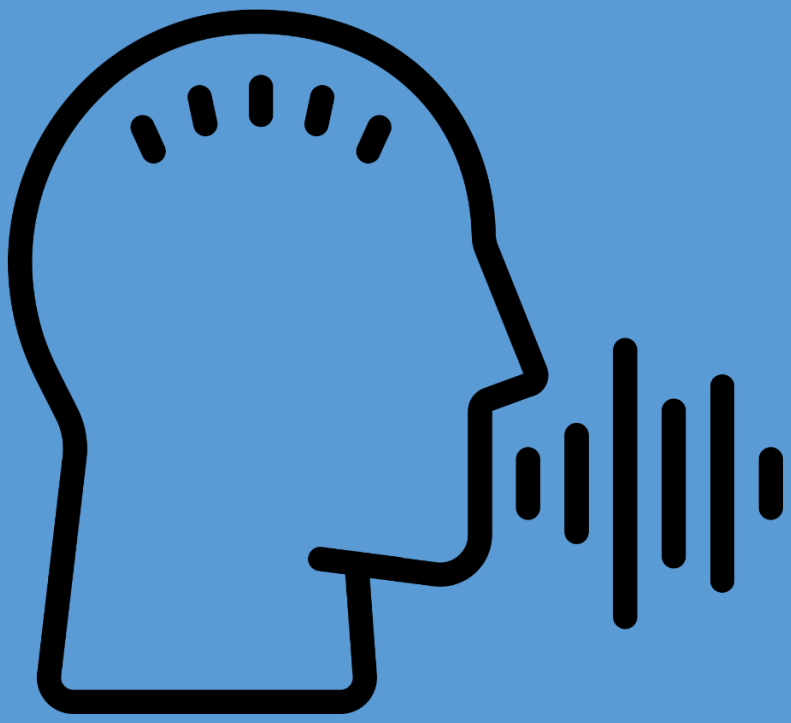
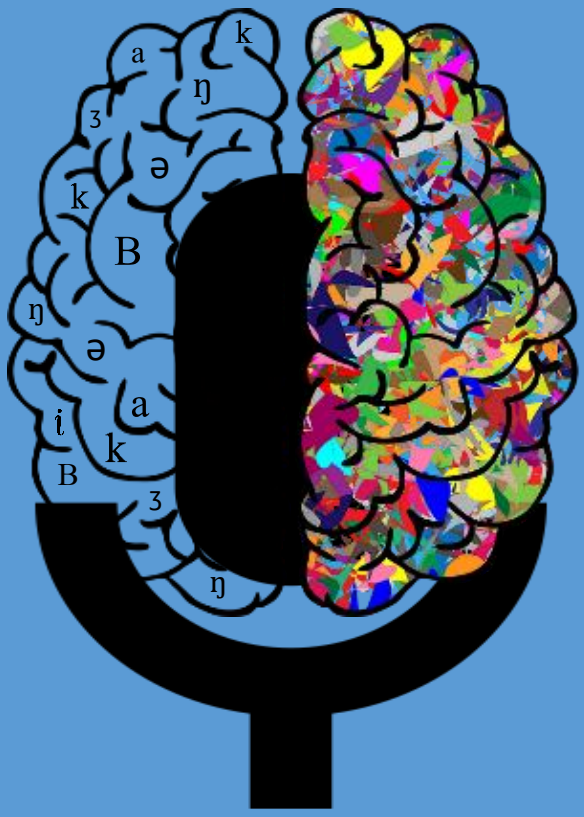
Phon Times

Improving Dutch phoneme recognition

By Robert Levenbach





TU Delft

Cognizant

PHON TIMES

Improving Dutch phoneme recognition

By

Robert Levenbach





in partial fulfilment of the requirements for the degree of

Master of Science

in Computer Science

at the Delft University of Technology,

to be defended publicly on Thursday, March 11th, 2021 at 11:00 AM.

Supervisor:	Dr. Odette Scharenborg	TU Delft
Thesis committee:	Dr. Odette Scharenborg	TU Delft
	Dr. Catharine Oertel	TU Delft

Dedication

This thesis would not have been possible without the time, support, and dedication of the people around me.

First, I would like to express my sincere gratitude to my supervisor Odette Scharenborg. In these extraordinary times of a global pandemic, Odette supported me throughout my work. Despite the lectures she had to give at the university, work she had to do for the board she is part of, giving her children home-schooling during lockdowns, and of course conducting her own research, during meetings she always stayed pleasant, informative, and funny. Additionally, I would like to thank Catharine Oertel for taking the time and effort of being part of the thesis committee.

Besides that, I would like to express my gratitude towards Siyuan Feng, who has co-supervised me. Siyuan especially introduced and guided me through the great world of Kaldi. Additionally, I would like to thank the members of the SALT (the language and speech group of the Technical University of Delft), who have given me interesting views and input on the work I have conducted and were also part of some fun Zoom-activities during the research period.

I would also like to thank Cognizant for the experience they have brought me. My gratitude is expressed towards Ton Vlot, Slava Keshkov, Eli Kling, and Hayo Rubingh for supervising me through the company and attending monthly meetings. Cognizant has shown me the world of consultancy and how to think about applying data science into real-world business cases. Besides that, Cognizant let me join the Cognizant Outreach Benelux volunteer team, where we conducted volunteer work by, for instance, encouraging and helping children from disadvantaged backgrounds to start a career in tech.

Finally, I would like to sincerely thank my parents, sister, other members of my family, and friends who have kept me sane and where always there for me during my thesis and the lockdowns.

Personal quote

| ɪf juː wɒnt tuː flaɪ wɪð ðiː iːgəlz, dəʊnt swɪm wɪð ðə dʌks |

Abstract

In this research, Dutch phoneme recognition (PR) is researched and improved. The last research on Dutch PR dates back to 1995. This research presents Dutch PR in modern daylight by researching state-of-the-art techniques found in research on other languages and implementing them on Dutch PR. The goal of this research is to find the current best architectures for Dutch phoneme recognition, analyse how those architectures perform on different speaking styles, and analyse the errors of the phonemes.

To achieve these goals, four state-of-the-art Neural Network architectures (BLSTM, OPGRU, TDNN-BLSTM, and TDNN-OPGRU) are built and trained on a Dutch dataset containing 15 different speaking styles. Then, the trained architectures are used for three experiments. The first experiment tests the architectures on a 15-speaking style containing the same 15 speaking styles as the training set. The results of that training set are used to calculate the phoneme error rate (PER), analyse the errors, and identify error-prone phonemes. The second and third experiments follow the same methodology on the same-trained architectures, however, are tested on a clear speaking style test set and a spontaneous speaking style test set. The results are used to analyse and compare the performances of the models on different speaking styles.

This research shows that the BLSTM and OPGRU are a poor choice for Dutch PR, as the results show relatively high PERs for these two architectures. The hybrid acoustic models TDNN-BLSTM and TDNN-OPGRU show to be good models for Dutch PR, as they gain low PERs. The results in this research set a new benchmark for Dutch PR, as the TDNN-BLSTM and TDNN-OPGRU show the lowest recognition accuracy found in existing Dutch PR literacy. Additionally, this research shows that recognising Dutch clear speech is less complex than recognising Dutch spontaneous speech, which is in line with existing word- and phoneme recognition research on other languages. Besides that, this research makes recommendations for which of the hybrid models is more effective to use when using the different speaking styles as input. Finally, this research finds the phonemes /I/, /a/, /k/, /s/, and the schwa as error-prone phonemes in Dutch PR, as they have high, above average individual PERs and contribute above-average to the overall PER on both the TDNN-BLSTM and TDNN-OPGRU.

This research also makes recommendations on where future Dutch PR could focus on. Setting a benchmark for Dutch PR, analysing different speaking styles, and identifying error-prone phonemes hopefully triggers future researchers on further improving Dutch PR.

Table of Content

ABSTRACT ..... **I**

TABLE OF CONTENT ..... **II**

LIST OF TABLES ..... **IV**

LIST OF FIGURES ..... **V**

1 INTRODUCTION **1**

1.1 RESEARCH OVERVIEW 4

1.2 REPORT SETUP 5

2 THE PROCESS AND HISTORY OF PHONEME RECOGNITION **6**

2.1 PHONEMES 7

2.2 SIGNAL PROCESSING, FEATURE EXTRACTION, AND FORCED ALIGNMENT 7

2.3 PRONUNCIATION LEXICON 9

2.4 PHONOTACTIC LANGUAGE MODELLING 9

2.5 ACOUSTIC MODELLING 10

2.5.1 Previous Work on DNN-HMMs *11*

2.6 DECODING 13

3 NEURAL NETWORKS AND THE ARCHITECTURES OF THIS RESEARCH **14**

3.1 BASICS OF NEURAL NETWORKS 14

3.2 TRAINING A NEURAL NETWORK 15

3.3 RECURRENT NEURAL NETWORKS 17

3.3.1 Sequential memory building *18*

3.3.2 Backpropagation through time and the vanishing gradient *21*

3.4 BEYOND VANILLA RECURRENT NEURAL NETWORKS 22

3.4.1 LSTMs and the basics of gated-RNNs *23*

3.4.2 Gated-RNN variants *25*

3.4.3 Bi-directional RNNs *26*

3.5 TIME-DELAY NEURAL NETWORKS 27

3.6 ARCHITECTURES OF THIS RESEARCH 29

3.6.1 Projected Bi-directional LSTM *29*

3.6.2 Output-gate projected GRU *29*

3.6.3 Hybrid models: Time-Delayed BLSTM and OPGRU *30*

4 EXPERIMENTAL SETUP **31**

4.1 Features used in this research 31
4.2 Lexicon of this research 31
4.3 Language model details 32
4.4 Acoustic model details 32
4.4.1 Baseline: BLSTM parameter setting 32
4.4.2 OPGRU parameter setting 32
4.4.3 TDNN-BLSTM parameter setting 32
4.4.4 TDDN-OPGRU parameter setting 33
4.5 Evaluation 33
4.5.1 Phoneme error rate 34
4.5.2 Confusion matrices 35
4.5.3 Individual phoneme error analysis 35
4.6 ASR toolkit: Kaldi 36
4.7 Dataset used in this research 36
4.7.1 Forced alignment 37
5 RESULTS 38
5.1 Phoneme error rates of all experiments 38
5.2 15-speaking style error analysis 39
5.3 Clear and spontaneous speaking style error analysis 42
5.4 Error-prone phonemes 44
5.4.1 Phonemes error analysis 15-speaking style test set 44
5.4.2 Phonemes error analysis clear and spontaneous speech 46
6 DISCUSSION 49
6.1 A split-decision: the best architecture for Dutch phoneme recognition 49
6.2 The big showdown: spontaneous versus clear speech 50
6.3 "Error, Error;" discussing the phoneme errors 51
7 CONCLUSION AND FUTURE WORK 53
BIBLIOGRAPHY 55

List of Tables

Table 1: Example lexicon of a phoneme recognition system 9

Table 2: Existing phoneme recognition architecture overview 12

Table 3: Existing word recognition architecture overview 12

Table 4: CGN phonetic lexicon and lexicon of this research 31

Table 5: Example of converting word-level transcriptions to phoneme-level transcriptions 32

Table 6: Example of a confusion matrix 35

Table 7: Overview of speaking styles in CGN, categorised in components a-o and with word-counts ('Totaal aantal woorden') differentiated between Flemish (VL) and Dutch (NL) 37

Table 8: Overview of training-, validation-, and test sets of this research 37

Table 9: Phoneme error rates of the BLSTM, OPGRU, TDNN-BLSTM, and TDNN-OPGRU on three different test sets 38

Table 10: Distribution of PER over the number of substitutions, insertions, and deletions over four architectures on the 15-speaking style 39

Table 11: 15-speaking style confusion matrices for BLSTM, OPGRU, TDNN-BLSTM, and TDNN-OPGRU 41

Table 12: Distribution of PER over the number of substitutions, insertions, and deletions for the TDNN-BLSTM and TDNN-OPGRU on the clear a spontaneous speech test sets 42

Table 13: Spontaneous and clear test set confusion matrices for the TDNN-BLSTM and TDNN-OPGRU 43

Table 14: Example of /s/-substitution 51

Table 15: Example of /@/-deletion 52

Table 16: Example of /@/-insertion 52

List of Figures

FIGURE 1: SEGMENTATION OF AN AUDIO WAVE INTO DISCRETE PHONEMES 2

FIGURE 2: DIFFERENT ACOUSTIC REPRESENTATIONS OF THE WORD "HELLO" 3

FIGURE 3: SUB-PROCESSES OF A PHONEME RECOGNITION SYSTEM 6

FIGURE 4: FEATURE EXTRACTION PROCESS SHOWING THE FRAMES AND FEATURES 8

FIGURE 5: ACOUSTIC MODELLING OF THE PHONEME SEQUENCE /J U:/ 10

FIGURE 6: HMM FOR THE PHONEME /J/ 11

FIGURE 7: A TYPICAL FEED-FORWARD NEURAL NETWORK WITH MULTIPLE HIDDEN LAYERS AND TWO TYPES OF ACTIVATION FUNCTIONS 14

FIGURE 8: LOGISTIC ACTIVATION FUNCTION 15

FIGURE 9: GRADIENT DESCENT AND DERIVATIVE OF THE LOSS FUNCTION J(W) W.R.T. THE WEIGHTS W 16

FIGURE 10: EXAMPLE SEQUENTIAL DATA TIME STEP T+1 17

FIGURE 11: EXAMPLE SEQUENTIAL DATA TIME STEP T 18

FIGURE 12: LEFT: ROLLED UP RNN ARCHITECTURE. RIGHT: UNROLLED RNN ARCHITECTURE 19

FIGURE 13: MEMORY CELL OF A RECURRENT NEURAL NETWORK 20

FIGURE 14: RNN WITH OUTPUT AND LOSS 21

FIGURE 15: TANH ACTIVATION FUNCTION AND THE DERIVATIVE 22

FIGURE 16: LEFT: ROLLED UP LSTM ARCHITECTURE RIGHT: UNROLLED LSTM ARCHITECTURE 23

FIGURE 17: LSTM CELL WITH DIFFERENT GATES, INPUT, AND OUTPUT 24

FIGURE 18: GATED RECURRENT UNIT CELL 25

FIGURE 19: PROTECTED GATED GRU CELL 26

FIGURE 20: BI-DIRECTIONAL LSTM ARCHITECTURE 27

FIGURE 21: TYPICAL TDNN ARCHITECTURE 28

FIGURE 22: PROTECTED-BLSTM CELL 29

FIGURE 23: EXAMPLE OF A HYBRID TDNN-BLSTM ARCHITECTURE 30

FIGURE 25: $PER_{PHONEMEX}$ RESULTS PER PHONEME ON THE 15-SPEAKING STYLE TEST SET ON THE TDNN-BLSTM AND TDNN-OPGRU 44

FIGURE 26: $PER_{PHONEMEX}$ RESULTS OF THE CLEAR AND SPONTANEOUS SPEECH TEST SET ON THE TDNN-BLSTM AND TDNN-OPGRU 46

FIGURE 27: CONTRIBUTIONTOPER$_{PHONEMEX}$ RESULTS OF THE CLEAR AND SPONTANEOUS SPEECH TEST SET ON THE TDNN-BLSTM AND TDNN-OPGRU 47

1 Introduction

One hundred harmonized muscles produce around fourteen different sounds per second whilst a human is speaking [1]. This naturally acquired, complex human motor ability produces an audio wave that contains sentences. These sentences are made up of words, which themselves are made up out of phonemes. Phonemes are defined as "the smallest units of sound that make one word different from another word" [2]. An example of a sentence is “Hello you!” which consists of the words “hello” and “you” and the phonetic sequence “/h e l ə ʊ j u: /”. The task of an automatic speech recognition (ASR) system is to transform the audio wave coming from the person’s mouth into a sequence of written discrete (i.e. separated) sentences/words/phonemes.

A popular form of an ASR system is a word recogniser. Word recognition (WR) systems transform an input speech wave into discrete written words. These words can then be used for a variety of tasks. These days, home- and phone assistants such as Alexa, Siri, and Google Assistant are being used by people all around the world. These systems commonly perform operations based on the recognition of words. This is just an example of a WR use-case; word recognisers can also perform many other tasks, like translation [3], as controller of command & control system [4], for dictation [5], in help desks [6], for chatbots [7], in-car assistants [8], among many others.

Another type of ASR system is a phoneme recogniser. Phoneme recognition (PR) systems are similar to WR systems, but where a WR system recognises words, the task of PR is to convert a speech signal into a discrete sequence of written phonemes. PR systems can also be used for a variety of tasks. For instance, when a sentence contains words that are not present in the vocabulary of a WR system (and therefore cannot be recognised), a PR can be used to create a sequence of phonemes as a replacement (i.e. an out-of-vocabulary word recognition system). This is useful when input contains new names, alternating pronunciations, or new words. Other tasks for PR are, for instance, speaker verification [9], evaluating the goodness of pronunciations (e.g. for when a person is learning a new language) [10], increasing speech intelligibility in noisy speech [11], identifying mispronounced phonemes for people with speech impairments [12], synchronising music with lyrics [13], language identification [14, 15], among many others.

This research will build a Dutch phoneme recogniser for Cognizant. Cognizant is an American product-led digital consultancy firm that provides IT services, IT consultancy, digital products, and operation services, focussing on Artificial Intelligence & Analytics, Digital Engineering & Strategy, Internet of Things, and Interactive Technologies [16]. The Dutch ASR industry is predicted to be worth around 16 billion dollars by 2021 [17]. Cognizant *Benelux*, for whom this research is conducted, is interested in the research and implementation of a *Dutch* PR, as they see the potential for future business and commercial use-cases in the Benelux. In the future, this can be shown to the clients of Cognizant as inspiration for business- and commercial applications.

When a PR application is built for business and commercial use, a correctly performing system is crucial. When recognition-inaccuracies are too apparent for the user, PR systems could lead to frustration, user-unfriendly experiences, ultimately resulting in unwanted products [18, 19]. Improvements in PR performance will benefit the performance of the aforementioned tasks, which will create more user-friendly applications and business value for commercial parties. As this research will build a business and user-oriented PR system, the recognition performance of the built PR is an important factor.

The performance of a PR system is commonly measured using the phone error rate (PER). This error rate is based on the Levenshtein distance and calculates how many phonemes are misclassified (as further discussed in Section 4.5). The common goal of researchers in PR is to minimise the PER. When

looking at existing Dutch *WR*, research is active on lowering the word error rate (the word equivalent of PER) [20, 21, 22, 23, 24]. However, when looking at existing Dutch PR, fewer papers are found. The only paper that exists in current research on improving Dutch PR dates back to 1995 [25]. That research gained a PER of >50%, meaning more than half of the phonemes were misrecognised. As Cognizant is interested in a Dutch PR system, the performance of the PR system is crucial for customer satisfaction, and no modern research on Dutch PR exists, this research will focus on improving Dutch PR.

As mentioned earlier, PR predicts a sequence of discrete written phonemes based on an input speech wave. An example of a speech wave is visualised in Figure 1. In Figure 1, a speech wave is visualised as an *acoustic representation* and is split into phonemes that represent the word 'hello'. As can be seen, speech is not organised like writing. Writing has the form of discrete entities (i.e. isolated from each other), while speech is formed by simultaneously woven adjacent phonemes. When looking at Figure 1, PR might not seem overly complex; after all, the PR should just be able to recognise what acoustic representation belongs to every phoneme in a language and then match them to the input speech wave. However, two characteristics make PR a more complex task.

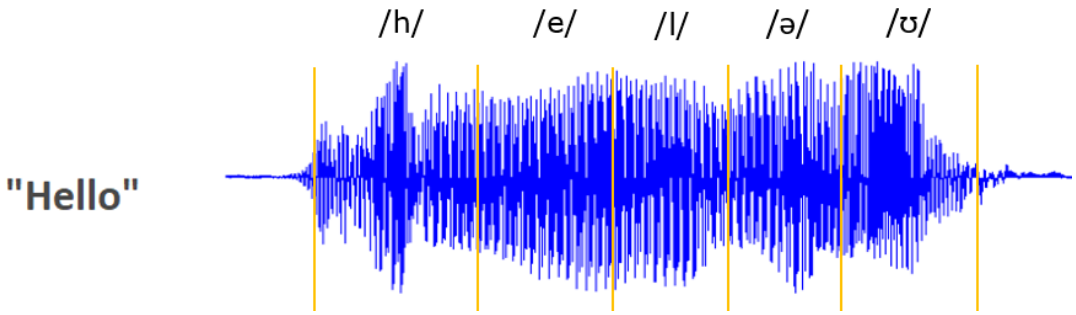


Figure 1: Segmentation of an audio wave into discrete phonemes

The first characteristic is that the speech wave of the same sequence of phonemes is different every time it is pronounced (by either the same or a different person, in the same or different context) [26]. This means that the blue speech wave of the word 'hello' in Figure 1 will look different if it would be pronounced again. This is visualised in Figure 2. The first speech wave in Figure 2 is spoken slowly and articulated well; the second one is mumbled and pronounced rapidly. As can be seen, the acoustic representations of both words, and thus phonemes, are different. The PR system must be able to match the speech waves to the correct phonemes and therefore, must be able to match different acoustic representations to the same phonemes.

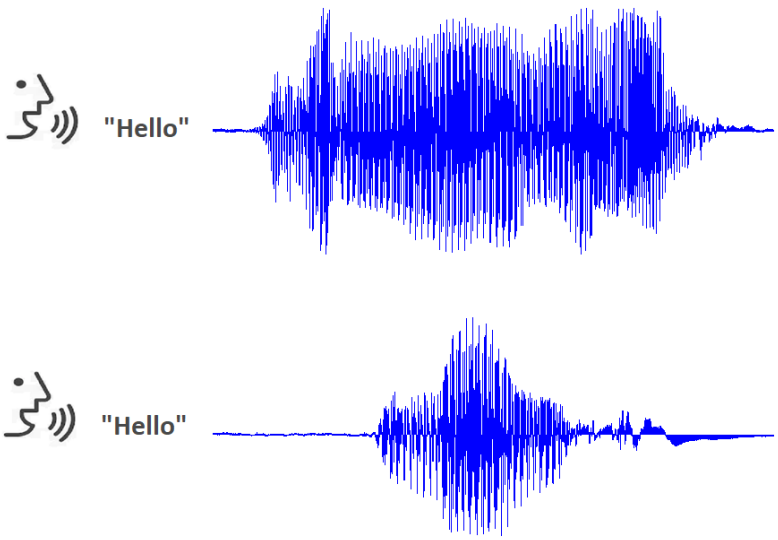


Figure 2: Different acoustic representations of the word "hello"

The second characteristic that makes PR complex is finding the boundaries between phonemes, as it is rare to find phonemes that are isolated from other phonemes. As the phonemes are close to each other and/or woven into each other, the gap between discrete phonemes is not always clear. For instance, when looking at Figure 2, the same five phonemes are pronounced. However, it is not directly clear where the phonemes start and end (e.g. where does the first phoneme start and ends?).

Searching through *all possible* (i.e. infinite) acoustic representations and the accompanying phonemes, while not knowing where the phonemes begin or end is too computationally expensive and time-consuming. Therefore, statistical models have been introduced in PR, which try to predict the most optimal match of phonemes with the frames (the whole process of how PR implements statistical models is further explained in Section 2). State-of-the-art PR research on languages such as English [27], Mandarin [28] [29], and Indian [30] [31] show that implementing Neural Networks (NN) architectures as statistical models show the best performance, gaining PERs of less than 15%. The aforementioned only existing research on Dutch PR did not implement NNs. Therefore, this research will be the first research to implement NNs in a Dutch PR system.

The performance of a PR does not only depend on the type of architecture, but also the type of speaking style that is used as input data [32]. Different speaking styles are, for example, news-bulletins (which is prepared, read, and well-articulated speech), guided interviews (which is semi-prepared, spontaneous, and fairly well-articulated speech), or conversations between friends (which is unprepared, spontaneous, and less well-articulated speech). Not every speaking style uses *canonical* pronunciation, meaning the pronunciation does not always follow the generally accepted rules of pronunciation. Differences in speaking styles result in variability in the articulation of phonemes. The variability could change due to several factors: the speech/speaking rate, read/unprepared speech, dialect, accent, mood, among other causes. Also, the context of the speech is important, as the quality of the microphone, environmental noise, number of people who partake in the conversation, etc. will affect the clarity of the pronunciations of the phonemes. Over 20 different speaking styles can be distinguished [32], however, in existing research a common way of splitting all these different speaking styles is into two main categories [32] [33] [34] [35]:

1) *Spontaneous speech*: consists of speech data that is produced in more or less unprepared situations (e.g. conversations between friends, spontaneous interviews, etc.). When speaking in this type of speaking style, people tend to take 'the path of the least resistance', while keeping 'the law of intelligibility' in mind [36]. This means that people tend to pronounce words as easy as possible while staying understandable. This results in a higher variation of pronunciations of phonemes, phonemes that have less distance between them, phonemes that are more often woven into each other, and more phoneme reductions. As explained earlier in this chapter, this leads to a higher complexity for the PR models to recognise the phonemes.
2) *Clear speech*[1]: consists of more prepared speech (e.g. professional speakers, news-bulletins, text read from paper, etc.). This type of speech has a higher clarity of articulation, less variation in pronunciation and less phoneme reduction when compared to spontaneous speech. This means that the phonemes are mostly pronounced and articulated as clear as possible, making it easier for models to recognise when compared to spontaneous speech.

Existing research on different speaking styles focusses on researching the recognition performance of different architectures using different speaking styles as input or, for instance, what phonemes are miss- or unrecognised when using different styles [37]. Other research focusses on designing PR systems that can recognise all or one specific style(s) specifically, to either compare speaking styles or to improve the performance of the aforementioned business- and commercial use-cases (which, in this case, only use certain speaking styles) [37]. Research on Japanese [33], German [34], and English [35] has shown that recognising clear speech is less complex than recognising spontaneous speech, due to the reasons mentioned in the bulletins above. This type of research does not exist for Dutch PR. Therefore, in this research, different speaking styles will be analysed.

1.1 Research overview

In this research, the aim is to improve Dutch phoneme recognition (PR). This will be done by implementing and training several state-of-the-art deep neural network architectures, analysing the errors, testing the same trained architectures on clear and spontaneous speech, and identifying error-prone phonemes.

As existing Dutch PR research is lacking, the choice of architectures in this research is made based on existing WR and PR research on other languages. One of the best performing architectures from existing research on English PR uses a light-gated gated recurrent unit (LiGRU) as NN architecture [38]. LiGRU could be implemented in this research; however, existing *WR* research shows that a variation of the LiGRU, namely the Output-Gated Protected GRUs (OPGRU), shows a better performing model while lowering computation time [39]. Therefore, this research will implement an OPGRU as NN architecture. In the same paper, an advanced version of the OPGRU is introduced, namely the time-delayed neural network OPGRU (TDNN-OPGRU). This advanced architecture combines two different NN architectures (i.e. TDNN and OPGRU) into a single NN. TDNN-OPGRUs improve performance, while also reducing computation time [39]. Therefore, TDNN-OPGRUs are also implemented in this research.

