2021•2022
Faculteit Industriële Ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Exploring Efficient Neural Architectures for Text-to-Speech Synthesis

PROMOTOR :
Prof. dr. ir. Ronald THOELEN
PROMOTOR :
dr. Mohammed Salah AL-RADHI
dr. Tamás Gábor CSAPO

Dean Reymen

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding UHasselt en KU Leuven

2021•2022

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Exploring Efficient Neural Architectures for Text-to-Speech Synthesis

PROMOTOR :

Prof. dr. ir. Ronald THOELEN

PROMOTOR :

dr. Mohammed Salah AL-RADHI

dr. Tamás Gábor CSAPO

Dean Reymen

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

UHASSELT KU LEUVEN

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Dean Reymen

EXPLORING EFFICIENT NEURAL ARCHITECTURES FOR TEXT-TO-SPEECH SYNTHESIS

SUPERVISOR

Dr. Csapó Tamás Gábor

CO-SUPERVISOR

Dr. Mohammed Salah Al-Radhi

BUDAPEST, 2022

Contents

STUDENT DECLARATION

I, **Dean Reymen**, the undersigned, hereby declare that the present MSc thesis work has been prepared by myself and without any unauthorized help or assistance. Only the specified sources (references, tools, etc.) were used. All parts taken from other sources word by word, or after rephrasing but with identical meaning, were unambiguously identified with explicit reference to the sources utilized.

I authorize the Faculty of Electrical Engineering and Informatics of the Budapest University of Technology and Economics to publish the principal data of the thesis work (author's name, title, abstracts in English and in a second language, year of preparation, supervisor's name, etc.) in a searchable, public, electronic and online database and to publish the full text of the thesis work on the internal network of the university (this may include access by authenticated outside users). I declare that the submitted hardcopy of the thesis work and its electronic version are identical.

Full text of thesis works classified upon the decision of the Dean will be published after a period of three years.

Budapest, 29 May 2022

...……
Dean Reymen

Abstract

Text-to-speech (TTS) is a computer-based technology that allows text to be read aloud, sounding like a human voice. Based on textual input, TTS synthesis generates a speech waveform. This study aims to explore the deep learning approach based on neural networks and investigate an efficient architecture to improve TTS synthesis. First, the current architecture of speech synthesis used in the Merlin toolkit will be investigated. Several speech syntheses, taken from the CMU_ARCTIC speech synthesis databases, will be run on this architecture, and objective and subjective evaluations will be used to evaluate the performance. For example, this study will examine the world and continuous vocoder already integrated into the Merlin toolkit, and a new vocoder, the ahocoder, will be integrated into the toolkit and evaluated afterwards. After that, both the configuration of the neural network will be investigated, think of the different types of neural networks and the number of hidden layers. The different optimization algorithms offered in Merlin will be investigated. The best-made model includes using the full dataset, a neural network with six hidden layers, and the stochastic gradient descent optimization algorithm. According to the objective evaluations, it can be concluded that the WORLD and continuous vocoder perform almost equally when comparing their MCDs. The ahocoder performs with an MCD of 6.063, which means that it is less accurate but still intelligible. The continuous vocoder performs best on the slt dataset with an MCD of 4.192. From the subjective evaluations, it can be concluded that the WORLD vocoder gives the best results, with a subjective rating of 74. In comparison, the implemented ahocoder has a rating of 62, which means that it is natural enough synthesized speech according to the listeners.

Samenvatting

Text-to-Speech (TTS) is een op de computer gebaseerde technologie die het mogelijk maakt tekst hardop voor te lezen, klinkend als een menselijke stem. Op basis van tekstuele input zorgt TTS synthese ervoor dat er een spraakgolfvorm gegenereerd wordt. Het doel van deze studie is het verkennen van de *deep learning* aanpak op basis van neurale netwerken en het onderzoeken van een efficiënte architectuur om de TTS synthese te verbeteren. Eerst zal de huidige architectuur van de spraaksynthese, gebruikt in de Merlin toolkit, onderzocht worden. Verschillende spraaksyntheses, afkomstig uit de CMU_ARCTIC spraaksynthese databanken, zullen op deze architectuur worden uitgevoerd, en de hand van een objectieve en subjectieve evaluatie zal de prestatie beoordeeld worden. Zo zal er in deze studie onderzoek worden gedaan naar de *world* en *continuous vocoder* die al geïntegreerd zijn in de Merlin toolkit en zal er een nieuwe *vocoder*, de *ahocoder*, geïntegreerd worden in de toolkit en zal deze achteraf geëvalueerd. Daarna zal zowel de configuratie van het neurale netwerk onderzocht worden, denk hierbij aan de verschillende type neurale netwerken en het aantal *hidden layers*, als de verschillende optimalisatie algoritmes aangeboden in Merlin onderzocht worden. Het best gemaakte model omvat het gebruik van de volledige dataset, een neuraal netwerk met zes *hidden layers*, en het *stochastic gradient descent*-optimalisatiealgoritme. Volgens de objectieve evaluaties kan worden geconcludeerd dat de *WORLD* en *continuous vocoder* vrijwel gelijk presteren bij de vergelijking van hun MCD's. Waarbij de *ahocoder* het slechts presteert met een MCD van 6.063, wat betekent dat het minder accuraat is maar wel nog verstaanbaar is. De *continue vocoder* presteert het beste op de slt dataset met een MCD van 4,192. Uit de subjectieve evaluaties kan worden geconcludeerd dat de *WORLD vocoder* het beste resultaat geeft, met een subjectieve waardering van 74. Terwijl de geïmplementeerde *ahocoder* een waardering van 62 heeft, wat betekent dat het een natuurlijk genoeg gesynthetiseerde spraak is volgens de luisteraars.

1 Introduction

Text-to-speech (TTS) is a computer-based technology that enables text to be read aloud, sounding like a human voice. TTS synthesis involves generating a speech waveform given textual input. TTS can be used for various things, including car navigation, train station announcements, telecoms response services, and e-mail reading. Nowadays, the goal of TTS is not to have machines talk but to make them sound like humans of different ages and gender. However, text-to-speech quality in conventional TTS systems is still far from natural-sounding. Deep learning is a new research direction in the machine learning area and has added a new perspective to the problem of speech synthesis.

The current thesis aims to explore the deep learning approach based on neural networks and investigate an efficient architecture to improve the TTS synthesis. First, the current architecture of the speech synthesis used in the Merlin toolkit will be examined. Several speech syntheses, taken from the CMU_ARCTIC speech synthesis databases, will be run on this architecture, and how they perform will be discussed. In addition, the efficiency of different vocoders will be explored. Three traditional vocoders will be researched, namely the world vocoder, the continuous vocoder and the ahocoder. Furthermore, the network architecture will be further investigated to see if the algorithm can be further optimized and if the number of hidden layers impacts the efficiency.

2 Fundamental theories

2.1 Machine learning

Machine learning (ML) is a type of artificial intelligence (AI) that aims to create systems that can learn from and improve processed data without being explicitly programmed [1]. Artificial intelligence is a broad phrase that refers to systems or technologies that are designed to emulate human intelligence. Even though machine learning and artificial intelligence are sometimes used interchangeably, they do not mean the same thing. One significant distinction is that, whereas machine learning is usually included in AI, AI is not necessarily included in machine learning.

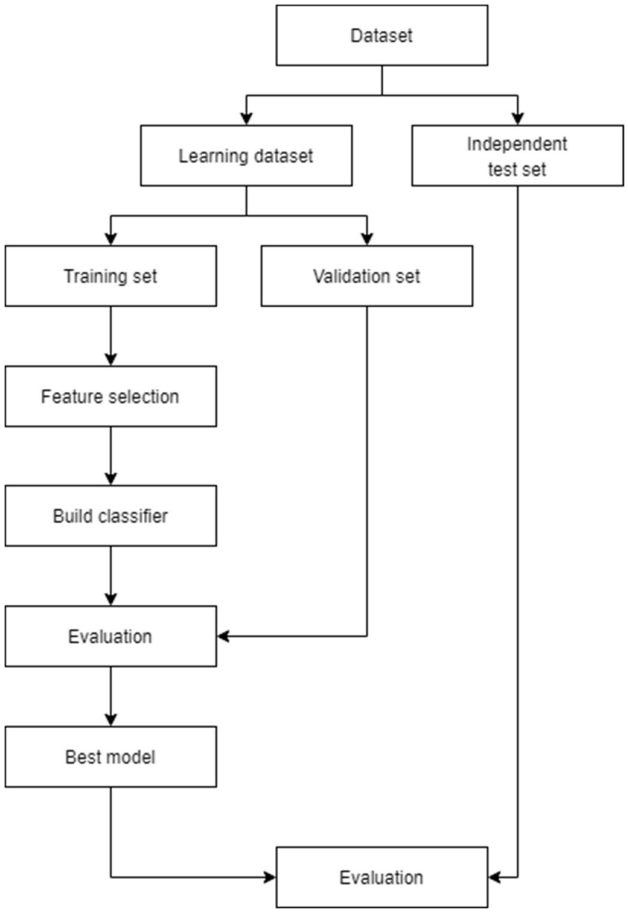


Figure 2.1: Machine learning flowchart [2]

In total, there are four types of algorithms used in machine learning: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. Roughly two of these types of algorithms are used for machine learning: supervised learning and unsupervised learning.

The difference between the two types is how data is processed to learn from it or make predictions with it [3].

2.1.1 Supervised machine learning

A data scientist serves as a middleman in this model, instructing the algorithm on which conclusions to make. The algorithm is 'trained' in supervised learning using a pre-labelled dataset. It has a predefined outcome, similar to how a toddler learns about different types of fruit by having them pointed out in a picture book.

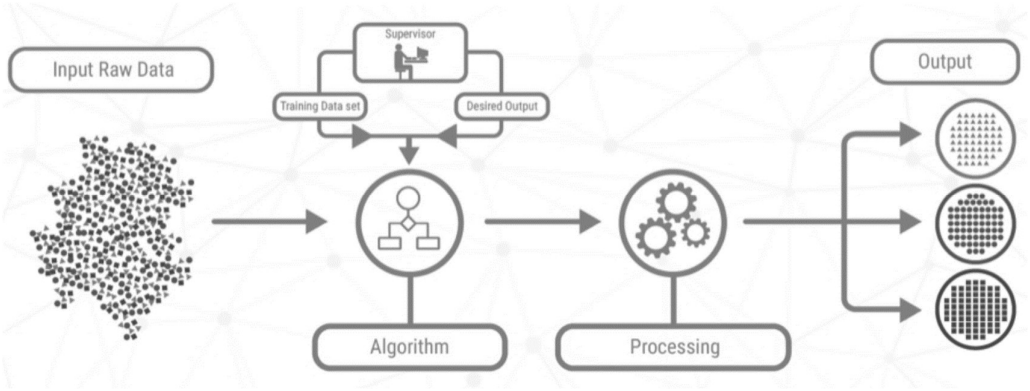


Figure 2.2: Supervised machine learning [4]

Algorithms for linear and logistic regression, multiclass classifiers, and support vector machines are examples of supervised machine learning.

2.1.2 Unsupervised machine learning

Unsupervised machine learning is a type of machine learning in which a computer learns to recognize complex processes and patterns without the assistance of a human. Unsupervised machine learning uses data without labels or a specified outcome.

Unsupervised machine learning, to continue with the "learning child" analogy, is analogous to a kid learning to distinguish fruit by seeing colours and patterns rather than memorizing the names spoken by the supervisor. The kid compares photographs and categorizes them into groups, each with its own label.

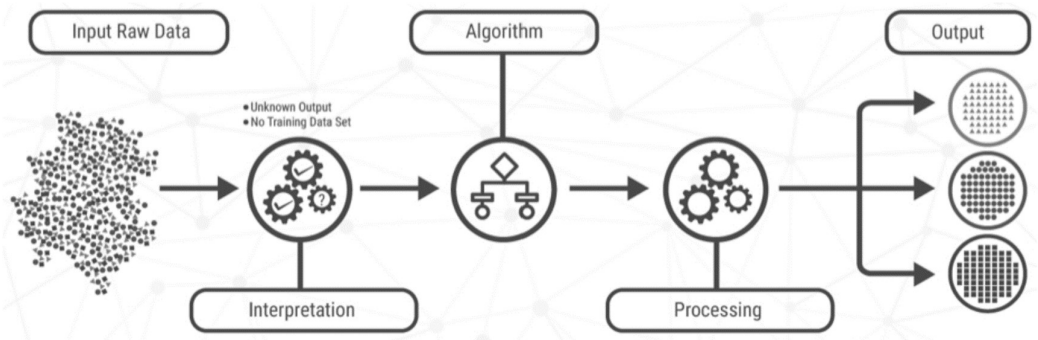


Figure 2.3: Unsupervised machine learning [4]

K-means clustering, main and independent component analysis, and association rules are some examples of unsupervised machine learning techniques.

2.1.3 Cost function

The cost function is used to express the algorithm's performance [5]. It will compare the predicted and actual outputs and calculate how accurate the model was in its prediction. If our predictions differ from the actual values, it returns a higher number. The cost function indicates how the model has improved as we adjust it to improve the predictions. The goal of all optimization strategies is to minimize the cost function.

2.1.4 Gradient descent

In order to attempt to minimize the cost function, gradient descent is used [6], [7]. The gradient descent optimization algorithm will attempt to minimize the cost function by moving in the steepest descent, defined by the negative of the gradient as seen in Figure 2.4.

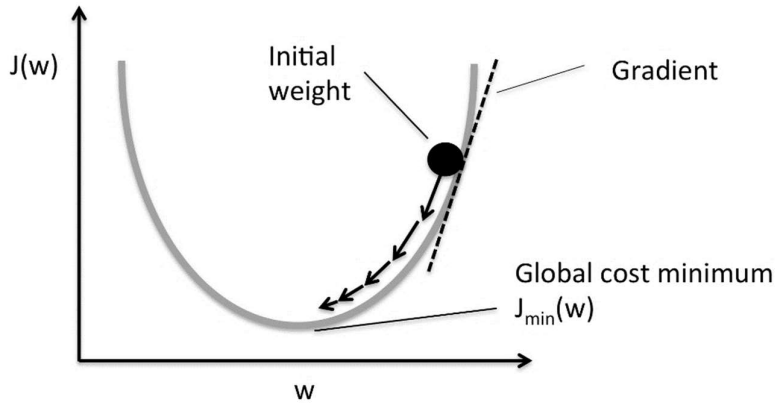


Figure 2.4: Visualisation of gradient descent [8]

2.1.4.1 Stochastic gradient descent

The most commonly used algorithm for solving optimization problems is stochastic gradient descent, abbreviated SGD [9]. The term "stochastic" refers to a system or process associated with a random probability. Consequently, a few random samples are chosen for each iteration in stochastic gradient descent rather than the entire dataset [10]. SGD does each iteration with only one sample, i.e., a batch size of one, making it less computationally intensive. To perform the iteration, the sample is randomly shuffled and selected. Because SGD selects only one sample from the dataset at random for each iteration, as illustrated in Figure 2.5, the path the algorithm takes to reach the minima is noisier than that of a standard gradient descent method.

