular data preprocessing method are used to enhance the models' performance. The goal of this study, however, is to investigate if it is possible to reach accuracy close to what is reported in the literature, with both models on a smaller dataset in order to assess the practicality and applicability of these models in real-life. Since companies usually have restricted resources at hand (e.g time and processing power), it is not always possible to train each model for a long time (e.g at least 14 days) or there might be no access to the powerful GPUs or TPUs. Therefore, in the proposed research, the plain LAS model as described in [15] and CNN-BLSTM model, are trained on LibriSpeech100 and the results are discussed in section 4.2.

The first selected model in this research is the LAS model which is reported to have the WER of 14.1% without LM on the Google voice search dataset [15]. The second model is inspired by [4] that uses the CNN-BLSTM model architecture on a very large dataset and achieves the accuracy of 5.33% WER. In addition to [4], researchers in [42] show that the combination of CNN and dense BLSTM layers on the LibriSpeech960 dataset can reach the WER of 3.51% on test-clean with LM. It can be seen that all of the models have been applied to very large datasets.

It is clear that there is a gap in the literature about the LAS and CNN-BLSTM models on a small dataset like LibriSpeech100. Yet, by this research, we want to cover this gap and explore the scalability of both models on smaller datasets.

2.8 Justification of Choises

Finding the state-of-the-art models and comparing them is not an easy task. Since each of the models that are reported to be the best, work on different datasets with a different architecture of networks in the models. Even those papers that report to work with a general *encoder-attention-decoder* model, use various modifications over the model structure. Moreover, even in the cases in which the models are relatively similar, the datasets are different. The frequently updated list of all the state-of-the-art models on speech recognition categorized by the dataset can be found in [1]. According to that list, by the time of writing this report, the most used models are *encoder-attention-decoder* and CNN based models. Therefore, we decided to pick the LAS and the CNN-BLSTM architectures to examine their accuracy on smaller datasets.

Table 2.3 demonstrates the list of all the papers which work on speech recognition with either *encoder-attention-decoder* or CNN- LSTM based models and the LibriSpeech dataset and reported the state-of-the-art results by the time of this research. Other architectures such as HMM-based models with transformers also show promising results [99]. However, they have not been used as much as the selected two models of this research.

Paper	Model	WER %	Publication Date
Amodei [4]	CNN + RNN	5.33 (960)	Dec 2015
The CAPIO [42]	CNN - Dense LSTM	3.51 (960)	Apr 2018
Zeyer [105]	Encoder-attention-decoder	4.87 (960)	Sep 2018
Sabour [77]	Reinforcement learning	4.5 (960)	Jan 2019
Zeghidour[104]	End-to-end CNN	3.26 (-)	Apr 2019
Irie [47]	CNN + Encoder-attention-decoder	12 (100)	Jul 2019
Luscher [61]	Encoder-attention-decoder	14.7 (100)	Jul 2019
Park [64]	CNN + Encoder-attention-decoder	2.8 (960)	Dec 2019

Table 2.3: The recent state-of-the-art researches on CNN-LSTM and encoder-attention-decoder models on speech recognition on the LibriSpeech dataset since 2015. The numbers in parenthesis in the WER column demonstrate if the results are from Librispeech960 or 100. In [104], the version of the Librispeech data set has not been mentioned.

The next step after selecting the models is the dataset selection which is important for having a fair comparison between the models. We decide to work on the LibriSpeech dataset which is a well-known non-telephony freely available dataset for the speech recognition tasks. After selecting the English dataset, we conduct a couple of experiments with both models to see which one is more accurate on the selected dataset. Then, we test both models on the Dutch Mozilla common voice dataset (a freely available open-source dataset). Moreover, in [75], the researchers mentioned that they also want to use the common voice Dutch dataset in the future which motivated us to pick this dataset. The details of the datasets are described in section 3.2.

[1]Adopted from https://github.com/syhw/wer_are_we

Chapter 3

Method

In this project, first, we want to compare two RNN based speech recognition systems and see which one is more accurate on the English dataset. Then we train both models on the Dutch dataset as well and will report their results based on the CER. For model selection, we use two state-of-the-art models. For the English dataset, the LibriSpeech dataset has been chosen [63]. LibriSpeech has different subsets that consist of 100, 360, 500 hours in addition to a complete version of LibriSpeech with 960h of speech.

We pick the 100h subset of LibriSpeech since with the resources we have at hand, especially the number and power of GPUs are limited. Another reason for this selection is that experimenting with smaller datasets allows us to examine the practicality and scalability of the models in the real world as well.

By the time of this research project, no experiment is done on the LibriSpeech100 dataset with the exact configuration of the LAS as mentioned in [15] and the CNN-BLSTM model defined in [4]. We first examine the accuracy of both models on LibriSpeech100 and then take the one with the best result for experiments on the Dutch dataset. The first architecture is the LAS [15] and the second model is a combination of different numbers of convolution and BLSTM layers and fully-connected neural networks inspired by [4] and [42]. To provide scientific research, the CRISP-DM methodology is followed.

Originally, the cross-industry process for data mining (CRISP-DM) methodology provides a structured approach to planning a data mining project. However, it has recently gained popularity and much attention among the AI community. It is a robust, practically effective, and flexible methodology. The CRISP-DM model is shown in figure 3.1.

The process is started with business understanding. Since this thesis is done as an internship in a financial company, the first step is to investigate to what extent a speech recognition system can add value to the company's current objectives and to understand what should be accomplished by this research from a business perspective. As a result of this stage, we can uncover that Dutch speech recognition should be the final goal of this research since more than 90% of the customers of the company are Dutch. In the project plan, we listed the steps in the project, together with their duration, required resources (e.g access to the server, access to GPUs, availability of data, etc.), inputs, and outputs.

In the second stage of the CRISP-DM process, we acquired the data. We searched for different English datasets for the speech recognition task. Among all the available data, we selected the LibriSpeech. First of all, it is the biggest freely available English dataset. Second, it has different subsets from which we could choose. We took the Librespeech100h based on the purpose of this research which is examining the scalability of state-of-the-art models on smaller datasets. For the Dutch dataset, we tried to get the data from the company to have a domain-specific dataset. However, the data was not labeled and it was not possible to use it in our work. Consequently, we worked with a Dutch dataset (Mozilla common voice) that is freely available on the internet. If we could have labeled data from the company, then we could train the model on 80% of that dataset, validate it on another 10% and test it on the last remaining 10%. Then, we could achieve a domain-specific Dutch speech recognition system. More explanations about the datasets will be given in section 3.2.

Based on the third stage of CRISP-DM and also as a first and fundamental step in speech recognition, we started data preprocessing. The details are given in chapter 2, section 2.3.

As the first step in modeling, we made decisions for the actual modeling technique that will be used in the research. For this, we did research on all the state-of-the-art models on the LibriSpeech dataset in speech recognition. These models are listed in table 2.3. As can be seen from the list, the models are either based on *encoder-attention-decoder* or CNN. Therefore, we decided to choose the LAS model as our first model and for the second model we chose CNN + BLSTM. After selecting the models, the next step is to implement them. We needed to specify the hyperparameters and their possible value options. Then we described the results of the models along with the interpretation of the results. According to the CRISP-DM method, in the next step, we assessed the models by summarizing the results of each experiment, listed the qualities of each experiment (e.g.in terms of accuracy) and ranked their quality in relationship with each other. As the last step of the modeling stage, according to the model assessment, we revised the parameter settings and tuned them for the next experiment. We iterated over the model building and model assessment steps until we found the best model configuration. All the documents of the revisions and assessments and also an elaboration on encountered difficulties during the training of the models are discussed in chapter 4.2.

The fifth stage of CRISP-DM is evaluation. During this step, we evaluate the final best results of both models among all the datasets in order to decide to what extent each model meets the company objectives.