As no Dutch NN PR research exists, a baseline model must be built to monitor and assess the performance of the aforementioned architectures. In [39], the OPGRU and TDNN-OPGRU are compared to a bi-directional long-short term memory (BLSTM) architecture as the baseline. Therefore, this research will also implement a BLSTM as a baseline model. Additionally, in this research, a TDNN-

[1] Clear speech in this context means speech that is more clearly pronounced (other research might refer to clear speech as speech with little contextual noise)

BLSTM will be implemented to assess the performance of the addition of TDNN layers to the baseline model.

The dataset that is used to train and test the architectures is the Corpus Gesproken Nederlands dataset [40]. This large, online available dataset consists of 900 hours of annotated Dutch and Flemish speech and contains 15 different speaking styles (see Section 4.7). The architectures will be trained on a 150-hour subset of the Dutch part of CGN containing all 15 speaking styles. The performance of the models will be tested in three experiments. The first experiment will test how well the architectures perform on a 15-speaking style containing the same 15 different speaking styles as the training set. Then, the second and third experiment will test how well the architectures generalise to a separate test set containing respectively spontaneous and clear speech.

The main evaluation metric that will be used to assess the performance of these architectures is the phoneme error rate (PER). As the goal of this research is to improve Dutch PR, further error analysis will be conducted to analyse where the errors originate and what phonemes are difficult to recognise by the models. This can be used for future research to see where improvements can be made.

The following research goal is formulated:

Research Goal: "Improving Dutch phoneme recognition"

To achieve this goal, the following sub-research questions are answered:

Sub-Research Question 1: "What is the best architecture for general Dutch phoneme recognition?"

Sub-Research Question 2: "How do different architectures perform on different speaking styles?"

Sub-Research Question 3: "What are error-prone phonemes in Dutch phoneme recognition?"

1.2 Report setup

This report is structured as followed. First, the process and accompanying history of PR is given (see Section 2). This is done by explaining all sub-processes individually and finding existing literature on those processes. A sub-process of PR found in Section 2 is acoustic modelling. In this research, acoustic modelling is done using NNs. NNs are a large part of this research. Therefore, the basics of NNs and the NNs used in this research are explained in Section 3. In Section 4, the experimental setup of this research gives the setup of building, training, and testing of the models used in this research. In the same chapter, the evaluation methodology for the models is given. Thereafter, the results are presented (see Section 54.7.1), which are further discussed in Section 6. In the discussion, the research questions of this research are answered, which are used to draw the conclusion and recommend future work (see Section 7).

2 The process and history of phoneme recognition

The task of phoneme recognition (PR) is to automatically convert a speech signal into a discrete sequence of written phonemes. This is done in several processes, which are visualised in Figure 3. These processes will now be briefly explained and more thoroughly throughout this chapter. It starts with a spoken sentence. This sentence is captured by a microphone and pre-processed using signal processing (see Section 2.1). This results in a computer-readable input sound wave *O*. This sound wave is transformed into a chosen numerical representations, building feature vectors *X* using feature extraction (see Section 2.1). These feature vectors are then fed into a decoder.

The goal of the decoder is to transcribe the feature vectors *X* into a sequence of phonemes *W*. The decoder aims to find the most likely sequence of phonemes *W* out of all possible sequences of phonemes in the language *L*, given feature vectors *X* (i.e. the probability $_{argmax}P(W|X)$). This likelihood is calculated by combining the entries of a lexicon *L* (see Section 2.3) into an acoustic model (i.e. the probability *P(X|W)*, see Section 2.3) and multiplying it with the phonotactic language model probability (i.e. the probability *P(W)*, see Section 2.4). This forms the following formula:

$$\underset{w \in L}{\operatorname{argmax}} P(W|X) = P(X|W)P(W)$$

On a high level, the lexicon determines what phonemes can be recognised by the PR, the acoustic model represents the relationship between the audio wave and the phonemes, and the language model models coherent sequences of phonemes (i.e. the order of phonemes). These combined elements can be used to find the best match of the speech wave and the phoneme sequence during decoding (see Section 2.6).

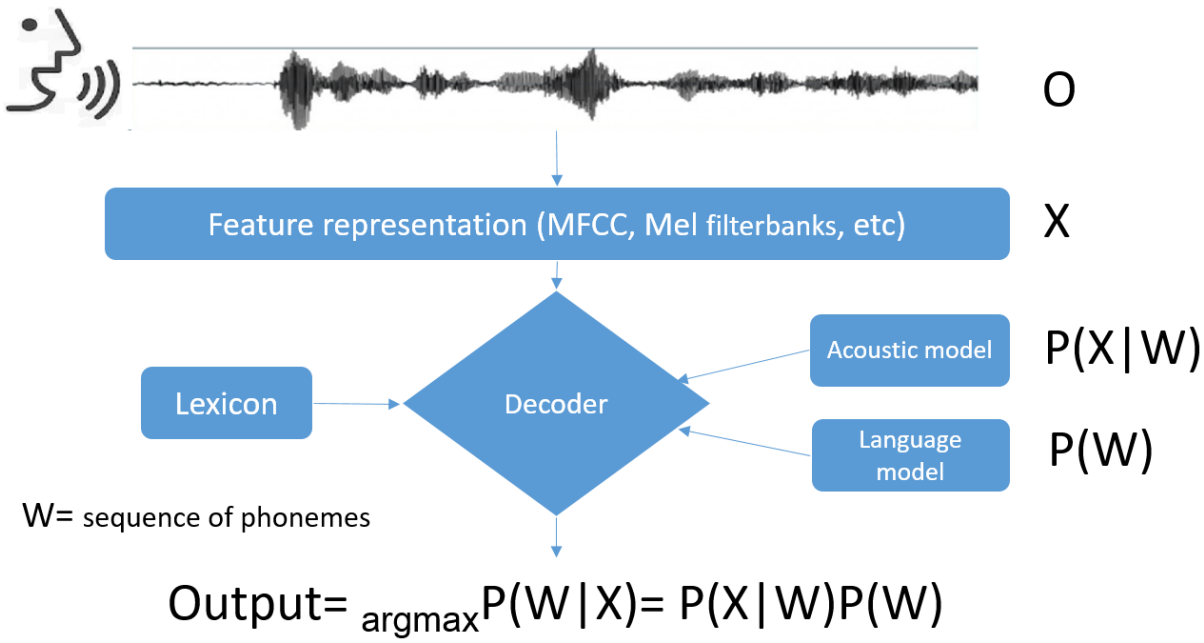


Figure 3: Sub-processes of a phoneme recognition system

2.1 Phonemes

Before the sub-processes of a phoneme recogniser are explained, the actual concept of a phoneme is given[2]. As defined earlier, phonemes are "the smallest units of sound that make one word different from other words" [2]. For instance, the words /f e l/ ('fel') and /v e l/ ('vel') are two separate words that are distinguishable by the substitution of one phoneme. Phonemes can be notated in several different ways (e.g. IPA, SAMPA, YAPA, CGN, among others) [41]. This research will use the notation that is used in CGN (see Section 4.7), which contains 40 phonemes [41].

Phonemes can be further categorised into vowels and consonants. Within vowels and consonants, several sub-categories exist. Grouping phonemes into these (sub-)categories might show patterns in the results of the performances of the architectures and between speaking styles (e.g. certain groups are easier to recognise by the models, certain categories are harder to recognise in clear speech, etc.). In this research, consonants are categorised into three sub-categories: plosives, fricatives, and sonorant consonants [42]. These are categorised based on the way and where they are produced by humans (for more information, please refer to [43]).

2.2 Signal processing, feature extraction, and forced alignment

The first part of building a PR system consists of two sub-process: signal processing and feature extraction. Signal processing captures the speech wave using a microphone and removes redundant sounds (e.g. background noise) [44]. The output is a computer-readable speech wave *O*. How signal processing precisely works falls outside the scope of this research (for more information, please refer to [44]).

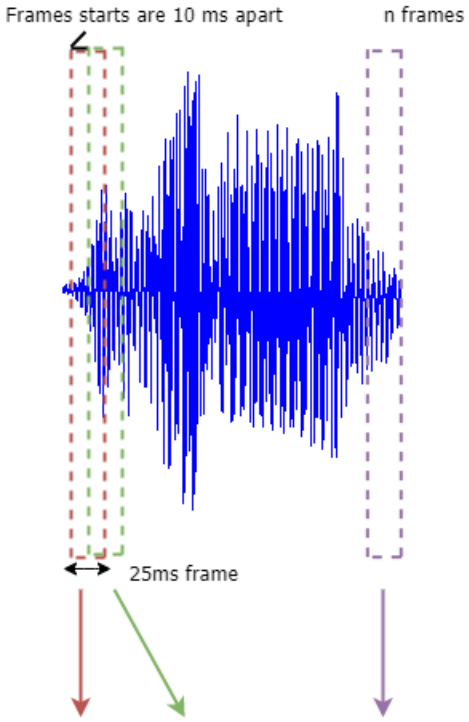
Thereafter, the captured speech signal *O* can be transformed into feature vectors *X* using feature extraction. This is visualised in Figure 4. Figure 4 shows how the input speech waveform *O* is split into *n* frames (commonly with a chosen width of 25ms). From every frame, a numerical vector *x* is extracted with vector length *d* (also known as the dimension of the feature). This vector is also known as a feature vector or simply feature [45]. When combining all *n* features from all *n* frames, a *(n x d)* matrix of features *X* is created, as visualised at the bottom of Figure 4.

The numerical representation of the features is a parameter choice. In existing SOTA PR research, different types of features are used. Commonly used features are Mel-frequency cepstrum (MFCC) and mel filterbank (MFB) features. As shown in SOTA PR research [46, 47, 48, 49] and a recent survey on PR [27], MFBs mostly yield better results. Therefore, in this research, MFB features are used. MFB feature extraction follows several subprocesses, which falls outside the scope of this project (for the full extraction method, please refer to [50]).

Besides MFB features, I-Vector[3] features are also extracted. I-vectors are vectors that are extracted not from the individual frames in an utterance, but from the whole utterance [51]. I-vectors try to capture the speaker and channel variability of a speech wave; difference utterances have different settings, changing factors like the recorder, speaker, environment, etc. (i.e. the context of the speech signal) [52]. I-vectors extract a fixed-length vector from every utterance to add information to the model about the channel and speaker variability. This addition of information has proven to yield better results in several phoneme recognition related research [53, 54, 55, 56]. Therefore, in this research, I-vector features are also extracted and used in combination with MFB features. The extraction of I-vectors is done in several subprocesses, which again, fall outside the scope of this project (for the full extraction method, please refer to [51]).

[2] This will explained for standard ABN Dutch

[3] The I stands for intermediate representation



Feature Set X			
Feature vector 1	**Feature vector 2**	**Feature vector ...**	**Feature frame n**
x11	x21	...	xn1
x12	x22	...	xn2
x1...	x2...	...	xn...
x1d	x2d	...	fnd

Figure 4: Feature extraction process showing the frames and features

2.3 Pronunciation lexicon

The pronunciation lexicon *L* (in short 'lexicon') has the role of specifying what phonemes are recognised by the system [57][4]. The lexicon is a list of phonemes. An example of a lexicon with five phonemes is given in Table 1. The goal is to build an acoustic model for every phoneme in the lexicon (see Section 2.5). Using those individual acoustic models, the probabilities $P(X|W)$ can be calculated in the HMM, where W is limited to only a sequence of phonemes that are combinations of entries of the lexicon (i.e. $W \in L$).

Table 1: Example lexicon of a phoneme recognition system

PR Lexicon entry	Phonemic representation
/e/	/e/
/ə/	/ə/
/ʊ/	/ʊ/
/h/	/h/
/l/	/l/

2.4 Phonotactic language modelling

The task of the language model (LM) in a PR is to model *P(W)*: the probability of a sequence of phonemes *W* occurring in a language [58]. Typically, in PR, this probability is calculated using n-gram models [59].

N-grams calculate the likelihood of a phoneme based on *N-1* previous phonemes. For instance, a bigram (i.e. 2-gram) assigns a probability of a phoneme based on the preceding phoneme, a trigram on the two preceding phonemes, and a 4-gram on the three preceding phonemes, etc. These probabilities can be found using the frequency of occurrences of the n-grams in the phoneme transcriptions of the training corpus. For instance, take a transcript that consists of one sentence: "/h/ /e/ /l/ /ə/ /ʊ/ /h/ /e/ /l/ /ə/ /ʊ/" ("Hello hello"). The chance of every phoneme occurring individually (i.e. unigram (= 1-gram)) would be:

$$P(\text{"}phoneme\text{"}) = (\#\ \text{"}phoneme\text{"})/(\#total\ number\ of\ phonemes\ in\ the\ transcript)$$

So for instance in the example transcript, the phoneme /e/ would get the following probability:

$$P(/e/) = 2/10 = .20$$

In this research, bigram models are used as they have shown to perform well for PR [60]. With bigrams, the probability of the current phoneme is calculated by the probability of the directly preceding phoneme occurring before the current phoneme, which is then divided by the probability of the current phoneme occurring in the transcriptions. For instance, the sequence of "/h/ /e/" in the example corpus would get a probability of:

$$P(/h/\ |\ /e/) = \frac{(P(/h/ \cap /e/)\ )}{(P(/e/))} = 1$$

P(W) can be calculated by combining all probabilities of the individual bigrams in the phoneme sequence *W*.

[4] In comparison, in a WR the lexicon would consists of words and their phonemic representations. In WR, this constrains the models to only those combinations of phonemes. In PR it does not, as all possible combinations of phonemes can be made. Therefore, the lexicon in PR is commonly referred to as dummy-lexicon

2.5 Acoustic modelling

The task of the acoustic model (AM) is to model *P(X|W)*; given a sequence of phonemes *W*, what is the probability of observing feature vectors *X* [61]. As explained in Section 1, this probability is estimated using statistical models.

Early effective statistical AMs in ASR were Gaussian Mixture Models (GMM) combined with Hidden Markov Models (HMM), creating GMM-HMMs. These were implemented for the first time in the 1980s [62]. GMM-HMMs have proven to work effectively for certain tasks, as they can model acoustic variability like pronunciation variability, accents, and dialects [63]. However, for tasks like PR, alternative models work more effectively, namely DNN-HMMs. With the introduction of deep neural networks (DNN) around 2006, DNN-HMMs gained popularity in PR research around 2010 [64, 65]. The way DNN-HMMs work and why they are effective for PR is further explained now.

As explained in Section 2.2, signal processing and feature extraction is used to create feature vectors *X*. The DNN-HMM acoustic model estimates the probability for every possible phoneme sequence *W* belonging to this extracted feature vectors *X*, where *W* is only made out of phonemes in the lexicon (see Section 2.3). An example of the phoneme sequence *W* of the word *'you'* is visualised in Figure 5. In Figure 5, the phoneme sequence /j u:/ is split into individual phonemes, which are then individually modelled as HMMs. The bottom of Figure 5 shows how the feature vectors of the speech signal are matched to the phonemes. This is works as followed.

The probability *P(X|/j u:/)* can be calculated by multiplying the probabilities of the individual phonemes /j/ and /u:/, i.e. $P(X_1|/j/)$ and $P(X_2|/u:/)$, where X_i denotes separate parts of the feature vectors *X*. These probabilities are estimated using the HMMs models of every phoneme.

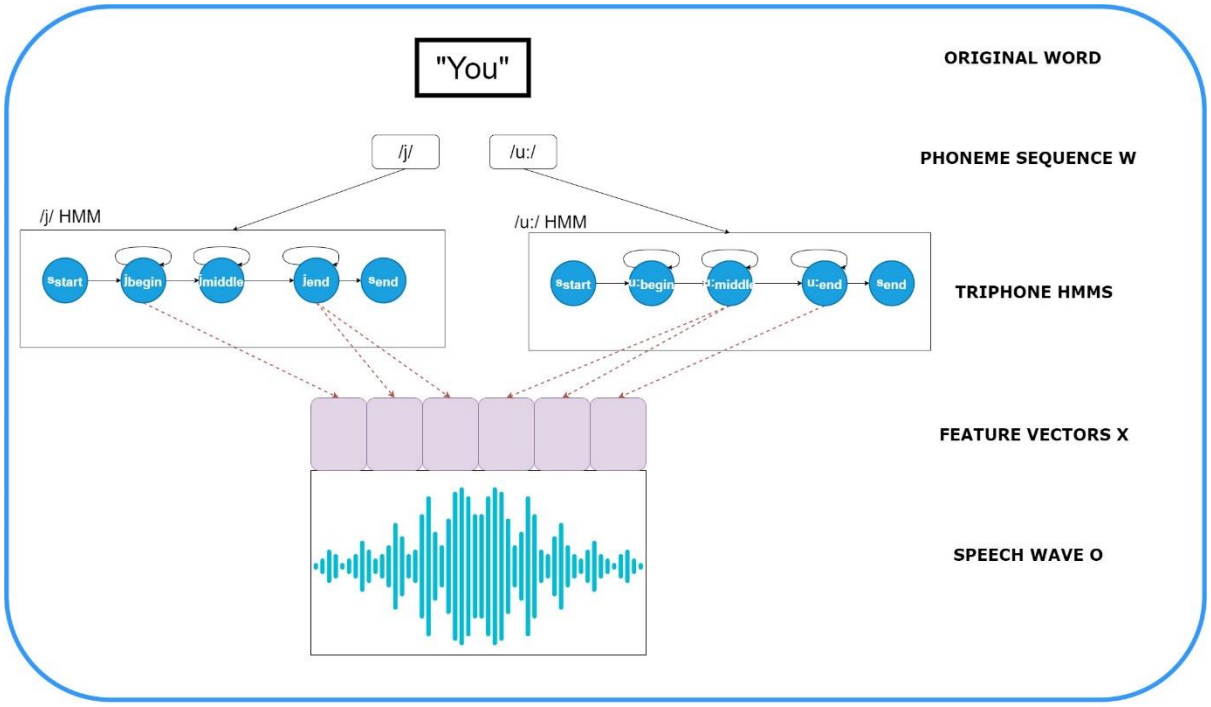


Figure 5: Acoustic modelling of the phoneme sequence /j u:/

The HMM of the phoneme /j/ in Figure 5 is further expanded in Figure 6. As can be seen in Figure 6, the phoneme is modelled as a triphone, meaning the phoneme is modelled in three sub-phones (beginning, middle, and end of the phoneme), where every sub-phone is a hidden state. Sub-phones enable the acoustic model to model context to the phonemes, as the feature vectors can now be matched with three different sub-phones, all representing the same phoneme but with different contexts. This is useful, as the acoustics of a phoneme can change based on the context of the phoneme (as mentioned in Section 1). Besides the three sub-phone states, a start and end state are added to the HMM.

The goal is now to estimate the probabilities in the HMM; the transition probabilities (between the hidden states) and the emission probabilities (between the hidden states and the features). The transition probabilities can be trained using a corpus of transcriptions and calculating the probability of one state following the other state based on the occurrences in the same corpus (for more information, please refer to [66]). More interesting for this research are the emission probabilities. These calculate the likelihood of an observation feature vector *x* given a sub-phone state (see the middle row in Figure 6). These likelihoods can be estimated using DNNs. How DNNs calculate the emission probabilities is further explained in Section 3. The emission probabilities and transition probabilities can now be used to calculate *P(X|W)*.

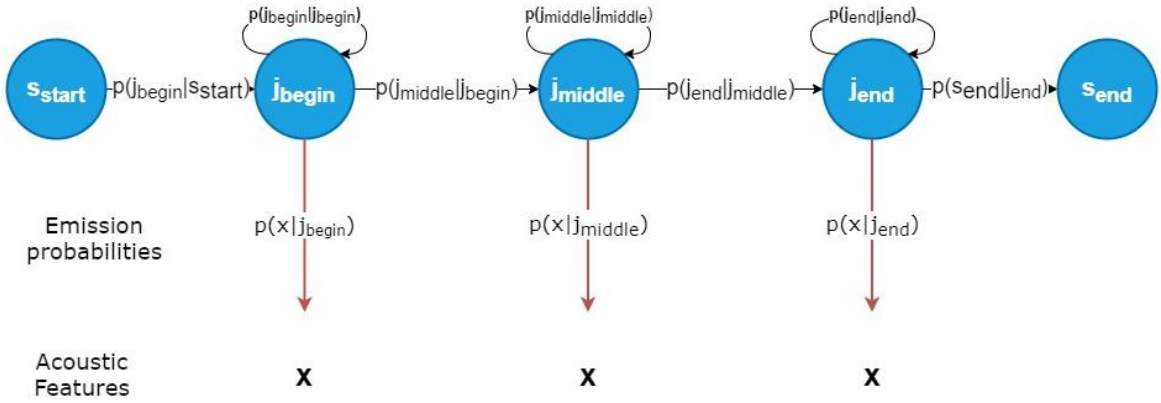


Figure 6: HMM for the phoneme /j/

When the same HMM model in Figure 6 is built for the phoneme /u:/ from Figure 5, the probabilities of both phonemes can be multiplied to get *P(X|/j u:/)*.

2.5.1 Previous Work on DNN-HMMs

Since the introduction of DNNs, research on improving ASR systems has focussed mostly on optimising the input and settings of the DNNs (i.e. finding the best architectures, parameters, optimisation functions, features, etc.) to calculate the emission probabilities more and more accurate [67, 68, 69, 70]. Table 2 shows an overview of existing research architectures on PR, with accompanying training dataset, language, error rate (i.e. phoneme error rate (PER)), and references to the original work. This is by far not an exhaustive overview, as a lot of other research has been conducted. What this figure does show, is popular ASR architectures over time with their PER. For instance, the introduction of recurrent neural networks (RNN) (e.g. Long-Short Term Memory RNNs (LSTM), (Bi-directional) LSTMs (BLSTM), Gated Recurrent Units (GRU)) around 2016 has shown great improvements in recognition rates. RNNs are a class of DNNs that perform well on sequential data. RNNs are used in this research and are further explained in Section 3.

Table 2: Existing phoneme recognition architecture overview

Architecture	Language(Dataset)	WER	Paper
DNN-HMM (2013)	English (MOCHA-TIMIT)	23.70%	[71]
Monophone DBN (2011)	English (Switchboard)	20.70%	[72]
BLSTM (2013)	English (TIMIT)	17.99%	[73]
LSTM (2013)	English (TIMIT)	17.70%	[74]
RNN (2014)	English (TIMIT)	18.57%	[75]
Li-GRU (2018)	English (TIMIT)	14.9%	[76]

Besides research on PR, it is also interesting to analyse research on WR, as this could give insights into whether advanced architectures are implemented in WR which have yet been implemented on PR. Research on WR shows trends as PR. In the same way as PR, NNs have led to major advances in the recognition rates of phoneme recognisers. Table 2 shows an overview of existing research architectures on WR, with accompanying training dataset and language, error rate (i.e. word error rate (WER)), and reference to the original work. Again, this is by far not an exhaustive overview, as substantial other research has been conducted on WE.

As for PR, for WR, the introduction of RNNs has shown improvements in recognition rates, as the (D)BLSTM and RNN show relatively lower WER scores. Additionally in Table 3 (and less popular in PR) is an improvement of performance made by hybrid acoustic models (e.g. RNN-LSTM, Time-Delayed LSTMs (TDNN-LSTM), and TDNN-OPGRU) around 2017. Hybrid-acoustic models combine different types of architectures to combine the strength that they both individually have (see Section 3.6.3). Hybrid models have shown great improvements in accuracy and computational speed, especially when trained on large datasets [77, 78]. Therefore, this research will also implement hybrid models, which and are further explained in Section 3.

Table 3: Existing word recognition architecture overview

Architecture	Language(Dataset)	WER	Paper
DNN-HMM (2013)	English (Switchboard)	18.40%	[79]
TDNN-HMM (2015)	English (Switchboard)	17.10%	[80]
BLSTM+MMI (2016)	English (Switchboard)	13.00%	[81]
RNN+LSTM (2016)	English (Switchboard)	12.20%	[82]
GRU (2017)	English (Wall Street Journal)	14.90%	[83]
LSTM-BLSTM (2017)	English (NIST)	9.10%	[84]
TDNN-LSTM (2019)	English (Switchboard+Fisher)	8.10%	[85]
TDNN-OPGRU (2019)	English (Switchboard+Fisher)	7.90%	[85]

The architectures implemented in this research are chosen based on existing research on both WR and PR. The motivation and choice of architectures can be found in Section 1.1. How they are built and work is further explained in Section 3.

2.6 Decoding

The goal of the decoder is to transcribe the feature vectors extracted from the speech wave into the most probable sequence of phonemes. The decoder creates a ranked-ordered N-best list for all possible phoneme sequences $W \in L$ where the top of the N-best list represents the sequence of phonemes with the highest probability $P(W|X)$. As can be noticed, going through all possible sequences becomes computationally expensive when the lexicon has a lot of entries. Therefore, the Viterbi search algorithm is used in this research to go through all possible sequences in polynomial time [86]. How the Viterbi search algorithm works falls outside the scope of this research, please refer to [86] for a more detailed explanation. From the output N-best list, typically the sequence with the highest probability is considered the recognised phoneme sequence. In Kaldi, decoding is done using an *HCLG* decoding graph (how the *HCLG* works falls outside the scope of this report. For a visual explanation, please refer to [87]).

3 Neural networks and the architectures of this research

As discussed in Section 1, this research builds the first Dutch phoneme recognition (PR) system that uses deep neural networks (DNNs) in the acoustic model. As explained in Section 2, NNs are used in PR to calculate the emission probabilities in the HMMs [88]. In this research, four NN architectures are implemented. The first architecture that is built as a baseline model is a protected bi-directional long-short term memory (BLSTMP) NN. Secondly, an open-gate protected gated recurrent unit (OPGRU) NN is built. Lastly, time-delay neural networks (TDNN) are combined with both the BLSTM and OPGRU, creating two hybrid acoustic models (respectively a TDNN-BLSTM and TDNN-OPGRU).

In this section, the characteristics and training of basic NNs (see Section 3.1), recurrent NNs (RNN) (see Section 3.3), advanced RNNs (see Section 3.4), and Time-delayed NNs (see Section 3.5) are given. Using that information, the NNs used in this research are explained (see Section 3.6).

3.1 Basics of neural networks

A typical, basic NN is a feed-forward NN (FFNN). In Figure 7, a multi-layered FFNN is given, with an input layer, output layer, hidden layers (between the input and output layer), nodes (the blue circles), weights (the connections between the nodes), biases (as small 'b-arrows'), and activation functions (abstractly visualised using lines within the nodes). The data is fed through the network in the direction of the connections (i.e. feed-forward).