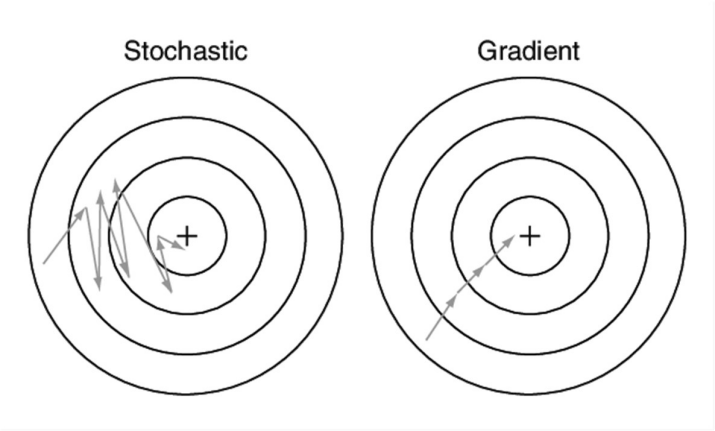


Figure 2.5: Stochastic gradient descent compared with gradient descent [11]

2.1.4.2 Adaptive movement estimation algorithm

The adaptive movement estimation algorithm, or Adam [12] for short, is an extension of the gradient descent optimization algorithm [13]. By reducing the number of function evaluations required to reach the optimum or improving the capability of the optimization algorithm by obtaining a better final result, the optimization process by using Adam can be accelerated [13]. The use of momentum and adaptive learning rates makes it so that adam can converge faster [14].

Adam combines adaptive gradients and root mean square propagation, two stochastic gradient descent methods [15]. Using a randomly selected data subset, the optimization algorithm creates a stochastic approximation.

2.1.4.3 Resilient backpropagation

Resilient backpropagation, or rprop for short, is a first-order optimization algorithm [16], [17]. The gradients are computed, and the step sizes are updated for each dimension individually in each iteration. By comparing the sign of the current and previous iteration gradient, the step size can

be determined. When both iterations have the same sign, this is the right direction, and the step will continue in the same direction, and the step size will be increased. If the sign of the last two iterations is different, we have missed the optimum, and the step will have to change direction, and the step will be reduced in size so that the optimum is not missed again. The change of direction of the step and increasing or decreasing the step can be seen in Figure 2.6.

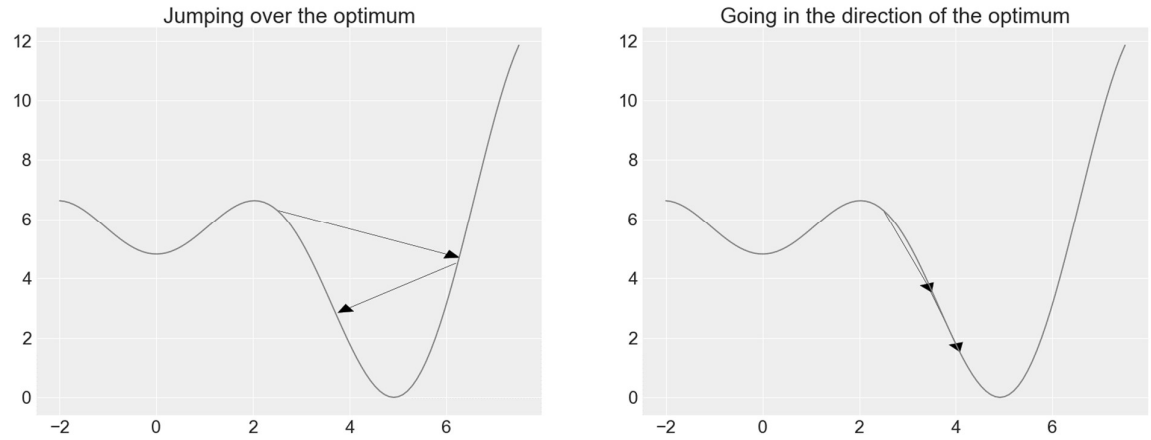


Figure 2.6: The gradient direction changes when jumping over optima [18]

2.1.5 Data set

In order to create a efficient systems that can learn from and improve processed data, the input data used to build the model is split into multiple data sets: test, training, and cross-validation.

2.1.5.1 Training set

The training dataset is used to train the model, i.e., the neural network. That is, based on the training dataset, the parameters of the neural network are determined [19].

2.1.5.2 Cross validation set

While tuning the model's hyperparameters, the validation data set allows for an unbiased evaluation of a model fit on the training data set [20]. In other words, the training set is used to fit the model, which is then used to predict the outputs for the validation set observations [21]. Based on this cross-validation set, the best hyperparameters [22] for the model can be determined.

Figure 2.7 shows an example of tuning the complexity of the neural network. Thus, it can be seen that with low complexity, high bias occurs. High bias can cause underfitting, which means that an algorithm can miss the relevant relations between features and target outputs [23]. However, as can be seen, too high a complexity is also not suitable because then high variance occurs. High variance can cause the data set on which the model is trained to be accurately represented, but when

this model is tested on an unknown data set, the output may be inaccurate [24]. This inaccuracy is due to overfitting.

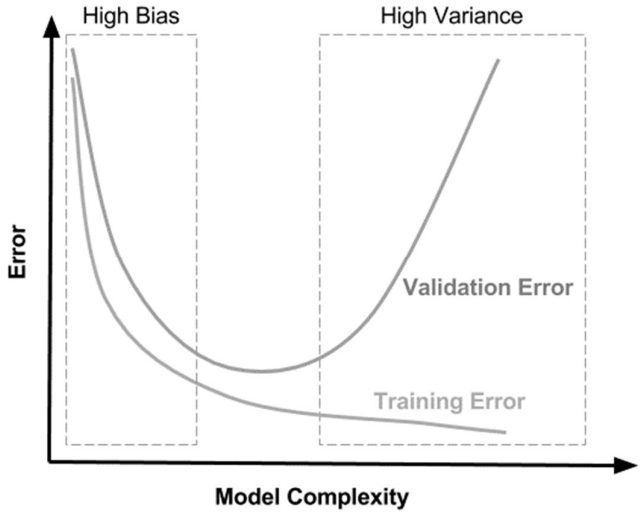


Figure 2.7: Bias and variance tradeoff [25]

2.1.5.3 Test set

The test data set is a portion of the input data that is used to provide an unbiased evaluation of a final model fit on the training data set [20].

2.2 Deep learning

Machine learning has a subfield called deep learning where artificial neural networks are used [26]. Deep learning allows computers to learn new skills by analyzing large amounts of data. Due to its versatility and effective results, deep learning is being used in various domains, including text generation, machine translation, speech synthesis, picture identification, and others [27]. Deep learning is a catch-all term for several types of neural networks and related algorithms that frequently consume raw input data.

2.2.1 Neural networks

Deep learning focuses on machine learning approaches using several layers of neurons in deep neural networks. Neural networks, meaning extremely elaborate and interconnected networks of neurons, are designed the same way as the human brain, and these neural networks mimic the human brain's behavior. The human brain, a biological neural network, allows computer programs to detect patterns and solve typical artificial intelligence, machine learning, and deep learning challenges.

When neural networks with multiple layers of neurons are implemented, deep learning occurs. A deep learning algorithm is defined as a neural network with more than one hidden layer or three layers in total, including inputs and outputs layers [28].

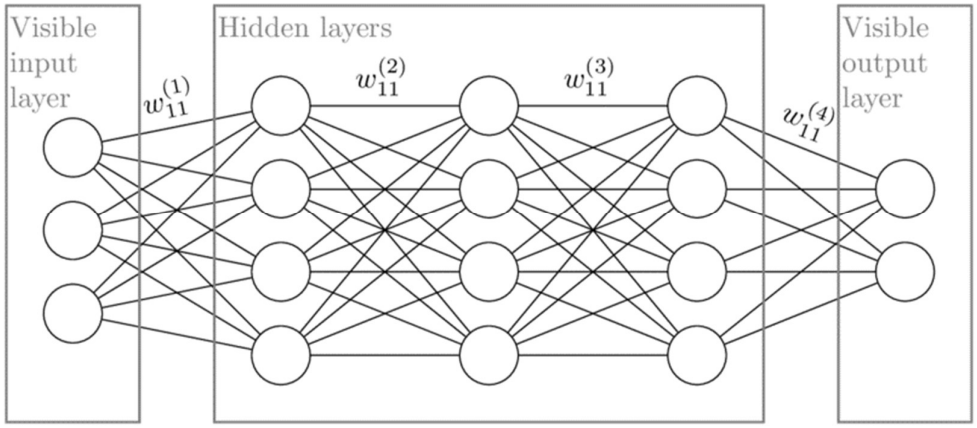


Figure 2.8: An example of a deep learning neural network with 3 hidden layers. Each layer is specified as a vector of binary components, with the edges between the vectors defined as a matrix of weight values. [29]

As seen in Figure 2.8, each neuron in a neural network assigns a weight to its input. The input is combined with weights that attenuate or amplify the input. The output of each layer is the input of the next layer. By combining weights with inputs, the network can rank and aggregate these inputs. Ultimately, the output is determined by the total of the weights. Neural networks are the backbone of deep learning algorithms [26].

Each node layer has an activation function that works on the previous layer's sum of inputs. The activation function determines whether or not the neuron is activated. It accomplishes this by applying a non-linear adjustment to the input, allowing the neural network to learn and perform more complex tasks [27].

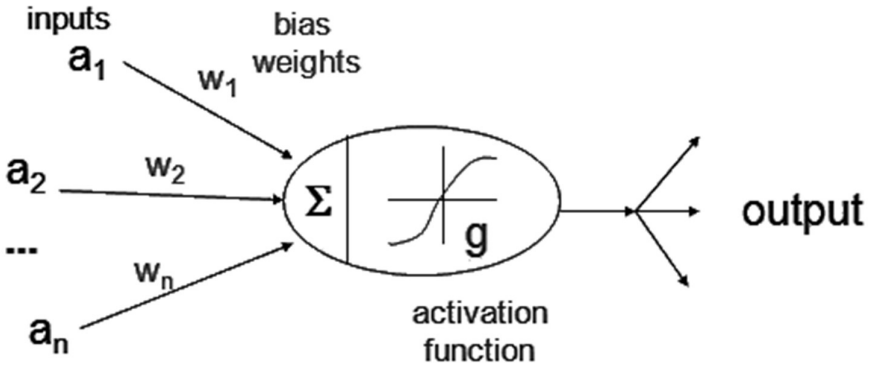


Figure 2.9: One single node with activation function [30]

2.2.1.1 Sigmoid activation function

A continuous space value can be converted to a binary value using the sigmoid function. Its non-linearity is its significant benefit over other steps and linear functions [31]. The function has an S form ranging from 0 to 1.

$$S(x) = 1 - \frac{1}{1 + e^{-x}} \tag{1.1}$$

As seen in Figure 2.10, when an independent variable z approaches negative infinity, the function return tends to be zero, and when z approaches positive infinity, it tends to be one.

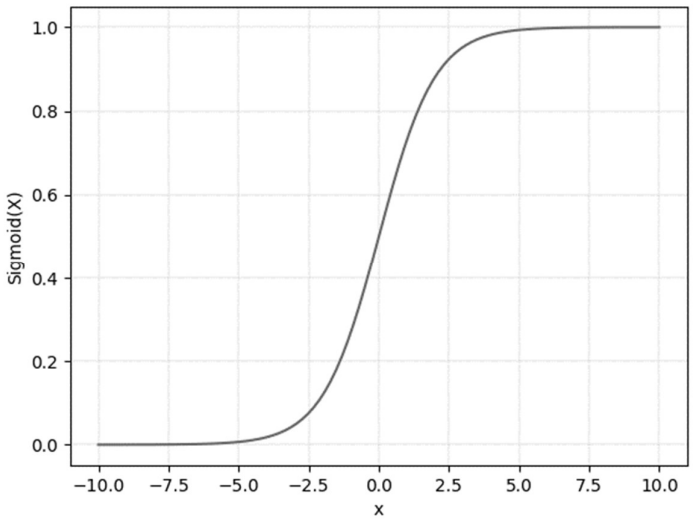


Figure 2.10: Plot of the sigmoid function

Some of its fundamental shortcomings are sharp damp gradients during backpropagation, gradient saturation, slow convergence, and non-zero centred output, allowing gradient updates to propagate in multiple directions [30].

2.2.1.2 TANH activation function

The hyperbolic tangent activation function is referred to as the tanh function. The function takes any real value as input and returns a value between minus one and one since the tanh function is similar to the sigmoid activation function and has the same S-shape [32].

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.2}$$

As seen in Figure 2.11, the more positive the variable, the more the function will tend to return one, but the more negative the variable, the more the function will tend to return minus one.

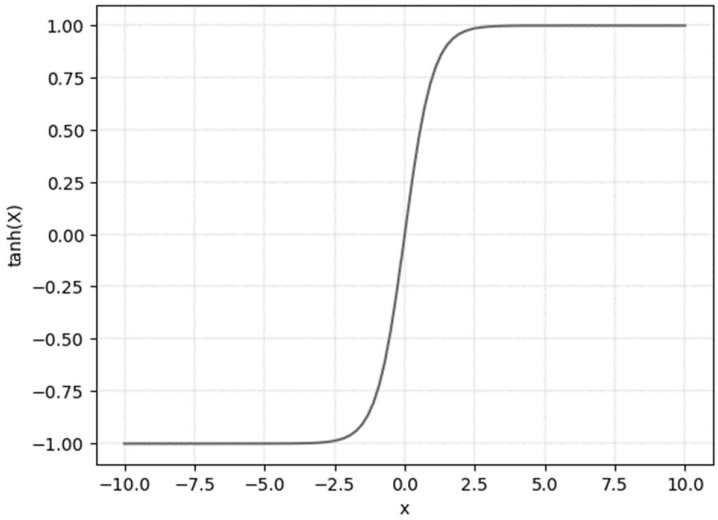


Figure 2.11: Plot of tanh function

2.2.2 Different types of neural networks

2.2.2.1 Feedforward neural network

The typical deep learning models are deep feedforward networks, also known as feedforward neural networks or multilayer perceptrons (MLPs) [33]. As seen in Figure 2.12, a feedforward neural network (FNN) is a type of artificial neural network in which nodes' connections do not form a cycle [34]. This means that the data always flows in one direction and never in the opposite direction.

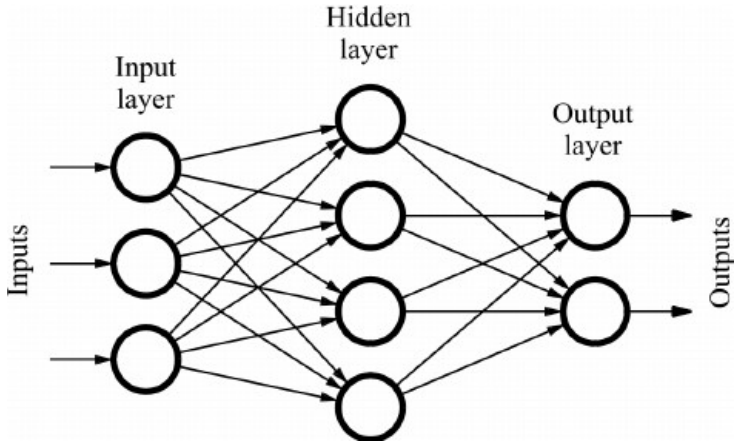


Figure 2.12: Sample of a feed-forward neural network [35]

The feedforward neural network was the first and most straightforward artificial neural network to be developed [36].