In the final stage of the method, after we decided which model aligns better with the company's goal, we can plan for the deployment of that model. We have not reached this stage so far. We need to summarize the deployment strategy which consists of the necessary steps and

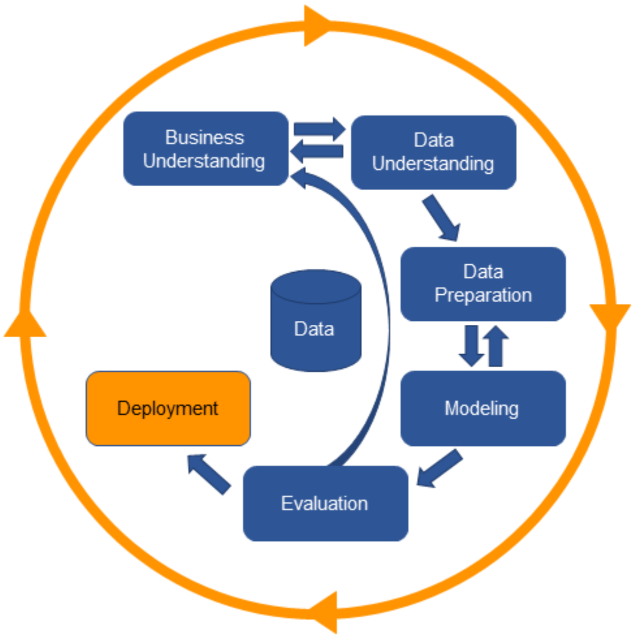


Figure 3.1: Deployment and management activities in CRISP-DM [91]

a description of how to perform them. After getting the final approval from the company we can start the deployment stage.

3.1 Research Pipeline

In this research, we try to find the answer to the research questions by taking five steps which are illustrated in figure 3.2. First, by investigating the literature, we will find two state-of-the-art deep learning architectures for speech recognition tasks. After that, we will find the available English and Dutch datasets for such tasks. The next step is to preprocess the data and after that, the preprocessed data will be fed into the implemented models and the experiments will start. By conducting rigorous experiments, we will try to find the best hyperparameters and network structure for both models in order to achieve the best results. After finding the best hyperparameters and structure we will train the models on the Dutch dataset and we will report the results. Finally, we will be able to determine the best model for the Dutch dataset.

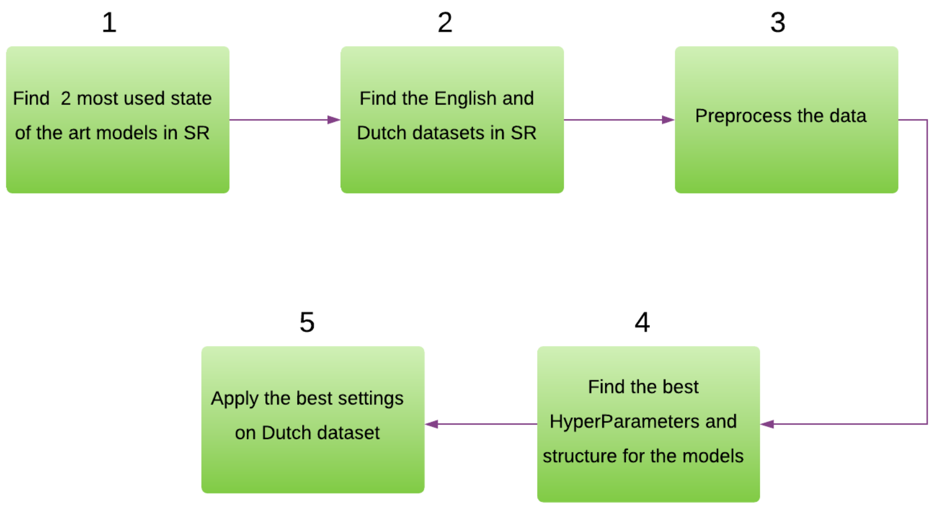

1
Find 2 most used state of the art models in SR
2
Find the English and Dutch datasets in SR
3
Preprocess the data
4
Find the best HyperParameters and structure for the models
5
Apply the best settings on Dutch dataset

Figure 3.2: The pipeline of steps from starting the research to finding the best results

3.2 Datasets

3.2.1 English Dataset

To find out which one of the two models gives better accuracy we need to compare them on same datasets. In this project, among all the freely available English datasets, the LibriSpeech corpus has been used [1]. There are some reasons why we used the LibriSpeech dataset. One of its advantages is that the acoustic models trained on LibriSpeech resulted in a lower error rate on the Wall Street Journal (WSJ) test sets than models trained on WSJ itself [63]. Moreover, WSJ is available to linguistic data consortium (LDC) members only; however, the LibriSpeech dataset can be freely acquired from the LibriSpeech corpus OpenSLR website. Another reason is that it consists of more hours of speech compared to other freely available datasets. Table 3.1 shows all the common publicly and privately available datasets which can be used for speech recognition or other speech activities [2]. Among various versions of LibriSpeech, which

[1]http://www.openslr.org/12
[2]Adopted from https://github.com/robmsmt/ASR_Audio_Data_Links

Name	Type	Size (Hours)	Access
LibriSpeech	Read	960	Free
TED-LIUM	Read	118	Free
Voxforge English	Read	130	Free
Common Voice v1	Read	500	Free
Tatoeba Audio Eng	Read	200	Free
Fisher	Conversational	2000	Paid - LDC
Switchboard Hub 500	Conversational	240	Paid - LDC
TIMIT	Read	5	Paid - LDC
Wall Street Journal (WSJ)	Read	80	Paid - LDC

Table 3.1: A list of English datasets for speech recognition.

consists of 100-360-400 and 960 hours, we trained our models on the LibriSpeech 100h train-clean. The validation set is dev-clean when testing on test-clean. The complete version of LibriSpeech contains 960 hours with separate train, dev and test sets. Each dataset contains splits for clean and noisy of read speech, and comes with official textual data as the labels to train the networks.

LibriSpeech is a project of the LibriVox and has been created from audiobooks, thus it is a dataset in which only one person speaks at a time. Although the main dataset in this research is the LibriSpeech100, to conduct fast training experiments we used some smaller subsets of the LibriSpeech. Because finding the best configuration of the network and the best hyper-parameters on a large dataset needs a lot of time and we need to train the network over and over again to see the effect of each hyper-parameter on the network. Consequently, we worked on the smaller subsets of the LibriSpeech100; "micro-Libri", "chopped-Libri", and "small-Libri" datasets. The "small-Libri" is a verified smaller version of the LibriSpeech 100 which contains 1519 training samples and is also available on the openslr website. Following is the description of the "micro-Libri" and "chopped-Libri" datasets which we made as two subsets of LibriSpeech100, in order to cover different requirements of the experiments in this research.

In general, in the original LibriSpeech dataset, the length of the samples is not similar. However, since we need to have a fixed input dimension for the networks, we applied the zero-padding strategy. By using zero-padding, we get the length of the longest sample and add zeros to any other shorter samples until it reaches the length of the longest one. As a result, this might affect the network's learning procedure due to the huge number of zeros that can be fed to the network. By creating a subset of the LibriSpeech dataset with samples of a similar length, we could avoid this phenomenon. Therefore, apart from the verified LibriSpeech100 and the small-Libri dataset, we also created another subset of LibriSpeech, and we named it as "micro-Libri". In this subset, we filtered out the samples which have a similar length (e.g 350-360 characters in each text). We named this dataset as "micro-Libri".

The "chopped-Libri" is the LibriSpeech100 but we chopped all the samples every 20 characters. All the samples which have the number of characters less than 20, remained as they are. Then we calculated the proportion of 20 on each sample and cut the spectrogram of that sample with the same proportion rate. For instance, if a sample contains 100 characters, it has five portions of 20 characters in each. Therefore, we chopped the spectrogram of this sample into five portions as well. Although the characters and the sounds might not match very precisely, this was the best way we found to cut the long sentences in the LibriSpeech dataset in which the samples are on average 15 seconds.

However, the final models will be trained on the LibriSpeech Train-clean-100 [63].

Name	Type	Size (Hours)	Access
Spoken Wikipedia corpora	Read	224	Free
Corpus Gesproken Nederlands	Read	900	Free
Mozilla commonvoice	Read	23	Free

Table 3.2: A list of Dutch datasets on speech recognition task.