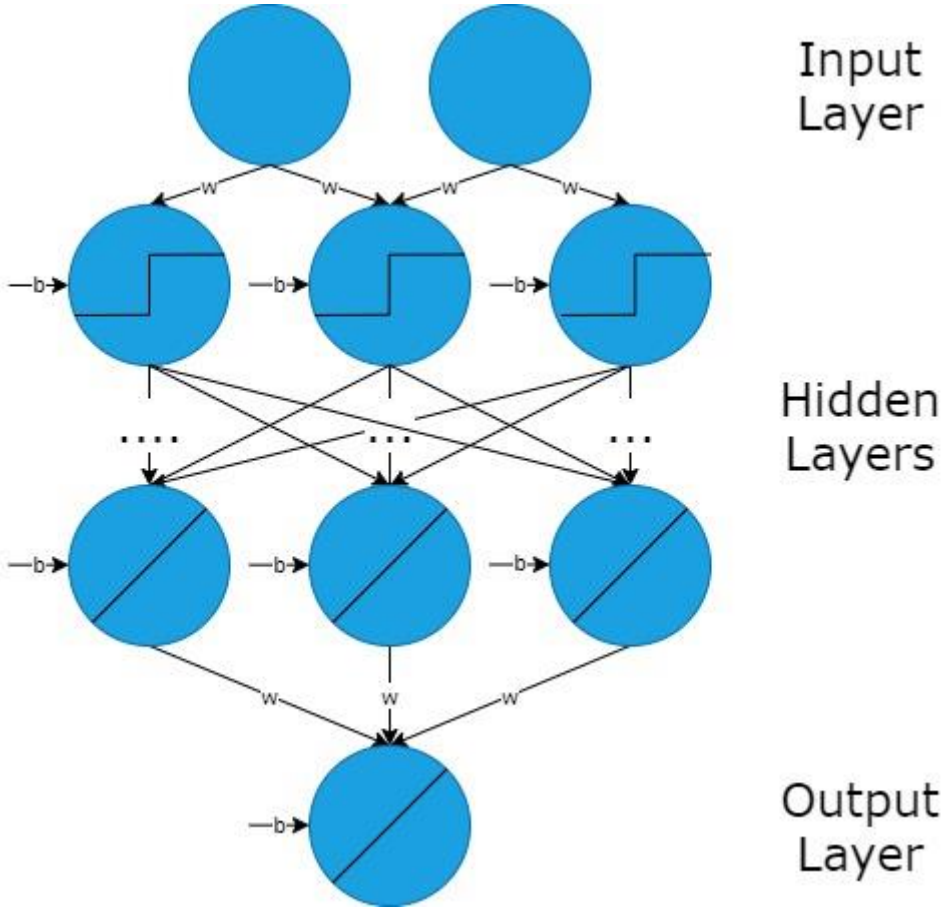


Figure 7: A typical feed-forward neural network with multiple hidden layers and two types of activation functions

An FFNN is a system of connected nodes (i.e. perceptrons). Every node takes input and generates output using the following formula:

Equation 1: Computation in a perceptron

$$y = \varphi(b + \sum_i x_n w_n)$$

Where the output *y* is calculated using the input x_n (which are either the input feature vectors or the output of previous nodes), which is multiplied with the weights w_n (every input x_n has an own weight). Then a bias *b* is added.

The outcome of this is fed into an activation function φ, which ultimately generates the output of a node in the following way. An example of an activation function is a logistic activation function, as visualised in Figure 8. An activation function takes the combination of weights and biases as input and outputs a value between the bounds of the activation function (e.g. in Figure 8 between [0,1]). For example in Figure 8, all input values smaller than -6 will get a 0 as output, while all values greater than 6 will get 1 as output. The values in between -6 and 6 get a value between 0 and 1. When the output is 1 the following connected nodes will get a 1 as input x_n (which again is multiplied by the weights and where a bias is added). For a 0 it is the opposite. The output(s) of the node(s) in the output layer is/are used to generate a prediction of the NN. Finding the optimal weights, biases, and activation functions is vital for the performance of the NN, as this will make the nodes in the network output some value accordingly, which will ultimately affect the output of the last layer, which is used to make the prediction of the NN.

Note that the input of the activation function here is a single value. Later in this research, activation functions are also used with a vector as input, where point-wise for every value is used as input in the activation function. This results in a vector with the same dimensions, only the values are squeezed between the output values of the activation function.

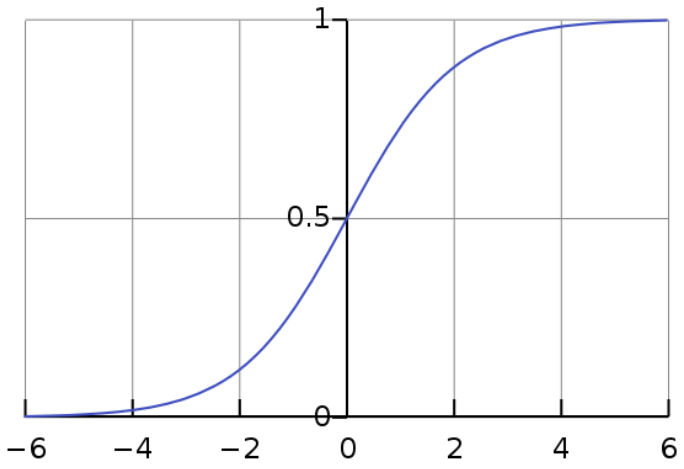


Figure 8: Logistic activation function

3.2 Training a neural network

A NN is an adaptive system; it can change its internal structure (i.e. weights and biases) by training on training data. The goal of changing the internal structure is to make as accurate as possible predictions for new, unseen input. To create accurate predictions, the weights and biases from Equation 1 must be trained. This is done in multiple steps, using forward- and backward propagation.

Forward propagation begins with initialising all weights and biases between the nodes, commonly done using random initialisation. Then during training, in every training step, input feature vectors $X=\{x_1, x_2,\ldots,x_n\}$ are inserted into the FFNN. Every feature vector is used as input one by one. Then, the calculations in every layer of the FFNN from the input layer to the output layer are performed. In the last layer, the output label is predicted (e.g. in PR the emission probabilities). This process is known as forward propagation. For every input feature vector, the predicted output label will be compared with the true output label (i.e. in PR the phoneme label). Using the difference between the true output and the calculated output, a loss is calculated using a designated loss function *J(w)*. The goal is to minimise the loss function, by changing and obtaining the optimal weights and biases.

To optimise all weights in an FFNN, *backpropagation of errors*, shortened backpropagation, is performed. This procedure will be explained using only the optimising of the weights as an example (the same procedure holds for the biases). Backpropagation is the process of calculating the derivatives of all individual weights with respect to the loss function and adjusting the weights accordingly (when optimising the biases: derivative with respect to the biases). The derivative of the loss function shows the rate with which the loss function is changing in value depending on a change in the weights (as visualised with a black dot and dashed line in Figure 9). This is also known as the gradient. Using this slope, it can be determined how the weights can iteratively be adjusted such that the loss of the FFNN moves in the right direction of the optimal value (a bigger gradient means bigger adjustment, smaller gradient means smaller adjustment). If done correctly, the loss function will move to its (local or global) optimum by adjusting the weight correctly. Optimally, the loss will be minimised at the end of the training (i.e. the gradient becomes minimum/zero, marked as $J_{min}(w)$ in Figure 9). This process is called gradient descent[5].

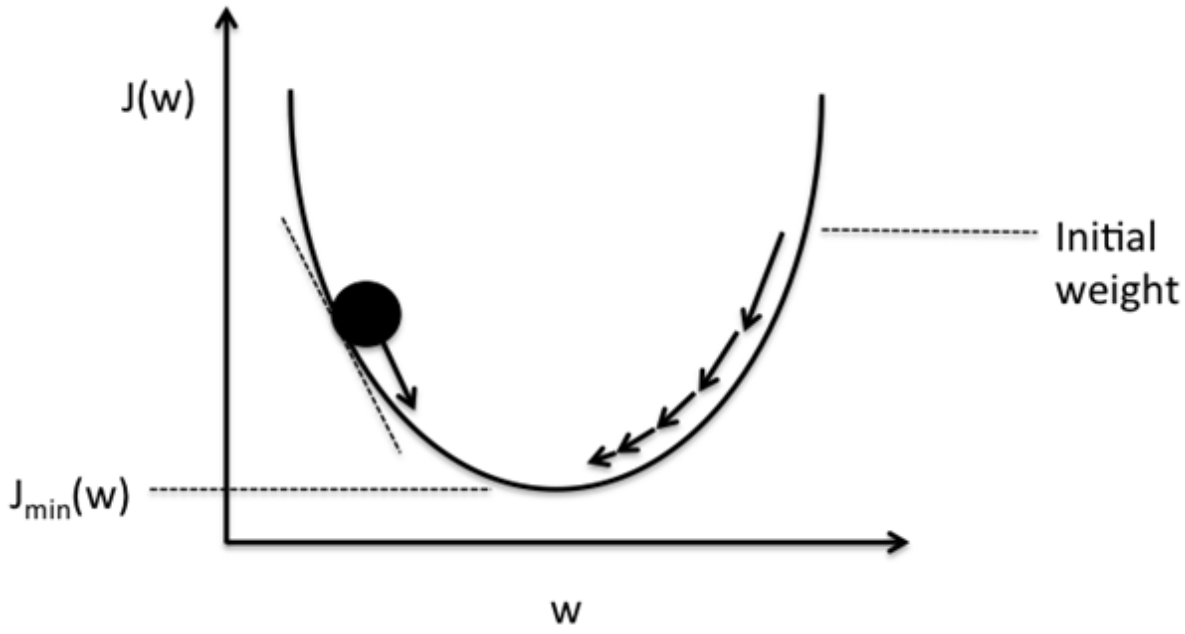


Figure 9: Gradient descent and derivative of the loss function J(w) w.r.t. the weights w

[5] Gradient ascend if the goal is to maximise the loss function

The process of backward propagation starts in the final layer *L* (i.e. output layer). Using the gradient of the loss functions, the weights of the last layer can be adjusted using gradient descent. Then, the weights of layer *L-1* (i.e. the layer before last layer *L,* hence *back*propagation) are adjusted in the same way. This process continues for all the following layers *L-2* to *L = 1*.

After backward propagation is done through all layers, new input can be fed into the NN to perform forward and backward propagation again. This repeats until all input feature vectors are parsed by the NN or a certain threshold is reached (e.g. minimum derivative, minimum loss, etc., which are set as parameters by the researcher).

3.3 Recurrent neural networks

In this research, recurrent neural networks (RNN) are used. RNNs are a different type of NN when compared to FFNN, as the RNNs are differently built architectures that can model sequential data more effectively, as further explained in this section [89]. Sequential data is data with an order in it (e.g. sentences, time-series data, stock data, etc.). Speech is also an example of sequential data models, and therefore, the use of RNNs for PR will be beneficial.

The idea behind RNNs and their capabilities of modelling sequential data are similar to how the human brain works. Say a human sees Figure 10, which is a screenshot taken from a video at time step *t + 1*. A human might not initially guess what the black object on the right side of the red box is.

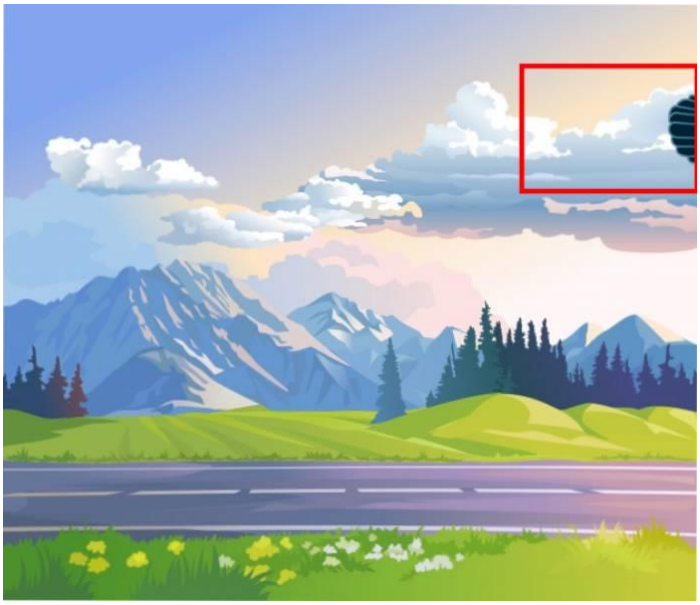


Figure 10: Example sequential data time step t+1

However, when the human sees the image taken from the same video at the previous time step *t* (found in Figure 11), guessing the black object in Figure 10 becomes easier.

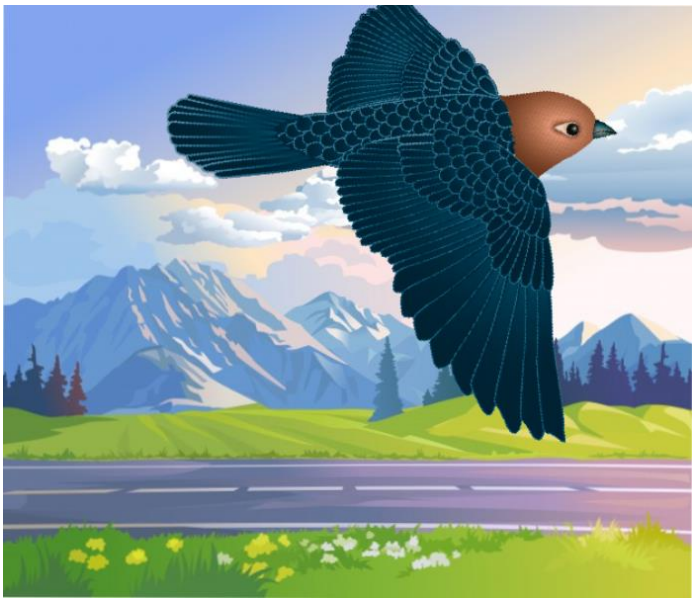


Figure 11: Example sequential data time step t

RNNs work similarly. RNNs take sequential data, split the sequence, and take every individual part of the sequence individually as input. Then for every individual part, the RNN makes a prediction. This prediction is made based on the input of that individual part and on the parts it has already parsed. For instance, when a PR has parsed three phonemes of the word 'hello' (i.e. it has parsed *{h,e,l}*), the RNN will want to predict the following phoneme as /ə/, based on the information it has parsed and information of the current input.

The example in Figure 10 and Figure 11 is not only an example of sequential data (i.e. a sequence of images from a video); it is also an example of how previous data in a sequence can be used to add information to predict the following part of a sequence. Remembering and storing information from parsed parts of a sequence is commonly referred to as sequential memory building. RNNs are good at modelling sequential data, as they contain *sequential memory*.

3.3.1 Sequential memory building

To make accurate predictions, the sequential memory of an RNN must be built. Training an RNN will first be explained on a higher level of abstraction and thereafter more in-depth. An RNN computes the feature vectors X_N which are extracted from the input speech wave (see Section 2.2). Every feature vector x taken from X_N is fed into the RNN one by one in order of the sequence of the time steps. The memory cell takes information from each feature vector and stores it in the sequential memory. This memory can then be used to make predictions based on unseen input (i.e. a bird in a video). A typical RNN architecture at time step t with feature vector x_t, memory state h_t (explained hereafter), and *without* output is visualised on the left of Figure 12. The loop on the left of Figure 12 represents the sequential memory that is looped through all time steps.

Sequential memory is the biggest characteristic of an RNN when compared to FFNNs. In FFNNs, the input is fed into the NN and forward and backward propagation is conducted. Then, a new input is fed into the NN and again backward- and forward propagation is conducted. The weights are adjusted based on the input, but on nothing else (i.e. not using information from the parsed sequence). The most important part missing in FFNNs, which gives them the lack of ability to store memory of the parsed sequence, is a state that can store information of the individual time steps and can be passed through the sequence. In RNNs, at every time step, a *memory state* h_t is updated within the memory cell. This memory state is a vector that contains parameters that are adjusted based on the information from all time steps it has processed. The memory state is passed along to the next time steps.

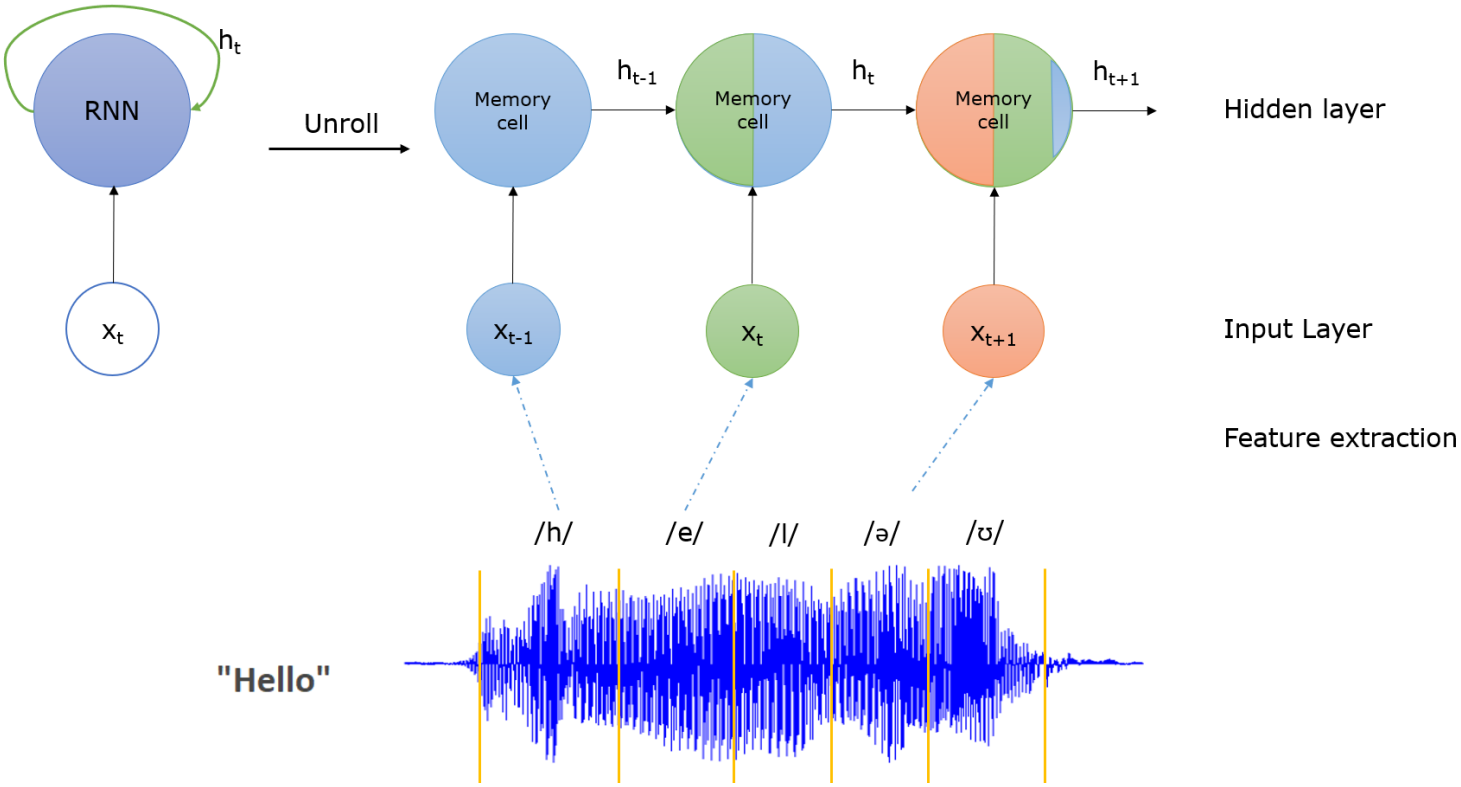


Figure 12: Left: Rolled up RNN architecture. Right: Unrolled RNN architecture

To make this easier to understand, the RNN on the left of Figure 12 is unrolled, as visualised on the right of Figure 12 (unrolled meaning: writing out the network for the whole sequence). Three of the time steps $T = \{t-1,t,t+1\}$ of the RNN are visualised in Figure 12[6]. As abstractly visualised with colours in Figure 12, every time step builds a memory state h containing information that is gained from the input and is passed on to the next time step(s).

The internal processes of the memory cell that build sequential memory h_t are visualised in Figure 13. The memory state of the previous RNN memory cell h_{t-1} is combined with the current input feature vector x_t. The combination of these two is used to update the memory state using the information of the current input x_t.

[6] Note that here, every time frame, and thus feature vector, belongs to one phoneme. It could also occur that multiple frames belong to a single phoneme

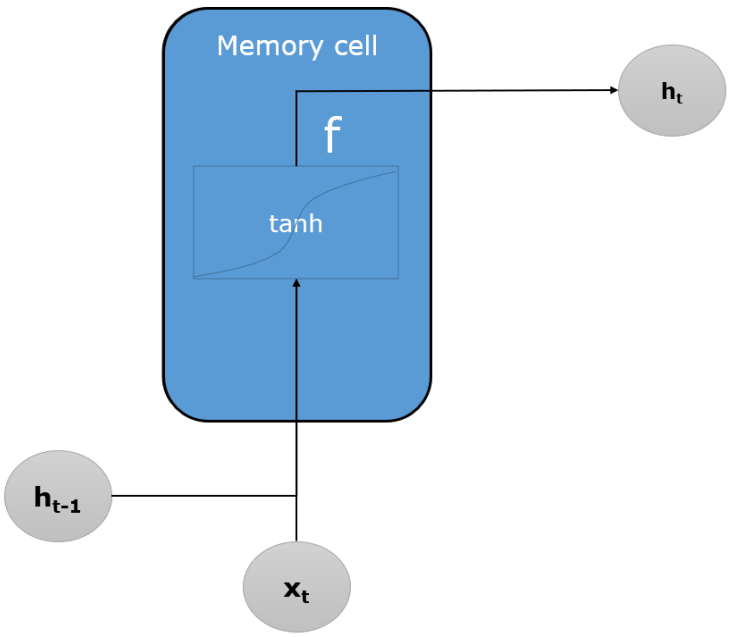


Figure 13: Memory cell of a recurrent neural network

The RNN in Figure 12 is an example of how an RNN processes information and 'kneads' (i.e. adjusts the content of, while keeping the previous information) the memory state with information from the parsed time steps. However, to make a prediction, output needs to be computed. How RNNs compute output is visualised in Figure 14. In Figure 14, a prediction y_t^{pred} is made at every time step based on the input and memory cell. In every memory cell, a NN is present, containing trained weight matrices W_n. The output y_t^{pred} of time step t is calculated using the following formula:

Equation 2: Output formula RNN

$$y_t^{pred} = \alpha(W_1.x_t + W_2.y_{y-1}^{pred}) + b_t$$

Where W_1 and W_2 represent weight matrices in the memory cell, x_t represents the current input, alpha represents an activation function, b_t a bias of the activation function, and $y^{pred}{}_{t-1}$ represents the previous output. Using the output, for every time step, a loss can be calculated using loss function J and the true label y_t, also visualised in Figure 14.

When all time steps and losses are computed, the forward propagation pass is completed. Then, using backpropagation, the parameters in the NNs in the memory cells can be optimised (similar to backpropagation in Section 3.2). These parameters are the weights and biases in Equation 2. As can be seen in Equation 2, the parameters are shared over time, as the output of a time step depends on the output of all previous time steps. As the parameters are passed through all time steps, an RNN back propagates *through time* (BPTT) [90].

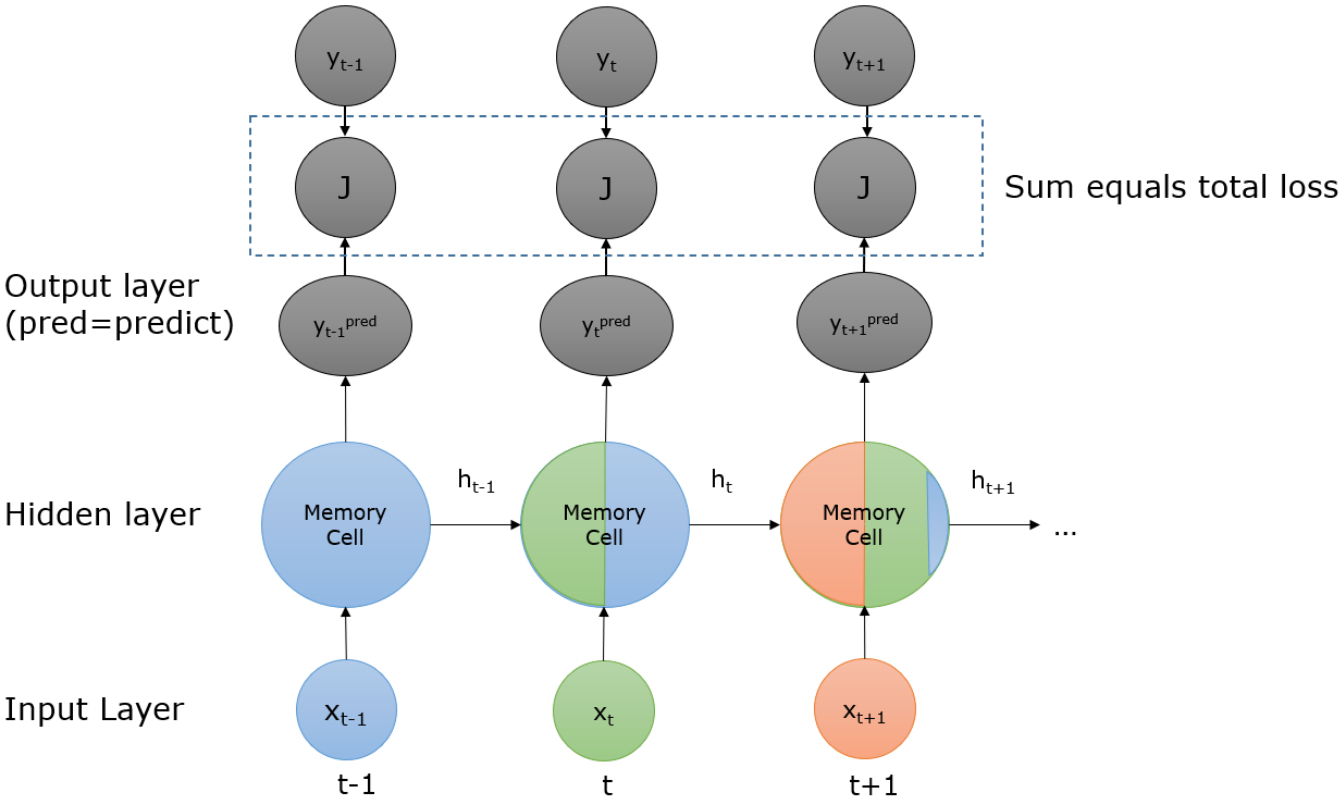


Figure 14: RNN with output and loss

3.3.2 Backpropagation through time and the vanishing gradient

The previous part has explained the strengths of basic RNNs and how they are trained. This research implements advanced RNNs. The reason this research implements advanced RNNs, is that basic RNNs suffer from the vanishing gradient problem during training. The problem of the vanishing gradient is explained in this part to understand why basic RNNs are less- and the advanced models are more effective for PR and thus this research.

As mentioned, RNNs are trained using BPTT. BBPT is similar to BP; however, wherein BP the weights and biases are adjusted using the derivative of the loss function at every time step individually, in BPTT the derivatives of the previous time steps are also used to adjust the weights and biases. BPTT works as follows. In the last layer *L,* at time step *t = T* (where *T* is the last time step of the sequence), the derivative of the loss function is simply taken on $\alpha(W_1.x_t)$ (first part of Equation 2, as y^{pred}_{t-1} equals 0). For the next layer *L-1,* at time step *t=T-1*, the output is calculated the same way as the previous layer, however, y^{pred}_{t-1} will not equal 0. This means that the function of the output of layer *L* is also included in the calculation in *L-1*. The derivative to calculate the gradient in layer *L-1* during BPTT is thus an activation function nested within an activation function, which means the derivatives are taken from functions within functions. Taking the derivative of a function within a function is derived by the chain rule. The chain rule states that the derivative of function *f(g(x))* is:

Equation 3: Chain Rule

$$f\big(g(x)\big) = f'(g(x)) \cdot g'(x).$$

For RNNs, this is a problem.