2.2.2.2 Recurrent neural network

Recurrent neural networks (RNNs) are identified by their feedback loops [37]. RNNs are often preferable for tasks that need sequential inputs, such as speech and language [38]. In order to predict the output of the layer, RNN makes it possible to save the previous output of a particular layer and feed it back to the input [39]. Advanced varieties of recurrent neural networks can produce the best results when processing sequential input such as text, audio, music, or video.

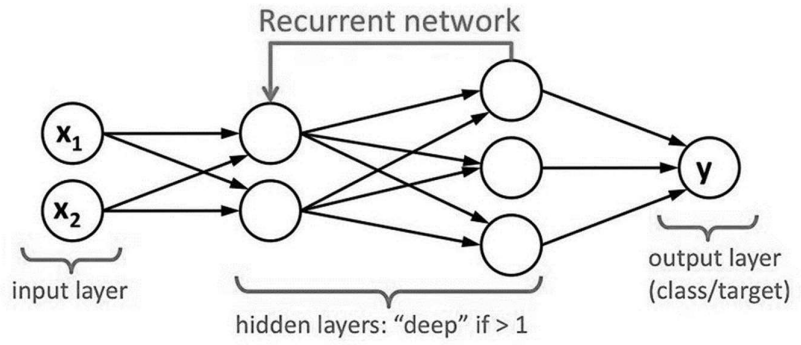


Figure 2.13: Recurrent neural network (RNN) [40]

Long short-term memory (LSTM) was introduced to alleviate the vanishing gradient problem [41], and it has since become one of the most widely used RNN architectures. By using LSTM, the vanishing problem can be avoided. Since errors can propagate backwards over an infinite number of virtual layers spread out in space. LSTM may learn tasks requiring memory of events that occurred hundreds or even millions of discrete time steps ago [36]. Therefore, it prevents backpropagated errors from vanishing or exploding [42].

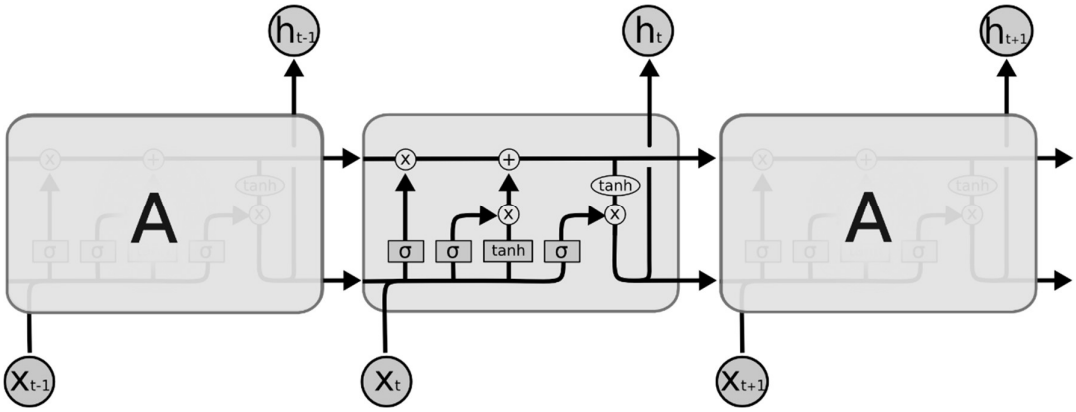


Figure 2.14: The repeating module in an LSTM contains four interacting layers [43]

2.3 Merlin

The Merlin toolkit [44] used in this paper is an open-source system for speech synthesis published by the Speech Research Center of Edinburgh University [45]. Merlin includes the extraction of characteristic acoustic parameters required for the vocoder, acoustic modelling learning function through a neural network, and vocoder function for voice waveform generation [46]. Merlin, based on the theano library, is written in Python and includes source code documentation and recipes for various system configurations. The toolkit is free software, distributed under an Apache License Version 2.0, allowing unrestricted commercial and non-commercial use alike [47].

As seen in Figure 2.15, the toolkit must be used in combination with a front-end text processor and a vocoder. Where the front-end makes sure the waveform is transformed into linguistic features and the vocoder makes sure the acoustic features are transformed into waveforms.

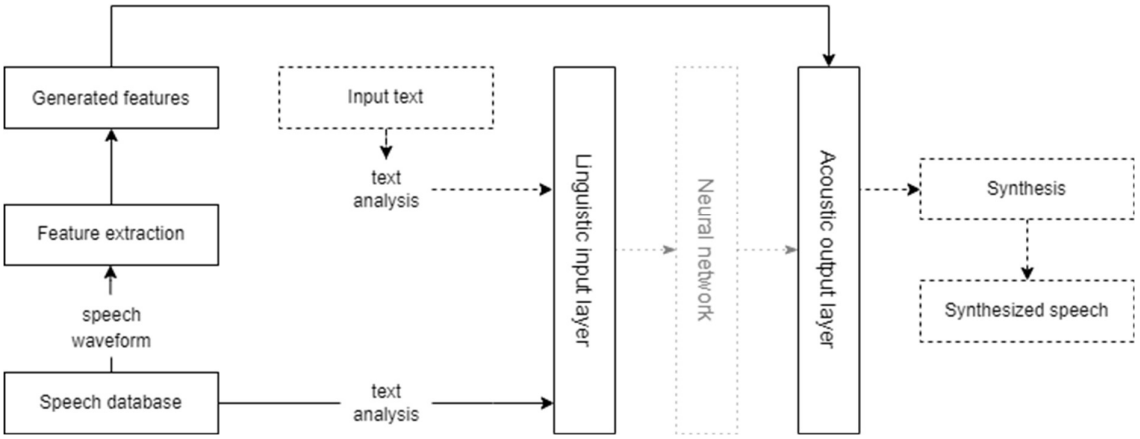


Figure 2.15: Overview of the used model in Merlin

2.3.1 Front-end

An external front-end, such as Festival, is required in Merlin [45]. For the time being, the front-end output must be formatted as HTS-style labels with state-level alignment. The toolkit transforms such labels into vectors of binary and continuous features for neural network input. Using HTS-style questions, the characteristics are extracted from the label files. If the HTS-like approach is not convenient, it is also possible to offer already-vectorized input features directly.

2.3.2 Vocoder

Vocoders analyze and synthesize human voice signals for audio data compression, multiplexing, voice encryption, and voice transformation.

2.3.2.1 WORLD

The WORLD vocoder is open-source speech analysis, modification, and synthesis software [48]–[50]. As seen in Figure 2.16, it can estimate the fundamental frequency (F0), aperiodicity, and spectral envelope and synthesize speech with only estimated parameters. Despite the development of various high-quality speech synthesis systems, real-time processing has proven difficult due to its high computational costs. This new speech synthesis technology has excellent sound quality, but it also processes information quickly [50]. The system's effectiveness was determined by comparing its output to natural speech, which included consonants. Its processing speed was also measured against that of traditional systems. The results showed that WORLD outperformed the other systems in terms of sound quality and processing speed. It was more than ten times faster than conventional systems, and the real-time factor (RTF) suggested that it could handle real-time processing.

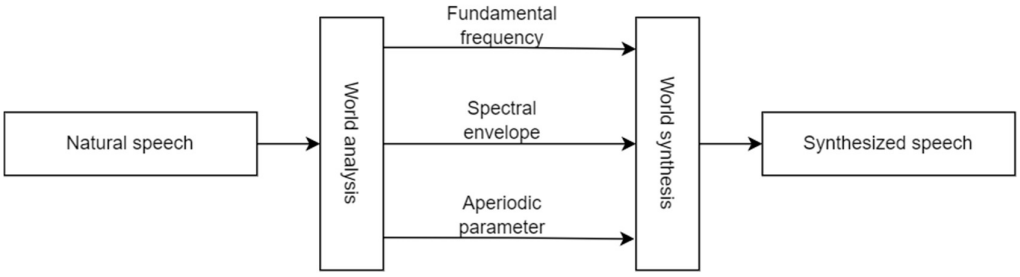
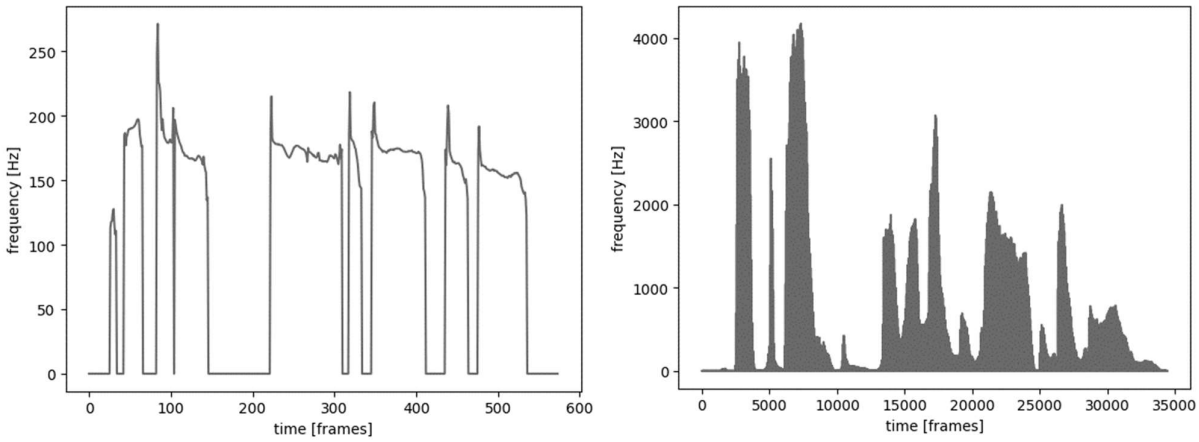
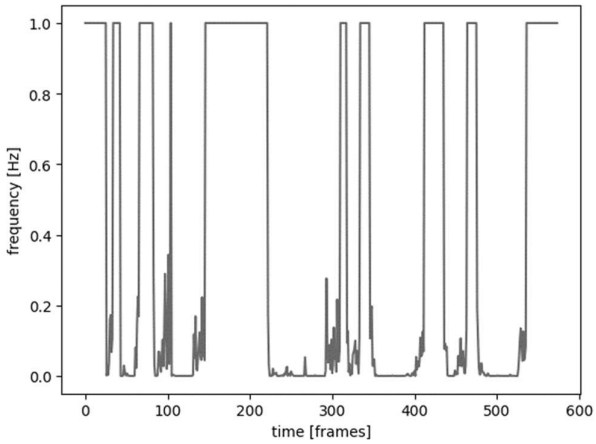


Figure 2.16: Simplified world vocoder workflow

Three algorithms for determining three speech parameters are included in WORLD, and a synthesis algorithm that uses these parameters as input. First, DIO [51], [52] is used to estimate the f0 contour. Second, the spectral envelope is calculated using CheapTrick [53], [54], which considers both the waveform and the F0 data. Third, PLATINUM [55] is utilized to estimate the excitation signal, then used as an aperiodic parameter.



(a) LF0 (b) MGC



(c) BAP

Figure 2.17: Example of features extracted by the WORLD vocoder

2.3.2.2 Continuous

Figure 2.18 and Figure 2.19 show the continuous vocoder's analysis and synthesis phases [56].

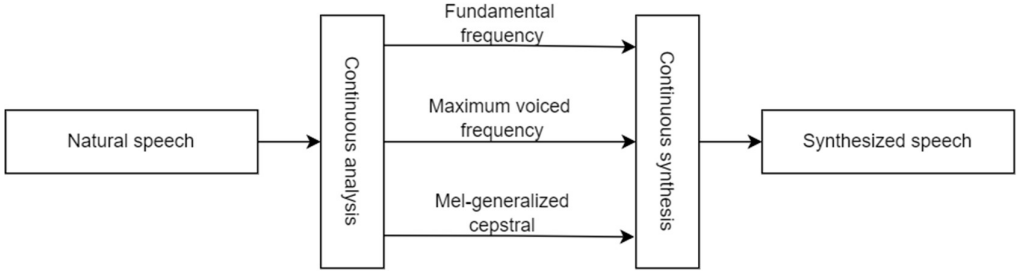


Figure 2.18: Simplified continuous vocoder workflow

The continuous fundamental frequency (contF0) is determined using a simple continuous pitch tracker on the input waveforms during the analysis phase [57]. This pitch tracker uses a linear dynamic system with Kalman smoothing to interpolate F0 in areas of creaky voice, as well as unvoiced sounds or silences. Another excitation parameter is the maximum voiced frequency (MVF), which uses amplitude and phase spectra to determine MVF decisions [58] using a maximum probability criterion. An an accurate and temporally stable spectral envelope estimation called CheapTrick [53] is used to achieve high-quality Mel-Generalized Cepstral analysis (MGC). The results are the contF0, MVF, and MGC parameter streams.