3.2.2 Dutch Dataset

To achieve Dutch speech recognition, both models have also been applied to a Dutch dataset. "Spoken Wikipedia corpora" (SWC), "spoken Dutch corpus" (CGN), and "Mozilla common voice" are the three verified Dutch datasets for speech recognition tasks. Table 3.2 lists the Dutch datasets in speech recognition tasks. The CGN dataset is the most complete one which consists of 900 hours of contemporary Dutch speech. However, training the models on such a large dataset requires a very long time and it was in conflict with the goal of this research which is examining the practicality of the models on small datasets. Therefore, the common voice Mozilla dataset has been selected for this research. The common voice is an open-source corpus created by Mozilla[3] and contains 23 hours of validated data.

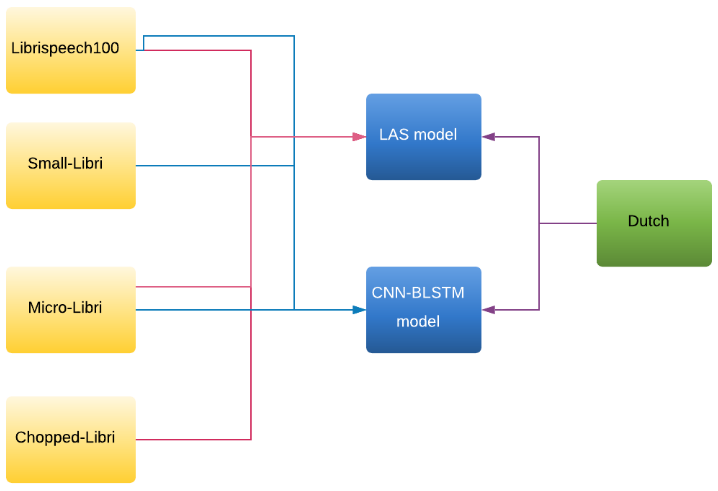


Figure 3.3: All the datasets that have been applied on two models in this research. The details of each dataset are described in section 3.2.

These English and Dutch datasets are chosen in order to study the impact of the different components of the two selected models at different scales of training data.

All the datasets which have been used in this research have been illustrated in figure 3.3.

3.3 Implementation Details

In this section, the implementation details that are used in this thesis research will be explained. As shown in chapter 2, after selecting the datasets and models, the first step of the whole pipeline in speech recognition for both of the models is data preprocessing. In this step, 40-dimensional log Mel-frequency filter bank features are computed every 25 milliseconds (ms) from the raw speech data. The window size of 25 ms is used because most of the non-silence

[3]https://voice.mozilla.org/de

phonemes in speech are shorter than 20 ms, thus, using a window size larger than 25 in feature extraction will not result in more improvements [88]. The output of the log-Mel filter bank will be the input for the LAS and CNN-BLSTM models.

In the CNN-BLSTM model, the input of the convolutional layer consists of feature maps with the time steps and frequency axes. Each feature map is formed by the Mel-filter bank output. The filter size of the 1-D convolutional layers is equal to five, as proposed in [4, 81]. Using convolutional layers not only improves the performance of recognition but also it can down-sample the input frames and reduce the computational complexity in the next layers. However, in this research the stride of one is used in CNN layers and down-sampeling is done by max-pooling.

Furthermore, a widely used frame-wise discriminative training criterion in speech recognition training is cross-entropy (CE) which aims to minimize the expected frame error [92]. In the CNN-BLSTM model, the RNN is trained with CE loss function which consists of a softmax over 30 dimensional outputs plus the negative log-likelihood. However, the LAS model showed better behavior with the negative log-likelihood as the training criterion.

We will now give more explanation on details of the experiments. The dropout equals to 0.5 and is used to every output of the recurrent layer as a regularization according to [31] and the initial learning rate is equal to 0.0001. Among different optimizers that have been used in this experimental research such as SGD, SGD with momentum and Adam optimizer [51], Adam gives better results. Therefore, we utilized a gradient descent optimization algorithm based on Adam for all of our models, which is pre-implemented in PyTorch. Data normalization has a significant effect on the accuracy of the models. Different batch sizes (1, 4, 8, 16, 64) have also been tried. However, for the RNNs, using a smaller number of batch sizes (e.g batch size equal to one) leads to better results in the accuracy of the models. Moreover, teacher Forcing is used in the LAS model to have the ground truth label as an additional input for our network.

We implemented both models using the *Pytorch* library. It offers highly optimized kernels for convolutional layers and very good initialization for LSTM layers. The output is a softmax layer computing a probability distribution over characters. Furthermore, label smoothing has not been applied in the final experiments, due to the instability observed in the training of primary experiments.

According to the letter vocabulary in English which contains 26 graphemes, the output size of our networks has been set to 30 and it is based on graphemes (characters). We used 26 standard English alphabets, one for the apostrophe, one for silence (#), and two for the encoding of the duplication (once or twice) of the previous letter known as special "repetition" graphemes [58].

Moreover, to have a fair comparison between the two models, we train the two models with identical hyperparameter settings, on the same dataset with the same preprocessing model. Therefore, we implement the CNN-BLSTM model from A to Z since it is not possible to use the available codes on the Github on this model due to the fact that there are incompatibility problems between the available codes on Github with our preprocessing steps, python's version and also the PyTorch framework which we use for the implementation of this research project

All the experiments of this research are conducted on four parallel GPUs to accelerate the training process. After finding the best settings of the hyperparameters and the best architecture for the CNN-BLSTM model, the training pipeline binds one process to one GPU in a high-performance computing (HPC) system provided by the company.

3.4 Evaluation Metrics

The models in this research are evaluated based on the character error rate (CER), because characters are the most generic and simple output units for generating texts [105]. Traditionally, *attention-based encoder-decoder* models are trained to output character-based (graphemes) units, in which the model is allowed to map the input audio features directly to the output word sequence, without using a hand-crafted pronunciation lexicon (PL). This is the reason why we call them end-to-end. Thus, it can also solve the out-of-vocabulary (OOV) problem [15]. Moreover, when a speech recognition model utilizes character-based output units, it means that they learn the AM, pronunciation model (PM), and LM within a single neural network [47]. In [77, 92] they also used CER to report the accuracy of their model for the speech recognition tasks.

In Addition, using phonemes has some downsides. For instance, having an additional PL and LM is required. Hence, it has been found that there will be no improvement over graphemes based on the experiments in [80]

Although it is also common to use words as output units due to the OOV problem, in this research, the word-level language model (WLM) is not used. Because it consumes a large number of parameters in the output Softmax layer. This makes the computation more expensive [105]. Moreover, WLM demands much more training data because it needs a huge number of labels. WLM needs a vocabulary size which is at least ten times larger than that of the character level [65]. Since in all of the experiments in this research, we used CER, therefore, our system is open-vocabulary. CER is derived from the Edit-Distance, also called Levenshtein distance [4]. It simply calculates the least number of edit operations important to change one string to get another as shown in equation 3.1.

$$CER = \frac{(i+s+d)}{n} \tag{3.1}$$

Here i represents the number of required insertions, d denotes the number of required deletions, s is the number of substitutions required to transform the system output to the desirable output. And n is the total number of characters in the target output [71].

To calculate the CER, first, we need a "correct" transcription baseline (usually human-made). Then the number of substitutions, insertions, and deletions will be counted compared to the baseline. Substitutions are usually words that have been misheard or misspelled. They are considered as wrong words. Having too few words is known as deletion mistake and having too many words is called insertion mistake [96].

3.5 Hardware Setup

In this project, the software has been built based on PyTorch and numpy. The experiments were performed on a server provided by the company in which this research has been done as an internship with four NVIDIA Tesla V100 GPUs for approximately 12 days.

The datasets were stored on the same server with 1.5 TB of memory to allow for a faster data stream during training. The server used an Intel Xeon Gold 6148 as a CPU. Utilizing HPC techniques leads to fast training of the networks. As a result of this efficiency, we were able to run the experiments in a couple of days that could take weeks previously. However, for the final experiments, the models have been trained for approximately one month until the networks converged. As a result of speeding up the calculation with these dedicated GPUs, we were able to iterate the networks more quickly to identify superior architectures and algorithms.

[4] http://www.levenshtein.net/index.html

Even using a simple GPU in the training resulted in remarkable improvement in performance [69]. Consequently, acquiring two or more GPUs scales the performance gain linearly [52, 20]. In this project, the model implementation for speech recognition utilized the previous model-parallelism structures [20].