The function where the derivative is taken from in RNNs during BPTT is commonly a *tanh* function and lies between [-1,1] (visualised as a blue line in Figure 15). This means that the gradient of this function will be between [0,1] (visualised as a magenta line in Figure 15). As the gradient of earlier layers in backpropagation is between [0,1], the gradient in the following stages of backpropagation will become smaller and smaller, as they are multiplied by a factor smaller than 1. When *T* is large, which it does for speech sequences, this results in a gradient that is a factor exponentiated to *T* and will eventually 'vanish' [91].

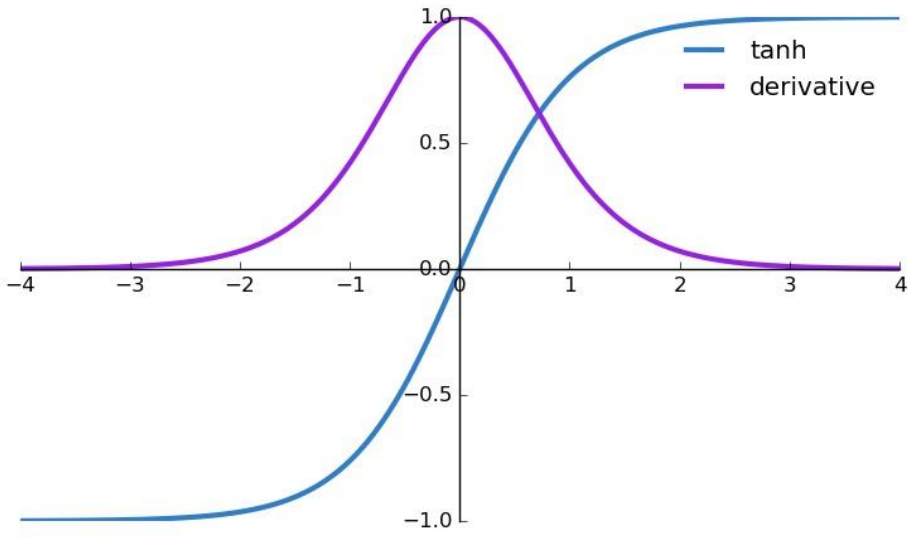


Figure 15: Tanh activation function and the derivative

The weights in the later stages of BPTT are adjusted based on the size of the gradient As the gradients vanish, the weights are adjusted only slightly/insignificantly. This results in long-term dependencies not being able to be learned. This phenomenon is called the vanishing gradient problem [92].

3.4 Beyond vanilla Recurrent Neural Networks

Researchers have designed advanced RNN models that solve the problem of the vanishing gradient. One advancement was made with the introduction of gates. Examples of RNNs with gates are long-short term memory RNNs (LSTM) [93] and Gated Recurrent Units RNNs (GRUs) [94]. As variants of these networks will be built in this research, the basics of gates, LSTMs, and GRUs are explained in this section[7]. The function and variety of gates are explained using an LSTM as an example (see Section 3.4.1). Thereafter, gates and GRUs are further explained (see Section 3.4.2).

Basic RNNs are unidirectional; a time step *t* is only predicted using the information of the parsed sequence from time steps *1* to *t-1*. However, the future context could also be important. This can be captured using bi-directional architectures. These are also used in this research and explained in Section 3.4.3.

[7] Note that the visualisations of the cells are simplifications to make the operations more clear. For the full cells and their internal operations, please refer to [178] for an illustrated guide and the references throughout this section for the original papers

3.4.1 LSTMs and the basics of gated-RNNs

Gates and gated-RNNs will be explained in this part using an LSTM as an example. In Figure 16, a typical LSTM architecture is given on the left and an unrolled LSTM is given on the right. Most components in Figure 16 are similar to the basic RNN components. The difference with basic RNNs is the addition of a cell state c_t. The cell state is a vector containing long- and short-term information on the sequence. Whereas in basic RNNs the complete 'internal memory' (i.e. h_t) is overwritten at every timestep and long-term dependencies are lost due to the vanishing gradient, in the LSTM it is decided what to keep and forget using gates. This information can then be stored in the cell state. This gives the LSTM the ability to model long-term dependencies more effectively, solving the vanishing gradient problem [95]. The overall idea is that the gates in the LSTM cells learn what data are important and must be kept, and what data are less important and can be forgotten. The process of how all states and gates operate and how long-term dependencies are stored is further explained now.

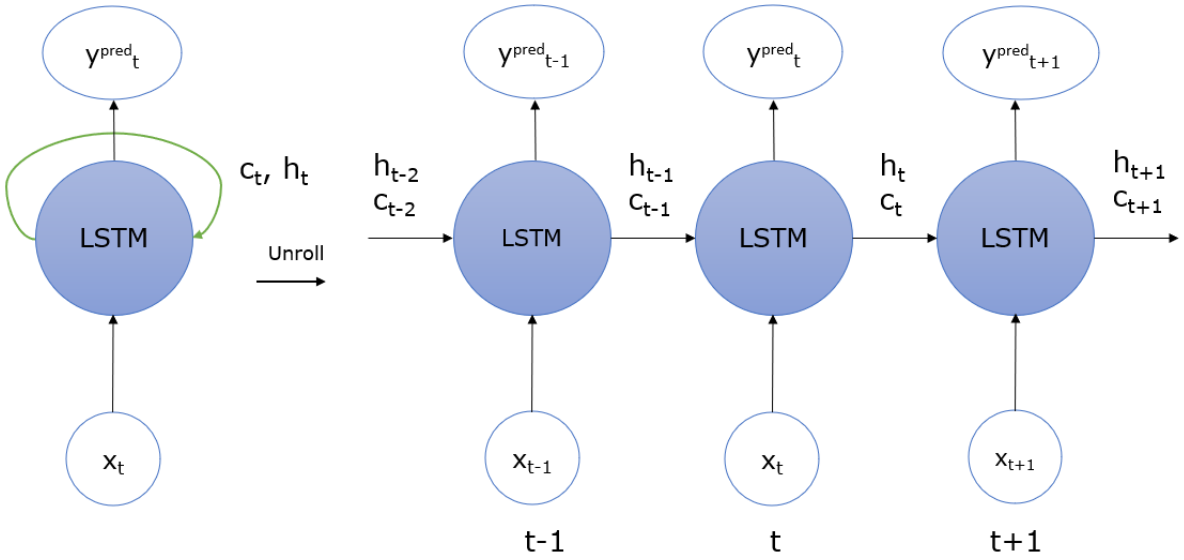


Figure 16: Left: Rolled up LSTM architecture
Right: Unrolled LSTM architecture

The gates in the LSTMs cell are visualised in Figure 17. The global operations performed to create the memory- and cell state are visualised in a memory cell in Figure 17. As can be seen in Figure 17, the LSTM memory cell contains three gates: the forget gate *f*, the input gate *i*, and the output gate *O*. For input of every cell, the hidden state of the previous LSTM cell h_{t-1} (which is a vector) is concatenated with the current input feature vector x_t. This concatenated vector is used as input in the gates to update the hidden state and cell state. The numbers in Figure 17 indicate the dimensions of the vectors that move over the lines. The dimensions chosen in Figure 17 are arbitrary, however, they are useful to indicate the input and output of the several functions of the LSTM cell.

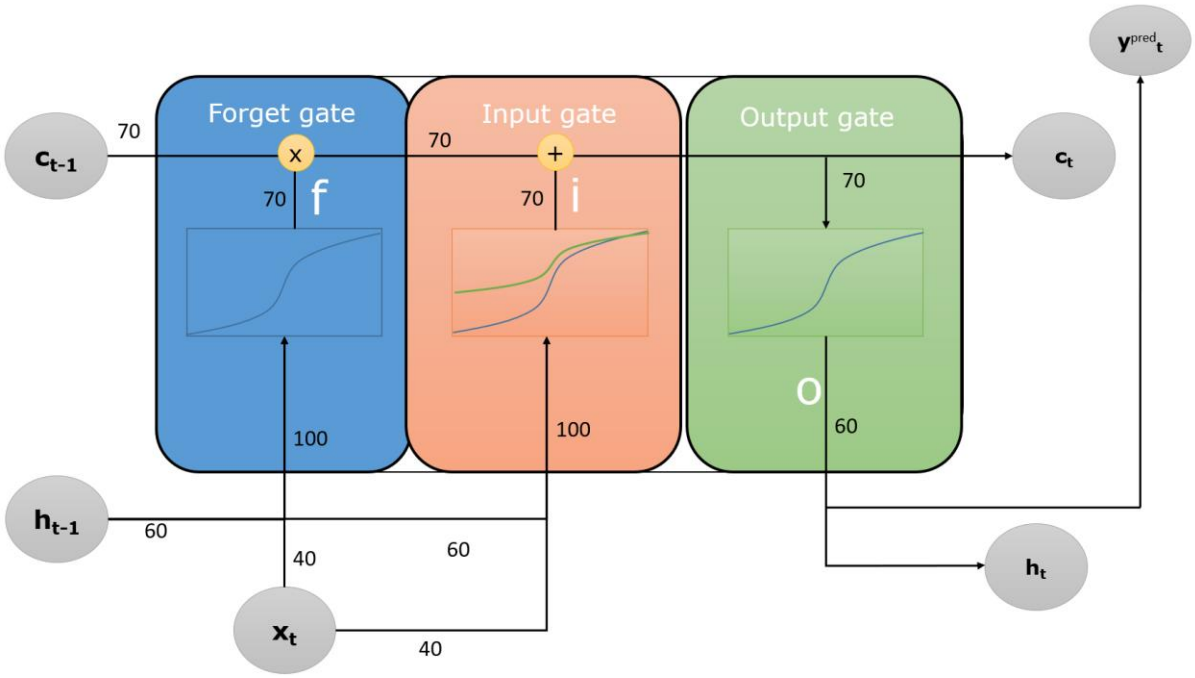


Figure 17: LSTM cell with different gates, input, and output. The numbers indicate the dimensions of the vectors that pass through the cell

The core idea is that the cell state *c* vector runs through the different gates while being updated by linear interactions ('x' and '+' circles) in the gates. This will be explained now (the full mathematical methodology falls outside the scope of this project. For the full equations, please refer to [93]).

In the forget gate, as visualised as a blue square in Figure 17, the LSTM decides what information to forget. It adjusts the values in the cell state accordingly using a trained sigmoid function. This trained sigmoid function outputs a vector with the same dimensionality as the cell-state and the values of the vector are between 0 and 1. This output vector is used to update the cell-state using point-wise multiplication, where the values closer to 0 forget the information in the cell-state and values closer to 1 keep the information in the cell-state.

After updating the cell-state on what to forget, the next step is to add new information to the cell-state. This happens in the input gate, visualised as the orange square in Figure 17. In this gate, the cell-state vector is updated on what information is useful to remember by combining the output vectors of a trained sigmoid and tanh function. This vector has the same dimensions as the cell-state and is added to the cell-state by point-wise addition. The cell-state is now point-wise updated on useful information to remember.

After updating the cell-state (which serves as long-term memory), the hidden state h_{t-1} must be updated to create h_t. h_t can be used to make predictions y_t^{pred} for the current input x_t (just like in basic RNNs). This update of h_{t-1} *to* h_t is performed in the output gate using a sigmoid function.

To put this all in the context of PR, the task of the LSTM is to predict all phonemes in a phoneme sequence. At the beginning of the phoneme sequence, the speaker might have pronounced an /x/ as a /ɣ/, which is the 'soft-g' in Dutch Southern accents. If the LSTM is trained to recognise accents and adjust its predictions accordingly, the cell-state is updated in the input gate. This can enable future predictions to recognise other Southern characteristics more effectively. However, if a non-Southern person starts speaking after a certain time in the speech segment, the cell-state must be updated. This is done by forgetting the Southern accent information in the forget gate and updating the accent style to the accent of the new speaker.

Ultimately, the cell-state can learn long-term memories, which solves the problem of the vanishing gradient descent (the mathematical reasoning behind solving the vanishing gradient falls outside the scope of this project, please refer to [96].

3.4.2 Gated-RNN variants

Another RNN variant that uses gates to forget and remember relevant information in the cell state is the Gated Recurrent Unit (GRU). The GRU is a simplified version of the LSTM [97]. An example of a GRU cell is visualised in Figure 18.

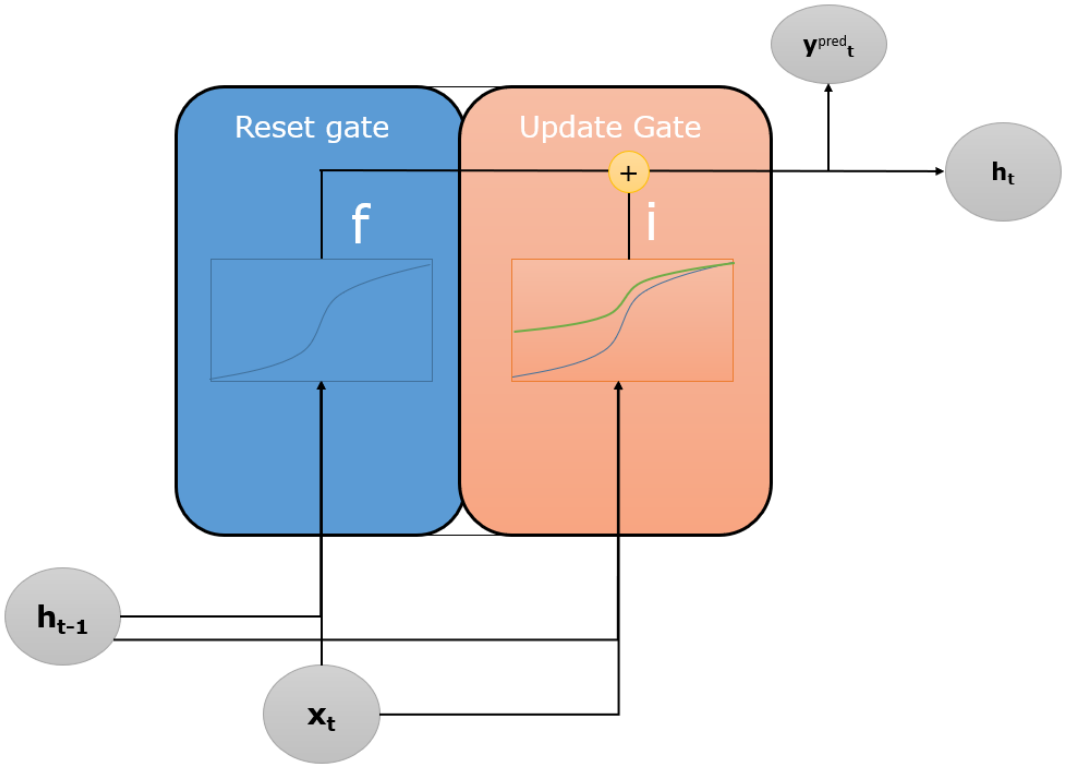


Figure 18: Gated Recurrent Unit cell

In Figure 18, a GRU is different to the LSTM, as it has two, instead of three gates: an update- and reset gate. The 'update gate' decides what information from previous time steps must be remembered for future predictions [67]. The 'reset gate' is similar to the 'forget gate'; it models how much information to forget. Finally, the cell state c_t is removed and all memory is now stored in the memory state h_t. This seems similar to the basic RNNs (as no cell-state is available for long-term dependencies), however, the gates solve the vanishing gradient by keeping the values between [-1,1] (in tanh function in the update gate).

Both the LSTM as the GRU perform well on sequential data and they can outperform each other depending on the task and dataset [98]. Since GRUs contain fewer gates, fewer parameters need to be trained, which will make training less computationally expensive but also making them more effective for smaller datasets when compared to LSTMs. However, when large datasets are available, LSTMs might outperform GRUs, as more parameters and thus more complex dependencies can be learned by the LSTM [98]. In this research, advanced versions of both LSTMs and GRUs are implemented.

In this project, gated-RNNs are implemented that contain protected gates. Protected gated RNNs are gated-RNNs that optimise speed and performance by adding an extra gate in the memory cell [99]. An example of an RNN with added protected gates is a GRU-projection (GRUP), as visualised in Figure 19.

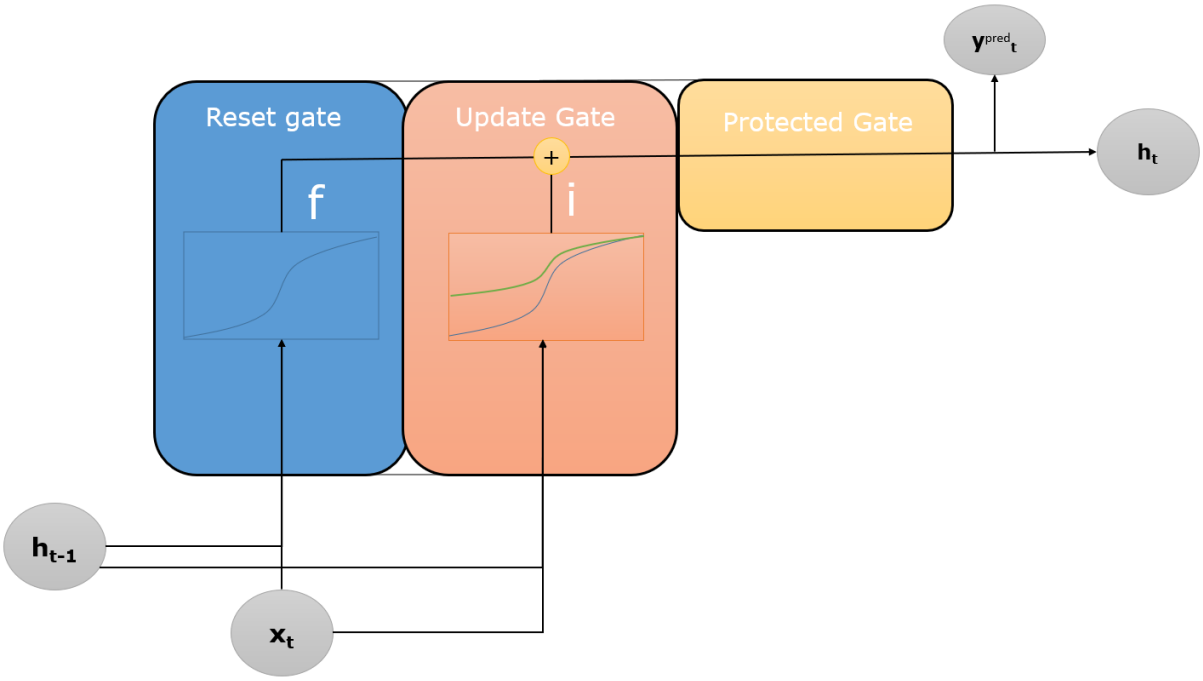


Figure 19: Protected Gated GRU cell

In RNNs, adding more memory cells to the NN architecture can increase performance. In all of these memory cells, a large number of weight matrices are present (see Equation 2). Let's say, per memory cell, *N* parameters are present in the weight matrices. As can be seen in Equation 2, weight matrices are multiplied with each other through every layer. Ultimately, the numbers of parameters in the total architecture grow in a square pattern, which results in computationally expensive operations [99]. To reduce this squared factor, protection layers are added to the memory cell. The protection layer is added at the end of a memory cell, where the 'normal' output of the memory cell is fed through the protection layer to reduce the number of parameters *N*, visualised in Figure 19. The result is a non-squared growth of parameters when memory cells are added. This has shown improved recognition results for ASR and PR and will thus be implemented in this research [100, 101].

3.4.3 Bi-directional RNNs

Besides protected gated-RNNs, this research will also implement a bidirectional RNN, i.e. the bidirectional LSTM (BSLTM). BLSTMs are architectures that are designed based on the idea that a time step in a sequence does not only depend on the previous context of the sequence but also future context. For example, the LSTM explained earlier is a unidirectional RNN that only preserves information from the past, as predictions in the time steps are only based on preceding time steps. An example of a situation where a BLSTM is useful is when it is trying to predict the next phoneme in the following sentence: "/h/ /e/…" Predicting the next phoneme in this sentence is a guess without any other context (e.g. /m/, /l/, /v/, etc.). However, when the part after this time step is predicted as: "…/ə/ /ʊ/." predicting the phoneme becomes easier (i.e./l/).

BLSTMs do this by combining two types of layers in RNNs, where one type of layer has as input the sequence in normal time-order, and the other has the sequence in reverse time-order. An example of a BLSTM network containing multiple LSTM cells is visualised in Figure 20. Then, the forward and backpropagation algorithm (as discussed in Section 3.1) is done twice; first on the forward RNN layers

and then on the reverse RNN layers. The output of the hidden states of both the past and future are combined and can be used to compute the output y^{pred}_t similarly as the output is predicted for the unidirectional LSTM.

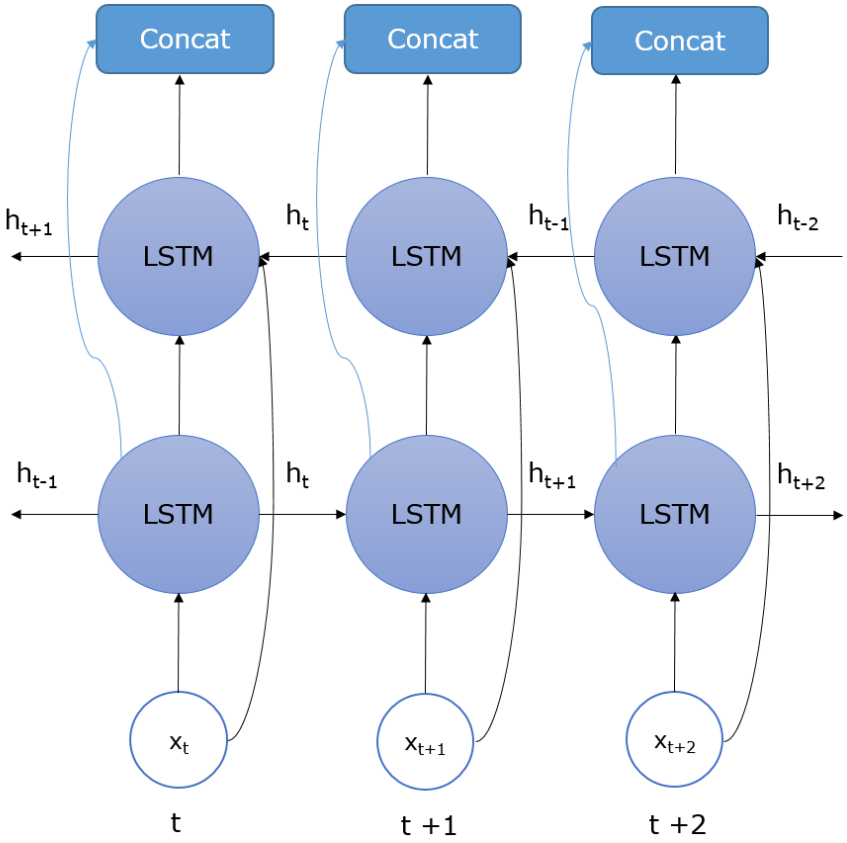


Figure 20: Bi-directional LSTM architecture

3.5 Time-delay Neural Networks

A different type of NN is the Time Delay Neural Networks (TDNN). Originally introduced in 1990, TDNNs are FFNNs that add temporal context when making a prediction [102]. Temporal context can be added in TDNNs as they have a delay element associated with them, which can capture contextual dependencies. Both RNNs and TDNNs compute an output based on the parsed sequence. The difference with RNNs is that RNNs use the complete parsed sequence (i.e. time steps *1 to t*), while TDNNs use time steps *t-r* to *t* (where *r* is a parameter of choice).

An example of a TDNN architecture can be found in Figure 21. The TDNN in Figure 21 maps a finite sequence $\{x_{t-r}, x_{t-\dots}, x_{t-2}, x_{t-1}, x_t\}$ to output y_t . Every input x_t trains weights similarly to how FFNNs are trained (as discussed in Section 3.1). The difference with basic FFNN is that for every time step *t*, not only the weight matrix of time step *t* is used to make the prediction y_t, but the average of *r* weights matrices of the *r* previous time steps are used. This means that the TDNN stores the weights of the *r* previous time steps to predict the current time step (i.e. the weights matrices of the previous time steps are 'delayed', hence the name Time *Delayed* Neural Networks). Using the average of the previous and current weight matrices, temporal context can be used to predict the output for time step *t*. In modern TDNNs, also future context can be added to gain additional temporal context [103].

The number of layers of the TDNN decides the length of the temporal context and is a parameter that can be optimised. The lower layers capture short-term context, while the deeper layers capture

longer-term context. The strength of TDNNs for PR comes from the characteristic that important information of a time step *t* lies in a relatively narrow context of that time step [104].

When comparing the performance of multi-layer TDNNs with other NN architectures containing only DNN or RNN layers, the latter have shown better results for ASR [105]. Recent work has shown that adding TDNN layers to existing gated-RNN models can improve recognition accuracy, as the TDNN layers can capture relatively short term-dependencies effectively, while the RNN layers can model the relatively long-short dependencies more effectively [8] [106]. These types of models are called hybrid models, which will be implemented in this research and are further explained in Section 3.6.3.

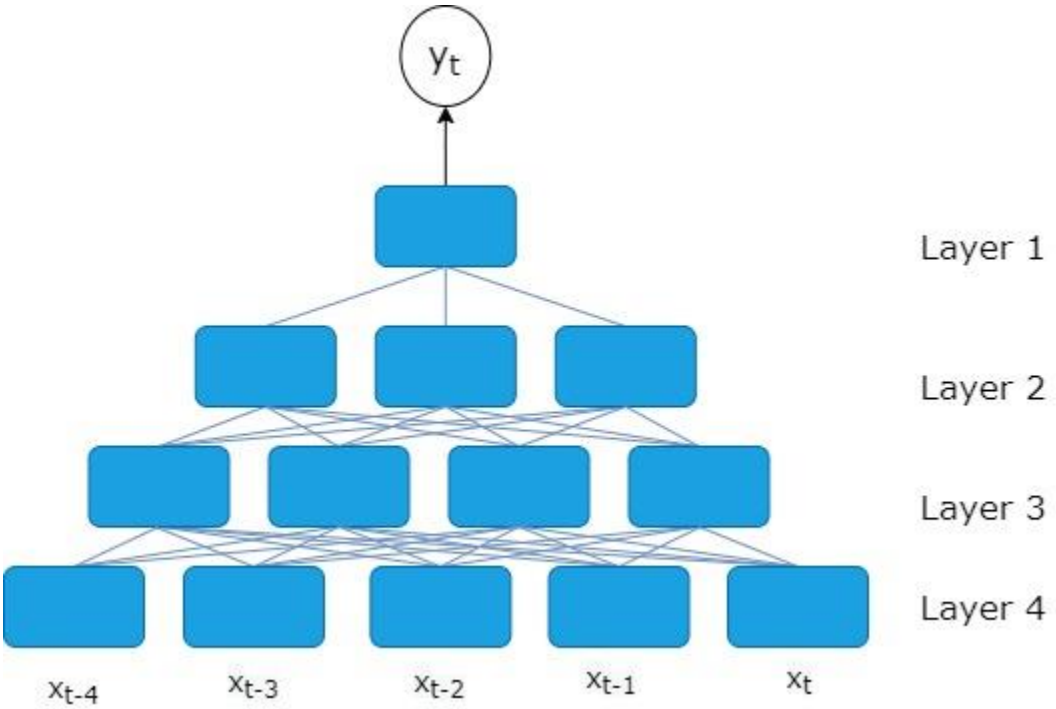


Figure 21: Typical TDNN Architecture

[8] Relatively when comparing the dependencies of TDNN and RNN layers with each other. When comparing TDNN and FFNN, the TDNN can capture long-term dependencies more effectively.