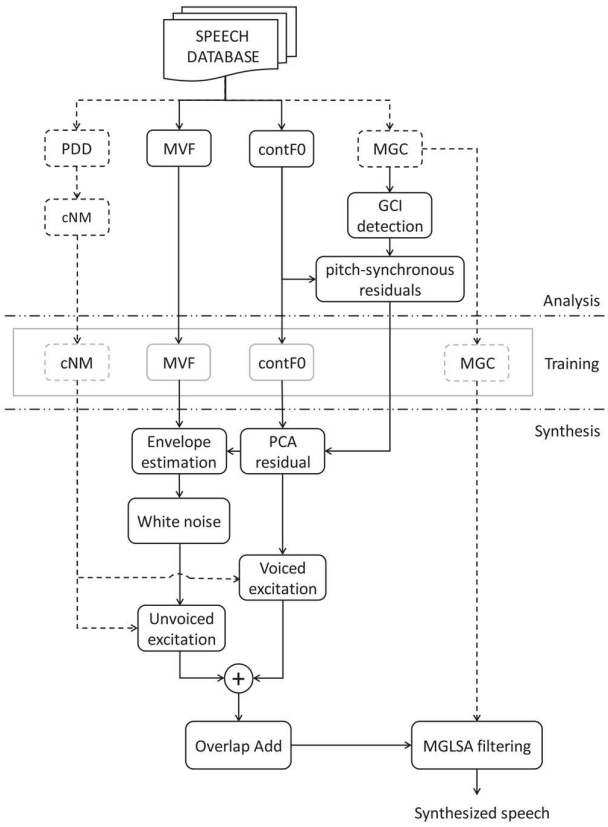
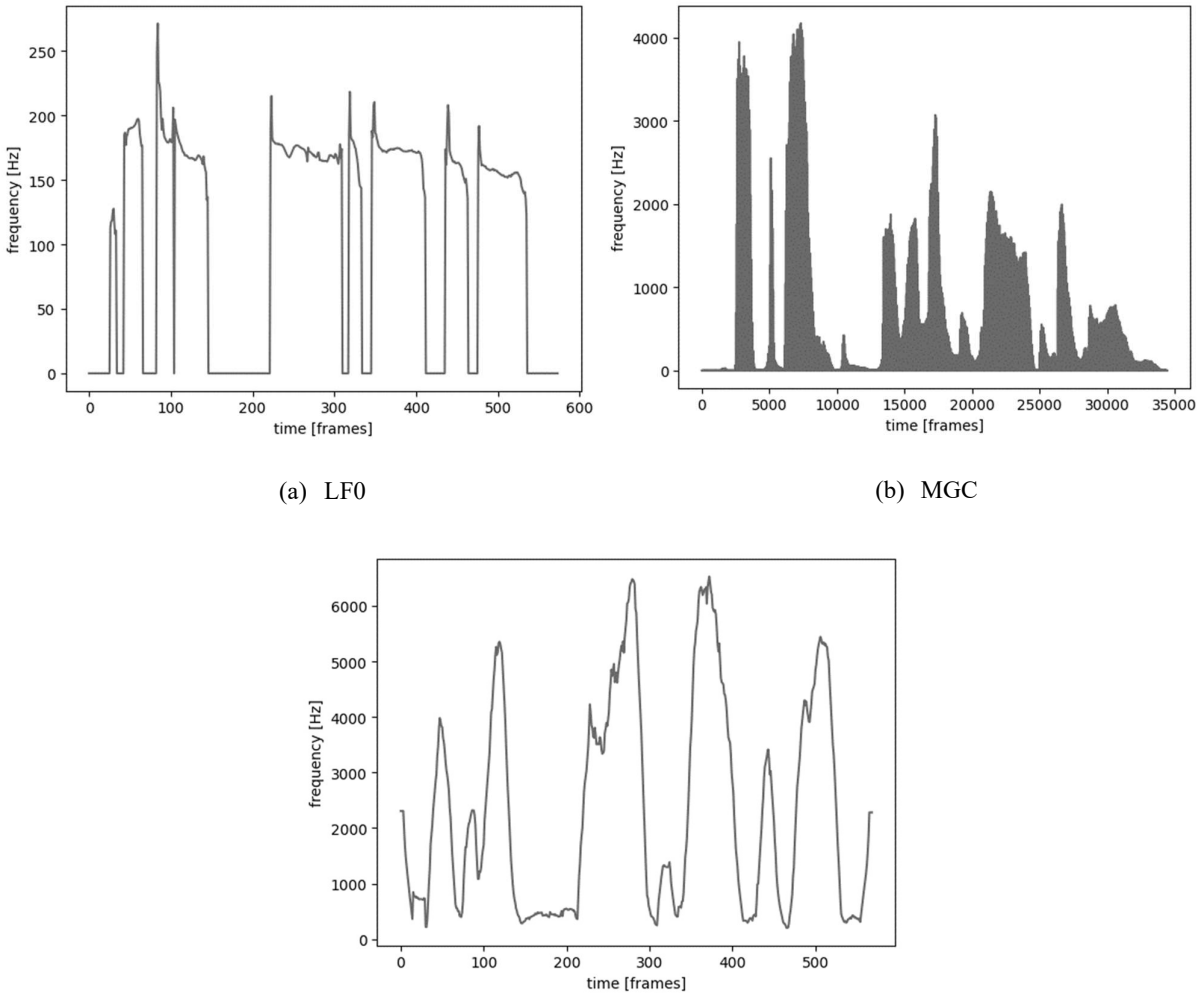


Figure 2.19: Schematic diagram of the developed continuous vocoder. Additions and refines are marked with dashed lines. [56]

The continuous vocoder can avoid per-frame voicing decisions, which can contribute to minimizing perceptual degradation caused by voicing decision errors [56]. Furthermore, it models the excitation using only two one-dimensional parameters, which is computationally feasible in deep neural network-based text-to-speech [59], [60]. The noise component in the continuous baseline vocoder, on the other hand, is still not accurately modelled, limiting the overall perceived quality.

In order to reduce the overall buzziness of the voice in the continuous baseline vocoder [61], a combination of the binary noise masking (bNM) and Phase Distortion Deviation (PDD) called continuous noise masking (cNM) is used. The PDD of the signal carries all of the crucial information relevant to the glottal pulses shape [62]. Moreover, noise masking is a fundamental technique to improve the performance of the speech synthesizer by reducing the number of noise artefacts in the time-frequency domain. Based on a simple measure of harmonicity, a bNM in the time-frequency space is used in [63]. However, since bNM might lack a minimum of randomness in the voiced segments because of forcing values below the threshold to zero [63], [64], bNM and PDD are combined. Therefore changes from 0 to 1 (or 1 to 0) are made in the cNM, rather than a binary 0 or 1 as in the bNM, and hence preserve the quality of the voiced segments.



(a) LF0

(b) MGC

(c) MVF

Figure 2.20: Example of features extracted by the continuous vocoder

2.3.2.3 Ahocoder

As seen in Figure 2.21, the ahocoder parameterizes speech waveforms into three streams: log-f0, cepstral representation of the spectral envelope, and maximum voiced frequency [65]. It provides high accuracy during analysis and high quality during reconstruction. For statistical parametric speech synthesis and voice conversion, it is adequate. It can also be used for simple speech manipulation and transformation (pitch level and variance, speaking rate or vocal tract length).

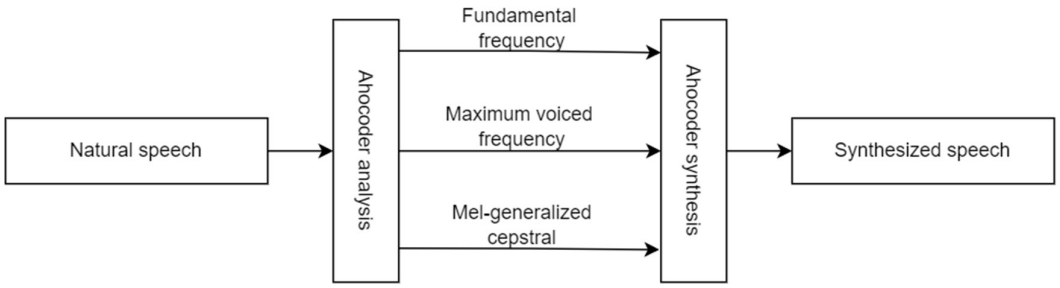
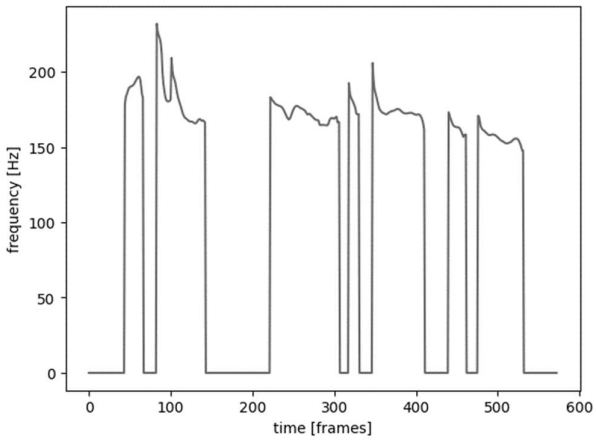
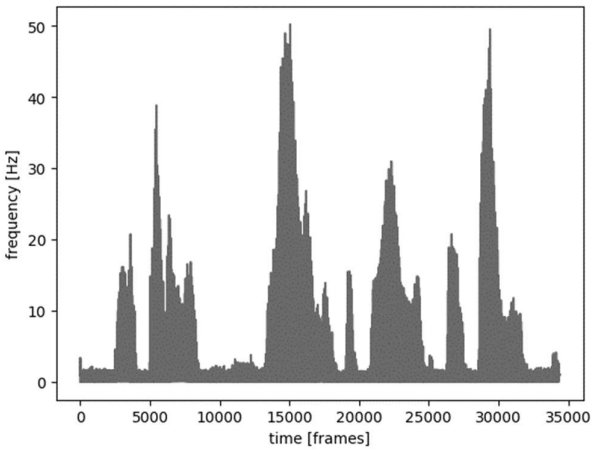


Figure 2.21: Simplified ahocoder vocoder workflow

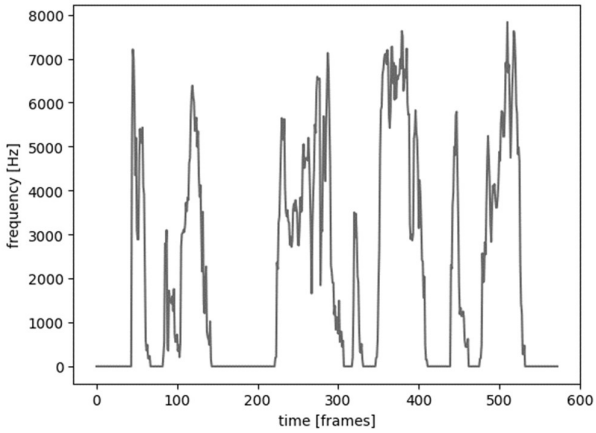
The way the three different parameter streams ($\log f_0$, MCEP coefficients and MVF) are obtained and discuss different issues arose during the development of the vocoder are described in [65, Sec. II], [65, Sec. III] and [65, Sec. IV]. The harmonics-plus-noise model (HNM) [66] based reconstruction procedure is described in [65, Sec. V]. The optimization of several aspects of the vocoding process in resynthesis tasks is devoted in [65, Sec. VI]. The performance of the optimized vocoder in synthesis is evaluated in [65, Sec. VII]. The final conclusions are summarized in [65, Sec. VIII].



(a) LF0



(b) MGC



(c) MVF

Figure 2.22: Example of features extracted by the ahocoder

Unlike other HNM-based systems, which reconstruct signals at the pitch-synchronous frame rate and frame length, the reconstruction method, in this case, has been intended to function at a constant frame rate, making it compatible with statistical synthesizer output.

3 Related work

Studies have attempted to research the improvement of the TTS synthesis using the Merlin toolkit. Merlin is a stable toolkit where text-to-speech can be applied to datasets in the English language and foreign languages. For example, [67] obtains accurate results with the Arabic language by examining DNN-based architectures and figured out that a class-specific modelling approach, which for each sound, uses the model that performs the best on the validation set, improves the results. [68] achieved first place in all three subjects, including naturalness, intelligibility, and MOS, with an optimized DNN-based speech synthesis system. The results of [68] indicated that using deeper architectures improves the synthesized speech quality. [68], [69] describe that less training data reduces speech quality.

It is possible to modify the Merlin toolkit with many different options. For example, several studies examine modifying the vocoder. A hierarchical encoder-decoder model is proposed in [70]. The model is designed to make better use of suprasegmental characteristics than traditional designs while also being computationally efficient. The suggested solution outperforms a traditional design requiring linguistic input to be at the same frame rate as the auditory input.

It is not always the case that a developed vocoder performs better than an old version. In the following study [71], a vocoder called Yang was developed where the results were worse than the already existing WORLD vocoder since the low accuracy of the spectral envelope.

The possibility of a WaveNet architecture as a statistical vocoder is researched in [72]. In this research it could be concluded that the WaveNet architecture gives a good result since only one hour of training data is enough for producing very good quality of speech.

Instead of a vocoder-based system, [73] uses a hybrid text-to-speech framework that uses a waveform generation method based on the natural speech waveform exemplars. Using a larger dataset that contains around four hours of training data, the examplar-based waveform generation variants were rated higher than the vocoder-based system.

4 Task analysis

There are several approaches to improving neural architectures' efficiency for text-to-speech synthesis. The various modifications and tests to research the approaches will be performed using the Merlin toolkit.

It is also possible to obtain better results by modifying the vocoder. Thus, the world, continuous and ahocoder will be integrated in the toolkit. The world and continuous vocoder have already been implemented in the Merlin toolkit. The code for the ahocoder needs to be written since the ahocoder is not implemented in the Merlin toolkit. In order to implement the ahocoder, the first step will be to look at how exactly this vocoder works. After that, the code for the analysis and synthesis step will have to be written. Once the vocoders are implemented in the toolkit and the demo datasets are tested. When these give a correct result the full dataset will be tested on the vocoders.

In order to optimize the use of the vocoders, the neural network architecture can be modified. These hidden layers influence the final result by changing the hidden layers, such as adding extra layers or using a different type of neural network. In addition, not only does the architecture affect the efficiency but also how this architecture is trained. The algorithm used to calculate the gradient descent also impacts efficiency. For example, different optimizers such as sgd, adam and rprop will be used.

The different datasets that will be used, consist of male and female speakers, and this will allow the research to determine which architectures work better on male or female speakers. Two males, awb and ksp, will be used, where awb is a Scottish English male and ksp is an Indian English male, and two English female speakers, slt and clb, will be used.

All these possibilities will be investigated, and the natural and synthesized speech will be compared using MCD, BAP, RMSE, correlation and VUV.

5 Implementation

In this section, there will be a discussion of the tools, dataset and optimizations used in this thesis. It will include a discussion of exactly how these datasets and optimizations were implemented and the problems and errors that come with this implementation.

5.1 Dataset

The CMU Arctic databases were designed for the purpose of speech synthesis research [74]. The database contains datasets of male and female speakers. In this thesis, two male (awb and ksp) and two female (clb and slt) speakers were used. Each dataset contains about 1500 out-of-copyright utterances, including 16KHz waveform and simultaneous EGG signals. Only the waveforms and the sentence prompt list are needed to perform speech synthesis.

5.2 Working environments

5.2.1 Google Colab

Google Colab is a Google Research product. Colab allows anyone to write and run arbitrary Python code in the browser, making it ideal for machine learning [75]. The choice to use Colab is because it allows Jupyter notebooks to be shared with others in an easy way.

In order to make sure that the toolkit would work on Colab the required libraries and tools had to be installed on this environment. Since Merlin works on Python versions 2.7 to 3.6, it is necessary to downgrade the environment.

```
sudo update-alternatives --config python3
```

By using the above code, the python version can be changed manually. In this case, python is downgraded from python 3.7 to python 3.6.

In addition, it is necessary to install the basic tools for Merlin. This was not without problems either, first the pip had to be updated, next it was necessary to install the corresponding packages. Before compiling the tools, it was necessary to manually install csh, manpages-pl, manpages-fr-extra, autotools-dev, automake, python3-numpy and python3-scipy. The Python package, cmake and bandmat, also needed to be installed

In addition, Colab is not an ideal environment to run the code. For example, Colab has an idle timeout of 90 minutes and an absolute timeout of 12 hours. As a result, it was impossible to train the neural network efficiently, and valuable time was lost.

5.2.2 TMIT Deep 1 server

As an alternative to Colab, a docker image running on a larger server of the TMIT department was chosen. The correct python version was already installed on this server. Only the additional packages had to be installed to be able to compile the tools within Merlin.

In order to run longer scripts in the background, tmux is used. Tmux is a program which runs in a terminal and allows multiple other terminal programs to be run inside it [76], and this made it possible to run multiple scripts simultaneously.

5.3 Implementation of the Merlin toolkit

As discussed earlier, the Merlin toolkit is used to perform speech synthesis. This section will discuss the practical part of Merlin, such as what code and scripts are used and what code is written.

Merlin includes many sample scripts to learn how to use the toolkit. Depending on the purpose of the speech synthesis, a different example script is used. For example, there is a sample script to perform speech synthesis on a mandarin dataset. There also is a sample script to use your dataset to develop a speech synthesis.