Chapter 4

Experiments and Results

In this chapter, the experiments and their results will be discussed. First, some experiments are conducted to find the best hyperparameters and settings for the LAS model. To overcome the time limitation and get to the results faster, we ran these experiments with less iterations (e.g. around 5000) on smaller datasets such as small-Libri or micro-Libri. After that, based on the results of those experiments we run the final experiments. To maximize the performance we trained both models on the larger English dataset (LibriSpeech100). The final experiments are trained for a longer schedule (approx. 30 days). In the end, the best model will be trained on the Dutch Mozilla common voice dataset.

4.1 Experiments

In the following tables, the configuration of different experiments which have been done during this research on the LAS and the CNN-BLSTM models have been listed. The structure of the LAS model remains as it has been defined in [15]. Therefore, for the LAS model we only focus on finding the best hyperparameters by conducting some experiments as mentioned in table 4.1 and 4.2. We decide to examine the selected hyperparameters based on what is suggested in the literature for the deep LSTM networks [72]. However, in the CNN-BLSTM model, we also need to find the best structure of the network. Therefore, for the CNN-BLSTM model, we use the results of the experiments on the LAS model as the best settings for hyperparameters and concentrate on finding the best architecture for the CNN-BLSTM model by conducting the experiments which have been listed in table 4.3, 4.4 and 4.5.

The results of the first three experiments in table 4.1 show that among the three gradient based optimizers, the Adam optimizer performs best. Then with this conclusion we tested the normalization. The fourth experiment shows that normalization has a huge effect on the

Exp	Opt	BS	Normalized	CER %
1	Adam	4	No	0.71
2	SGD	4	No	0.86
3	Momentom 0.01	4	No	0.84
4	Adam	4	Yes	0.56
5	**Adam**	**1**	**Yes**	**0.32**

Table 4.1: Experiments which have been done on the LAS model to find the best hyperparameters on the LibriSpeech100 dataset with the learning rate of 0.0001. Opt denotes optimizer, BS = batch size, and Normalized denotes data normalization. The results are reported in character error rate (CER).

Exp	Opt	BS	DS	Normalized	CER %
1	**Adam**	**1**	**LibriSpeech100**	**Yes**	**0.32**
2	Adam	1	Chopped-Libri	Yes	0.79
3	Adam	1	micro-Libri	Yes	0.45
4	Adam	1	Dutch	Yes	0.47

Table 4.2: Experiments which have been done on the LAS model to study the effect of the different sizes of datasets on the accuracy. Opt denotes optimizer, BS = batch size, DS means Dataset, and Normalized denotes data normalization. The learning rate is equal to 0.0001.

Exp	Max Pool	ReLU	FC	BS	# BLSTMs	Init.	CER %
1	No	No	No	16	1	random	0.85
2	Yes	No	No	16	3	random	0.85
3	Yes	No	No	64	3	random	0.85
4	**Yes**	**No**	**No**	**16**	**5**	**random**	**0.83**
5	Yes	No	No	64	5	random	0.9
6	Yes	No	No	16	7	random	0.85

Table 4.3: Experiments which have been done on the BLSTM model on the small-Libri dataset. The reported CER is for the dev set. Max-pooling is used on top of the BLSTM layers. FC denotes the fully connected layer and # BLSTMs shows the number of BLSTM layers. No CNN layers used in these experiments. Init. represent weight initialization which prevents exploding or vanishing of the layer activation outputs. In this thesis random and Kaiming initializaion is used.

accuracy of the model. By the last experiment of table 4.1, it is concluded that using a batch size of one is influential on the accuracy of the model.

In table 4.2, when we figured out the best configuration of the hyper parameters, we applied the model on different datasets. The effect of the size of the dataset can be seen from the results of this table.

Considering the results of the experiments on the LAS model, we find the best setting for the hyperparameters. Therefore, when we start to implement the CNN-BLSTM model, we use those best hyperparameters and the focus for the CNN-BLSTM model is to find the best structure of the networks.

From the experiments in table 4.3, it can be deduced that using five layers of BLSTM performs best. It also shows that using smaller batch sizes leads to better results. Thus, in the final experiments we will use five BLSTM layers and will try smaller batch sizes.

In the first experiment of table 4.4, we wanted to see the effect of adding one CNN layer to one BLSTM layer with the Kaiming initialization. In the second experiment we changed the

Exp	Max Pool	ReLU	FC	# CNNs	BS	# BLSTMs	Init.	CER %
1	Yes	Yes	Yes	1	16	1	Kaiming	0.83
2	Yes	Yes	Yes	3	16	1	Kaiming	0.83
3	**Yes**	**Yes**	**Yes**	**3**	**16**	**5**	**Kaiming**	**0.81**

Table 4.4: Experiments which have been done on the CNN-BLSTM model on the small-Libri dataset. The reported CER is for the dev set. ReLU has been used after the BLSTM layers on top of the Max-pooling and the fully-connected is the final layer. In the BLSTM layers, the hidden dimension of each direction is equal to 256. FC denotes the fully connected layer with the dimension of 30, # CNNs shows the number of CNN layers and # BLSTMs shows the number of BLSTM layers used in the architecture and Init. represent data initialization.

Exp	Max Pool	ReLU	FC	# CNNs	BS	# BLSTMs	Init.	Dataset	CER %
1	No	No	No	0	16	1	random	small-libri	0.85
2	Yes	Yes	Yes	0	1	1	Kaiming	small-libri	0.80
3	Yes	Yes	Yes	2	1	1	Kaiming	micro-libri	0.78
4	**Yes**	**Yes**	**Yes**	**2**	**1**	**2**	Kaiming	**micro-libri**	**0.74**
5	Yes	Yes	Yes	2	1	2	Kaiming	LibriSpeech100	0.98
6	Yes	Yes	Yes	2	1	2	Kaiming	Dutch	0.80
7	Yes	Yes	Yes	3	1	5	Kaiming	LibriSpeech100	0.83
8	Yes	Yes	Yes	3	1	5	Kaiming	Dutch	0.76

Table 4.5: Experiments which have been done on the CNN-BLSTM model on different dataset. According to the results of previous experiments the initialization is Kaiming. ReLU has been used after the BLSTM layers on top of the Max-pooling and the fully-connected is the final layer. FC denotes the fully connected layer, # CNNs shows the number of CNN layers and # BLSTMs shows the number of BLSTM layers used in the architecture.

Experiment	Model	Dataset	CER %
1	LAS	LibriSpeech100	0.32
2	LAS	Dutch	0.47
3	CNN-BLSTM	Dutch	0.76
4	CNN-BLSTM	LibriSpeech100	0.83

Table 4.6: The summary of the best results of all the experiments which have been done with both models on the Dutch and LibriSpeech100 datasets. The Batch size is set to 1.

number of CNN layers to three and it can be seen that with only one layer of BLSTM the result is not improved. Moreover, based on the results of the experiments in table 4.3, we concluded that using five BLSTM layers performs better. Therefore, for the last experiment of table 4.4, five BLSTM layers on top of the three CNN layers are used and the accuracy of the model increased.

From table 4.5 it can be seen that the result of the 2CNN-2BLSTM architecture on the LibriSpeech100 dataset is not as good as the one on the micro-libri dataset. This can be explained by the effect of the number of zeros which are fed to the network, by using the LibriSpeech100. Since the length of the samples in the complete LibriSpeech100 are different, to unify the input size, zeros are added to the shorter samples so that all the inputs become equal in length. Therefore, because of the huge number of zeroes in the input, CNN-BLSTM model can not learn very well, which shows the sensitivity of the CNN-BLSTM model to the structure of the dataset. Moreover, it shows that the because of the simple structure of the CNN-BLSTM model it can learn simple datasets like micro-libri better than the bigger LibriSpeech100.

The best results from all the experiments on the main English and Dutch datasets are reported in table 4.6. As a result, the sensitivity of the models to the size of the dataset will be specified. Moreover, by comparing the results of the two models, we will find out which of these two models provide more accurate transcription from the sound signals on the English and the Dutch datasets. Furthermore, we can measure the accuracy of models on the Dutch and English datasets separately and see if they keep the same quality in both datasets. In the tables, we reported test set results, validated by the dev-clean set.

After conducting all these 23 experiments, the best hyperparameters and the model structure was known.