3.6 Architectures of this research

In this part, the four NN architectures that are implemented in this research are presented. First, a BLSTM is built as the baseline (see Section 3.6.1). The second architecture is an OPGRU model (see Section 3.6.2). Finally, both the BLSTM and OPGRU are combined into hybrid acoustic models using TDNN layers, as TDNN-BLSTMs and TDNN-OPGRUs (see Section 3.6.3)

3.6.1 Projected Bi-directional LSTM

The projected bi-directional BLSTM (BLSTMP) is a combination of bi-directional a BLSTM with added protected gates within the LSTM layers. An example of a basic BLSTMP can be found in Figure 22. The BLSTMP cell is composed of a BLSTM cell and protected gates, which functions and uses have both previously been explained in Section 3.4. In the rest of this report, BLSTMPs are referred to as BLSTMs, as this is commonly done in existing work[9].

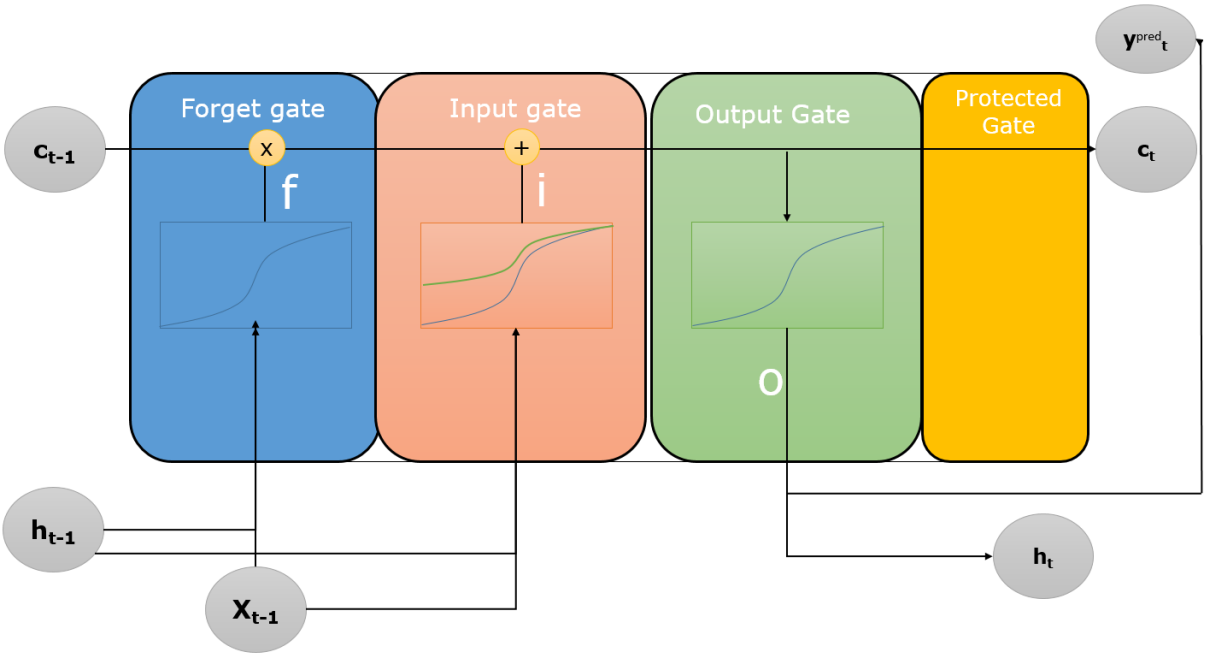


Figure 22: Protected-BLSTM cell

3.6.2 Output-gate projected GRU

The second type of architecture that is investigated is the Open-gate Projected Gated Recurrent Unit (OPGRU). This is an enhanced version of GRU, as explained and visualised in Section 3.4.2. Protected GRUs (PGRU) accelerate decoding and increase decoding accuracy by adding protected gates [106]. However, when compared to the addition of protected gates in BLSTMPs, in the case of PGRUs, the addition of protected gates is not only beneficial, it also produces problems [106]. As explained before, a reset gate in a GRU gives the model the ability to delete past, redundant memory. A problem that arises with PGRUs is that the reset gate and protected gate cause the PGRU to saturate (i.e. reset gates remove all information from previous layers). This results in a badly performing model [106]. Open-gate PGRUs (OPGRU) solve this problem by replacing the reset gate with an output gate, which regulates the flow information by limiting the number of states that can be adjusted in every layer. This has resulted in improved performance [106]. However, improved performance is only met when

[9] E.g. in all Kaldi related research, the basic Kaldi scripts are used. These contain protected layers, yet the research still names them just BLSTM

a large dataset is used to train the model, as this type of structured requires complex parameter training. As this research uses a large dataset, an OPGRU is implemented.

3.6.3 Hybrid models: Time-Delayed BLSTM and OPGRU

Hybrid acoustic models are NNs that combine different types of architecture layers. An example is visualised in Figure 23. In Figure 23, a TDNN-BLSTM hybrid model can be found, which is built by combining forward and backward layers of the BLSTM on top of TDNN layers. The input is first fed into the TDNN layers, which will process the information and parse it to the next layers. Using hybrid models, different strengths of the individual models can be utilised in a single NN. For instance, the strengths of BLSLTMs and OPGRUs are to capture long-term dependencies, while TDNNs can capture detailed information in a narrower context. Combining both has proven to increase accuracy when compared to the networks individually, while also reducing training time [77, 78].

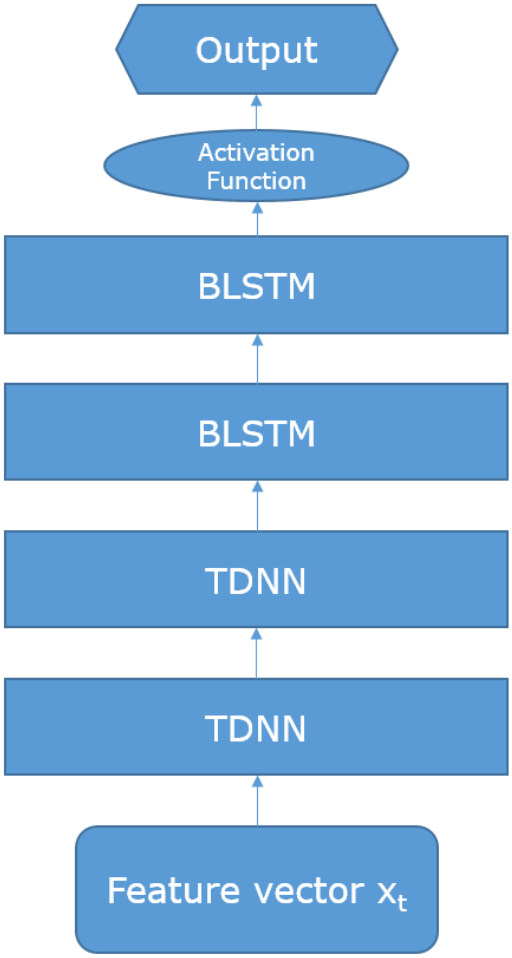


Figure 23: Example of a hybrid TDNN-BLSTM architecture

4 Experimental setup

In this section, the experimental setup of this research is given. First, the features used in this research are given (see Section 4.1). Then, the lexicon (See Section 4.2) and the language model (see Section 4.3) details are given. After this, the (parameter) settings of the neural networks are provided (see Section 4.4). Thereafter, the evaluation metrics and error analysis methodology is described (see Section 4.5). Then, the ASR toolkit Kaldi, used in this research, is given (see Section 4.6). Finally, the dataset specifications are given (see Section 4.7)

4.1 Features used in this research

This research uses two different features: mel-filterbanks and I-Vector features (see Section 2.2). The 40-dimensional MFB features and the 100-dimensional I-vector features are both extracted using existing Kaldi scripts.

The MFB and I-vector features are concatenated, meaning that the two separate feature vectors are stacked on top of each other creating 140-dimensional feature vectors. These feature vectors are then used as input.

4.2 Lexicon of this research

As discussed in Section 2.3, the lexicon in this research only consists of individual phonemes. The complete lexicon is shown in Table 3. This notation is based on CGN phonemic notation (examples of CGN phonemes in words can be found in [107]). Additional entries are 'Spoken Noise' ([SPN]), which can be used to transcribe parts of speech where there is noise but no actual phonemes and the silence (<SIL>), which is used to indicate silences in the speech audio.

Table 4: CGN phonetic lexicon and lexicon of this research

<SIL>	A+	u	[SPN]	z	w	i:	r:
@	E+	y	G	j	a:	j:	s:
A	Y+	k	h	l	Y:	k:	t:
E	a	p	x	m	d:	m:	u:
I	e	t	f	n	e:	n:	v:
O	i	b	v	r	g:	o:	w:
Y	o	d	s	N	h:	p:	z:

4.3 Language model details

This research uses a bigram model for the language model (LM). The bigram LM used in this research is trained on all 1.1 million-text transcriptions of CGN. Since this research focuses on PR, phoneme-level transcriptions are needed for the LM to calculate the probability of the phoneme bigrams. CGN only contains word-level transcriptions (see Section 4.7); therefore, a script was built that translated the word-level transcriptions into phonemic transcriptions. An example can be found in Table 5, where the word-level sentence "Hello you" is transformed into the phoneme-level transcription /h e l ə ʊ j u:/, using the pronunciations that come with the CGN corpus of both words. The resulting phonemic transcriptions are fed into the training procedure of the bigram.

Table 5: Example of converting word-level transcriptions to phoneme-level transcriptions

Word-level transcription	CGN Lexicon entries	Phone-level transcription
"Hello you"	Hello - /h e l ə ʊ/	/h e l ə ʊ j u:/
	You - /j/ u:/	

4.4 Acoustic model details

In this part, the parameters for the neural networks are given. All architectures use the LF-MMI objective function (i.e. loss function) during training, as it has proven to train faster and yield better results for PR when compared to the more commonly chosen cross-entropy objective function (for a more detailed explanation, please refer to [101]).

4.4.1 Baseline: BLSTM parameter setting

The parameters for the baseline BLSTM are based on related work:

- 3 BLSTM layers [108]
- Amount of cells per BLSTM layer (Cell-Dimension): 1024 [109]
- Each layer equal amount of cells [110]
- Projected recurrence and projected non-recurrence: 256 [111]
- Dropout schedule: 0,0@0.20,0.3@0.50,0 [112]
- L2 Regularisation: 0.00005 [113]
- Mini-batch: 128 [110]
- Epochs=6

4.4.2 OPGRU parameter setting

The parameters for the OPGRU are based on related work:

- 3 OPGRU layers [108]
- Amount of cells per OPGRU layer (Cell-Dimension): 1024 [109]
- Projected recurrence and projected non-recurrence: 256 [111]
- Dropout schedule: 0,0@0.20,0.3@0.50,0 [112]
- L2 Regularisation: 0.00005 [113]
- Mini-batch: 128 [110]
- Epochs=6

4.4.3 TDNN-BLSTM parameter setting

The parameters for the TDNN-BLSTM are based on related work:

- 3 LSTM layers [108]
- 7 TDNN layers [114]

- Amount of cells per OPGRU layer (Cell-Dimension): 1024 [109]
- Each layer equal amount of cells [110]
- Dropout schedule: 0,0@0.20,0.3@0.50,0 [112]
- Mini-batch: 128 [110]
- Epochs=6
- L2 Regularisation: 0.00005 [113]
- Mini-batch: 128 [110]

4.4.4 TDDN-OPGRU parameter setting

The parameters for the TDNN-OPGRU are based on related work:

- 3 OPGRU layers [108]
- 7 TDNN layers [114]
- Context: 5 for 5
- Amount of cells per OPGRU layer (Cell-Dimension): 1024 [109]
- Each layer equal amount of cells [110]
- Dropout schedule: 0,0@0.20,0.3@0.50,0 [112]
- Mini-batch: 128 [110]
- Epochs=6
- L2 Regularisation: 0.00005 [113]

4.5 Evaluation

Evaluation is conducted in three parts. First, the phoneme error rates (PER) of the architectures are calculated to analyse the overall performance of the architectures (see Section 4.5.1). In the second part, the PER is decomposed into the percentage phoneme insertions, deletions, and substitutions. Both the overall PER as the decomposed PER are used to analyse the weaknesses/strengths of the different architectures (answering sub-research question 1).

Thereafter, in the third and the last part of the evaluation, individual phonemes are analysed using confusion matrices (see Section 4.5.2) and phoneme error analysis (see Section 4.5.3). The confusion matrices show the number of deletions and insertions of the individual phonemes and with which phonemes they are substituted. The error analysis can show which phonemes are identified as error-prone phonemes (answering sub-research question 3). The results can be used to provide suggestions for future research on improving phoneme recognition.

All the above evaluation parts will first be conducted on the 15 speaking style test set. Thereafter the same methodology is used to compare the performances of the models on the clear and spontaneous speech test sets (answering sub-research question 2). The results of the different speaking styles produce insights into what architectures are more effective for specific applications.

4.5.1 Phoneme error rate

The main evaluation metric that is used to evaluate the performance of the architectures is the PER. The PER is based on the "Levenshtein Distance" metric, which is a metric that determines the difference between two sequences; in this research the difference between the predicted and ground-truth transcription.

The "Levenshtein Distance" metric is calculated using three components: the number of deletions, substitutions, and insertions needed to match the predicted output to the ground-truth transcription. Deletions are those phonemes that are not present in the ground truth but are present in the predicted phoneme sequence. Deletions are those phonemes that are not present in the ground truth but are not present in the predicted phoneme sequence. Insertions are those phonemes that are present in the ground truth but are not present in the predicted phoneme sequence. The number of substitutions represents the number of phonemes that must be changed for other phonemes in the ground-truth transcription to match the predicted transcription. The final PER is calculated using the following formula:

Equation 4: Phoneme error rate

$$PER = \frac{\#Substitutions_{phoneme_{all}} + \#Insertions_{phoneme_{all}} + \#Deletions_{phoneme_{all}}}{\#occurrencesingroundtruth\ _{phoneme_{all}}}$$

This formula shows the PER as the sum of the number of substitutions, insertions, deletions for all phonemes divided by the number of occurrences of *all* phonemes in the ground truth.

Besides that, the percentages of substitutions, insertions, and deletions, are calculated separately, which will provide insights into the strengths and weaknesses of the various architectures. The three metrics are calculated using the following three formulas:

Equation 5: Decomposed PER: %Substitutions, %Insertions, %Deletions

$$\%Substitutions = \frac{\#Substitutions_{phoneme_{all}}}{\#occurrencesingroundtruth\ _{phoneme_{all}}}$$

$$\%Insertions = \frac{\#Insertions_{phoneme_{all}}}{\#occurrencesingroundtruth\ _{phoneme_{all}}}$$

$$\%Deletions = \frac{\#Deletions_{phoneme_{all}}}{\#occurrencesingroundtruth\ _{phoneme_{all}}}$$

The sum of these percentages per architecture equals the total PER of that architecture.

4.5.2 Confusion matrices

The errors of the individual phonemes are further analysed using confusion matrices, which visualise the insertions, deletions, and substitutions per individual phoneme in a matrix. An example can be found in Table 6, where each row and column represents a phoneme. The diagonal gives the count of correctly classified phonemes. Rows show the phoneme in the ground truth transcriptions, and the columns show the number of that particular phoneme being classified as the phoneme of the column (i.e. the number of substitutions). The second to last column indicates deletions (e.g. in Table 6: from all the errors of phoneme /a/, 20% is due to deletions). The last column indicates the number of insertions. Finally, all rows are normalised.

Table 6: Example of a confusion matrix

	/a/	**/b/**	**/c/**	**<del>**	**<ins>**
/a/	0.5	0.1	0.0	0.2	0.2
/b/	0.1	0.6	0.1	0.2	0.0
/c/	0.0	0.1	0.7	0.5	0.1

The results of the three experiments of the four models in this research can be used to make confusion matrices. These matrices are used to compare the phoneme errors of the different architectures and different speaking styles.

4.5.3 Individual phoneme error analysis

The main goal of this project is to research how Dutch PR can be improved. The third sub-research question aims to find error-prone phonemes. Error-prone phonemes are found in this research using two metrics: $PER_{phonemex}$ and $\%ContributionToPER_{phonemex}$.

Individually badly performing phonemes can be found using the $PER_{phonemex}$ statistic. The $PER_{phonemex}$ statistic calculates the percentage of phoneme *x* being wrongly predicted against the number of occurrences of the phoneme *x* in the ground truth. The $PER_{phonemex}$ statistic is calculated using the following formula:

Equation 6: Individual phoneme error rate of a phoneme x

$$PER_{phoneme_x} = \frac{\#Substitutions_{phoneme_x} + \#Insertions_{phoneme_x} + \#Deletions_{phoneme_x}}{\#occurrencesingroundtruth\ _{phoneme_x}}$$

Where the enumerator sums the number of errors of phoneme *x*, representing the number of times phoneme *x* is classified wrongly and $\#phonemesingroundtruth_{phonemex}$ represents the number of times phoneme *x* is present in the ground truth. For example, if the phoneme */a/* has 100 appearances in the test set and 10 of those are classified wrongly, the result is $PER_{/a/}$ = 10/100=10%.

Improving the performance of some phonemes with a high $PER_{phonemex}$ might not always result in as high improvement of the overall PER performance. This is because the contribution of phonemes with little occurrences in the transcriptions have little contribution to the overall PER of the models (see Equation 4). Therefore, also the $\%ContributionToPER_{phonemex}$ statistic is used to identify error-prone phonemes. This statistic calculates the total percentage each phoneme contributes to the overall PER of the architectures, by dividing the number of errors of a phoneme *x* by the total number of errors of all phonemes. $\%ContributionToPER_{phonemex}$ is calculated using the following formula:

Equation 7: Individual phoneme x contribution to the overall error of the models

$$\%ContributionToPER_{phoneme_x} = \frac{\#Substitutions_{phoneme_x} + \#Insertions_{phoneme_x} + \#Deletions_{phoneme_x}}{\#Substitutions_{phoneme_{all}} + \#Insertions_{phoneme_{all}} + \#Deletions_{phoneme_{all}}}$$

Where the enumerator sums the number of errors of phoneme *x* and the denominator sums the number of errors of all phonemes in the test set (equal to the numerator in Equation 4). For instance, if there are 100 phonemes misclassified in the total test set, and phoneme /a/ is misclassified 40 times, $\%ContributionToPER_{/a/}$ is 40/100=40%.

4.6 ASR toolkit: Kaldi

The Kaldi toolkit is a state-of-the-art ASR toolkit written mainly in Shell bash and C++. It is used to create the forced-alignments, extract the features, build the language model, build and train the acoustic model architectures, and perform decoding [115].

4.7 Dataset used in this research

The dataset used in this research is the Corpus Gesproken Nederlands (CGN) [40]. This large, online available dataset consists of 900 hours of word-level annotated Dutch and Flemish Dutch speech. This dataset can also be used for PR research, as forced alignment procedures (see Section 4.7.1) will transform these word-level transcriptions into time-labelled phoneme transcription.

For training, a subset of 140 hours of the Dutch part of CGN is used (due to computational purposes, the full Dutch part is not used). CGN contains 15 different speaking styles, ranging from spontaneous conversations to reciting the written text, from news bulletins to solemn speeches. The full table of components is given in Table 7. The training set used to train all NNs contains all 15 components/speaking styles of CGN. For testing, 3 different test sets are used: a test set containing all 15 CGN speaking styles (from hereon called *15-speaking style test set*), a test set containing only clear speech (news bulletins in CGN, 'component k' in Table 7, from hereon called the *clear test set*[10]), and a test set containing only spontaneous speech (conversations in CGN, 'component a' in Table 7, from hereon called the *spontaneous test set*).

[10] Clear speech in this context means speech that is more clearly pronounced (other research might refer to clear speech as speech with little contextual noise)

Table 7: Overview of speaking styles in CGN, categorised in components a-o and with word-counts ('Totaal aantal woorden') differentiated between Flemish (VL) and Dutch (NL)

	Component	Totaal aantal woorden	VL	NL
a.	Spontane conversaties ('face-to-face')	2.626.172	878.383	1.747.789
b.	Interviews met leraren Nederlands	565.433	315.554	249.879
c.	Telefoondialogen opgenomen m.b.v. platform	1.208.633	465.096	743.537
d.	Telefoondialogen opgenomen m.b.v. minidisc recorder	853.371	343.167	510.204
e.	Zakelijke onderhandelingen	136.461	0	136.461
f.	Interviews en discussie uitgezonden op radio en televisie	790.269	250.708	539.561
g.	Discussie, debatten, vergaderingen (m.n. politieke)	360.328	138.819	221.509
h.	Lessen	405.409	105.436	299.973
i.	Spontane commentaren (o.a. sport) uitgezonden op radio en televisie	208.399	78.022	130.377
j.	Actualiteitenrubrieken en reportages uitgezonden op radio en televisie	186.072	95.206	90.866
k.	Nieuwsbulletins uitgezonden op radio en televisie	368.153	82.855	285.298
l.	Beschouwingen en commentaren uitgezonden op radio en televisie	145.553	65.386	80.167
m.	Missen, lezingen, plechtige toespraken	18.075	12.510	5.565
n.	Colleges, voordrachten, lezingen	140.901	79.067	61.834
o.	Voorgelezen teksten	903.043	351.419	551.624
Totaal		**8.916.272**	**3.261.628**	**5.654.644**

None of the training-, test- and evaluation sets contain any overlapping speakers or utterances. The full overview of training- and test sets is given in Table 8. All phonemes with less than 25 occurrences in the ground truth of the *test set* are removed from the result analysis of this report[11]. This will make the confusion matrices smaller and will remove large, sparse rows and columns from the confusion matrices. The final analyses in this report are based on 34 phonemes.

Table 8: Overview of training-, validation-, and test sets of this research

Dataset	Speaking Styles	Utterances
Training set	15 speaking styles of CGN	200.000
Test set #1	15 speaking styles of CGN	400
Test set #2: spontaneous speech	Component-a: spontaneous conversations	400
Test set #3: clear speech	Component-k: news bulletins	400

4.7.1 Forced alignment

Timed-aligned (i.e. when does every phoneme begin an end), phonemically annotated, and segmented speech corpora are needed to train the NNs. CGN only contains word-level transcriptions without time-labels. To gain the phone-level time-aligned transcriptions, forced alignment is conducted. For the full process explanation of forced alignment, please refer to [116]. The result of the forced alignment process is that every feature vectors now has an accompanying phonemic label, which can be used to train the NNs.

The process of forced alignment can be performed using predefined scripts in Kaldi.

[11] They are present in the training- and test set, however are removed from the result analysis in *this report*

5 Results

In this section, the results of the experiments in this research are given. First, the overall phoneme error rates (PER) of the 15-speaking style test set, clear test set, and spontaneous test set on the four architectures are given (see Section 5.1). Then, the errors of the architectures on the 15-speaking style test set are further analysed by inspecting where the errors originate (i.e. substitutions, insertions, and/or deletions) (see Section 5.2). These results can be used to answer the first sub-research question: '*What is the best architecture for general Dutch phoneme recognition?*'

The same analyses are used to error-analyse the performances of the architectures on the clear and spontaneous speech test sets (see Section 5.3). Those results can be used to answer the second sub-research question: *"How do different architectures perform on different speaking styles?"*

Finally, the errors of the phonemes from the 15-speaking style test set and the two speaking style test sets are further analysed using individual phoneme error-analysis and confusion matrices (see Section 5.4). Those results can be used to answer the third sub-research questions: '*What are error-prone phonemes in Dutch phoneme recognition?*'

5.1 Phoneme error rates of all experiments

Table 9 shows the PER results of all three speaking style experiments in the columns and every row depicts the architecture on which the experiment is conducted.

Table 9: Phoneme error rates of the BLSTM, OPGRU, TDNN-BLSTM, and TDNN-OPGRU on three different test sets

Architecture	PER All speaking styles	PER Spontaneous Speech	PER Clear Speech
BLSTM	35.82%	43.12%	11.75%
OPGRU	30.79%	38.51%	10.16%
TDNN-BLSTM	18.36%	23.42%	5.71%
TDNN-OPGRU	18.09%	23.18%	6.29%

As can be seen in Table 9, the BLSTM model performs worst on the 15-speaking style test set, gaining the highest PER. The OPGRU shows an improvement when compared to the baseline, gaining a relative improvement of 14.04% (5.03% absolute). When looking at the hybrid acoustic models, the PER improves a lot when compared to the baseline and OPGRU. The TDNN-BLSTM shows a relative improvement of 48.74% (17.46% absolute) when compared to the baseline model. The best performing architecture is the TDNN-OPGRU, resulting in a relative improvement of 49.50% (17.73% absolute) when compared to the baseline.

The PER results of the separate speaking style experiments show similar patterns to the PER results of the 15-speaking style. The baseline BLSTM, on both spontaneous and clear speech, obtained the highest PERs for both spontaneous speech and clear speech. Again, the OPGRU outperforms the baseline, lowering the PER by 10.69% relatively (4.61% absolute) for spontaneous speech and 13.53% relatively (1.59% absolute) for clear speech. Again, both hybrid models show a relatively large decrease in PER when compared to the baseline and OPGRU. The TDNN-BLSTM improves the PER with 45.69% relatively (19.70% absolute) for spontaneous speech and 51.40% relatively (6.04% absolute) for clear speech. The TDNN-OPGRU improves the PER with 46.24% relatively (19.94% absolute) for spontaneous speech and 46.47% relatively (5.46% absolute) for clear speech.

When comparing the PER performances of the architectures on clear and spontaneous speech, it shows that, for all architectures, spontaneous speech is harder to recognise than clear speech, with a

74% lower PER averaged over all architectures. Moreover, while the TDNN-BLSTM outperforms the TDNN-OPGRU on the clear speech test set, the pattern is reversed for the spontaneous speech test set.

5.2 15-speaking style error analysis

Before analysing and comparing the origins of the errors of the four architectures, one thing must be noted that emerged from the results and occur in all test sets. A factor that contributes to the PER of all models is the '[UNK]' symbol in the ground truth word-level transcript. This symbol represents words or other speech-related sounds in the original word-level transcription that were transcribed as 'unknown'. This results in a single '/UNK/' phoneme (i.e. the symbol for an unknown phoneme) in the true phonetic transcription. However, in the audio, the (multiple) phoneme(s) of the full-unknown words are still pronounced. This results in the phoneme recogniser still trying to recognise all phonemes of this unknown word (if phonemes are present). This results in one of those predicted phonemes to count as a substitution (as it should be '/UNK/'), and the additionally recognised phonemes to count as insertions. Fortunately, there are not a lot of unknown words present in the transcriptions, so the contribution to the PER is not very large. This problem can be solved in future work by either identifying unknown words and ignoring the errors or make more accurate phonemic transcriptions on the positions of the unknown words.

The percentage insertions, deletions, and substitution for each of the four architectures are found in Table 10. The confusion matrices of the BLSTM, OPGRU, TDNN-BLSTM, and TDNN-OPGRU on the 15-speaking style are found in Table 11. When looking at Table 10 and the confusion matrix in Table 11, the following observations are made.