5.3.1 SLT Arctic

Merlin recommends testing the SLT Arctic demo first to ensure all libraries, tools and packages are installed correctly. The demo is run by using the run_demo script.

```
./run_demo.sh
```

However, this demo did not work the first time. For example, an error was generated when calculating the MCD.

```
TypeError: No loop matching the specified signature and casting was found for ufunc
add
```

The error is related to the version of NumPy and SciPy. The version of NumPy is downgraded from version 1.19.2 to version 1.16.4, and SciPy upgraded to version 1.2.3.

By updating both Python packages, the error is resolved. Now that we know the program works, exactly what steps are being performed can be examined to reuse the program with another dataset.

```
./01_setup.sh voice_name
```

In order to run the setup, one parameter is required. This parameter is the name of the voice. If the parameter matches the preexisting datasets, the setup will ensure the dataset is downloaded. The following datasets are available in the setup: slt_arctic_demo, slt_arctic_full, awb_arctic_demo, awb_arctic_full. The bdl_arctic_full dataset for this example is no longer available.

The setup creates the necessary folders, and moves the downloaded data in the dataset to the correct place. It also creates the global configuration file. This configuration file contains the information the speech synthesis needs. For example, it contains the partition between the train, validate and test set to train the model.

```
./02_prepare_conf_files.sh conf/global_settings.cfg
```

The parameter given is the path to the global configuration file. In this step, the configuration files are created for the acoustic and duration model and the synthesis.

```
./03_train_duration_model.sh conf/duration_voice_name.conf
```

This step ensures that the duration model is generated and trained. The duration model is trained with the state-aligned data. This model trains the state-level durations and is needed as an input to the acoustic model to predict the speech parameters at the synthesis time [77].

The main Merlin script (*run_merlin.py*) that is used to train the neural network, is fed into another script (*submit.py*). This script makes sure that the job runs on the CPU or GPU. The main script used to train the neural network is *train_DNN* which is shown below with its arguments and defaults [78].

```
def train_DNN(train_xy_file_list,   # training file list
           valid_xy_file_list,      # validation file list
           nnets_file_name,         # filename for DNN we save to disk
           n_ins,                   # input feature dimensionality
           n_outs,                  # output feature dimensionality
           ms_outs,                 # multistream_outs
           hyper_params,            # hyperparameters for training and architecture
           buffer_size,             # training buffer size
```

```
            plot=False,              #   create   plot   of   (train/dev)   training
convergence
            var_dict=None,           # load covariance matrix
            cmp_mean_vector = None,  # cmp == audio features used in HTS training
            cmp_std_vector = None,   # cmp == audio features used in HTS training
            init_dnn_model_file = None):  # DNN model with which we initialize new
DNN
```

Once the duration model is trained, the next step is to train the acoustic model.

```
./04_train_acoustic_model.sh conf/acoustic_voice_name.conf
```

Training the neural network for the acoustic model is identical to training the duration model. Nevertheless, since the acoustic configuration file is used, the neural network will comply with the configuration of the acoustic model. Also, in this case, the model will be trained in *run_merlin.py* and the function *train_DNN* to be exact. The Merlin script will be plugged into another script and will thus be run on the CPU or GPU.

```
./05_run_merlin.sh                                  conf/test_dur_synth_voice_name.conf
conf/test_synth_voice_name.conf
```

The final step is the synthesis. This step can be divided into two steps. First, there will be a synthesis for the durations and second the acoustic synthesis.

Now that the different steps have been mastered, it is possible to research the different configurations to improve the synthesis's efficiency. One of the first steps to get a better result is to enlarge the size of the dataset. The demo dataset contains only 60 samples, while the full dataset contains about 1132 samples. This adjustment is accomplished by downloading and implementing a different dataset.

As a standard, the neural network consists of 6 TANH layers with a layer size of 1024, and this is referred to as the DNN in this paper. The following change that will be implemented is modifying the neural network. Instead of the six TANH layers, five TANH layers with 1024 layers followed by an LSTM layer with 512 layers will be used, and this is referred to as the LSTM in this paper. This modification is accomplished by modifying the configuration files for the duration, acoustic, and synthesis models. It is essential to adjust the layout of the neural network and set the sequential training parameter to true.

Now that we know how to modify the neural network and implement other pre-made datasets into the model, it is possible to perform speech synthesis for different gender. Therefore,

we will use a male speaker, namely the dataset of US English by Indian English male (ksp) from the CMU Arctic database.

5.3.2 Build your own voice

Since Merlin's example script, slt arctic uses the pre-made datasets. The correct labels and features in these datasets are already generated. Therefore the ksp dataset cannot be executed in slt arctic. Therefore, another script must be used to generate the features and labels. Within Merlin, an example script is provided for this purpose, namely 'build your own voice'.

This script contains the same steps as the script discussed above.

```
./01_setup.sh voice_name
./04_prepare_conf_files.sh conf/global_settings.cfg
./05_train_duration_model.sh conf/duration_voice_name.conf
./06_train_acoustic_model.sh conf/acoustic_voice_name.conf
./07_run_merlin.sh                                  conf/test_dur_synth_voice_name.conf
conf/test_synth_voice_name.conf
```

However, there are some steps missing that have not yet been explained. Namely, two steps are missing: the step to generate the linguistic features and the step to generate the acoustic features.

```
./02_prepare_labels.sh database/wav database/txt.data database/labels
```

This step will generate the linguistic features. Depending on the two inputs, the waveforms and the text file, the labels will be determined. State align labels or phone align labels will be generated depending on the global configuration.

```
./03_prepare_acoustic_features.sh database/wav database/feats
```

In this step, the acoustic features will be generated. What kind of features these depend on the vocoder set in the global configuration file.

Changing the neural network on which the model is trained can be changed in the same way as the 'slt arctic' script. In other words, the LSTM layout can be applied to the ksp dataset.

5.3.3 Vocoders

Now that it is possible to use different neural network configurations and other datasets, changing the vocoder will be the next step. Since the world vocoder is already implemented in the

Merlin toolkit, also the continuous vocoder is already implemented in a modified Merlin toolkit [79].

5.3.3.1 Ahocoder

Before the ahocoder is implemented in the Merlin toolkit, the coder and decoder for the ahocoder will be looked at individually.

```
ahocoder16 filein filef0 filecc [filefv] [params]
```

The coder ensures that the waveform is converted to the following features: fundamental frequency, mel-cepstral coefficients and maximum voiced frequency. In addition, it is possible to change parameters that affect the generated features. The parameters are shown below with their standard values.

```
--LFRAME=80  Frame shift (samples)
--CCORD=39   Order of the cepstral representation
--CCMETH=0   Method for cepstral coeff. extraction in voiced frames:
             0 = harmonic analysis + interpolation + mcep analysis
             1 = harmonic analysis + mel regularized discrete cepstrum
             2 = efficient (slightly less accurate) method
--F0MIN=60   Lowest detectable pitch (Hz)
--F0MAX=500  Highest detectable pitch (Hz)
--F0LOAD=0   Using external pitch detector?
             0 = use the default pitch analysis method (based on
                 autocorrelation plus dynamic programming)
             1 = load pitch file if it already exists. Otherwise, use
                 the default pitch analysis method.
```

In order to use the vocoder in the Merlin toolkit, it is important that the size of the features the acohoder outputs match with the features that are used in Merlin. Therefore, the coder parameters for determining the highest detectable pitch (F0MAX) and the order of cepstral representation (CCORD) were changed from their default values. Thus, the F0MAX was changed from a default value of 500 to 350, and the CCORD from a default value of 39 to 59.

```
ahodecoder16 filef0 filecc [filefv] fileout [params]
```

The decoder converts the fundamental frequency, the cepstral coefficients, and the maximum voiced frequency optionally to a waveform. There are also optional parameters for the decoder.

First, a tool or script will need to be written to ensure that all acoustic features are generated. This script is written in Python, which is the standard for the Merlin toolkit.

```
def get_features(wav_path, basefilename):
    in_wav = wav_path + basefilename + '.wav'
    in_lf0i = lf0_path + basefilename + '.lf0'
    in_mgci = mgc_path + basefilename + '.mgc'
    in_mvfi = mvf_path + basefilename + '.mvf'

    # Get Features
    os.system('./ahocoder_64/ahocoder16_64 ' + in_wav + ' ' + in_lf0i + ' ' + in_mgci
+ ' ' + in_mvfi + ' --CCORD=59 --CCMETH=1 --F0MIN=60 --F0MAX=350')

    return 0
```

The code above is part of the code used to generate the acoustic features. For each waveform in a directory, the get_features function is executed. Since the source code of the ahocoder is not available, it is only possible to call the ahocoder script through the system and thus execute it.

The same principle as above is used in the speech synthesis of the ahocoder. Before the speech synthesis is performed, the three features are determined based on the linguistic features and the trained neural network. The same principle as above is used in the speech synthesis of the ahocoder. Before the speech synthesis is performed, the three features are determined based on the linguistic features and the trained neural network. Since these features are all in the same directory, it is essential to make sure that the three features that belong together are found and then perform speech synthesis on these features.

```
for lf0_file in os.listdir(gen_path):
    if '.lf0' in lf0_file:
        basefilename = lf0_file[:-4]
        wav_file = basefilename + '.wav'
        for mgc_file in os.listdir(gen_path):
            if '.mgc' in mgc_file and basefilename == mgc_file[:-4]:
                for mvf_file in os.listdir(gen_path):
                    if '.mvf' in mvf_file and basefilename == mvf_file[:-4]:
                        print(lf0_file, mgc_file, mvf_file)

                        os.system('./ahocoder_64/ahodecoder16_64  '  +  gen_path  +
lf0_file + ' ' + gen_path + mgc_file + ' ' + gen_path + mvf_file + ' ' + gen_path +
wav_file)
```

Apart from the analysis scripts and synthesis scripts, adjusting Merlin's configuration file is crucial. The world vocoder uses bap, lf0 and mgc features. In contrast, the ahocoder uses lf0, mgc and mvf features, and this mvf feature is not implemented by default in the configuration.

```
def wavgen_ahocoder(gen_dir):
    import os
    cur_dir = os.getcwd()
    os.chdir("../../../misc/scripts/vocoder/ahocoder/")
    command = 'python3 aho_speech_synthesis.py ' + gen_dir + '/'
    os.system(command)
    os.chdir(cur_dir)

    return 0
```

In addition, a section of code had to be added to the generate.py file in the utils directory. Here, the directory where the acoustic features are generated is passed along to the speech synthesis of the ahocoder, where the features are converted to a waveform as described earlier.

6 Model evaluation methods

In order to evaluate the TTS models, five metrics [80] are used. The smaller the MCD, BAP, RMSE and VUV, the better, and the higher the correlation, the better.

In addition, the spectrogram of the generated waveforms will also be examined.

6.1 Objective model evaluation methods

6.1.1 MCD

Mel cepstral distortion (MCD) measures how different two MCDs are [81]. The MCD between synthesized and natural mel cepstral sequences is used to assess the quality of parametric speech synthesis systems, including statistical parametric speech synthesis systems. The idea is that the smaller the MCD between synthesized and natural mel cepstral sequences, the closer synthetic speech is to reproducing natural speech [82]. It is not a perfect statistic for evaluating the quality of synthetic speech, but it is frequently used in conjunction with other indicators.

6.1.2 BAP

BAP stands for band aperiodicity of speech signals, where “aperiodicity” is defined as the power ratio between the speech signal and the aperiodic component of the signal [83]. Since this power ratio depends on the frequency band, the aperiodicity should be given for several frequency bands.

6.1.3 RMSE

The root mean square error (RMSE) is a common statistical tool for evaluating model performance. The RMSE measures the distance between the predicted and the expected output of a model and is the square root of the mean of the square of all of the error.

6.1.4 Correlation

A correlation coefficient indicates the strength and direction of a relationship between variables. It is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations.

6.1.5 VUV

The voiced/unvoiced (VUV) analysis metric estimates a cut-off frequency for the voiced and unvoiced part of a signal, in analogy with the production model of vocal sounds [84]. The estimation is based on going through each natural and synthesis waveform frame. When the natural waveform voiced and the synthesis waveform is unvoiced, it will count as an error. So this means that the smaller the VUV is, the more accurate the obtained result is.

6.1.6 Spectrogram

The sound spectrograph is a wave analyzer which produces a permanent visual record showing the distribution of energy in both frequency and time [85].

In order to plot spectrograms, Parselmouth is used [86]. Parselmouth is a Python library for the Praat software [87].

6.2 Subjective model evaluation methods

6.2.1 Speakers testing

A web-based MUSHRA (MUlti-Stimulus test with Hidden Reference and Anchor) listening test is used to determine which vocoder is closest to natural speech [88]. The listeners were asked to score the naturalness of each stimulus in comparison to the reference, which was a natural voice, on a scale of 0 to 100, where 0 is highly unnatural, and 100 is highly natural. For each participant, the utterances were given in a different order.

The online listening test was performed by a total of 7 participants, all of whom were men, between the ages of 21 and 25 (mean age: 22 years). They were primarily from an engineering background, and only one of them spoke English natively, and none of them had any hearing problems. The exam took an average of 11 minutes to complete. The listening test samples can be found online [89].

7 Evaluation

This section will describe the evaluation of the different models made. A distinction will be made between the objective and the subjective evaluation.

7.1.1 Effect of training data size

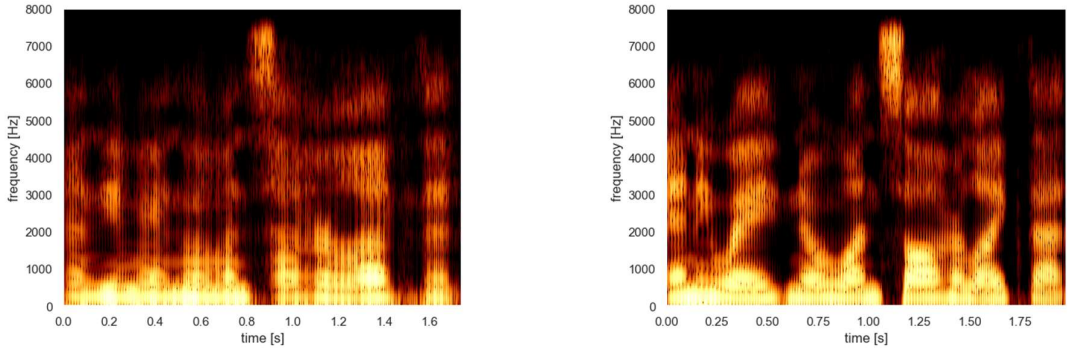
We first examine the effect of the amount of training data used in the model. Two DNN-based TTS systems with vocoder are trained, where one uses the demo dataset, and the other uses the full dataset. The demo dataset contains about 60 waveforms that can be used to train where the divide between train, validation and test set is 50, 10 and 10. In comparison, the full dataset contains 1132 waveforms divided into 1000, 66, and 66.

Table 7.1 shows that a significant improvement in synthesized speech quality is achieved by using a larger dataset.