4.2 Results

4.2.1 Results of the LAS Model

Given the results of the experiments in table 4.1, in the first three experiments all the other hyperparameters were fixed except the optimizer. As a result, applying the Adam optimizer shows a better performance than SGD and SGD with momentum. After finding the best optimizer, the next step is to find the effect of data normalization. Comparing experiment number 1 and 4 gives the answer that normalization has a significant effect on the accuracy of the LAS model.

Furthermore, a smaller learning rate on the chopped-Libri dataset was also examined but it was not effective, therefore, we used the learning rate equal to 0.0001 in all the experiments. We also tried differnet batch sizes and the batch size equal to one gives the best results. As a conclusion, the Adam optimizer, with a learning rate of 0.0001 and batch size equal to one is the best setting for the hyperparameters.

LAS on Dutch

Figure 4.1 illustrates the Character Error Rate (CER) and loss for the LAS model on the Dutch dataset. The batch size is equal to one, and the data has been normalized. The learning rate is 0.0001 and the Adam optimizer has been used. As it can be seen, both CER and loss converged so we did not continue the experiment any longer. The best result is 47% CER. In table 4.7 some output of the system has been demonstrated. The errors are more related to the names which show that adding an LM will help to improve the results.

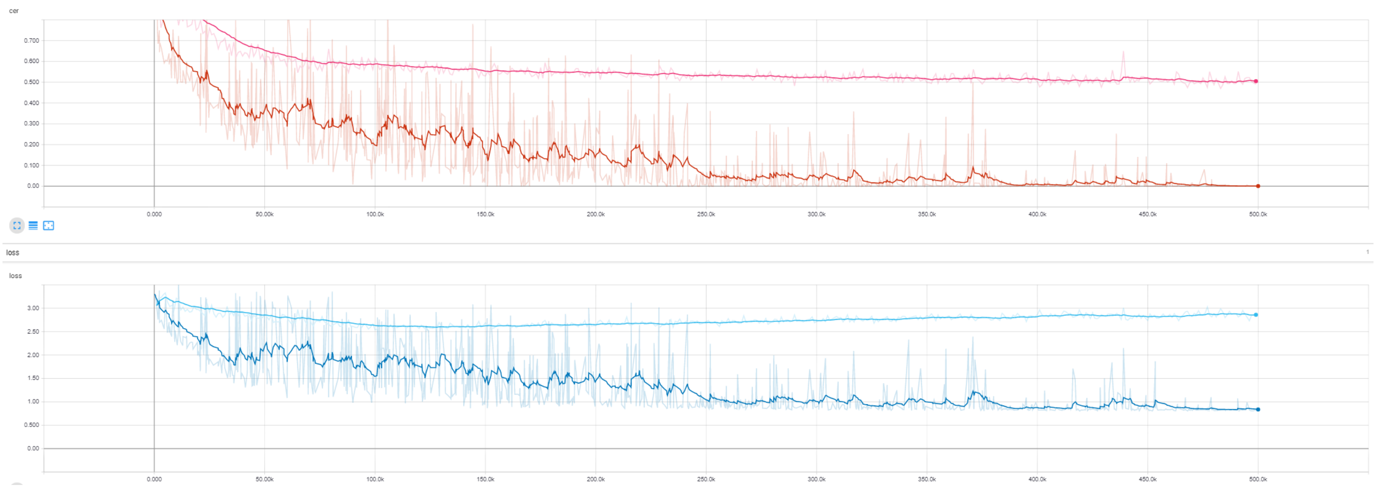


Figure 4.1: Character Error Rate (CER) and Loss of the LAS model on the Dutch dataset. The x axis is the number of iterations and the y axis is CER/Loss rate. Pink is the validation CER, whereas red describes the train CER. In the Loss graph, light blue represents the validation loss and dark blue shows the training loss.

LAS on LibriSpeech100

Figure 4.2 demonstrates the result of experiment number 1 from table 4.6 which is the final experiment with the LAS model on LibriSpeech100. Like the final configuration of the LAS model on the Dutch dataset, here also the batch size is equal to one, and the data has been normalized. The learning rate is 0.0001 and the Adam optimizer has been used. The iteration numbers of the experiment was first set to 500K steps, but since the model did not converge

Target	Waar ben ik?
Output	Waar gan je niet?
Target	Tot ziens
Output	Dat zients
Target	Ken jij Jonas?
Output	Ken jij een has?

Table 4.7: The target and the output of the LAS model on the Dutch dataset.

Target	TO PICKLE EGGS
Output	TO PICKLE EGGS
Target	SHILOH
Output	SHALLOW
Target	THE TWENTIES
Output	THE TWENTIES
Target	OTTO WINKED AT ME
Output	I DON'T WINK IT MEAN

Table 4.8: The target and the output of the LAS model on the LibriSpeech100 dataset.

at 500K steps, we continued the experiment until 1 million iterations. It took 1 month until it reached 1M iterations. The output of the network can be seen in table 4.8. At best, the system reached 32% CER.

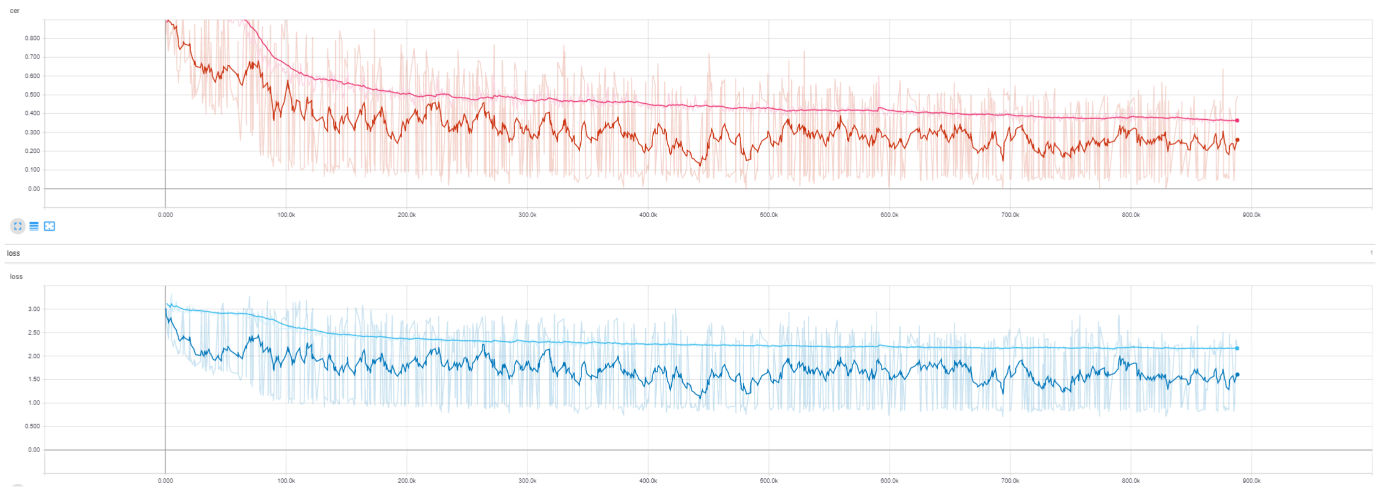


Figure 4.2: CER and Loss of the LAS model on the LibriSpeech100 dataset. Light blue is the validation loss, whereas dark blue describes the train loss. Pink is the validation CER, and red describes the train CER. The x axis is the number of iterations and the y axis is CER/Loss rate.

4.2.2 Results of the CNN-BLSTM Model

In this section the results of the experiments with the CNN-BLSTM model will be reported according to tables 4.3, 4.4, and 4.5.

CNN-BLSTM on LibriSpeech100 and its subsets

In the experiments below, the batch size of one and the Kaiming initialization is used and maxpooling, ReLU and a fully connected layer with 30 hidden units are applied on top of the

BLSTM layers.

Figure 4.3 shows the graph of the 3CNN-5BLSTM model on the LibriSpeech100 dataset. The result is from running the experiment after 17 days of training. Comparing the result of this experiment to the one with 2CNN and 2 BLSTM in the same iteration step, shows that the 3CNN-5BLSTM structure is more accurate than the 2CNN-2BLSTM.