Table 10: Distribution of PER over the number of substitutions, insertions, and deletions over four architectures on the 15-speaking style[12]

	BLSTM	OPGRU	TDNN-BLSTM	TDNN-OPGRU
%insertions	5.35%	9.36%	9.44%	8.58%
%deletions	19.33%	9.16%	3.23%	3.99%
%substitutions	11.13%	12.27%	5.70%	5.52%

Most errors of the baseline BLSTM model are due to deletions, while the fewest are due to insertions. The confusion matrices show that all phonemes are incorrectly classified at least 20% of the occurrences.

The errors in the OPGRU model are primarily due to substitutions. Besides that, the OPGRU had a similar number of insertions and deletions. When compared to the BLSTM, the OPGRU shows a relatively lower percentage of deletions, but a relatively higher number of insertions and substitutions. The lower number of deletions indicates the number of phonemes in the output transcriptions is closer to the ground-truth; however, the high number of substitutions and insertions means the predicted phonemes are inaccurate. The confusion matrices show that similar to the BLSTM, almost all phonemes are incorrectly classified at least 20% of the occurrences (only /v/ has a 19.2% error).

[12] The overall PER of the architectures are not equal to the sum of the percentage of substitutions, deletions, and insertions in the table, as the overall PER calculation includes all phonemes and the individual percentages in this table do not do not.

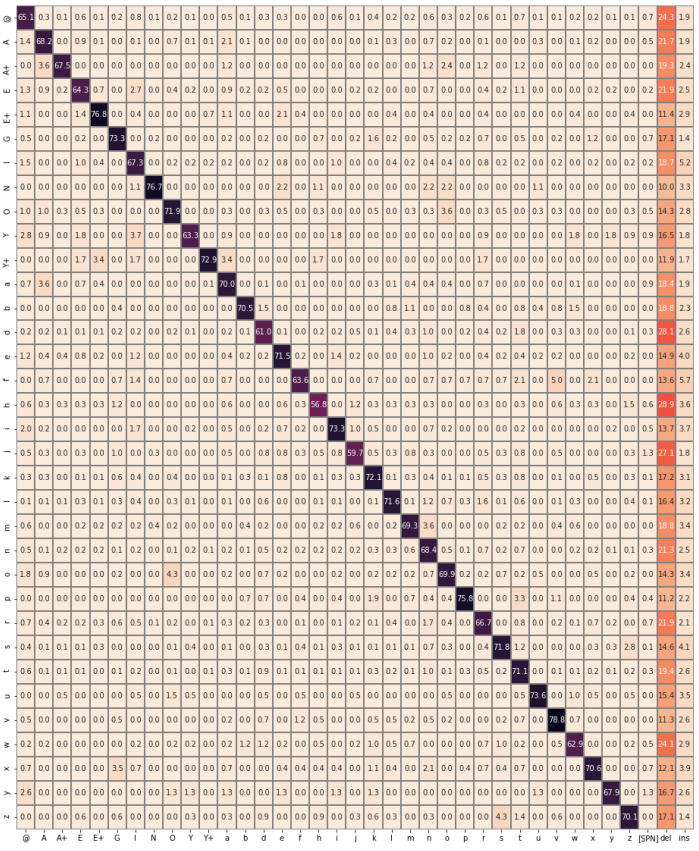
The errors of both the TDNN-BLSTM and TDNN-OPGRU are similar. Most errors of the TDNN-BLSTM and TDNN-OPGRU are primarily due to insertions, while the fewest are due to deletions. When compared to the BLSTM, the TDNN-BLSTM and TDNN-OPGRU show to have a relatively higher number of insertions, while having relatively lower deletions and substitutions when compared to the BLSTM. The confusion matrices show that the phonemes are incorrectly classified between 10% and 20% of the times (two phonemes had a PER of <10%: /p/ and /v/). This means that the number of phonemes in the output transcriptions is closer to the ground-truth and the predicted phonemes are predicted more accurate when compared to the BLSTM.

When comparing the TDNN-BLSTM with the TDNN-OPGRU, the results are similar. Small differences can be found; the TDNN-BLSTM needs more substitutions and insertions, while the TDNN-OPGRU needs more deletions.

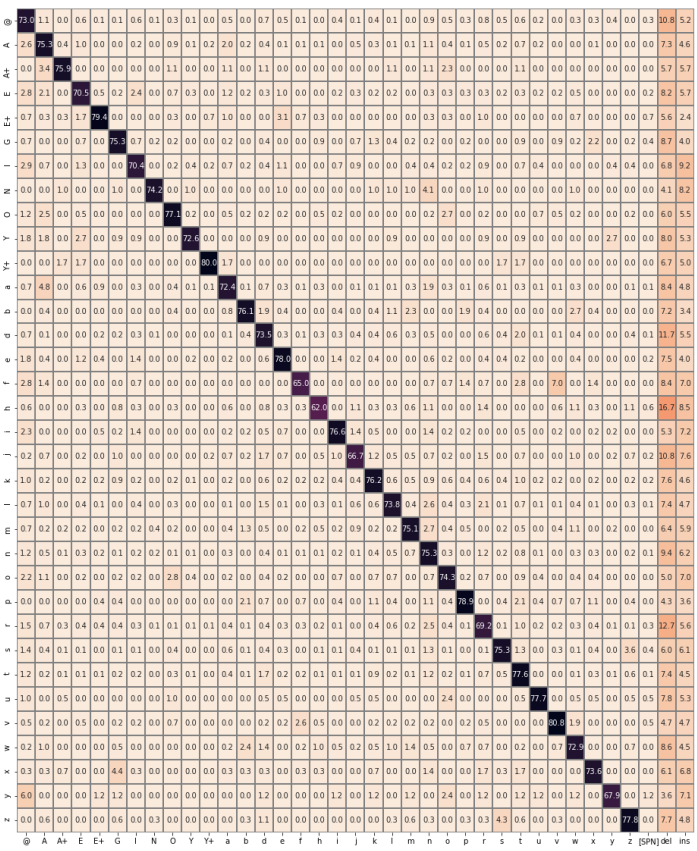
The BLSTM and OPGRU have shown relatively high errors when compared to the TDNN-BLSTM and TDNN-OPGRU. As the goal of this research is to improve Dutch PR, the speaking style analysis in Section 5.3 and the error-analysis in Section 5.4 will only be conducted on the TDNN-BLSTM and TDNN-OPGRU. This will assess the performances of the highest performing models, which will show where future work can focus on whilst improving Dutch PR.

Table 11: 15-speaking style confusion matrices for BLSTM, OPGRU, TDNN-BLSTM, and TDNN-OPGRU

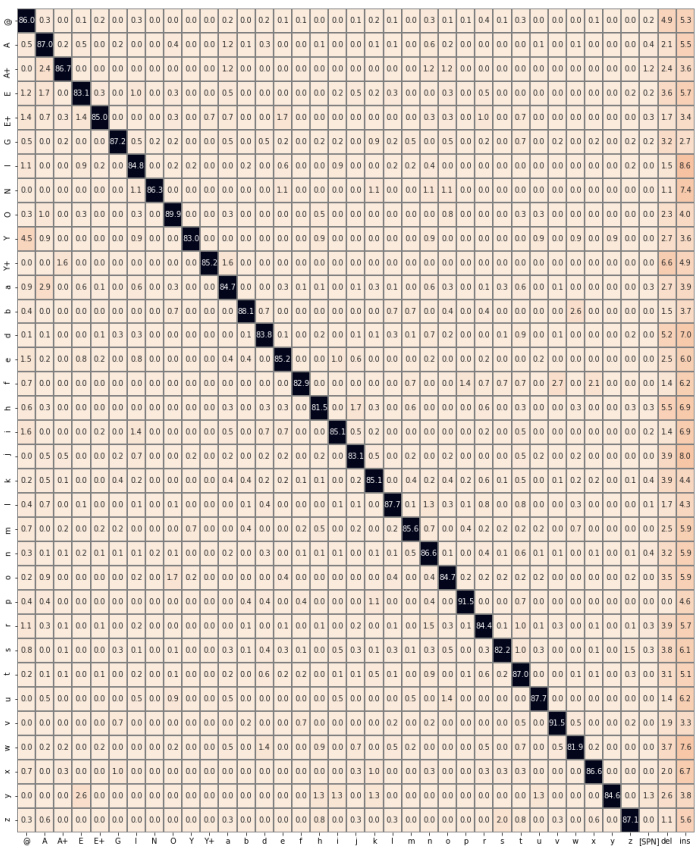
BLSTM



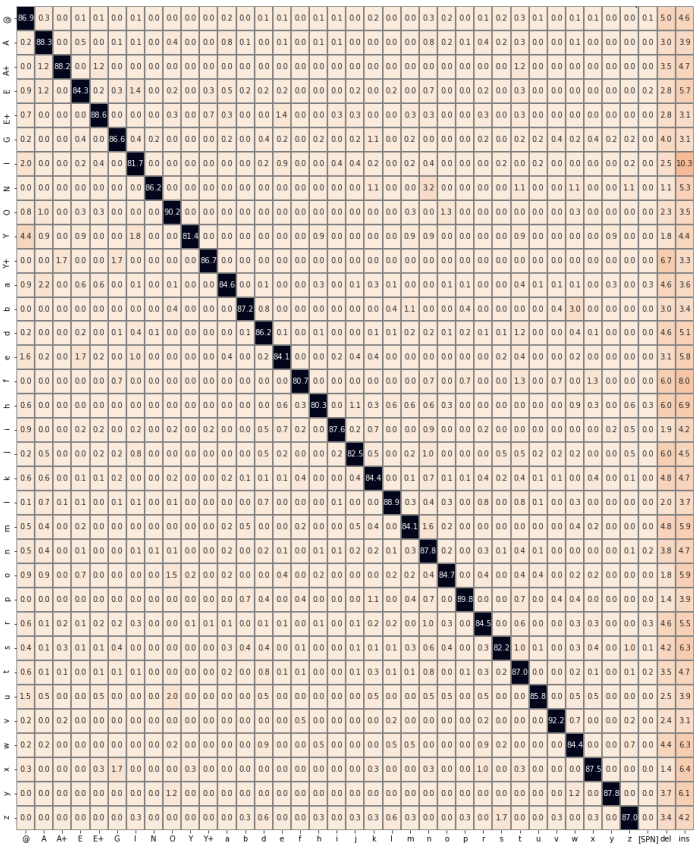
OPGRU



TDNN-BLSTM



TDNN-OPGRU



5.3 Clear and spontaneous speaking style error analysis

In this part, the results of the error analysis of the TDNN-BLSTM and TDNN-OPGRU on the separated test sets containing spontaneous speech and clear speech are presented.

The percentage insertions, deletions, and substitution and confusion matrices for the TDNN-BLSTM and TDNN-OPGRU on the clear and spontaneous test sets are found in respectively Table 12 and Table 13.

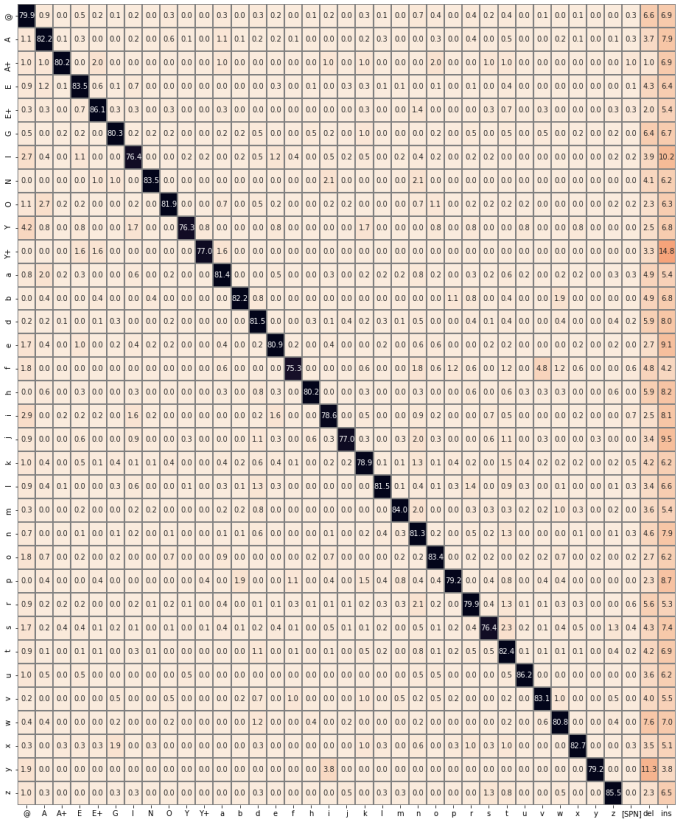
Table 12: Distribution of PER over the number of substitutions, insertions, and deletions for the TDNN-BLSTM and TDNN-OPGRU on the clear a spontaneous speech test sets

	Spontaneous speech test set		Clear speech test set	
	TDNN-BLSTM	TDNN-OPGRU	TDNN-BLSTM	TDNN-OPGRU
%substitutions	11.21%	10.58%	2.75%	2.86%
%insertions	4.84%	5.52%	1.06%	1.28%
%deletions	7.37%	7.19%	1.91%	2.15%

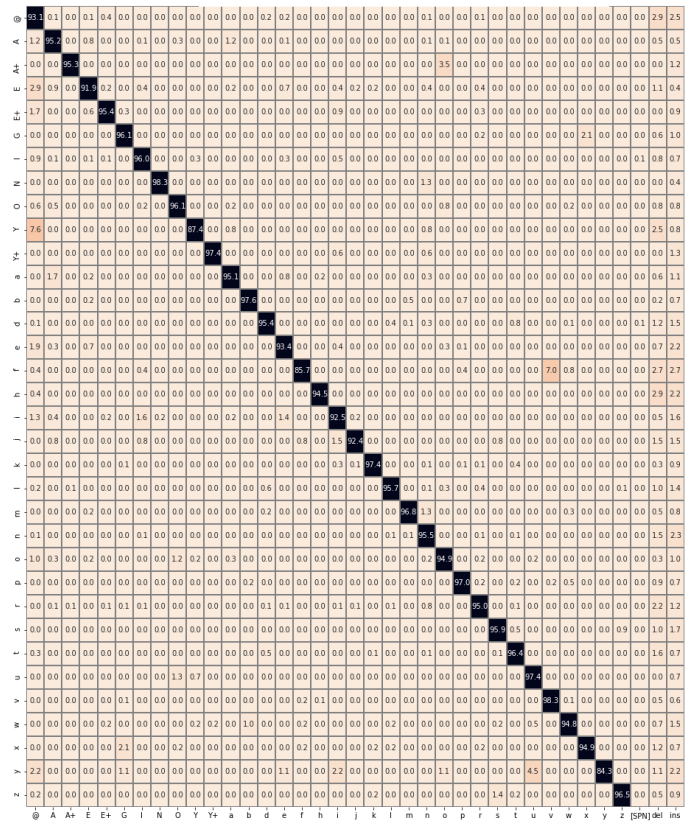
When looking at Table 12 and the confusion matrices in Table 13, the errors of both the TDNN-BLSTM and TDNN-OPGRU are similar for both speaking styles. For spontaneous speech, both the TDNN-BLSTM and the TDNN-OPGRU have a relatively high number of substitutions, a medium number of deletions, and a relatively low number of insertions. For clear speech, the number of substitutions is the highest, followed by the number of deletions, and the lowest percentage of the PERs come from insertions. The confusion matrices on clear speech of both matrices show that all phonemes are incorrectly classified between 13% and 25% of the occurrences. The confusion matrices on spontaneous speech of both matrices show that all phonemes are incorrectly classified between 2% and 13% of the occurrences.

Table 13: Spontaneous and clear test set confusion matrices for the TDNN-BLSTM and TDNN-OPGRU

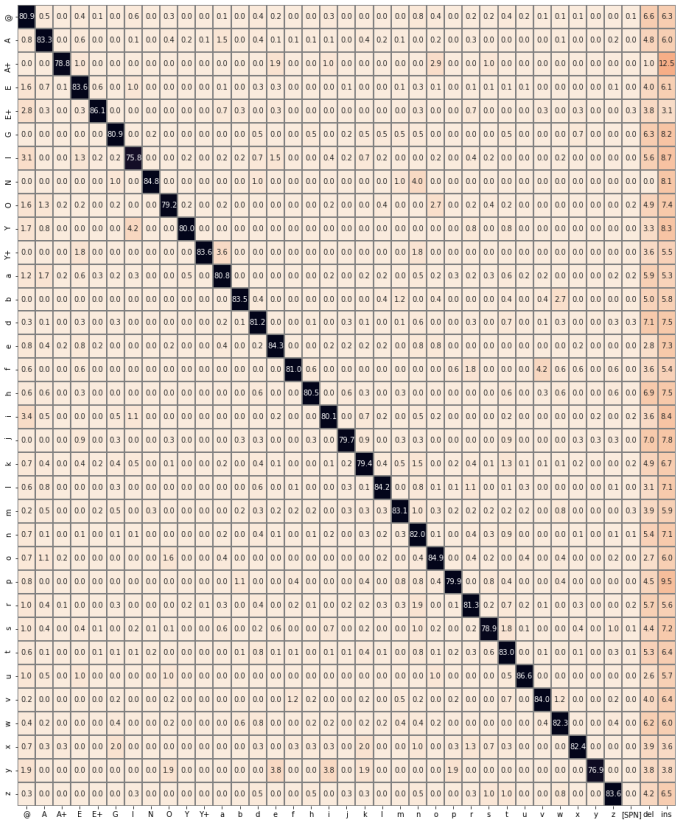
TDNN-BLSTM Spontaneous speech



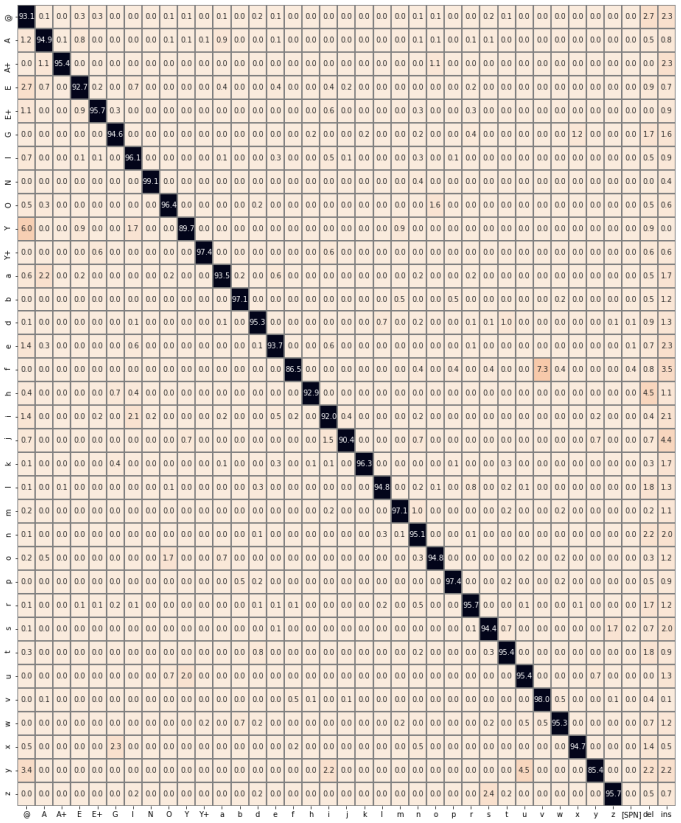
TDNN-BLSTM Clear speech



TDNN-OPGRU Spontaneous speech



TDNN-OPGRU Clear speech



5.4 Error-prone phonemes

The errors of the two best-performing architectures (TDNN-BLSTM and TDNN-OPGRU) are further analysed by analysing the errors of the individual phonemes and their contributions to the overall PER. This is done separately for the 15-speaking style test set (see Section 5.4.1) and the clear and spontaneous test sets (see Section 5.4.2).

5.4.1 Phonemes error analysis 15-speaking style test set

Figure 24 shows the $PER_{phonemex}$ (dashed lines) and the $\%ContributionToPER_{phonemex}$ (solid lines) [13]. The TDNN-BLSTM results are visualised in blue and the TDNN-OPGRU results are visualised in red. The average $PER_{phonemex}$ of both architectures (14.22%) is shown as a linear, green dashed line. The average $\%ContributionToPER_{phonemex}$ of both architectures (2.88%) is plotted as a linear, yellow line. Phonemes of the same type are ordered in the phoneme (sub)categories (see Section 2.1) on the x-axis to analyse if there are certain groups of phonemes that are more or less difficult to recognise.

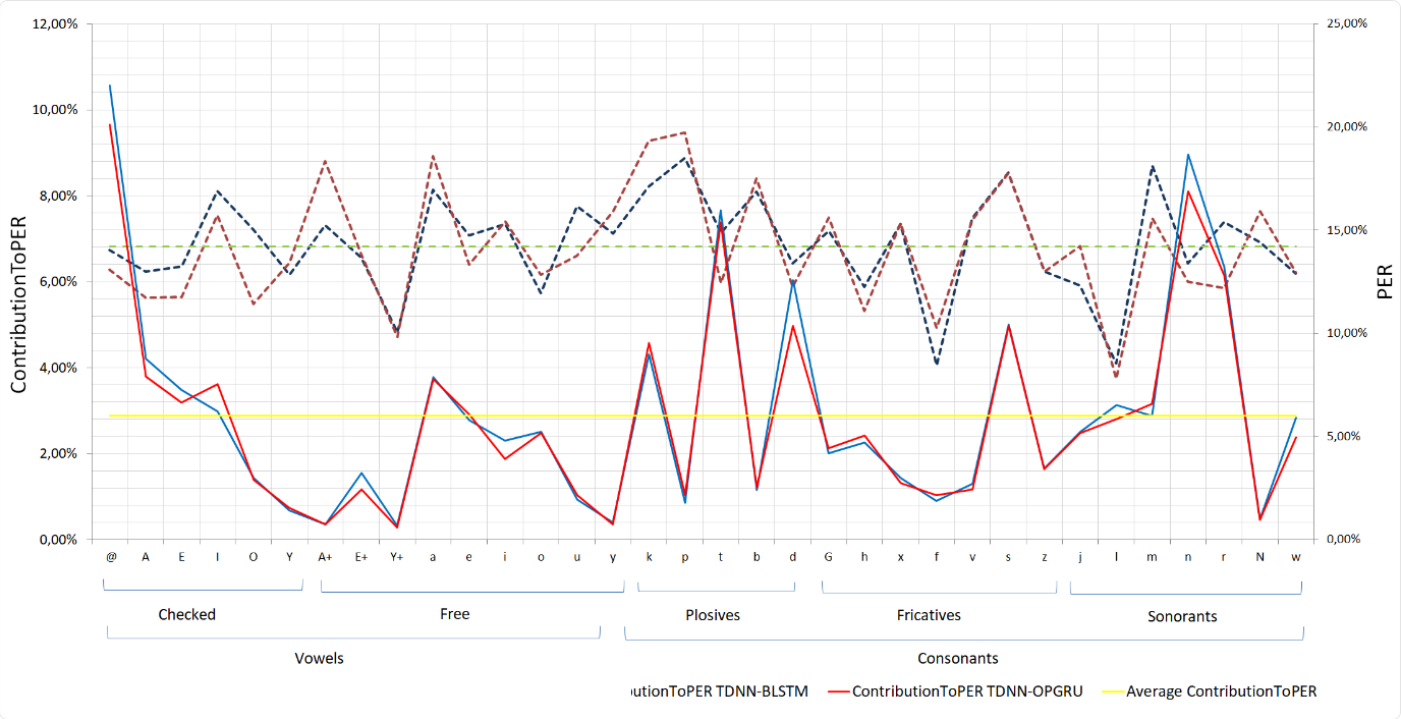


Figure 24: $PER_{phonemex}$ results per phoneme on the 15-speaking style test set on the TDNN-BLSTM and TDNN-OPGRU

Before the error-prone phonemes are identified, several observations can be made from Figure 24. The first observation is made by focussing on the phoneme groups. When analysing the phoneme groups, no clear pattern can be found as to what architecture is better or worse at recognising vowels or consonants, as for both groups at least one phoneme has an above-average error on both groups. The same holds for the sub-categories of the vowels and consonants.

The second observation is made by focussing solely on the $ContributionToPER_{phonemex}$ metric. The phonemes of both architectures have very similar $ContributionToPER_{phonemex}$ results, as the delta between the solid lines stays small (average delta = 0.12%). This means the same phonemes contribute similarly to the overall PER of both architectures. Besides that, three phonemes stand out with relatively high peaks in the graph on both architectures: the /@/ (i.e. the schwa), /t/, and /n/. This

13

means the contributions of these phonemes to the overall PER are relatively high compared to the other phonemes.

The third observation is made by focussing solely on the $PER_{phonemex}$ metric. The $PER_{phonemex}$ metric has less similar results when comparing the results of the phonemes of both architectures, as the variance of the deltas between the dashed lines is relatively high. The TDNN-BLSTM shows to have a relatively larger PER for the phonemes /A/, /E/, /u/, /t/, /m/, and /r/ when compared to the TDNN-OPGRU. The TDNN-OPGRU shows to have a larger PER for the phonemes /A+/, /a/, /k/, /p/, /j/, and /N/ when compared to the TDNN-BLSTM.

Error-prone phonemes are identified as the phonemes that have an above average $PER_{phonemex}$ and an above average $ContributionToPER_{phonemex}$ on both architectures. The error-prone phonemes found are /I/, /a/, /k/, and /s/.

Besides the identified error-prone phonemes, there are also phonemes with an above-average result on both metrics on either the TDNN-BLSTM or TDNN-OPGRU. For instance, the TDNN-BLSTM shows an above-average error on both metrics for the /t/ and the /r/, while the TDNN-OPGRU shows above average on both metrics for the /m/.

5.4.2 Phonemes error analysis clear and spontaneous speech

For the speaking style analysis, the error-prone phonemes are identified for the phonemes with an above-average $PER_{phonemex}$ and $ContributionToPER_{phonemex}$ on both architectures for both speaking styles. Both metrics will be discussed separately, as they will otherwise clutter the graphs.

Figure 25 shows the results of the $PER_{phonemex}$ metric for clear and spontaneous speech experiments on the TDNN-BLSTM and TDNN-OPGRU[14]. The solid lines represent clear speech results, while the dashed lines represent spontaneous speech results. The colour red represents the TDNN-OPGRU and the colour blue represents the TDNN-BLSTM (both spontaneous and clear speech). The horizontal lines represent the $PER_{phonemex}$ averages of both architectures (yellow represents the clear speech average, green represents the spontaneous speech average). Besides that, similar types of phonemes are shown next to each other to find if certain groups are more or less difficult to recognise.