Table 7.1: Comparison of MCD, BAP, RMSE, correlation and VUV of the WORLD vocoder with slt dataset between the demo and full dataset

Dataset	MCD (dB)	BAP (dB)	RMSE (Hz)	CORR	VUV (%)
Demo	6,586	0,259	**15,309**	**0,701**	8,821
Full	**5,234**	**0,167**	20,353	0,603	**6,575**

Figure 7.1 also shows this improvement in speech quality. For example, it can be seen that the audio with the demo dataset is more faded than with the full dataset. This loss of quality can also be seen in Table 7.1 since the VUV is higher in the demo dataset than in the full dataset.



(a) Demo dataset

(b) Full dataset

Figure 7.1: Spectrograms of the slt voice and world vocoder with the sentence Anyway, no one saw her like that' with different datasets

7.1.2 Effect of hidden layers in neural network

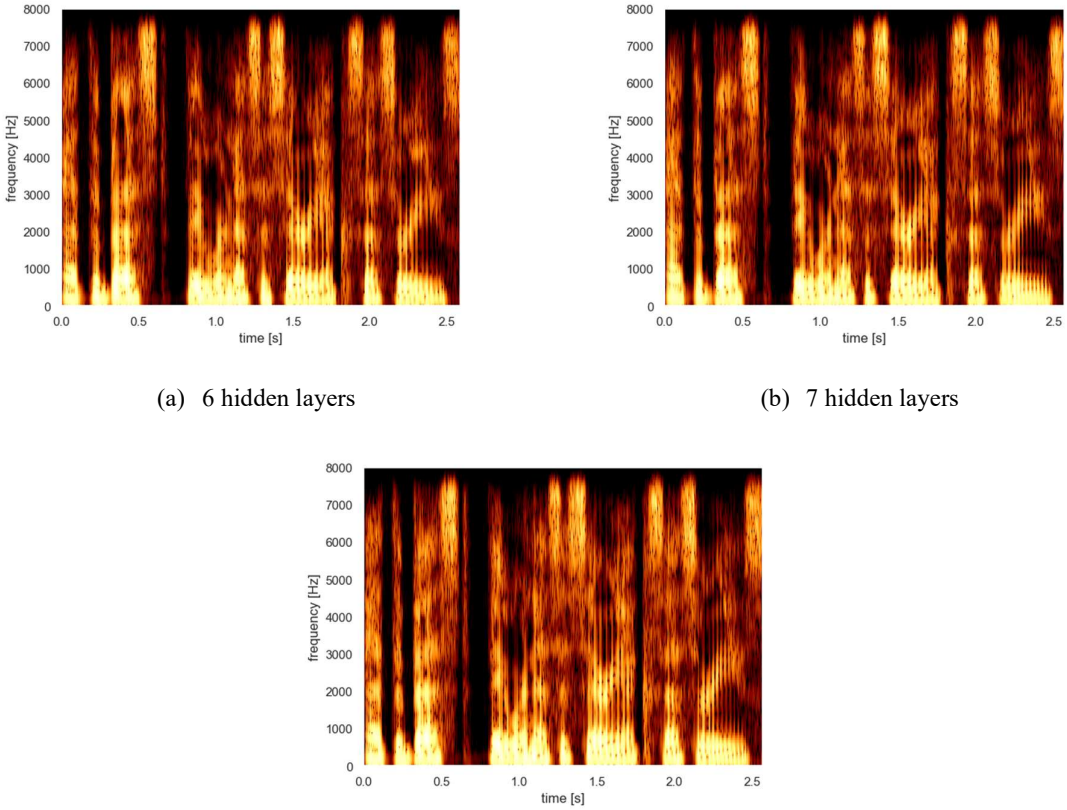
For the duration and acoustic models in previous experiments, 6-layer DNN was used. The next experiment will be conducted to research the effect of DNN architecture on the quality of the TTS system. It should be noted that the full dataset will be used from now on.

Table 7.2 shows that the results of the TTS systems with different DNN architectures differ negligibly, meaning that using more than six layers is not worth the extra computational time. This result corresponds to the result obtained in [68] , where it has been observed that the quality does not much improve after more than four hidden layers.

Table 7.2: Comparison of MCD, RMSE, correlation and VUV of the continuous vocoder with slt dataset between the amount of hidden layers in the neural network

Amount of hidden layers	MCD (dB)	RMSE (Hz)	CORR	VUV (%)
6	4,912	12,539	0,747	24,109
7	4,881	**12,481**	**0,750**	24,109
9	**4,834**	12,606	0,745	24,109

As seen in Figure 7.2, the spectrograms confirm this result since the spectrograms are practically identical.



(a) 6 hidden layers

(b) 7 hidden layers

(c) 9 hidden layers

Figure 7.2: Spectrograms of the slt voice and continuous vocoder with the sentence 'At the best, they were necessary accessories' with different amounts of hidden layers in the neural network

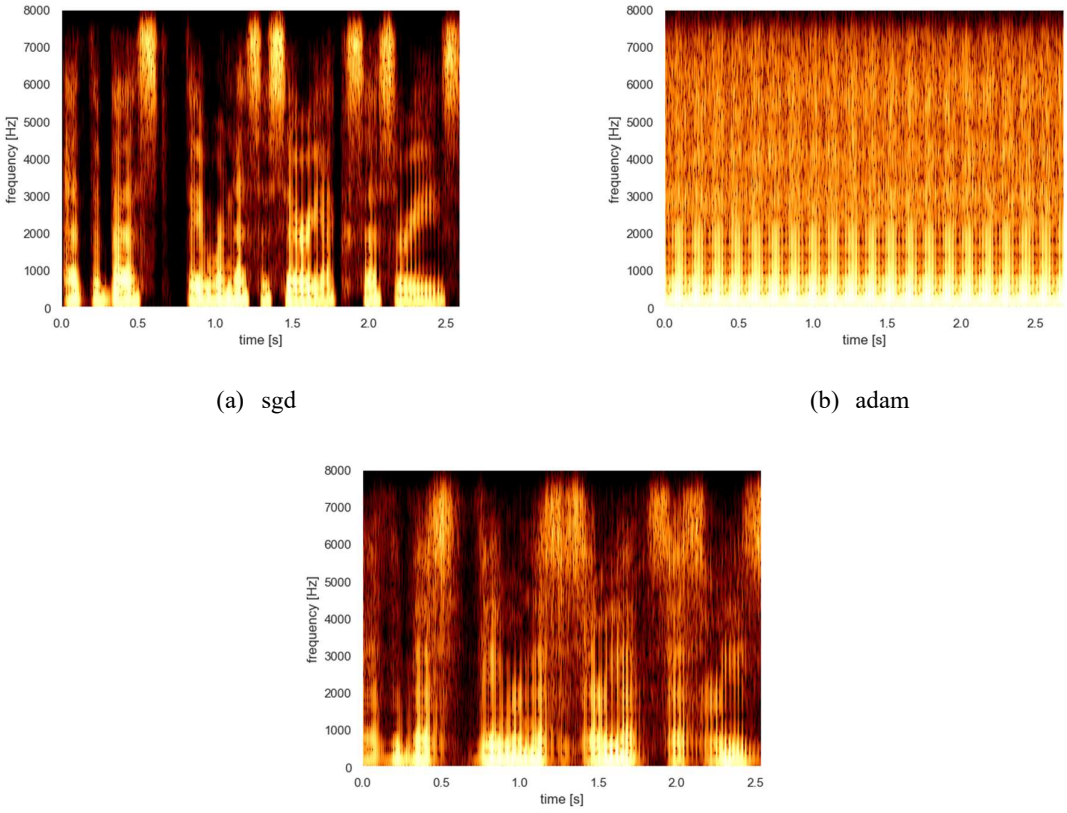
7.1.3 Effect of different optimization algorithms

The effect of using different optimization algorithms is discussed in this section. The optimization algorithms offered by the Merlin toolkit are the sgd, adam and rprop algorithms. In Table 7.3, it can be concluded that the sgd algorithm provides the best speech quality on the dataset.

Table 7.3: Comparison of MCD, RMSE, correlation and VUV of the ahocoder with slt dataset between different optimizers

Optimizer	MCD (dB)	RMSE (Hz)	CORR	VUV (%)
sgd	**6,063**	**14,878**	**0,692**	25,957
adam	11,512	20,641	0,174	25,957
rprop	8,052	16,473	0,601	25,957

In addition, it is unmistakable that the adam algorithm works inadequately as seen in Table 7.3 and Figure 7.3. The synthesized speech is completely washed out in the spectrogram, and the natural speech is impossible to recognize.



(a) sgd

(b) adam

(c) rprop

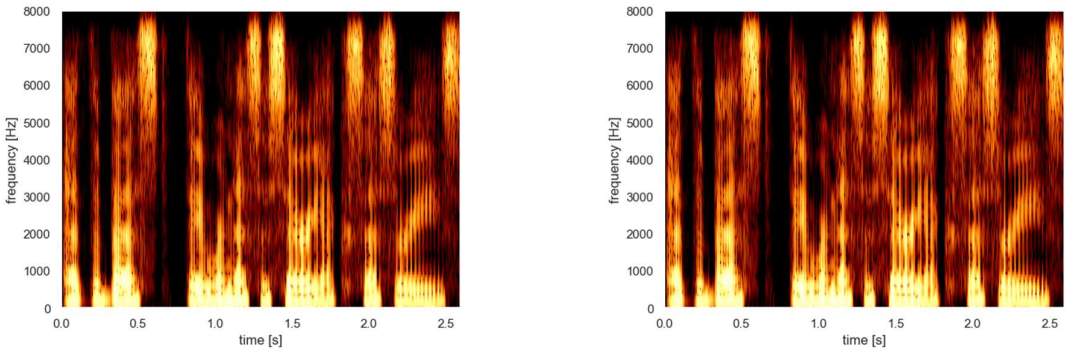
Figure 7.3: Spectrograms of the slt voice and ahocoder with the sentence 'At the best, they were necessary accessories' with different optimizers

7.1.4 Effect of different extractions of the cepstral coefficients on the ahocoder

Since the ahocoder offers several options for extracting the cepstral coefficients, this section explores the effect of these different types of extraction. The difference between performing harmonic analysis, then interpolation, then mcep analysis and performing harmonic analysis, then mel regularized discrete cepstrum was examined. From both Table 7.4 and Figure 7.4, it can be concluded that there is little to no difference between the two methods.

Table 7.4: Comparison of MCD, RMSE, correlation and VUV of the ahocoder with slt dataset between different methods to extract the cepstral coefficients where method 1 is harmonic analysis then interpolation then mcep analysis and method 2 is harmonic analysis then mel regularized discrete cepstrum

Method	MCD (dB)	RMSE (Hz)	CORR	VUV (%)
1	**6,063**	**14,878**	**0,692**	25,957
2	6,104	14,916	0,692	25,957



(c) Harmonic analysis then interpolation then mcep analysis

(d) Harmonic analysis then mel regularized discrete cepstrum

Figure 7.4: Spectrograms of the slt voice and ahocoder with the sentence 'At the best, they were necessary accessories' with different extractions of the cepstral coefficients

7.1.5 Effect of different vocoders

The current model obtained from previous experiments will be discussed before discussing the effect of the three different vocoders. The current model includes using the full dataset, a neural network with six hidden layers, and the sgd optimization algorithm. In addition, beyond changing the vocoders, the focus will also be on investigating the type of neural network; for example, a DNN and LSTM will be used. The effect of changing the vocoder and the type of neural network will be examined on both the slt and ksp datasets.

Table 7.5 and Table 7.6 shows that based on the MCD, LSTM performs poorly on both datasets. The model used for the ahocoder underperforms compared to the world and continuous vocoder. Nevertheless, it can be shown in Figure 7.7 and Figure 7.11 that the ahocoder will still be recognizable. When Table 7.5 and Table 7.6 are compared it is noticable that all vocoders perform better on the slt dataset. The most significant difference is the continuous vocoder, which performs about the same on the ksp dataset as the world vocoder. However, the continuous vocoder performs better on the slt dataset than the world vocodor.

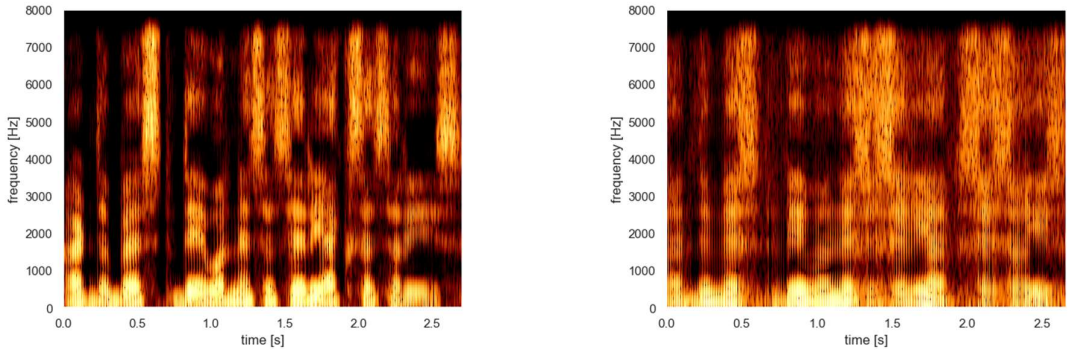
Table 7.5: Comparison of MCD, RMSE, correlation and VUV between the different vocoders and models of the ksp dataset where the shows bold font shows the best performance

Vocoder	WORLD		Continuous		Ahocoder	
Model	DNN	LSTM	DNN	LSTM	DNN	LSTM
MCD (dB)	**5,234**	7,28	5,271	7,352	6,199	8,303
RMSE (Hz)	20,353	23,831	19,834	23,628	**14,878**	19,425
CORR	0,603	0,48	0,635	0,459	**0,686**	0,503
VUV (%)	**6,575**	15,044	31,861	31,861	30,035	30,035

Table 7.6: Comparison of MCD, RMSE, correlation and VUV between the different vocoders and models of the slt dataset where the shows bold font shows the best performance

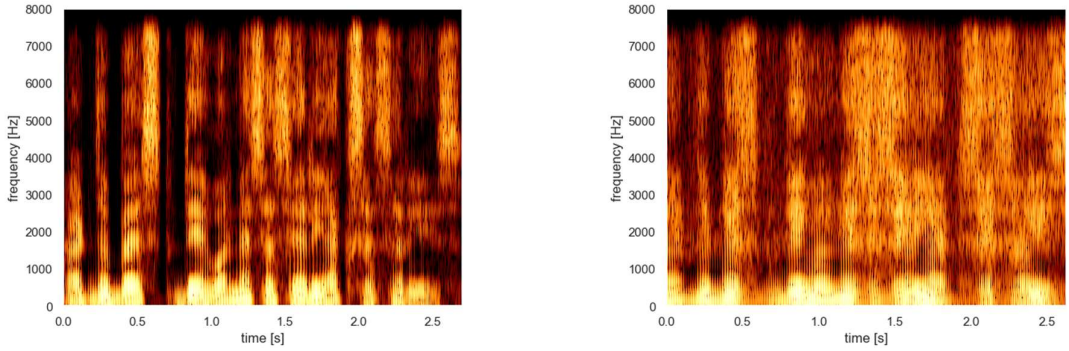
Vocoder	WORLD		Continuous		Ahocoder	
Model	DNN	LSTM	DNN	LSTM	DNN	LSTM
MCD (dB)	4,923	7,116	**4,192**	7,144	6,063	8,451
RMSE (Hz)	17,668	15,361	**12,539**	15,768	14,878	17,923
CORR	0,648	0,637	**0,747**	0,627	0,692	0,559
VUV (%)	**4,345**	11,625	24,109	24,109	25,957	25,957

In the spectrograms, it can be concluded that the synthesis speech is more faded than the natural speech, where the WORLD vocoder is the most closely resembling the original, then the continuous vocoder and then the ahocoder.