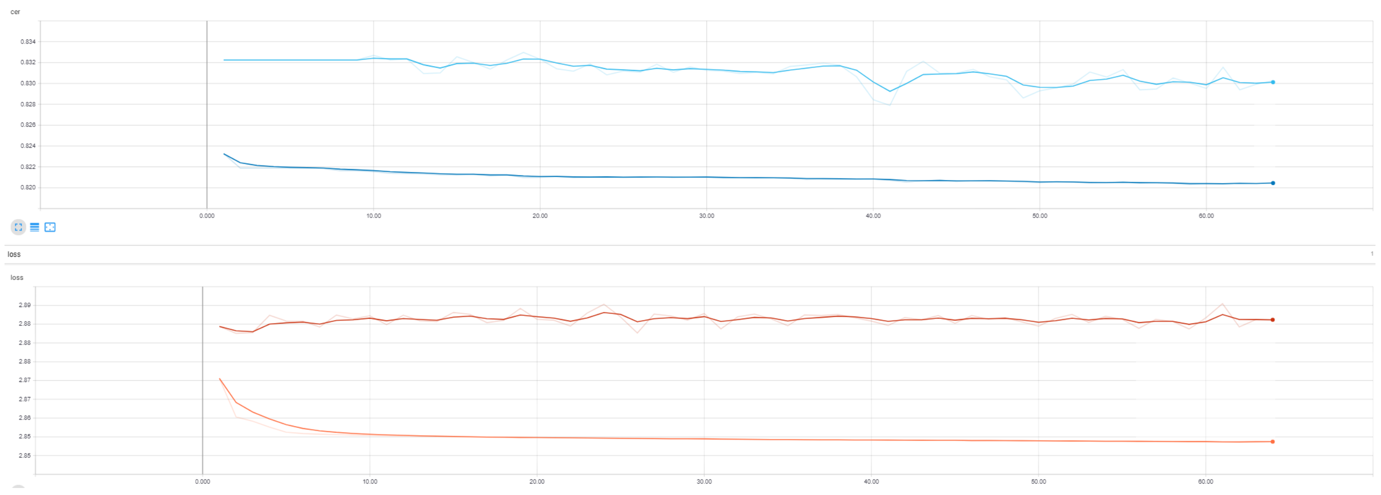


Figure 4.3: CER and loss on the LibriSpeech100 dataset with 3CNN and 5BLSTM layers with the batch size of 1. Light blue is the validation CER and dark blue describes the train CER. Red is the validation loss, and orange describes the train loss. The x axis is the number of iterations and the y axis is CER/Loss rate. At the best, it reached to the 0.83% CER.

CNN-BLSTM on Dutch

The following graphs demonstrate the results of the experiments of the CNN-BLSTM model on the Dutch dataset.

The results of the 3CNN-5LSTM on the Dutch dataset can be found in figure 4.4.

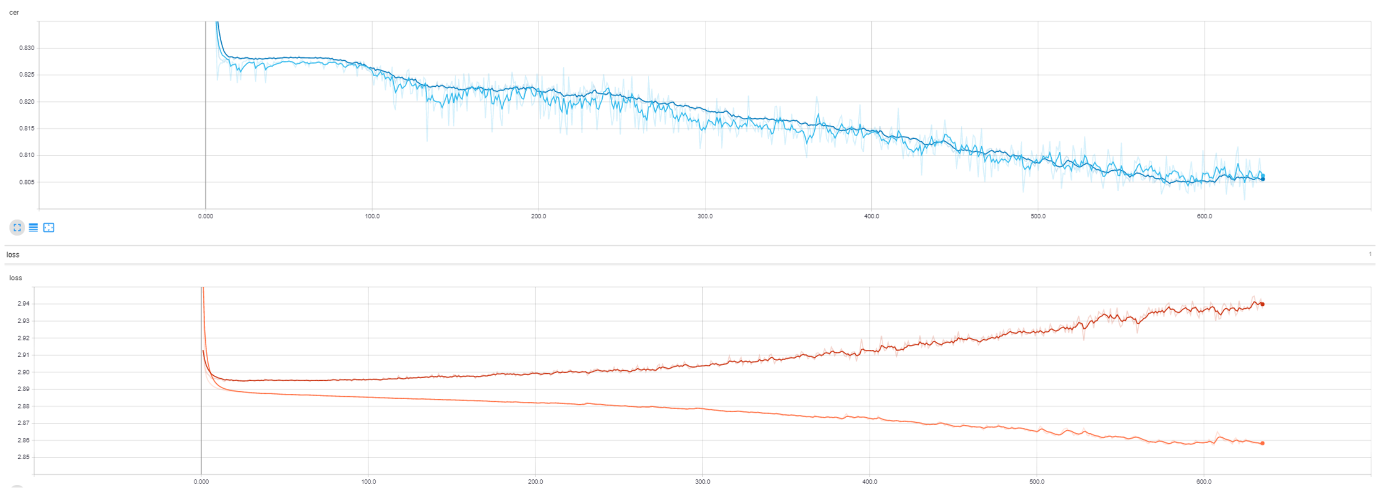


Figure 4.4: CER and loss on the Dutch dataset with 3CNN and 5BLSTM layers with the batch size of 1. Light blue is the validation CER and dark blue describes the train CER. Red is the validation loss, and orange describes the train loss. The x axis is the number of iterations and the y axis is CER/Loss rate. At the best, it reached to the 0.76% CER.

4.3 Model Comparisons

All the performances have been reported by evaluating the checkpoint with best dev-clean performance. It can be deduced from the results that although we used the same datasets with the same data preprocessing on both models to make sure that the comparison is fair, the LAS model showed better results by a large margin on all the English and Dutch datasets even without any language models. It shows the significant power of the end-to-end attention-based encoder-decoder models over simple LSTM-based models.

Chapter 5

Discussion and Conclusion

In this chapter, we first discuss the obstacles and the reasons behind the differences between the results of this research compared to other research reported in the literature are explained. Then we will answer the previously proposed questions through the knowledge gained with the experiments. At the end, potential future works for getting better results and also applying this system as a use case in the sponsored company are discussed.

5.1 Discussion

After all, in spite of all the experiments we ran in this research, the results are not as accurate as those claimed in the literature. Here we will explain the possible reasons for this performance difference.

During this research project, we worked on two LSTM-based models for the speech recognition task and the important features that have been noticed are as follows:

1. The size of the dataset has a big influence on the whole training process. All the other research on the LAS model with the same structure as we used in this project, worked on much bigger datasets than what we used. Librispeech 960h, Baidu dataset, and Google voice search dataset are some examples which are at least 10 times bigger than LibriSpeech100. In addition to a bigger dataset, in some of the research data augmentation has also been applied which adds to the amount of training data. We found out that the LAS model is strongly sensitive to the size of the training dataset and with smaller datasets, the accuracy may be affected significantly. This finding questions the practicality of the LAS model in real-life.

2. The length of the input sequence affects training speed time. Training time is slower when the input data is longer due to the iterative multiplications over time. In the LibriSpeech the utterances are usually about 15 seconds long which according to the literature this is considered as a long sample.

Despite various solutions to these issues like parallelizing data across multiple GPUs [89] and precise network initializations [54], RNN-based models still suffer from intensive computational procedures and a demanding training process.

On the other hand, there are some studies which work on the LibriSpeech100 but in order to get an acceptable result, they either change the LAS model architecture or use some pre-training strategies. For instance, in some papers like [47, 64] a CNN is added before the encoder in the LAS Model. However, in this research, the goal was to find out the accuracy of the pure LAS model as defined in [15] on smaller datasets to examine the practicality and scalability of the LAS model on different sizes of datasets.

Furthermore, following the results of [104] and [64], different preprocessing procedures for extracting the features in the network may affect the accuracy of speech recognition systems with challenging recording conditions.

As also mentioned in [47], for successfully training the LAS model, reducing the input frame rate in the encoder is important, especially for the LibriSpeech dataset in which the utterances are mostly long (approx. 15s). Therefore, we assume that having an encoder with two layers of convolution will result in a total time reduction factor of four and might give a better result than the basic LAS with a time reduction factor of two [15]. So far we realized that adding CNN to the BLSTM layer helps with the accuracy. Therefore, it might also be efficient in the LAS model as well. The authors of [64, 47] have used CNNs before the actual LAS model and reported a promising result. They also use different sizes of the LAS model components on Librispeech 960 and report better results. Consequently, we can also conclude that attention-based models are very sensitive to the amount of training data and the model configuration. It can be deduced from the results of our experiments that the performance of the attention-based models dramatically decreases when the amount of the training data is reduced and no pre-training or data augmentation strategy is used.