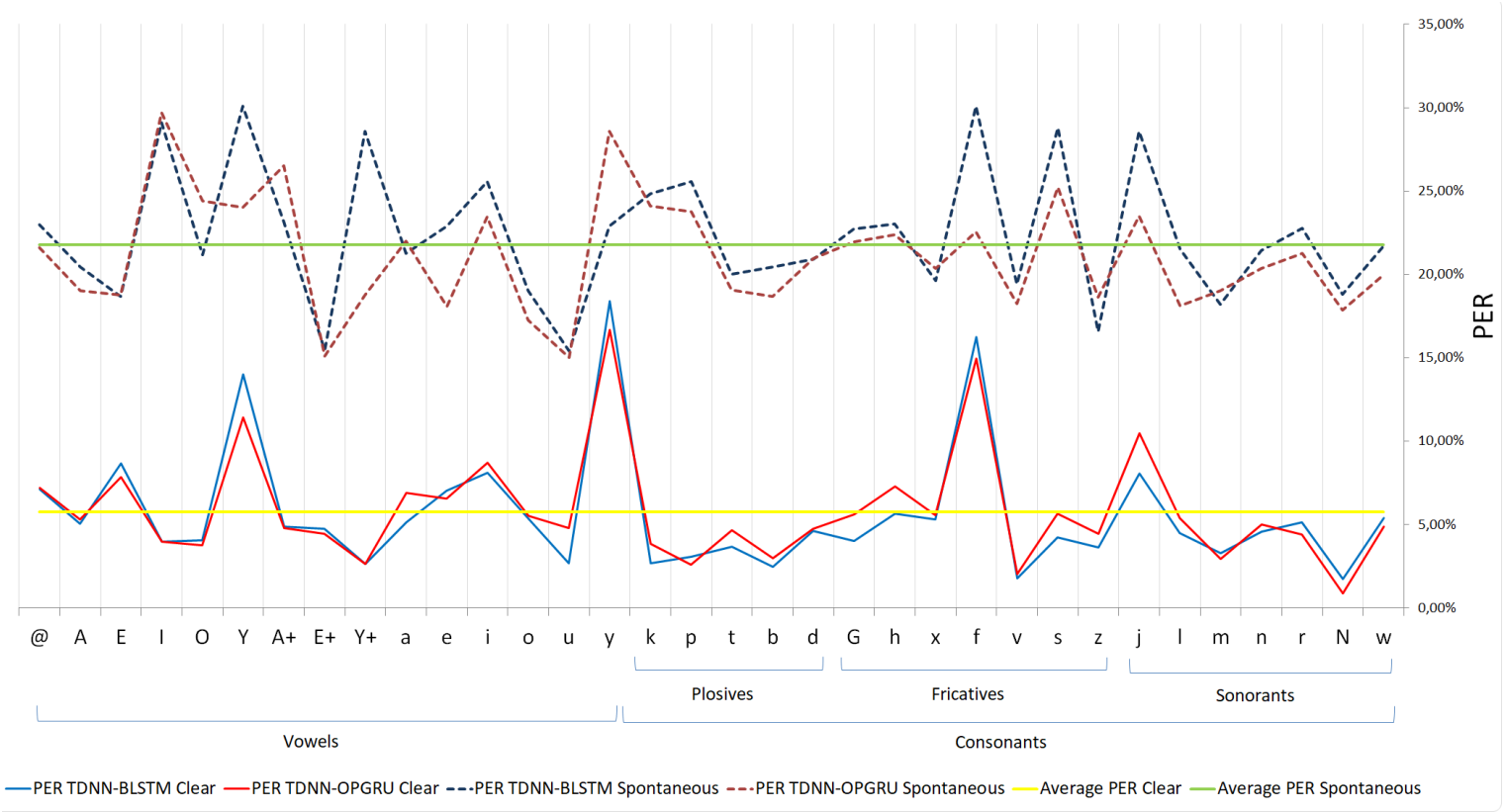


Figure 25: $PER_{phonemex}$ results of the clear and spontaneous speech test set on the TDNN-BLSTM and TDNN-OPGRU

Before the error-prone phonemes are identified, several observations can be made from Figure 25. First, the dashed, spontaneous speech line is above the solid, clear speech line. This means the architectures have more difficulty recognising the phonemes in spontaneous speech when compared to clear speech.

When looking at the phonemic categories in Figure 25, the second observation that can be made is that all plosive consonants are easily recognised in clear speech, as they all have PERs lower than the

[14] No trend should be extracted from the graph. The lines are used to compare the architectures and find patterns between phonemic groups

average. Besides that, no clear patterns can be found in the phonemic categories. Every category has at least one highly underperforming phoneme for both speaking styles.

When analysing the phonemes with an above-average $PER_{phonemex}$ on clear speech, it can be observed that both architectures share the same types of phonemic errors, as both dashed lines in the graph have a similar pattern and the delta between the lines stays fairly equal (maximum delta = 1.42% for the /r/). Besides that, three phonemes stand out in clear speech: the /Y/, /y/, and /f/ with high peaks in the graph, meaning the individual PERs of these phonemes are relatively high.

For spontaneous speech, the results for the two architectures are less similar. For instance, the TDNN-BLSTM model seems to have more difficulty recognising the /Y/, /Y+/, /e/, /f/, and /j/ compared to the TDNN-OPGRU. On the other hand, the TDNN-OPGRU has more difficulty recognising the /y/ compared to the TDNN-BLSTM model.

Figure 26 shows the results of the $ContributionToPER_{phonemex}$ metric for the clear and spontaneous speech test sets on the TDNN-BLSTM and TDNN-OPGRU[15]. When looking at Figure 26, several observations can be made.

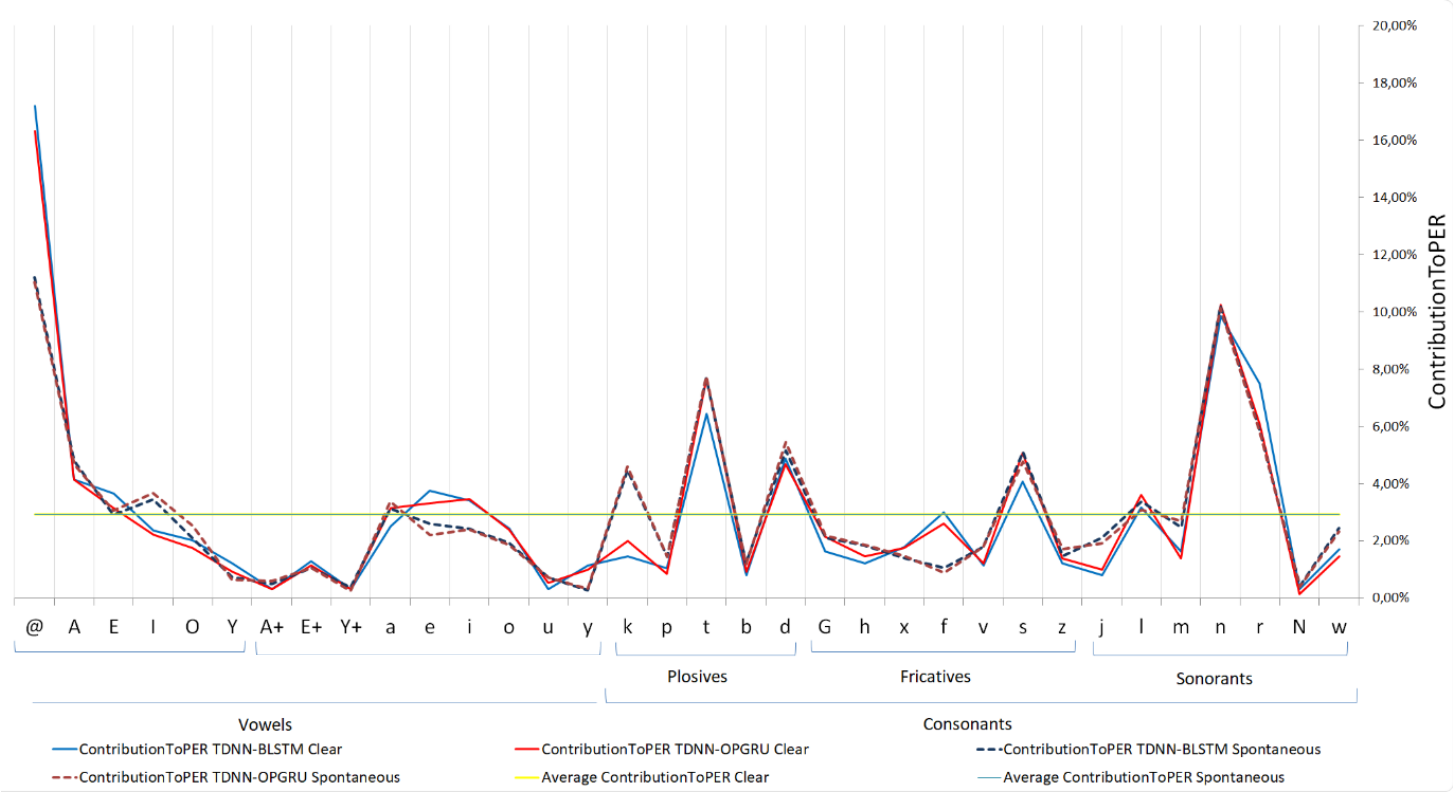


Figure 26: $ContributionToPER_{phonemex}$ results of the clear and spontaneous speech test set on the TDNN-BLSTM and TDNN-OPGRU

The first observation that is made is that the size of the contribution to the overall PER is fairly similar for almost all phonemes when comparing speaking styles, as both the dashed lines and the solid lines show a highly similar pattern and the deltas between the dashed lines and solid lines are relatively small. This means that, between architectures, there is little difference in what phonemes contribute to the PER most. The large difference in overall PER between the speaking styles is therefore mainly due to differences in the individual errors of the phonemes. This is further discussed in Section 6.2. The only larger deltas between the solid and dashed lines are the /@/ and /f/ which contribute

[15] No trend should be extracted from the graph. The lines are used to compare the architectures and find patterns between phonemic groups

relatively more to the PER on clear speech than on spontaneous speech. The /k/ contributes relatively more to the PER on spontaneous speech than on clear speech.

Another observation that can be made are the differences in phonemes with above-average values on both metrics between the speaking styles. The phonemes that have an above-average $PER_{phonemex}$ and an above-average $ContributionToPER_{phonemex}$ are the /@/, /E/, /e/, and /i/ for clear speech and the /@/, /I/, and /k/ for spontaneous speech. As can be seen, only the /@/ is shared between the speaking styles. This makes the /@/ interesting and will therefore be added to the list of error-prone phonemes of Dutch PR.

6 Discussion

This research was conducted to improve Dutch phoneme recognition. The goal of this research is stated as follows:

Main Research Goal: "Improving Dutch phoneme recognition"

To achieve this goal, this research aims to answer the following sub-research questions:

Sub-Research Question 1: "What is the best architecture for general Dutch phoneme recognition?"

Sub-Research Question 2: "How do different architectures perform on different speaking styles?"

Sub-Research Question 3: "What are error-prone phonemes in Dutch phoneme recognition?"

To answer these questions, four neural networks architectures (BLSTM, OPGRU, TDNN-BLSTM, and TDNN-OPGRU) were built and trained. These architectures were been trained on a 140-hour training CGN dataset containing 15 different speaking styles using the Kaldi ASR toolkit. Using the trained models, three experiments were conducted. The models were first tested on a 15-speaking style test set (containing the same 15 speaking styles as the training set), secondly on a test set containing spontaneous speech and lastly on a test set containing clear speech. The results of the models were analysed in multiple steps. First of all, the overall performances of the models were measured using the phone error rate (PER). Then, the PER was decomposed in substitutions, insertions, and deletions to see where the errors originate and compare architectures. These results can be used to answer the first sub-research question (see Section 6.1). Next, the same analyses were used for the clear and spontaneous speech experiments. Those results can be used to answer the second sub-research question (see Section 6.2). Finally, individual phoneme error metrics and confusion matrices were used to analyse the errors of the phonemes of both the 15-speaking style test set and specific speaking style test sets. The results can be used to answer the third research question (see Section 6.3).

6.1 A split-decision: the best architecture for Dutch phoneme recognition

Using the decoding result of the four neural networks, the first sub-research question can be answered. With a PER of 35.82%, the BLSTM showed to be ineffective for Dutch PR. The same holds for the OPGRU with only a 5.03% absolute PER improvement. Both TDNN-BLSTM and TDNN-OPGRU show a relatively large improvement of the PER by ~45% relatively when compared to the baseline BLSTM. This shows that the addition of TDNN layers greatly benefits Dutch PR. The strength of the hybrid models is the combination of capturing short-term context (TDNN layers) and long-term context (RNN layers).

The TDNN-BLSTM and TDNN-OPGRU are not able to recognise errorless, as the overall PERs on the 15-speaking styles are still higher than 18%. This is due to multiple reasons. The first reason is that the ground-truth transcriptions are not fully accurate. The second reason is that the 15-speaking style contains different speaking styles, some of which are more difficult to recognise than others (as further discussed in Section 6.2). The third reason is that the models have difficulties identifying and differentiating certain phonemes (as further discussed in Section 6.36.2).

To summarise, both the TDNN-BLSTM and TDNN-OPGRU have shown to be the best architectures for Dutch PR at this moment in time. For the 15-speaking style, the TDNN-OPGRU outperformed the TDNN-BLSTM.

The architecture choice of this research was based on existing English WR research which also implemented a BLSTM, OPGRU, and TDNN-OPGRU [39]. Even though the error rates cannot be directly

compared, the architecture performances and the role of the TDNN are assessed. In [39], the researchers showed that the addition of TDNN-layers to a gated-RNN can reduce the error-rate 3-7% relatively. This research has shown that the TDNN layers can reduce the error-rate by 40-50% relatively. Even though the error-rates cannot be directly compared, the addition of TDNN layers seems to benefit PR relatively more than WR. This could mean two things. First of all, this could indicate that PR depends more on relatively shorter-term dependencies when compared to WR, as the strength of TDNNs is to model relatively shorter-term dependencies more effectively when compared to the BLSTM and OPGRU. Secondly, it could mean that the Dutch language depends more on shorter-term dependencies, as the other research is performed on English datasets. This should both be researched in future work to make any concise conclusions. This could be done by training and testing the models of this research on different languages.

Besides that, when comparing the lowest PER of this research (5.71%) with the lowest PER results of existing research on other languages (14.9% (Light-GRU: paper [117], PR survey: [27]), the results of this research are relatively low. Even though not directly comparable (different dataset, architecture, language), the results of this research do indicate that the hybrid TDNNs are effective for PR. This should be researched on other languages with similar dataset sizes to make any concise conclusions.

6.2 The big showdown: spontaneous versus clear speech

When looking at the PER results of the spontaneous and clear speech test sets, the hybrid models outperformed the BLSTM and OPGRU models. When comparing the hybrid models on spontaneous speech, the TDNN-OPGRU again outperformed the TDNN-BLSTM. For clear speech, the TDNN-BLSTM outperformed the TDNN-OPGRU. The differences for both speaking styles are however small, and therefore, no clear recommendation on which architecture is better for either speaking style is made. The results of all architectures clearly show that recognising spontaneous speech is more difficult than recognising clear speech. This is in line with existing speaking style research on English [35], German [34], and Chinese [118], which all showed lower recognition errors for clear speech when compared to spontaneous speech.

The differences between speaking styles come from the outset of the people speaking. When speaking in a more clear speaking style, the speaker tries to be more articulate and canonical. When people speak more spontaneous, they tend to take ‘the path of the least resistance’, while keeping ‘the law of intelligibility’ in mind [36]. This means people tend to speak so the listening party understands what they are saying, while preferably only pronouncing those phonemes that cause what they are saying to stay intelligible. This can cause a change in the acoustic characteristics of the phonemes.

In [33], researchers investigated what effects the different speaking styles have on the acoustic characteristics of Japanese phonemes. They showed three effects that can change the acoustic characteristics. First, they showed the variance of the pronunciations of phonemes becomes higher when speech becomes more spontaneous. Secondly, they showed that the distance between phonemes becomes smaller when speech becomes more spontaneous and the sound of consecutive phonemes might be woven into each other. Thirdly, they showed that phoneme reduction occurs more frequently in spontaneous speech when compared to clear speech. In [33], the researchers concluded that three effects make spontaneous speech more complex to recognise than clear speech, which was also shown on other languages [119, 41, 57]. Several phenomena that cause the three above changes in the acoustic characteristics of the phonemes are further discussed in Section 6.3 on a phoneme-level.

The above reasons could explain the higher error found in this research for spontaneous speech when compared to clear speech. This should be validated in future research by further analysing the effects Dutch speaking styles have on the acoustic characteristics of the phonemes.

6.3 "Error, Error;" discussing the phoneme errors

The phoneme error-analysis was conducted on the two highest-performing architectures (TDNN-BLSTM and TDNN-OPGRU), to show where the errors lie and to give room for future work to improve the recognition. The error-analysis results showed the errors of the individual phonemes on both architectures. Then, error-prone phonemes are identified as the phonemes that have an above-average individual PER and an above-average individual contribution to the overall PER on both the TDNN-BLSTM and TDNN-OPGRU. The following phonemes are identified as error-prone phonemes for Dutch PR: /I/, /a/, /k/, and /s/ for the 15-speaking style test set and the/@/ (i.e. schwa) for the clear and spontaneous test sets. Besides the error-prone phonemes, the other phonemes in the results also showed errors. To gain more insights into reasons why these phoneme errors have occurred, the errors are now further analysed.

Section 6.2 shows that speaking styles can affect the acoustic characteristics of phonemes when comparing different speaking styles. In the following parts, several phenomena are explored that change the acoustic characteristics of phonemes. These phenomena can be used to hypothesise why the errors in the models of this research occur. Examples of these phenomena are given below with examples from the predicted transcriptions of the models. Having said that, these examples do not prove anything, they are merely examples of the errors in this research and potential reasons behind the errors. Future research should focus on quantifying the phenomena and acoustically analysing Dutch PR to prove the hypotheses.

An example of a phenomenon is assimilation. Assimilation in speech causes the sound (e.g. vowels or consonants) to change due to certain neighbouring phonemes [120]. Assimilation usually occurs because two phonemes share common phonetic features (e.g. similar-sounding phonemes, similar way of production by the human, etc.) [120]. An example taken from the predicted transcriptions that could be the cause of assimilation can be found in Table 14. In Table 14, the sound of the /s/ is recognised as the sound of a /z/. The /s/ and /z/ are often studied as pair in Dutch phonology, as the pronunciation of both is similar, the only difference being that the vocal cords vibrate when the /z/ is pronounced[16]. When phonemes share phonetic-features and assimilation occurs, an effect is that the sounds of one phoneme might change into the sound of the other phoneme (an example is shown in Table 14, where the /s/ takes on the sound of the /z/) [120]. As the models recognise acoustics, assimilation might cause misrecognitions. This will ultimately lead to more substitutions.

Table 14: Example of /s/-substitution

Word-level transcription	"Als ze dat"							
True phoneme output	A	l	s	z	@	d	A	t
Predicted phoneme output	A	l	z	z	e	d	A	t

The second example of a phenomenon that changes the acoustic characteristics of phonemes is elision, which is another word for the omission of sounds (e.g. vowels or consonants) in speech [121]. An example from the output of this research that could be caused by elision can be found in Table 15. In Table 15, the last vowel /@/ of the word "redelijk" is not recognised. Elision occurs more often in unstressed syllables, as the aforementioned 'path of the least resistance' causes weak phonemes to

[16] You can feel your throat vibrate when saying /z/, those are the vocal cords

be skipped (or 'glided over'), causing a weaker sound of pronunciation [122]. When the pronunciation becomes weak, the models will have more trouble recognising them. This will ultimately lead to more deletions.

Table 15: Example of /@/-deletion

Word-level transcription	"redelijk"							
True phoneme output	r	e	d	@	l	@	k	
Predicted phoneme output	r	e	t	@	l		k	

The last example of a phenomenon that changes the acoustic characteristics of phonemes is epenthesis. Epenthesis causes one or more sounds (e.g. vowels or consonants) to be added in a word. Epenthesis occurs mainly between consonant clusters to facilitate pronunciation, occurs more frequently in certain accents and dialects, and appears with similar frequency in different speaking styles [124]. An example of an epenthesis of a schwa can be found in Table 16. In Table 16, the schwa is added in the consonant cluster /l p/. As the models might recognise the acoustics of the inserted sounds, epenthesis might cause misrecognitions. This will ultimately lead to more insertions.

Table 16: Example of /@/-insertion

Word-level transcription	"help"				
True phoneme output	h	e	l		p
Predicted phoneme output	h	e	l	@	p

The predicted transcriptions of the models in this research are compared to the canonical ground-truth transcriptions. As people in all speaking styles do not necessarily speak canonically, the difference in the pronounced phonemes in the speech waves and the ground-truth transcriptions could be a partial explanation for the errors in this research[17]. This is strengthened as more similar examples like in Table 14, Table 15, and Table 16 were found in the results. Again, this should be proven in future work by quantifying the phenomena and analyse the acoustic characteristics of the phonemes.

The above phenomena occur more frequently in (but are not exclusive to) spontaneous speech when compared to clear speech [119]. Therefore, it is also hypothesised that the above phenomena are the reason the errors of the spontaneous test sets are higher than the errors of the clear test set. This hypothesis is strengthened, as the speaking style error analysis showed similar results for each phoneme on the percentages that they individually contributed to the overall PER. This could indicate that similar errors occur between speaking styles (e.g. assimilation, elision, and epenthesis of the same phonemes), however, more frequently in spontaneous speech when compared to clear speech.

[17] Partial because the models are also not perfect

7 Conclusion and future work

In this research, Dutch phoneme recognition (PR) is researched and improved. The following research goal was formulated*: "Improving Dutch phoneme recognition."* To improve Dutch PR, three sub-goals were formulated: finding the most optimal acoustic model architecture for general Dutch PR, analysing the performance of the most optimal architectures on different speaking styles, and identifying error-prone phonemes. Existing research on improving Dutch PR is limited, the latest research dating back to 1995. This research presents Dutch PR in modern daylight by implementing the latest neural network techniques, setting a benchmark, analysing different speaking styles, and indicating where improvements can be made. This will hopefully set up future research on Dutch PR. This is useful, as it will initiate the use of more effective applications and use-cases that use Dutch PR.

This is the first research for Dutch PR that implements Deep Neural Network-Hidden Markov Models in the acoustic models. This research has built a baseline Bi-directional Long Short Term Memory (BLSTM) and three additional advanced Neural Network architectures: an Open-Gate Protected Gated Recurrent Unit (OPGRU) and two hybrid-acoustic models, the Time-delayed BLSTM (TDNN-BLSTM), and a Time-Delayed OPGRU (TDNN-OPGRU). The architectures have all been trained on a 140-hour subset of the Dutch part of the CGN dataset. This dataset contains 15 different speaking styles. The trained models were tested on three different test sets: one containing the same 15 different speaking styles as in the training set, one containing only spontaneous speech, and one containing only clear speech. The features used were mel-filterbank features and the language model was a bigram language model.

The results of finding the optimal architecture have shown that the BLSTM and OPGRU are not good for Dutch PR, both showing high PER scores. The hybrid acoustic models containing TDNN layers reduce the PER results and ultimately have shown to perform best on Dutch PR. The individual phoneme analysis shows that the TDNNs can distinguish and recognise phonemes more effectively, resulting in fewer deletions, substitutions, and insertions. If a 15-speaking style test set is used as input, the TDNN-OPGRU outperforms the TDNN-BLSTM.

Future work should focus on optimising the parameter settings of the architectures using ablation studies. Additionally, future work on finding the optimal Dutch PR configurations should also focus on using different features (e.g. MFCC, PLP, LPC), other language model settings (e.g. higher n-grams or NN language models), different architecture models, and/or different combinations of NN models. Furthermore, future work could focus on comparing the architectures of this research with results of the same architectures on other languages, using similar size training sets.

When comparing the performance of the architectures on the different speaking styles, the results of all architectures clearly show that recognising spontaneous speech is much more difficult than recognising clear speech. As shown, this is in line with existing research on spontaneous versus clear speech on other languages. If a PR is used for a particular speaking style, the architecture choice should depend on the type of input speaking style. The TDNN-BLSTM performed better on clear speech, while the TDNN-OPGRU performed better on spontaneous speech.

Error-prone phonemes of the TDNN-BLSTM and TDNN-OPGRU were identified as phonemes that had an above-average individual PER and an above-average individual contribution to the overall PER. The phonemes /I/, /a/, /k/, /s/, and the schwa were found as error-prone phonemes. These error-prone phoneme errors are shared between architectures. If recognising particular phonemes is important, the choice of a model should depend on the individual phoneme error-analysis, as the two models outperform each other on different phonemes. Dutch PR can be improved by focussing on finding ways of improving the recognition rates of the error-prone phonemes, either by optimising the

training data or by building dedicated models that focus on one or more of the error-prone phonemes.

Several phenomena shortly elaborated in the discussion of this research hypothesised potential causes of the individual phoneme errors. Future work could focus on proving the hypothesis by quantifying the discussed phenomena and performing acoustic analysis on the (errors of the) phonemes.

This research was conducted in name of Cognizant. The models of this research can show their clients how Dutch PR has evolved and what it is capable of. This can be used to create ideas for future implementation and business use-cases. Future steps of Cognizant should focus on analysing the decoding time to ensure higher customer satisfaction and creating ideas to implement the PR in (ideas were discussed, however, due to confidentiality, cannot be further elaborated).

To conclude, the goal of this research was to improve Dutch PR. It has done so in three ways. First, this work has set the new benchmark for Dutch PR by setting the lowest recognition accuracies of Dutch PR systems found in literacy. Secondly, this work has analysed how the different speaking styles perform on the models. Thirdly, the individual phoneme errors of all of the experiments have been analysed. The results of this research will hopefully enable future researchers to analyse and improve Dutch PR even further. To all future Dutch PR researchers out there, you will have Phon Times doing so.

Bibliography

[1] Duffy, J. R. (2000). Motor speech disorders: clues to neurologic diagnosis. In Parkinson's Disease and Movement Disorders (pp. 35-53). Humana Press, Totowa, NJ..

[2] Cambrdige Dictionary, "dictionary.cambridge.org," Cambridge, 2020. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/phoneme. [Accessed 03 03 2020].

[3] Wahlster, W. (Ed.). (2013). Verbmobil: foundations of speech-to-speech translation. Springer Science & Business Media..

[4] Andrew, F. (2013). U.S. Patent No. 8,620,667. Washington, DC: U.S. Patent and Trademark Office..

[5] Basma, S., Lord, B., Jacks, L. M., Rizk, M., & Scaranelo, A. M. (2011). Error rates in breast imaging reports: comparison of automatic speech recognition and dictation transcription. American Journal of Roentgenology, 197(4), 923-927..

[6] Di Fabbrizio, G., Dutton, D. L., Gupta, N. K., Hollister, B. B., Rahim, M. G., Riccardi, G., ... & Schroeter, J. (2011). U.S. Patent No. 7,869,998. Washington, DC: U.S. Patent and Trademark Office..

[7] Abdul-Kader, S. A., & Woods, J. C. (2015). Survey on chatbot design techniques in speech conversation systems. International Journal of Advanced Computer Science and Applications, 6(7)..

[8] Williams, K. J., Peters, J. C., & Breazeal, C. L. (2013, June). Towards leveraging the driver's mobile device for an intelligent, sociable in-car robotic assistant. In 2013 IEEE intelligent vehicles symposium (IV) (pp. 369-376). IEEE..

[9] J. B. Hampshire and A. H. Waibel, "A novel objective function for improved phoneme recognition using time-delay neural networks," in IEEE Transactions on Neural Networks, vol. 1, no. 2, pp. 216-228, June 1990, doi: 10.1109/72.80233..