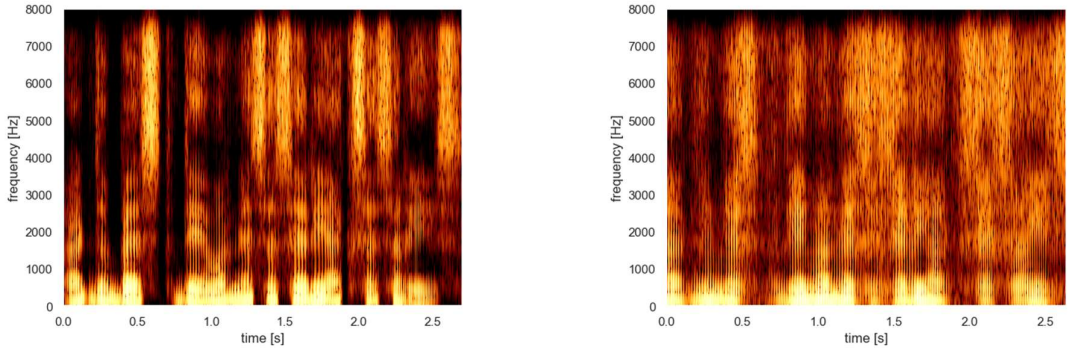
(a) DNN (b) LSTM

Figure 7.5: Spectrogram of the ksp voice with the sentence 'At the best, they were necessary accessories' with WORLD vocoder



(a) DNN (b) LSTM

Figure 7.6: Spectrogram of the ksp voice with the sentence 'At the best, they were necessary accessories' with continuous vocoder



(a) DNN (b) LSTM

Figure 7.7: Spectrogram of the ksp voice with the sentence 'At the best, they were necessary accessories' with ahocoder

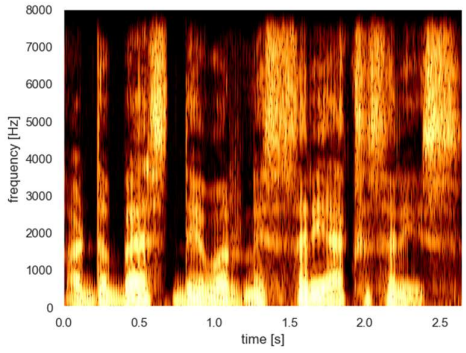
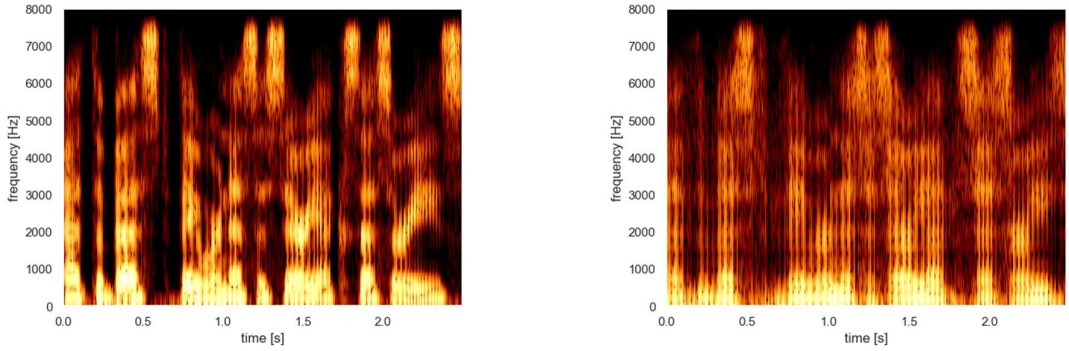
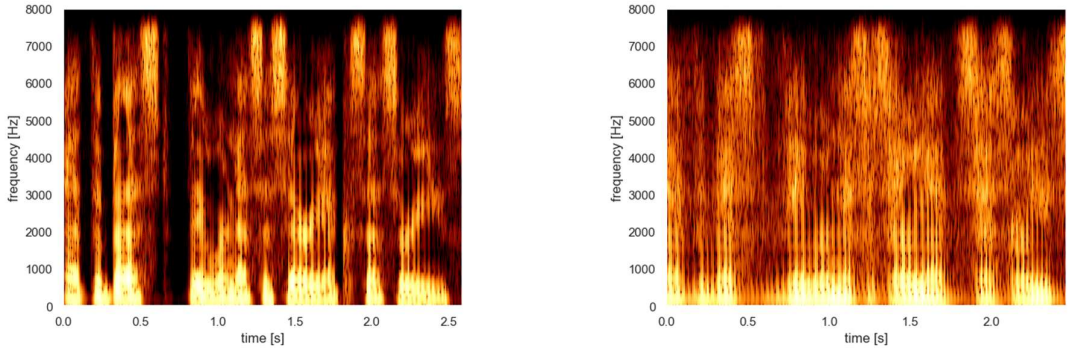


Figure 7.8: Spectrogram of the ksp voice with the sentence 'At the best, they were necessary accessories' of the natural voice



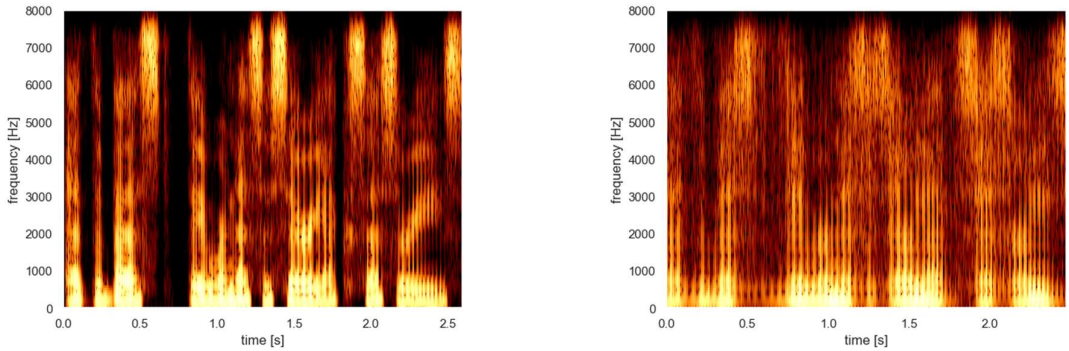
(a) DNN (b) LSTM

Figure 7.9: Spectrogram of the slt voice with the sentence 'At the best, they were necessary accessories' with WORLD vocoder



(a) DNN (b) LSTM

Figure 7.10: Spectrogram of the slt voice with the sentence 'At the best, they were necessary accessories' with continuous vocoder



(a) DNN (b) LSTM

Figure 7.11: Spectrogram of the slt voice with the sentence 'At the best, they were necessary accessories' with ahocoder

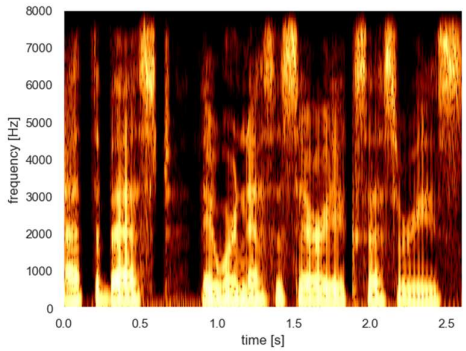
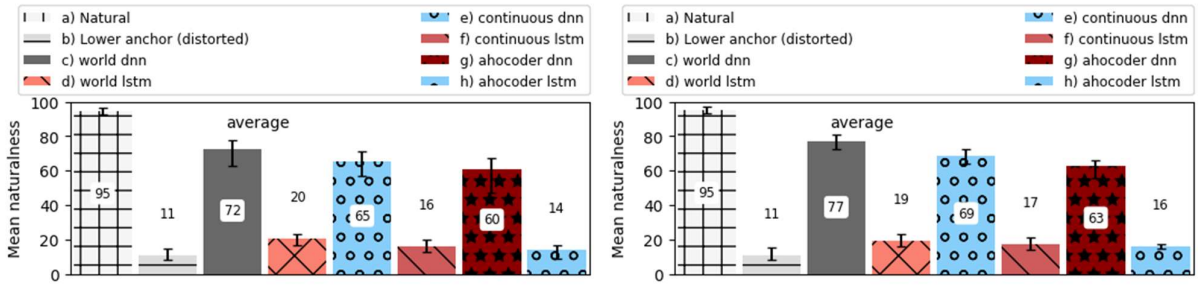


Figure 7.12: Spectrogram of the slt voice with the sentence 'At the best, they were necessary accessories' of the natural voice

7.2 Subjective evaluation

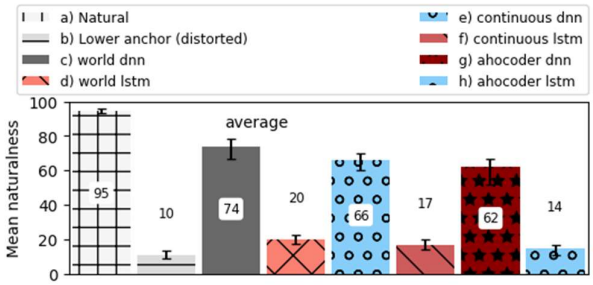
Figure 7.13 distinguishes the evaluation of the male and female speakers and the combination of both speakers. According to the listeners, the WORLD vocoder is the most natural of the models made. However, the scores of the continuous vocoder and ahocoder are not far from the performance of the WORLD vocoder. However, it is clear that when the models use an LSTM neural network, the result is unnatural, meaning that by using LSTM, the quality is significantly reduced. It can also be seen that the vocoders perform better on a female dataset.

The ahocoder, has an average mean naturalness of 62, meaning the use of the ahocoder can generate an intelligible sound.



(a) Male speaker

(b) Female speaker



(c) Overall

Figure 7.13: Results of the subjective evaluation for the naturalness question. Higher value means larger naturalness. Error bars show the boot-strapped 95% confidence intervals.

7.3 Summary

According to the objective evaluations, the WORLD and continuous vocoder perform roughly evenly when comparing their MCDs. The ahocoder has an MCD of 6.063, which suggests it is less accurate but still understandable. With an MCD of 4.192, the continuous vocoder performs best on the slt dataset. According to the subjective ratings, the WORLD vocoder produces the best results, with a subjective rating of 74. In comparison, the implemented ahocoder has a rating of 62, indicating that listeners find the synthesized speech to be natural enough.

8 Conclusion

The paper discusses the research of the deep learning approach based on neural networks and investigates an efficient architecture to improve the TTS synthesis. The main idea was to look at new and old ways to generate a more efficient model. For example, the study looked at the number of layers in a neural network and the type of neural network. In addition, three vocoders were examined, the WORLD vocoder, the continuous vocoder and the ahocoder. Objective and subjective evaluations allowed the different models to be compared.

The best-created model includes using the full dataset, a neural network with six hidden layers, and the sgd optimization algorithm. It can be concluded from the objective evaluations that the WORLD and continuous vocoder perform almost equally when comparing their MCDs. The MCD of the ahocoder is slightly outside this range, but not far enough to say that it is a inaccurate model. From the subjective evaluations, it can be concluded that the WORLD vocoder gives the best results, with a subjective rating of 74. The implemented ahocoder has a rating of 62, meaning it is natural enough synthesized speech according to the listeners. From this, it can be concluded that the ahocoder is correctly integrated into the Merlin toolkit.

8.1 Future work

It is necessary for future work to further investigate the neural network architecture for the ahocoder. For example, it would perhaps be possible to use other types of neural networks provided by the Merlin toolkit, such as BLSTM or GRU, to improve speech quality.

In addition, the following paper [46] shows that the use of LSTM and adam optimization algorithm provided a good result, unlike the results obtained in this paper. As a result, it would be good to re-examine LSTM and adam.

9 Acknowledgement

Without the help of numerous people, this thesis would not have been possible. Many thanks to Massimo Morello, who read through all of my modifications and helped me make sense of it all. I want to thank my mom, dad and brother for motivating me to keep going and making sure I would bring the thesis to a good end.

I would like to express my gratitude to my supervisor, Dr. Mohammed Salah Al-Radhi, for helping me with this project and bringing the thesis to a successful end. Without him, this would not have been possible. I learned a lot while working on the thesis thanks to him. In addition, I would also like to thank Dr Tamás Gábor Csapó for giving me the information I needed when I asked for it. Thanks to the Budapest University of Technology and Economics for offering me the possibility to have an Erasmus.

Finally, I want to express my gratitude to numerous friends who have supported me throughout this long process.

References

[1] "What Is Machine Learning and Why Is It Important?" https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML (accessed May 22, 2022).

[2] M. Chiesa, G. Maioli, G. I. Colombo, and L. Piacentini, "GARS: Genetic Algorithm for the identification of a Robust Subset of features in high-dimensional datasets," *BMC Bioinformatics*, vol. 21, no. 1, Feb. 2020, doi: 10.1186/s12859-020-3400-6.

[3] "Wat is machine learning?" https://www.oracle.com/nl/data-science/machine-learning/what-is-machine-learning/ (accessed May 22, 2022).

[4] "Supervised vs. Unsupervised Machine Learning." https://chisoftware.medium.com/supervised-vs-unsupervised-machine-learning-7f26118d5ee6 (accessed May 22, 2022).

[5] "Types of Cost Function Machine Learning." https://www.analyticsvidhya.com/blog/2021/02/cost-function-is-no-rocket-science/ (accessed May 28, 2022).

[6] "Gradient Descent Algorithm." https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21 (accessed May 28, 2022).

[7] "Gradient Descent." https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html (accessed May 28, 2022).

[8] "What are gradient descent and stochastic gradient descent?" https://sebastianraschka.com/faq/docs/gradient-optimization.html (accessed May 28, 2022).

[9] N. Ketkar, "Stochastic Gradient Descent," *Deep Learning with Python*, pp. 113–132, 2017, doi: 10.1007/978-1-4842-2766-4_8.

[10] "Stochastic Gradient Descent (SGD)." https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/ (accessed May 28, 2022).

[11] K. A. Carpenter, D. S. Cohen, J. T. Jarrell, and X. Huang, "Deep learning and virtual drug screening," *Future Medicinal Chemistry*, vol. 10, no. 21, pp. 2557–2567, Nov. 2018, doi: 10.4155/FMC-2018-0314.

[12] D. P. Kingma and J. Lei Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”.

[13] “Code Adam Optimization Algorithm From Scratch.” https://machinelearningmastery.com/adam-optimization-from-scratch/ (accessed May 28, 2022).

[14] X. Chen, B. Karimi, W. Zhao, and P. Li, “On the Convergence of Decentralized Adaptive Gradient Methods”.

[15] “What is Adaptive Moment Estimation (ADAM).” https://blog.marketmuse.com/glossary/adaptive-moment-estimation-adam-definition/ (accessed May 28, 2022).

[16] M. Riedmiller and H. Braun, “RPROP - A Fast Adaptive Learning Algorithm,” 1992.