5.1.1 Resource Limitations

The resource constraints were the main limitation of this master thesis. On the one hand, the available computational resources (e.g. GPU, RAM) limited the speed of the experiments. On the other hand, the time constraint limited the total run-time of the iterations of the experiments.

The RNN implementation has the problem of exceeding the cache memory. Although there are some solutions such as multi-time step parallelization, due to their complex input-output dependencies, LSTM networks do not permit these types of parallelization. As a result, we face a high memory consumption with LSTMs. This issue is the main reason for multiple forced stops during the run-time of the training process so that the experiment could be re-run after the memory was free.

5.2 Conclusion

5.2.1 Answers to the Supportive Research Questions

What are the two most used, state-of-the-art deep learning architectures in speech recognition tasks?

By conducting a rigorous literature review, it is realized that the *encoder-attention-decoder* models and the models with the combination of *CNN and LSTM networks* are the most used architectures among all the other state-of-the-art architectures.

What are the freely available datasets in English and Dutch languages for the speech recognition task?

As mentioned in chapter 3, section 3.2, the LibriSpeech dataset is the largest freely available data set in English. This dataset has been made specifically for speech recognition and many studies in this field have been done with the LibreSpeech dataset. This dataset contains subsets with different numbers of hours of speech (100, 360, 500, and 960). For the Dutch dataset, the Mozilla dataset from the common voice project is selected. The other two datasets, SWC and

CGN contain 224 and 900 hours of speech, respectively. However, the common voice Mozilla dataset is chosen in order to examine the models on a small dataset. Moreover, the common voice dataset can be easily integrated in our data preprocessing step. The common voice dataset is also open-source and freely available.

What are the effects of the size of the dataset on the accuracy of the models?

Given the results from all the experiments that have been conducted during this research, it is now obvious that compared to the CNN-BLSTM model the LAS model has a better performance on all the datasets that have been used in this research.

To get a more convenient result, after finding the best hyperparameters, we trained the LAS model on the LibriSpeech100 for approximately 30 days with 1 million iterations. The one million iterations has been used because the model did not converge on fewer iteration steps (e.g. 500,000 steps), thus, we extended the iteration steps.

Tuning the hyperparameters of both models was proved to be very tricky. Therefore, we needed to find the best combination of settings of the batch size, optimizers, learning rate, normalization, initialization, and regularization strategies. We first needed to fix all the hyperparameters and changed only one of them to find the best setting for each hyperparameter. To do so, every time we needed to train our networks separately to see which one gave better results. After finding the best setting for each hyperparameter, we trained the models with this best setting of the hyperparameters all together.

Although the LibriSpeech100 is the smallest version of the dataset, with our resources, training the network even in this setting is very time taking. Therefore, we worked on some smaller datasets (subsets of LibriSpeech100) to decrease the training time of the experiments. As a result, tuning the hyperparameters on the small datasets on average takes about four days which is much lower than training on the complete LibreSpeech100 with at least 20 days of training. However, there is always a trade-off between choosing a smaller dataset and running more experiments or having a more accurate network with fewer experiments because the network is also sensitive to the size of the dataset.

Furthermore, the limited available computing resources such as the number and performance of GPUs impose a constraint on the experiment running time and the number of possible iterations per hour. With a higher number of GPUs or better-performing ones, the result of the experiments can be improved in the same period of time. However, since the environment for all the experiments was the same, the answers to the research questions remain the same in any case.

5.2.2 Answer to the Main Research Question

Which deep learning architecture for speech recognition tasks performs best on relatively small Dutch datasets?

After all, given all the experiments which have been conducted in this thesis research, we conclude that the LAS model is significantly more accurate than the CNN-LSTM model on both LibriSpeech and the Dutch dataset. With the LAS model we reach 32% CER on the LibriSpeech100 and 47% CER on the Dutch dataset. The CNN-BLSTM model achieved the accuracy of 83% CER on the LibriSpeech100 and 76% CER on the Dutch dataset. Considering the achieved results, it is concluded that the LAS model performs best on both datasets. Furthermore, we noticed that due to the less complexity of the structure of the CNN-BLSTM model, it does not have the capacity of being trained end to end and adding a language model would be beneficial.

5.3 Future Work

Considering the limitations of this research, in the future, we would like to investigate the accuracy of the two models on the Dutch conversational telephony dataset and perform more hyperparameter tuning. We are also interested in using the dense CNN-LSTM model on the Librispeech 100 and Dutch datasets.

Based on our results and what has been reported in the literature, it can be deduced that augmentation increases the accuracy of the trained network. Acquiring a larger network and training the network for a longer time until it completely converges will be also beneficial.

To deal with the problem of poor performance on long sentences, the authors of [7, 17, 68] provide some solutions for the machine translation problem. These solutions can be tried on speech recognition in future research as well to assess the effects.

Finally, according to the evaluation stage of the CRISP-DM method, the next step would be to test the LAS model in an experimental environment for the real application and with real data from the company.

Bibliography

[1] Amani Al-Ajlan and Achraf El Allali. CNN-MGP: Convolutional neural networks for metagenomics gene prediction. *Interdisciplinary Sciences: Computational Life Sciences*, 11(4):628–635, Dec 2019.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

[3] Shreya Amin. Speech recognition is hard, 2019. https://towardsdatascience.com/speech-recognition-is-hard-part-1-258e813b6eb7.

[4] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[5] Ahmad Asadi and Reza Safabakhsh. The encoder-decoder framework and its applications. In *Deep Learning: Concepts and Architectures*, pages 133–167. Springer, 2020.

[6] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

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

[8] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

[9] Janet M Baker, Li Deng, James Glass, Sanjeev Khudanpur, Chin-Hui Lee, Nelson Morgan, and Douglas O'Shaughnessy. Research developments and directions in speech recognition and understanding, part 1 [DSP education]. volume 26, pages 75–80. IEEE, 2009.

[10] Ravi Bandakkanavar. Speech recognition, 2017. https://krazytech.com/technical-papers/speech-recognition.

[11] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012.

[12] Yishay Carmiel. Deep learning revolutionizes conversational AI, 2017. https://www.oreilly.com/ideas/deep-learning-revolutionizes-conversational-ai.

[13] Asli Celikyilmaz, Li Deng, and Dilek Hakkani-Tür. Deep learning in spoken and text-based dialog systems. In Li Deng and Yang Liu, editors, *Deep Learning in Natural Language Processing*, pages 49–78. Springer, 2018.

[14] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

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

[16] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.

[17] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[18] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv preprint arXiv:1412.1602*, 2014.

[19] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.

[20] Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Andrew Ng. Deep learning with COTS HPC systems. In *International conference on machine learning*, pages 1337–1345, 2013.

[21] Pina Colarusso, Linda H. Kidder, Ira W. Levin, and E. Neil Lewis. Raman and infrared microspectroscopy. In *Encyclopedia of Spectroscopy and Spectrometry*, pages 1945 – 1954. Elsevier, Oxford, 1999.

[22] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[23] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2011.

[24] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Large vocabulary continuous speech recognition with context-dependent DBN-HMMS. In *ICASSP*, pages 4688–4691, 2011.

[25] Ken H Davis, R Biddulph, and Stephen Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, 1952.

[26] Li Deng and Yang Liu. *Deep Learning in Natural Language Processing*. Springer Publishing Company, Incorporated, 1st edition, 2018.

[27] Patrick Doetsch, Albert Zeyer, Paul Voigtlaender, Ilia Kulikov, Ralf Schlüter, and Hermann Ney. RETURNN: The RWTH extensible training framework for universal recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5345–5349. IEEE, 2017.

[28] Niklas Donges. Recurrent neural networks and LSTM, 2018. https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5.

[29] Niklas Donges. Connectionist temporal classification, September 2018. https://machinelearning-blog.com/2018/09/05/753/.