[10] Witt, S. M., & Young, S. J. (2000). Phone-level pronunciation scoring and assessment for interactive language learning. Speech communication, 30(2-3), 95-108..

[11] Petkov, P. N., Henter, G. E., & Kleijn, W. B. (2013). Maximizing phoneme recognition accuracy for enhanced speech intelligibility in noise. IEEE transactions on audio, speech, and language processing, 21(5), 1035-1045..

[12] Halpern, B. M., van Son, R., Brekel, M. V. D., & Scharenborg, O. (2020). Detecting and analysing spontaneous oral cancer speech in the wild. arXiv preprint arXiv:2007.14205..

[13] Gruhne, M., Dittmar, C., & Schmidt, K. (2007, September). Phoneme Recognition in Popular Music. In ISMIR (pp. 369-370)..

[14] Matejka, P., Schwarz, P., Cernocký, J., & Chytil, P. (2005). Phonotactic language identification using high quality phoneme recognition. In Ninth European Conference on Speech Communication and Technology..

[15] Zissman, M. A. (1995, May). Language identification using phoneme recognition and phonotactic language modeling. In 1995 international conference on acoustics, speech, and signal processing (Vol. 5, pp. 3503-3506). IEEE..

[16] Cognizant, "Cognizant homepage," Cognizant, [Online]. Available: https://www.cognizant.com/en_us.

[17] C. Lin, “TrendForce Says Global Market Value for Voice Recognition Will Register CAGR of 43.64% During 2016~2021 as Applications for AI Expands,” TrendForce , 19 01 2017. [Online]. Available: https://www.trendforce.com/presscenter/news/20170119-9637.html.

[18] T. Cox, “Siri and Alexa Fails: Frustrations With Voice Search,” themanifest, 08 01 2020. [Online]. Available: https://themanifest.com/digital-marketing/resources/siri-alexa-fails-frustrations-with-voice-search.

[19] Cowan, B. R., Pantidi, N., Coyle, D., Morrissey, K., Clarke, P., Al-Shehri, S., ... & Bandeira, N. (2017, September). " What can i help you with?" infrequent users' experiences of intelligent personal assistants. In Proceedings of the 19th International C.

[20] Bijani, M. (2020). The Evaluation of Automatic Speech Recognition of Non-native Dutch Speech for PostNL (Master's thesis)..

[21] Yilmaz, E., Ganzeboom, M. S., Cucchiarini, C., & Strik, H. (2017). Multi-stage DNN training for automatic recognition of dysarthric speech..

[22] Ahmadi, N. (2020). A comparative study of state-of-the-art speech recognition models for English and Dutch (Doctoral dissertation)..

[23] Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 18(5-6), 602-610..

[24] Röpke, W., Radulescu, R., Efthymiadis, K., & Nowé, A. (2019). Training a Speech-to-Text Model for Dutch on the Corpus Gesproken Nederlands. In BNAIC/BENELEARN..

[25] D. H. a. P. L. C. a. K. H. Beun, “PHONEME-BASED AUTOMATIC SPEECH RECOGNITION: towards a demonstrator for information retrieval, using Dutch hi-fi speech,” *Proceedings in institute of phonetic sciences, University of Amsterdam,* vol. 19, pp. 126-134, 1995.

[26] Schoepflin, J. R. (2012). Back to basics: Speech audiometry. Adelphi University. Retrieved on, 23(12), 20-45..

[27] M. S. A. W. F. W. &. T. Y. Hunter van Geffen, “A Review Of Deep Neural Network-based Phoneme Recognition Systems,” TUDelft, 2020.

[28] C.-H. a. J. B.-H. Lee, “A survey on automatic speech recognition with an illustrative example on continuous speech recognition of Mandarin,” *International Journal of Computational Linguistics \& Chinese Language Processing,* vol. 1, no. 1, pp. 1-36, 1996.

[29] Y. a. F. P. Liu, “Pronunciation modeling for spontaneous Mandarin speech recognition,” *International Journal of Speech Technology,* vol. 7, no. 2-3, pp. 155-172, 2004.

[30] S. T. a. E. e. a. Swarna, “A comprehensive survey on bengali phoneme recognition,” *arXiv preprint arXiv:1701.08156,* 2017.

[31] A. a. S. P. Garg, “Survey on acoustic modeling and feature extraction for speech recognition,” *International Conference on Computing for Sustainable Global Development (INDIACom),* vol. 3, pp. 2291-2295, 2016.

[32] Llisterri, J. (1992, July). Speaking styles in speech research. In Workshop on Integrating Speech and Natural Language..

[33] Nakamura, M., Iwano, K., & Furui, S. (2008). Differences between acoustic characteristics of spontaneous and read speech and their effects on speech recognition performance. Computer Speech & Language, 22(2), 171-184..

[34] Slobada, T., & Waibel, A. (1996, October). Dictionary learning for spontaneous speech recognition. In Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96 (Vol. 4, pp. 2328-2331). IEEE..

[35] Zissman, M. A., & Singer, E. (1994, April). Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling. In Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing.

[36] M. v. Oostendorp, “De kleinste klank van het Nederlands,” Meertens Knaw, 2020. [Online]. Available: https://www.meertens.knaw.nl/medewerkers/marc.van.oostendorp/propedeuse/5.stomme-e.html.

[37] Benzeghiba, M., De Mori, R., Deroo, O., Dupont, S., Erbes, T., Jouvet, D., ... & Rose, R. (2007). Automatic speech recognition and speech variability: A review. Speech communication, 49(10-11), 763-786..

[38] Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(2), 92-102..

[39] Cheng, G., Povey, D., Huang, L., Xu, J., Khudanpur, S., & Yan, Y. (2018). Output-Gate Projected Gated Recurrent Unit for Speech Recognition. In Interspeech (pp. 1793-1797)..

[40] Oostdijk, N. H. J. (2000). Het corpus gesproken nederlands..

[41] Unknown, “Fonemische Transcriptie,” http://lands.let.ru.nl/, 04 April 2000. [Online]. Available: http://lands.let.ru.nl/cgn/doc_Dutch/topics/version_1.0/annot/phonetics/fon_verantw_nl.pdf.

[42] Mees, I., & Collins, B. (1983). A phonetic description of the vowel system of Standard Dutch (ABN). Journal of the International Phonetic Association, 13(2), 64-75..

[43] Verhoeven, J. (2002). The sound system of Dutch in a general phonetic perspective. Antwerp Papers in Linguistics, 100, 165-176..

[44] Picone, J. W. (1993). Signal modeling techniques in speech recognition. Proceedings of the IEEE, 81(9), 1215-1247..

[45] Bishop, C. M. (2006). Pattern recognition and machine learning. springer..

[46] Toth, L. (2015). Phone recognition with hierarchical convolutional ´ deep maxout networks. EURASIP Journal on Audio, Speech, and Music Processing, 2015(1), 25..

[47] Graves, A., Mohamed, A. R., & Hinton, G. (2013B, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE..

[48] Graves, A., Jaitly, N., & Mohamed, A. R. (2013, December). Hybrid speech recognition with deep bidirectional LSTM. In 2013 IEEE workshop on automatic speech recognition and understanding (pp. 273-278). IEEE..

[49] Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Bengio, C. L. Y., & Courville, A. (2017). Towards end-to-end speech recognition with deep convolutional neural networks. arXiv preprint arXiv:1701.02720..

[50] Wu, Y. P., Mao, J. M., & Li, W. F. (2016, December). Robust speech recognition by selecting mel-filter banks. In 2nd Annual International Conference on Electronics, Electrical Engineering and Information Science (EEEIS 2016). Atlantis Press..

[51] Senior, A., & Lopez-Moreno, I. (2014, May). Improving DNN speaker independence with i-vector inputs. In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 225-229). IEEE..

[52] N. D. a. S. Shum, *Low-dimensional speech representation based on Factor Analysis and its applications,* Interspeech: Spoken Language System Group MIT Computer Science and Artificial Intelligence Laboratory , 2011.

[53] Richardson, F., Reynolds, D., & Dehak, N. (2015). Deep neural network approaches to speaker and language recognition. IEEE signal processing letters, 22(10), 1671-1675..

[54] D'Haro, L. F., Glembek, O., Plchot, O., Matějka, P., Soufifar, M., Cordoba, R., & Černocký, J. (2012). Phonotactic Language Recognition using iVvectors and Phoneme Posteriogram Counts. In Thirteenth Annual Conference of the International Speech Communicat.

[55] Lei, L., & Kun, S. (2017). Speaker Recognition Using Wavelet Packet Entropy, I-Vector, and Cosine Distance Scoring. Journal of Electrical and Computer Engineering, 2017..

[56] Aizat, K., Mohamed, O., Orken, M., Ainur, A., & Zhumazhanov, B. (2020). Identification and authentication of user voice using DNN features and i-vector. Cogent Engineering, 7(1), 1751557..

[57] Adda-Decker, M., & Lamel, L. (2000). The use of lexica in automatic speech recognition. In Lexicon Development for Speech and Language Processing (pp. 235-266). Springer, Dordrecht..

[58] J. Meyer, “https://jrmeyer.github.io/,” 17 april 2019. [Online]. Available: https://jrmeyer.github.io/asr/2019/08/17/Kaldi-troubleshooting.html. [Accessed 2020].

[59] Jiang, H. (2005). Confidence measures for speech recognition: A survey. Speech communication, 45(4), 455-470..

[60] Pinto, J., Garimella, S., Magimai-Doss, M., Hermansky, H., & Bourlard, H. (2010). Analysis of MLP-based hierarchical phoneme posterior probability estimator. IEEE Transactions on Audio, Speech, and Language Processing, 19(2), 225-241..

[61] K. Samudravijaya, “Automatic Speech Recognition,” Tata Institute of Fundamental Research Archives, 2004.

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

[63] Padmanabhan, J., & Johnson Premkumar, M. J. (2015). Machine learning in automatic speech recognition: A survey. IETE Technical Review, 32(4), 240-251..

[64] Deng, L., Hinton, G., & Kingsbury, B. (2013, May). New types of deep neural network learning for speech recognition and related applications: An overview. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 8599-8603). IE.

[65] Mohamed, A. R., Dahl, G., & Hinton, G. (2009, December). Deep belief networks for phone recognition. In Nips workshop on deep learning for speech recognition and related applications (Vol. 1, No. 9, p. 39)..

[66] Chavan, R. S., & Sable, G. S. (2013). An overview of speech recognition using HMM. International Journal of Computer Science and Mobile Computing, 2(6), 233-238..

[67] Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(2), 92-102..

[68] Toth, L. (2015). Phone recognition with hierarchical convolutional ´.

[69] Nayak, S., Bhati, S., & Murty, K. S. R. (2017, November). An investigation into instantaneous frequency estimation methods for improved.

[70] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. In Advances in neural information processing systems (pp. 577-585)..

[71] Canevari, C., Badino, L., Fadiga, L., & Metta, G. (2013). Relevanceweighted-reconstruction of articulatory features in deep-neuralnetwork-based acoustic-to-articulatory mapping. In Interspeech (pp. 1297-1301).

[72] Mohamed, A.; Dahl, G.; Hinton, G. (2011). Acoustic Modeling using Deep Belief Networks", IEEE Transactions on Audio, Speech, and Language Processing, 2011, ISSN 1558-7916..

[73] Graves, A., Jaitly, N., & Mohamed, A. R. (2013, December). Hybrid speech recognition with deep bidirectional LSTM. In 2013 IEEE workshop on automatic speech recognition and understanding (pp. 273-278). IEEE..

[74] Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE..

[75] Chorowski, J., Bahdanau, D., Cho, K., & Bengio, Y. (2014). End-to-end continuous speech recognition using attention-based recurrent NN: First results. arXiv preprint arXiv:1412.1602..

[76] Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(2), 92-102..

[77] S. Feng and T. Lee, “Improving cross-lingual knowledge transferability using multilingual TDNN-BLSTM with languagedependent pre-final layer,” in Accepted by INTERSPEECH, 2018.

[78] P. Smit, S. Gangireddy, S. Enarvi, S. Virpioja, and M. Kurimo, “Aalto system for the 2017 Arabic multi-genre broadcast challenge,” in Proc. ASRU, 2017, pp. 353–359..

[79] Veselý, K., Ghoshal, A., Burget, L., & Povey, D. (2013, August). Sequence-discriminative training of deep neural networks. In Interspeech (Vol. 2013, pp. 2345-2349)..

[80] Peddinti, V., Povey, D., & Khudanpur, S. (2015). A time delay neural network architecture for efficient modeling of long temporal contexts. In Sixteenth Annual Conference of the International Speech Communication Association..

[81] Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., ... & Khudanpur, S. (2016, September). Purely sequence-trained neural networks for ASR based on lattice-free MMI. In Interspeech (pp. 2751-2755)..

[82] The IBM 2016 English Conversational Telephone Speech Recognition System.

[83] Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2017). Improving speech recognition by revising gated recurrent units. arXiv preprint arXiv:1710.00641..

[84] Han, K. J., Chandrashekaran, A., Kim, J., & Lane, I. (2017). The CAPIO 2017 conversational speech recognition system. arXiv preprint arXiv:1801.00059..

[85] Cheng, G., Zhang, P., & Xu, J. (2019). Automatic Speech Recognition System with Output-Gate Projected Gated Recurrent Unit. IEICE Transactions on Information and Systems, 102(2), 355-363..

[86] Forney, G. D. (1973). The viterbi algorithm. Proceedings of the IEEE, 61(3), 268-278..

[87] V. Panayotov, "Decoding graph construction in Kaldi:," blogspot, 11 06 2012. [Online]. Available: https://vpanayotov.blogspot.com/2012/06/kaldi-decoding-graph-construction.html.

[88] L. D. D. Y. G. D. A.-r. M. N. J. A. Geoffrey Hinton, "Deep Neural Networks for Acoustic Modeling," 2012.

[89] Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE..

[90] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550-1560..

[91] Pascanu, R., Mikolov, T., Bengio, Y., On the difficulty of training Recurrent Neural Networks, Feb. 2013 - https://arxiv.org/pdf/1211.5063.pdf.

[92] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 6, no. 2, pp. 107-116, 1998.

[93] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780..

[94] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078..

[95] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780..

[96] Bayer, Justin Simon. Learning Sequence Representations. Diss. München, Technische Universität München, Diss., 2015, 2015..

[97] Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259..

[98] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555..

[99] Jia, Y., Wu, Z., Xu, Y., Ke, D., & Su, K. (2017). Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano's Continuous Note Recognition. Journal of Robotics, 2017..

[100] Jia, Y., Wu, Z., Xu, Y., Ke, D., & Su, K. (2017). Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano's Continuous Note Recognition. Journal of Robotics, 2017..

[101] Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., ... & Khudanpur, S. (2016, September). Purely sequence-trained neural networks for ASR based on lattice-free MMI. In Interspeech (pp. 2751-2755)..

[102] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. IEEE transactions on acoustics, speech, and signal processing, 37(3), 328-339..

[103] Kipyatkova, I. (2017, September). Experimenting with hybrid TDNN/HMM acoustic models for Russian speech recognition. In International Conference on Speech and Computer (pp. 362-369). Springer, Cham..

[104] Liu, B., Zhang, W., Xu, X., & Chen, D. (2019, May). Time Delay Recurrent Neural Network for Speech Recognition. In Journal of Physics: Conference Series (Vol. 1229, No. 1, p. 012078). IOP Publishing..

[105] Sugiyama, M., Sawai, H., & Waibel, A. H. (1991, June). Review of TDNN (Time Delay Neural Network) architectures for speech recognition. In 1991., IEEE International Sympoisum on Circuits and Systems (pp. 582-585). IEEE..

[106] Cheng, G., Povey, D., Huang, L., Xu, J., Khudanpur, S., & Yan, Y. (2018). Output-Gate Projected Gated Recurrent Unit for Speech Recognition. In Interspeech (pp. 1793-1797)..

[107] RU, “Fonemische Transcriptie,” RU, 04 04 2000. [Online]. Available: http://lands.let.ru.nl/cgn/doc_Dutch/topics/version_1.0/annot/phonetics/fon_verantw_nl.pdf.

[108] Cheng, G., Povey, D., Huang, L., Xu, J., Khudanpur, S., & Yan, Y. (2018). Output-Gate Projected Gated Recurrent Unit for Speech Recognition. In Interspeech (pp. 1793-1797)..

[109] van Niedek, T., Heskes, T., & van Leeuwen, D. (2016). Phonetic Classification in TensorFlow..

[110] Mohamed, A.-r., Dahl, G. E., & Hinton, G. (2012). Acoustic modelling using deep belief networks. Audio, Speech, and Language Processing, IEEE Transactions on, 20 (1), 14–22..

[111] Cheng, G., Povey, D., Huang, L., Xu, J., Khudanpur, S., & Yan, Y. (2018). Output-Gate Projected Gated Recurrent Unit for Speech Recognition. In Interspeech (pp. 1793-1797)..

[112] Cheng, G., Peddinti, V., Povey, D., Manohar, V., Khudanpur, S., & Yan, Y. (2017, August). An Exploration of Dropout with LSTMs. In Interspeech (pp. 1586-1590)..

[113] o Vinyals, O. (2014). Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 ..

[114] Peddinti, V., Wang, Y., Povey, D., & Khudanpur, S. (2017). Low latency acoustic modeling using temporal convolution and LSTMs. IEEE Signal Processing Letters, 25(3), 373-377..

[115] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Vesely, K. (2011). The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding (No. CONF). IEEE Signal Processing Society..

[116] E. Chodroff, “Kaldi Tutorial,” 13 11 2018. [Online]. Available: https://eleanorchodroff.com/tutorial/kaldi/index.html.

[117] Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(2), 92-102..

[118] Li, A., Zheng, F., Byrne, W., Fung, P., Kamm, T., Liu, Y., ... & Chen, X. (2000). CASS: A phonetically transcribed corpus of Mandarin spontaneous speech. In Sixth International Conference on Spoken Language Processing..

[119] Ward, W. (1991, April). Understanding spontaneous speech: The Phoenix system. In Acoustics, Speech, and Signal Processing, IEEE International Conference on (pp. 365-367). IEEE Computer Society..

[120] RDG, “Connected speech and coarticulation,” RDG, N.D.. [Online]. Available: http://www.personal.rdg.ac.uk/~llsroach/phon2/asscoareli-into.htm. [Accessed 11 12 2020].

[121] Collins, B. D., & Mees, I. (2003). The phonetics of English and Dutch. Brill..

[122] Global English, “Elision,” Global English, Unknown. [Online]. Available: https://www.global-english.com/custom/courses/pdf/elision-1.pdf. [Accessed 11 01 2021].

[123] Huang, Y. (2009). Phoneme recognition using neural network and sequence learning model (Doctoral dissertation, Ohio University)..

[124] Romdhani, S. (2015). Implementation of dnn-hmm acoustic models for phoneme recognition (Doctoral dissertation, University of Waterloo)..

[125] Nakagawa, S. (2002). A survey on automatic speech recognition. IEICE TRANSACTIONS on Information and Systems, 85(3), 465-486..

[126] Deng, L., Hinton, G., & Kingsbury, B. (2013, May). New types of deep neural network learning for speech recognition and related applications: An overview. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 8599-8603). IE.

[127] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554..

[128] Kaldi, “Decoding graph construction in Kaldi,” 2020. [Online]. Available: https://kaldi-asr.org/doc/graph.html.

[129] Chen, G. (2016). A gentle tutorial of recurrent neural network with error backpropagation. arXiv preprint arXiv:1610.02583..

[130] Colah, “Understanding LSTM Networks,” 27 08 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[131] Adda-Decker, M., & Lamel, L. (2000). The use of lexica in automatic speech recognition. In Lexicon Development for Speech and Language Processing (pp. 235-266). Springer, Dordrecht..

[132] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Silovsky, J. (2011). The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding (No. CONF). IEEE Signal Processing Society..

[133] Veselý, K., Ghoshal, A., Burget, L., & Povey, D. (2013, August). Sequence-discriminative training of deep neural networks. In Interspeech (Vol. 2013, pp. 2345-2349)..

[134] McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., & Sonderegger, M. (2017, August). Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In Interspeech (Vol. 2017, pp. 498-502)..

[135] Chegg, “Phonetics and Phonology - Coarticulation Effects,” Chegg, 2020. [Online]. Available: https://www.chegg.com/flashcards/phonetics-and-phonology-coarticulation-effects-459a0edf-47ac-49af-b92e-044deb6981fc/deck.

[136] Unknown, “Connected speech and coarticulation,” RDG, Unknown. [Online]. Available: http://www.personal.rdg.ac.uk/~llsroach/phon2/asscoareli-into.htm.

[137] Coetsem, Frans van1951Over en naar aanleiding van een Nieuwfriese AnlautwijzigingHandelingen der Zuidnederlandse Maatschappij voor Taal- en Letterkunde584-93.

[138] R. v. S. a. L. C. Pols, “AN ACOUSTIC PROFILE OF CONSONANT REDUCTION,” Amsterdam.

[139] TFCS, “Consonant to Consonant Linking,” TFCS, [Online]. Available: https://tfcs.baruch.cuny.edu/consonant-to-consonant-linking/#:~:text=Rule%201%3A%20When%20the%20final,for%20extra%20time%2C%20then%20released.

[140] D. Schreier, “https://benjamins.com/catalog/dia.17030.sch,” *Diachronica,* vol. 36, no. 1, p. 37–65, 2020.

[141] Warner, N., Jongman, A., Cutler, A., & Mücke, D. (2001). The phonological status of Dutch epenthetic schwa. Phonology, 387-420..

[142] Ileriseviye, “ileriseviye.wordpress.com,” ileriseviye, [Online]. Available: https://ileriseviye.wordpress.com/2011/02/17/speech-recognition-in-1920s-radio-rex-the-first-speech-recognition-machine/. [Accessed 03 03 2020].

[143] R. Pieraccini, “From AUDREY to Siri: Is speech recognition a solved problem?,” *{International Computer Science Institute at Berkeley,* p. http://www. icsi. berkeley. edu/pubs/speech/audreytosiri12. pdf, 2020.

[144] Samudravijaya, K. (2004). Automatic Speech Recognition. Tata Institute of Fundamental Research Archives..

[145] Prasad, M., van Esch, D., Ritchie, S., & Mortensen, J. F. (2019). Building Large-Vocabulary ASR Systems for Languages Without Any Audio Training Data. In INTERSPEECH (pp. 271-275)..

[146] Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. arXiv preprint arXiv:1402.1128..

[147] e. a. Jan Vanek, “Tuning of Acoustic Modeling and Adaptation Technique for a Real Speech Recognition Task,” in *Statistical Language and Speech Processing*, Bohemia, 2019, pp. 235-245.

[148] Pinto, J., Garimella, S., Magimai-Doss, M., Hermansky, H., & Bourlard, H. (2010). Analysis of MLP-based hierarchical phoneme posterior probability estimator. IEEE Transactions on Audio, Speech, and Language Processing, 19(2), 225-241..

[149] Forsberg, M. (2003). Why is speech recognition difficult. Chalmers University of Technology..

[150] Kajarekar, S., Malayath, N., & Hermansky, H. (1999). Analysis of sources of variability in speech. In Sixth European Conference on Speech Communication and Technology..

[151] Joyce, J. (2003). Bayes' theorem..

[152] Juang, B. H., & Rabiner, L. R. (1991). Hidden Markov models for speech recognition. Technometrics, 33(3), 251-272..

[153] Seshardi, N., & Sundberg, C. E. W. (1993). U.S. Patent No. 5,208,816. Washington, DC: U.S. Patent and Trademark Office..

[154] Delgado-Martins, M. R., & Freitas, M. J. (1991). Temporal Structures of Speech:" Reading News on TV". In Phonetics and Phonology of Speaking Styles..

[155] Cuevas, L. A., Ayuso, M. J. M., & Dauden, G. M. (1991). Analysis of the Spanish sequence" de" in content words and in function words in continuous speech. In Phonetics and Phonology of Speaking Styles..

[156] Bergem, D. R. V. (1991). Acoustic and lexical vowel reduction. In Phonetics and Phonology of Speaking Styles..

[157] Beinum, F. J. K. V. (1990). Spectro-temporal reduction and expansion in spontaneous speech and read text: the role of focus words. In First International Conference on Spoken Language Processing..

[158] Duez, D. (1991). Some factors affecting F2-patterns in spontaneous speech in French. In Phonetics and Phonology of Speaking Styles..

[159] Llisterri, J. (1992, July). Speaking styles in speech research. In Workshop on Integrating Speech and Natural Language..

[160] Uribe, M. C., & Corugedo, S. G. F. (1991). The construction of a corpus of spoken spanish: Phonetic and phonological parameters. In Phonetics and Phonology of Speaking Styles..

[161] Lacheret-Dujour, A. (1991). Phonological variations in read speech, reduction phenomena and speaker classes: Do allophonic choices represent speaking style?. In Phonetics and Phonology of Speaking Styles..

[162] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550-1560..

[163] Nakamura, M., Iwano, K., & Furui, S. (2008). Differences between acoustic characteristics of spontaneous and read speech and their effects on speech recognition performance. Computer Speech & Language, 22(2), 171-184..

[164] Booij, G. (1999). The phonology of Dutch. Oxford University Press..

[165] Adank, P. M. (2003). Vowel Normalization. A Perceptual acoustic study of Dutch Vowels. sn: sl..

[166] Booij, G. (1999). The phonology of Dutch. Oxford University Press..

[167] Adank, P. M. (2003). Vowel Normalization. A Perceptual acoustic study of Dutch Vowels. sn: sl..

[168] NOOTEBOOM, S. G., AND DOODEMAN, G. J. N. Production and perception of.

[169] Adank, P., Van Hout, R., & Smits, R. (2004). An acoustic description of the vowels of Northern and Southern Standard Dutch. The Journal of the Acoustical society of America, 116(3), 1729-1738..

[170] van Oostendorp, M. (2006). Topics in the phonology of Dutch. Unpublished manuscript, Meertens Instituut, The Netherlands..

[171] Sugiyama, M., Sawai, H., & Waibel, A. H. (1991, June). Review of tdnn (time delay neural network) architectures for speech recognition. In 1991., IEEE International Sympoisum on Circuits and Systems (pp. 582-585). IEEE..

[172] Thomas, E. R. (2003). Secrets revealed by Southern vowel shifting. American Speech, 78(2), 150-170..

[173] Farrington, C., Kendall, T., & Fridland, V. (2018). Vowel dynamics in the Southern vowel shift. American Speech: A Quarterly of Linguistic Usage, 93(2), 186-222..

[174] Mees, I., & Collins, B. (1983). A phonetic description of the vowel system of Standard Dutch (ABN). Journal of the International Phonetic Association, 13(2), 64-75. Retrieved February 22, 2021, from http://www.jstor.org/stable/44526698.

[175] TIMIT, D. Acoustic-phonetic continuous speech corpus cd-rom. Document NISTIR, 4930, 1-1..

[176] Canavan, Alexandra, David Graff, and George Zipperlen. CALLHOME American English Speech LDC97S42. Web Download. Philadelphia: Linguistic Data Consortium, 1997..

[177] Grondelaers, S., Van Hout, R., & Steegs, M. (2010). Evaluating regional accent variation in Standard Dutch. Journal of Language and Social Psychology, 29(1), 101-116..

[178] M. Phi, “Illustrated Guide to LSTM’s and GRU’s: A step by step explanation,” 24 09 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Accessed 05 05 2020].