[17] M. Riedmiller and H. Braun, “Direct adaptive method for faster backpropagation learning: The RPROP algorithm,” in *1993 IEEE International Conference on Neural Networks*, 1993, pp. 586–591. doi: 10.1109/icnn.1993.298623.

[18] “RProp.” https://florian.github.io/rprop/#citation-1 (accessed May 28, 2022).

[19] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[20] “What is the Difference Between Test and Validation Datasets?” https://machinelearningmastery.com/difference-test-validation-datasets/ (accessed May 28, 2022).

[21] G. James, D. Witten, T. Hastie, and R. Tibshirani, “An Introduction to Statistical Learning with Applications in R Second Edition,” 2021.

[22] M. Claesen and B. de Moor, “Hyperparameter Search in Machine Learning,” 2015, Accessed: May 28, 2022. [Online]. Available: https://www.codalab.org/competitions/2321.

[23] R. Kohavi and D. H. Wolpert, “Bias Plus Variance Decomposition for Zero-One Loss Functions,” 1996.

[24] “What Is the Difference Between Bias and Variance?” https://www.mastersindatascience.org/learning/difference-between-bias-and-variance/ (accessed May 28, 2022).

[25] L. Freda, “Bias and variance tradeoff.” http://www.luigifreda.com/2017/03/22/bias-variance-tradeoff/ (accessed May 28, 2022).

[26] “Wat is deep learning? De betekenis uitgelegd.” https://www.sdim.nl/helpcentrum/begrippenlijst/deep-learning/ (accessed May 22, 2022).

[27] A. Rysbekova, T. Gábor, and C. Budapest, “Comparison Of UTI-EMA-MRI-LIP video and its application for Deep Learning-based Articulation-To-Speech Synthesis.”

[28] N. Kriegeskorte and T. Golan, “Neural network models and deep learning,” *Current Biology*, vol. 29, no. 7, pp. R231–R236, Apr. 2019, doi: 10.1016/J.CUB.2019.02.034.

[29] J. Adcock *et al.*, “Advances in quantum machine learning,” Dec. 2015, Accessed: May 23, 2022. [Online]. Available: http://arxiv.org/abs/1512.02900

[30] “Introduction to Activation Function for Deep Learning.” https://www.analyticsvidhya.com/blog/2021/04/neural-networks-and-activation-function/ (accessed May 23, 2022).

[31] S. Sharma, S. Sharma, and A. Athaiya, “ACTIVATION FUNCTIONS IN NEURAL NETWORKS,” 2020. [Online]. Available: http://www.ijeast.com

[32] P. Sibi, S. A. Jones, and P. Siddarth, “ANALYSIS OF DIFFERENT ACTIVATION FUNCTIONS USING BACK PROPAGATION NEURAL NETWORKS,” *J Theor Appl Inf Technol*, vol. 31, no. 3, 2013, [Online]. Available: www.jatit.org

[33] “Deep Learning: Feedforward Neural Network.” https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7 (accessed May 23, 2022).

[34] A. Zeil, R. O. Verlag, and M. Wien, “Simulation neuronaler Netze.”

[35] J. P. Davim, *Machining of hard materials*. Springer London, 2011. doi: 10.1007/978-1-84996-450-0.

[36] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/J.NEUNET.2014.09.003.

[37] “What are Neural Networks? | IBM.” https://www.ibm.com/cloud/learn/neural-networks (accessed May 23, 2022).

[38] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature 2015 521:7553*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[39] “Recurrent Neural Network (RNN) Tutorial.” https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn#what_is_a_recurrent_neural_network_rnn (accessed May 23, 2022).

[40] M. V. MISHRA, SMT. M. AGARWAL, and N. PURI, “COMPREHENSIVE AND COMPARATIVE ANALYSIS OF NEURAL NETWORK,” *INTERNATIONAL JOURNAL OF COMPUTER APPLICATION*, vol. 2, no. 8, 2018, doi: 10.26808/RS.CA.I8V2.15.

[41] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[42] J. Hochreiter, “DIPLOMARBEIT IM FACH INFORMATIK Untersuchungen zu dynamischen neuronalen Netzen,” 1991.

[43] “Understanding LSTM Networks.” https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed May 23, 2022).

[44] “This is now the official location of the Merlin project.” https://github.com/CSTR-Edinburgh/merlin (accessed May 27, 2022).

[45] Z. Wu, O. Watts, and S. King, “Merlin: An Open Source Neural Network Speech Synthesis System,” Sep. 2016, pp. 202–207. doi: 10.21437/ssw.2016-33.

[46] J. Hong and C. Kwon, “Performance comparison of various deep neural network architectures using Merlin toolkit for a Korean TTS system,” *Phonetics and Speech Sciences*, vol. 11, no. 2, pp. 57–64, Jun. 2019, doi: 10.13064/KSSS.2019.11.2.057.

[47] “Merlin.” https://www.cstr.ed.ac.uk/projects/merlin/ (accessed May 23, 2022).

[48] M. Morise, “Implementation of sequential real-time waveform generator for high-quality vocoder; Implementation of sequential real-time waveform generator for high-quality vocoder,” 2020. [Online]. Available: https://github.com/mmorise/World

[49] M. Morise, “D4C, a band-aperiodicity estimator for high-quality speech synthesis,” *Speech Communication*, vol. 84, pp. 57–65, Nov. 2016, doi: 10.1016/J.SPECOM.2016.09.001.

[50] M. Morise, F. Yokomori, and K. Ozawa, “WORLD: A vocoder-based high-quality speech synthesis system for real-time applications,” *IEICE Transactions on Information and Systems*, vol. E99D, no. 7, pp. 1877–1884, Jul. 2016, doi: 10.1587/TRANSINF.2015EDP7457.

[51] M. Morise, H. Kawahara, and H. Katayose, “Fast and Reliable F0 Estimation Method Based on the Period Extraction of Vocal Fold Vibration of Singing Voice and Speech,” Feb. 2009. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=15165

[52] M. Morise, H. Kawahara, and T. Nishiura, “Rapid F0 estimation for high-SNR speech based on fundamental component extraction,” *Trans. IEICEJ*, vol. 93, pp. 109–117, 2010.

[53] M. Morise, “CheapTrick, a spectral envelope estimator for high-quality speech synthesis,” *Speech Communication*, vol. 67, pp. 1–7, Mar. 2015, doi: 10.1016/J.SPECOM.2014.09.003.

[54] M. Morise, “Error Evaluation of an F0-Adaptive Spectral Envelope Estimator in Robustness against the Additive Noise and F0 Error,” *IEICE Transactions on Information and Systems*, vol. E98.D, no. 7, pp. 1405–1408, Jul. 2015, doi: 10.1587/TRANSINF.2015EDL8015.

[55] M. Morise, “PLATINUM: A method to extract excitation signals for voice synthesis system,” *Acoustical Science and Technology*, vol. 33, no. 2, pp. 123–125, Mar. 2012, doi: 10.1250/AST.33.123.

[56] M. S. Al-Radhi, T. G. Csapó, and G. Németh, “Continuous noise masking based vocoder for statistical parametric speech synthesis,” *IEICE Transactions on Information and Systems*, vol. E103D, no. 5, pp. 1099–1107, May 2020, doi: 10.1587/transinf.2019EDP7167.

[57] P. N. Garner, M. Cernak, and P. Motlicek, “A simple continuous pitch estimation algorithm,” *IEEE Signal Processing Letters*, vol. 20, no. 1, pp. 102–105, Jan. 2013, doi: 10.1109/LSP.2012.2231675.

[58] T. Drugman and Y. Stylianou, “Maximum voiced frequency estimation: Exploiting amplitude and phase spectra,” *IEEE Signal Processing Letters*, vol. 21, no. 10, pp. 1230–1234, 2014, doi: 10.1109/LSP.2014.2332186.

[59] M. S. Al-Radhi, T. G. Csapó, and G. Németh, “Deep Recurrent Neural Networks in Speech Synthesis Using a Continuous Vocoder,” 2017, doi: 10.1007/978-3-319-66429-3_27.

[60] M. S. Al-Radhi and G. Németh, “Continuous vocoder in feed-forward deep neural network based speech synthesis Signal processing View project Medical imaging for acoustic-to-articulatory inversion View project,” 2017. [Online]. Available: https://www.researchgate.net/publication/321527303

[61] M. S. Al-Radhi, T. G. Csapó, and G. Németh, “Time-Domain Envelope Modulating the Noise Component of Excitation in a Continuous Residual-Based Vocoder for Statistical Parametric Speech Synthesis,” in *Proc. Interspeech 2017*, 2017, pp. 434–438. doi: 10.21437/Interspeech.2017-678.

[62] G. Degottex and D. Erro, “A uniform phase representation for the harmonic model in speech synthesis applications,” *Tijdschrift voor Urologie*, vol. 2014, no. 1, pp. 1–16, Jan. 2014, doi: 10.1186/S13636-014-0038-1/FIGURES/8.

[63] G. Degottex, P. Lanchantin, and M. Gales, “A Log Domain Pulse Model for Parametric Speech Synthesis,” *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 26, no. 1, pp. 57–70, Jan. 2018, doi: 10.1109/TASLP.2017.2761546.

[64] W. Yang and R. Yantorno, “Improvement of MBSD by scaling noise masking threshold and correlation analysis with MOS difference instead of MOS,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2, pp. 673–676, 1999, doi: 10.1109/ICASSP.1999.759756.

[65] D. Erro, I. Sainz, E. Navas, and I. Hernaez, “Harmonics plus noise model based vocoder for statistical parametric speech synthesis,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 2, pp. 184–194, 2014.

[66] Y. Stylianou, “Harmonic plus noise models for speech, combined with statistical methods, for speech and speaker modification,” *Ph. D thesis, Ecole Nationale Superieure des Telecommunications*, 1996.

[67] I. Zangar, Z. Mnasri, V. Colotte, D. Jouvet, and A. Houidhek, “Duration modeling using DNN for Arabic speech synthesis Duration modeling using DNN for Arabic speech synthesis Duration modeling using DNN for Arabic speech synthesis,” 2018, Accessed: May 25, 2022. [Online]. Available: https://hal.inria.fr/hal-01889917

[68] T. van Nguyen, B. Q. Nguyen, K. H. Phan, and H. van Do, “Development of Vietnamese Speech Synthesis System using Deep Neural Networks,” *Journal of Computer Science and Cybernetics*, vol. 34, no. 4, pp. 349--363, Jan. 2019, doi: 10.15625/1813-9663/34/4/13172.

[69] Tijana Delić, Milan Sečujski, and Siniša Suzić, “A Review of Serbian Parametric Speech Synthesis Based on Deep Neural Networks,” *Telfor Journal*, vol. 9, 2017.

[70] S. Ronanki, O. Watts, and S. King, “A Hierarchical Encoder-Decoder Model for Statistical Parametric Speech Synthesis,” 2017, doi: 10.21437/Interspeech.2017-628.

[71] M. Morise and Y. Watanabe, “Sound quality comparison among high-quality vocoders by using re-synthesized speech,” *Acoustical Science and Technology*, vol. 39, no. 3, pp. 263–265, 2018, doi: 10.1250/ast.39.263.

[72] N. Adiga, V. Tsiaras, and Y. Stylianou, “ON the use of wavenet as a statistical vocoder,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Sep. 2018, vol. 2018-April, pp. 5674–5678. doi: 10.1109/ICASSP.2018.8462393.

[73] C. Valentini-Botinhao, O. Watts, F. Espic, and S. King, “Examplar-Based Speechwaveform Generation for Text-To-Speech,” *2018 IEEE Spoken Language Technology Workshop, SLT 2018 - Proceedings*, pp. 332–338, Feb. 2019, doi: 10.1109/SLT.2018.8639679.

[74] J. Kominek and A. W. Black, “The CMU Arctic speech databases,” *SSW5-2004*, pp. 223–224, 2004.

[75] “Google Colab.” https://research.google.com/colaboratory/faq.html (accessed May 19, 2022).

[76] “tmux Wiki.” https://github.com/tmux/tmux/wiki (accessed May 19, 2022).

[77] S. Ronanki, Z. Wu, O. Watts, and S. King, “A Demonstration of the Merlin Open Source Neural Network Speech Synthesis System”, Accessed: May 20, 2022. [Online]. Available: http://hdl.handle.net/10283/786.

[78] “Getting started with the Merlin Speech Synthesis Toolkit.” http://jrmeyer.github.io/tts/2017/02/14/Installing-Merlin.html (accessed May 20, 2022).

[79] “Deep learning for Text-to-Speech with continuous speech analysis and synthesis system based on Merlin.” https://github.com/malradhi/merlin (accessed May 21, 2022).

[80] H. Zen, A. Senior, and M. S. Google, “STATISTICAL PARAMETRIC SPEECH SYNTHESIS USING DEEP NEURAL NETWORKS”.

[81] R. F. Kubichek, “Mel-Cepstral distance measure for objective speech quality assessment,” *IEEE Pac Rim Conf Commun Comput Signal Process*, pp. 125–128, 1993, doi: 10.1109/PACRIM.1993.407206.

[82] “Mel cepstral distortion (MCD) computations in python.” https://github.com/MattShannon/mcd (accessed May 22, 2022).

[83] M. Morise, “D4C, a band-aperiodicity estimator for high-quality speech synthesis,” *Speech Communication*, vol. 84, pp. 57–65, Nov. 2016, doi: 10.1016/J.SPECOM.2016.09.001.

[84] P. Dai, S. Tamá Gá bor Csapó, and M. Salah Al-Radhi, “Investigation of F0 stimation algorithms and their applications in Text-to-Speech and Ultrasound-to-Speech synthesis.”

[85] W. Koenig, H. K. Dunn, and L. Y. Lacy, “The Sound Spectrograph,” *J Acoust Soc Am*, vol. 18, no. 1, p. 19, Jun. 2005, doi: 10.1121/1.1916342.

[86] Y. Jadoul, B. Thompson, and B. de Boer, “Introducing Parselmouth: A Python interface to Praat,” *Journal of Phonetics*, vol. 71, pp. 1–15, Nov. 2018, doi: 10.1016/J.WOCN.2018.07.001.

[87] P. Boersma and D. Weenink, “Praat: doing phonetics by computer [Computer program].” 2021.

[88] I. Recommendation, “1534:‘Method for the subjective assessment of intermediate audio quality.’” June, 2001.

[89] “Subjective listening test.” http://leszped.tmit.bme.hu/dr2022/ (accessed May 26, 2022).

List of figures

List of tables

Abbreviations

TTS	Text-to-Speech
ML	Machine learning
AI	Artificial Intelligence
MLP	Multilayer perceptron
FNN	Feed forward neural network
RNN	Recurrent neural network
LSTM	Long short-term memory
F0	Fundamental frequency
RTF	Real-time factor
MVF	Maximum voiced frequency
MGC	Mel-Generalized Cepstral analysis
bNM	Binary noise masking
PDD	Phase distortion deviation
cNM	Continuous noise masking
HNM	Harmonics-plus-noise model
MCD	Mel cepstral distortion
BAP	Band aperiodicity
RMSE	Root mean square error
VUV	Voiced/unvoiced
sgd	Stochastic gradient descent
adam	Adaptive Movement Estimation algorithm
rprop	resilient backpropagation