[30] Dan Ellis and Nelson Morgan. Size matters: An empirical study of neural network training for large vocabulary continuous speech recognition. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 2, pages 1013–1016. IEEE, 1999.

[31] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

[32] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention, please! a critical review of neural attention models in natural language processing. *arXiv preprint arXiv:1902.02181*, 2019.

[33] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: continual prediction with LSTM. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[35] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[36] 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. ACM, 2006.

[37] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

[38] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with bidirectional LSTM. In *Automatic Speech Recognition and Understanding Workshop*, 2013.

[39] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[40] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[41] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93:1–93:42, 2019.

[42] Kyu J Han, Akshay Chandrashekaran, Jungsuk Kim, and Ian Lane. The CAPIO 2017 conversational speech recognition system. *arXiv preprint arXiv:1801.00059*, 2017.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

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

[46] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[47] Kazuki Irie, Rohit Prabhavalkar, Anjuli Kannan, Antoine Bruguier, David Rybach, and Patrick Nguyen. Model unit exploration for sequence-to-sequence speech recognition. *arXiv preprint arXiv:1902.01955*, 2019.

[48] Benyamin Jafari. Difference between feedback RNN and LSTM/ GRU. Cross Validated. https://stats.stackexchange.com/q/420172 (version: 2019-08-14).

[49] Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems*, pages 5067–5075, 2016.

[50] Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke. Application of pretrained deep neural networks to large vocabulary speech recognition. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

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

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[53] James Le. The 3 deep learning frameworks for end-to-end speech recognition that power your devices, 2019. https://heartbeat.fritz.ai/the-3-deep-learning-frameworks-for-end-to-end-speech-recognition-that-power-your-devices-37b891ddc380.

[54] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[55] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[56] Deng Li and Douglas O'Shaughnessy. Speech processing: a dynamic and optimization-oriented approach. *Marcel Dekker Inc*, 2003.

[57] Jinyu Li, Li Deng, Reinhold Haeb-Umbach, and Yifan Gong. *Robust automatic speech recognition: a bridge to practical applications*. Academic Press, 2015.

[58] Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert. Letter-based speech recognition with gated ConvNets. *arXiv preprint arXiv:1712.09444*, 2017.

[59] Kate Loginova. Attention in NLP, 2018. https://medium.com/@joealato/attention-in-nlp-734c6fa9d983.

[60] Ramón López-Cózar, Zoraida Callejas, David Griol, and José F Quesada. Review of spoken dialogue systems. *Loquens*, 1(2):012, 2014.

[61] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. RWTH ASR systems for LibriSpeech: Hybrid vs attention-w/o data augmentation. *arXiv preprint arXiv:1905.03072*, 2019.

[62] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE transactions on audio, speech, and language processing*, 20(1):14–22, 2011.

[63] 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.

[64] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[65] Jinhwan Park, Yoonho Boo, Iksoo Choi, Sungho Shin, and Wonyong Sung. Fully neural network based speech recognition on mobile and embedded devices. In *Advances in Neural Information Processing Systems*, pages 10620–10630, 2018.

[66] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2, 2012.

[67] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[68] Jean Pouget-Abadie, Dzmitry Bahdanau, Bart Van Merrienboer, Kyunghyun Cho, and Yoshua Bengio. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. *arXiv preprint arXiv:1409.1257*, 2014.

[69] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880. ACM, 2009.

[70] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199. IEEE, 2017.

[71] Janet Read, Emanuela Mazzone, and Matthew Horton. Recognition errors and recognizing errors–children writing on the tablet pc. In *IFIP Conference on Human-Computer Interaction*, pages 1096–1099. Springer, 2005.

[72] Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*, 2017.

[73] Steve Renals, Nelson Morgan, Hervé Bourlard, Michael Cohen, and Horacio Franco. Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–174, 1994.

[74] Tony Robinson, Mike Hochberg, and Steve Renals. The use of recurrent neural networks in continuous speech recognition. In *Automatic speech and speaker recognition*, pages 233–258. Springer, 1996.

[75] Willem Röpke, Roxana Radulescu, Kyriakos Efthymiadis, and Ann Nowé. Training a speech-to-text model for dutch on the corpus gesproken nederlands. *CEUR Workshop Proceedings*, 2019.

[76] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[77] Sara Sabour, William Chan, and Mohammad Norouzi. Optimal completion distillation for sequence learning. *arXiv preprint arXiv:1810.01398*, 2018.

[78] Md Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. *Speech Communication*, 54(4):543–565, 2012.

[79] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for LVCSR. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8614–8618. IEEE, 2013.

[80] Tara N Sainath, Rohit Prabhavalkar, Shankar Kumar, Seungji Lee, Anjuli Kannan, David Rybach, Vlad Schogol, Patrick Nguyen, Bo Li, Yonghui Wu, et al. No need for a lexicon? evaluating the value of the pronunciation lexica in end-to-end models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5859–5863. IEEE, 2018.

[81] Tara N Sainath, Andrew W Senior, Oriol Vinyals, and Hasim Sak. Convolutional, long short-term memory, fully connected deep neural networks, April 7 2016. US Patent App. 14/847,133.

[82] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584. IEEE, 2015.

[83] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[84] Haşim Sak, Oriol Vinyals, Georg Heigold, Andrew Senior, Erik McDermott, Rajat Monga, and Mark Mao. Sequence discriminative distributed training of long short-term

memory recurrent neural networks. In *Fifteenth annual conference of the international speech communication association*, 2014.

[85] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[86] Karl Wilmer Scholz, James S Irwin, and Samir Tamri. Dialogue flow interpreter development tool, 2006. US Patent 7,024,348.

[87] Ben J Shannon and Kuldip K Paliwal. A comparative study of filter bank spacing for speech recognition. In *Microelectronic engineering research conference*, volume 41, 2003.

[88] William Song and Jim Cai. End-to-end deep neural network for automatic speech recognition. *Standford CS224D Reports*, 2015.

[89] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[90] Aarne Talman, Anssi Yli-Jyrä, and Jörg Tiedemann. Sentence embeddings in NLI with iterative refinement encoders. *arXiv preprint arXiv:1808.08762*, 2018.

[91] James Taylor. Fixing deployment and iteration problems in CRISP-DM, 2017. https://www.kdnuggets.com/2017/02/fixing-deployment-iteration-problems-crisp-dm.html.

[92] Xu Tian, Jun Zhang, Zejun Ma, Yi He, Juan Wei, Peihao Wu, Wenchang Situ, Shuai Li, and Yang Zhang. Deep LSTM for large vocabulary continuous speech recognition. *arXiv preprint arXiv:1703.07090*, 2017.

[93] Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *arXiv preprint arXiv:1704.01631*, 2017.

[94] Gokhan Tur and Li Deng. Intent determination and spoken utterance classification. *Spoken language understanding: systems for extracting semantic information from speech. Wiley, Chichester*, pages 93–118, 2011.

[95] Naomi van der Velde. Innovative uses of speech recognition today, October 2018. https://www.globalme.net/blog/new-technology-in-speech-recognition.

[96] Jeroen van Hoek. Choosing a speech-to-text service, 2018. https://medium.com/artificial-industry/choosing-a-speech-to-text-service-e11a27b20660.

[97] Oriol Vinyals, Suman V Ravuri, and Daniel Povey. Revisiting recurrent neural networks for robust ASR. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4085–4088. IEEE, 2012.

[98] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[99] Yongqiang Wang, Abdelrahman Mohamed, Duc Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, et al. Transformer-based acoustic modeling for hybrid speech recognition. *arXiv preprint arXiv:1910.09799*, 2019.

[100] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[101] Robert Andrew Wilson and Frank C Keil. *The MIT encyclopedia of the cognitive sciences*. MIT press, 2001.

[102] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[103] Dong Yu, Frank Seide, and Gang Li. Conversational speech transcription using context-dependent deep neural networks. In *ICML*, pages 437–440, 2012.

[104] Neil Zeghidour, Qiantong Xu, Vitaliy Liptchinsky, Nicolas Usunier, Gabriel Synnaeve, and Ronan Collobert. Fully convolutional speech recognition. *arXiv preprint arXiv:1812.06864*, 2019.

[105] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney. Improved training of end-to-end attention models for speech recognition. *arXiv preprint arXiv:1805.03294*, 2018.

[106] Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.