



M Ű E G Y E T E M 1 7 8 2

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

Felipe Lopes Franklin

**INVESTIGATION OF ENHANCED
LEARNING ABILITIES IN NEURAL
TEXT-TO-SPEECH SYNTHESIS WITH
MODIFIED TACOTRON 2**

SUPERVISOR

Dr. Al-Radhi Mohammed Salah

BUDAPEST, 2021

Contents

Summary.....	5
Sumário.....	6
Chapter 1 Introduction.....	7
1.1 Related Work.....	9
Chapter 2 Text-To-Speech	11
2.1 Preprocessor	12
2.2 Encoder-Decoder.....	13
2.3 Vocoder	14
Chapter 3 Deep Neural Networks.....	15
3.1 Artificial Neural Networks	15
3.2 Convolutional Neural Networks	17
3.3 Recurrent Neural Networks	20
3.3.1 Bidirectional Recurrent Neural Networks (BRNN).....	22
3.3.2 Long short-term memory RNN (LSTM RNN).....	22
3.3.2 Bidirectional Long short-term memory RNN (BLSTM)	26
Chapter 4 Methodology	27
4.1 Data	27
4.2 Tacotron 2.....	28
4.2.1 Preprocessing.....	29
4.2.2 Encoding stage.....	29
4.2.3 Attention network	30
4.2.4 Decoding stage.....	30
4.3 WaveNet.....	31
4.4 WaveGlow.....	31
4.4 Griffin-Lim	32
Chapter 5 Experiments and Evaluation.....	33
5.1 Setup.....	33
5.2 Mel spectrogram prediction	33
5.2.1 Tacotron 2 TensorFlow version	34
5.2.2 Tacotron 2 PyTorch version	36
5.3 Speech Synthesis.....	36

5.3.1 Griffin-Lim synthesis	37
5.3.2 WaveNet synthesis	38
5.3.3 WaveGlow synthesis	39
5.4 Objective Measurements	40
Chapter 6 Conclusion	41
List of Figures	42
References	43

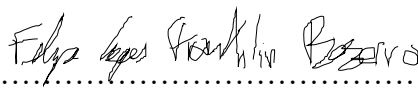
STUDENT DECLARATION

I, **Felipe Lopes Franklin**, the undersigned, hereby declare that the present BSc 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, 10 December 2021


.....
Felipe Lopes Franklin

Summary

Text-to-speech (TTS) synthesis is the generation of speech waveform, given textual input. There are several uses in today's world: car navigation, announcements in railway stations, response services in telecommunications, and e-mail reading are some examples. As the technology evolves, more natural and personalized speech can be produced. Recent advances in deep learning have shown impressive results in the domain of speech synthesis. However, the quality of text-to-speech in the conventional TTS systems is still far from natural-sounding utterances.

This thesis demonstrates the state-of-the-art technologies in text-to-speech synthesis for the Brazilian-Portuguese language. Analyzing the literature about end-to-end TTS systems, a few publicly available frameworks were chosen to test their performance in BR-PT language. In the experiments, text-to-speech systems were built using the spectrogram prediction network Tacotron 2, which is capable of learning and producing spectrograms from only text input. Different vocoders were tested, which are the ones generating speech from the spectrograms generated. In this work, three state-of-the-art neural vocoders, WaveNet, Waveglow and the Griffin-Lim, were selected for comparison.

Sumário

A síntese de texto para fala (TTS) é a geração da forma de onda da fala, dada a entrada textual. Existem vários usos no mundo de hoje que são: navegação automível, anúncios em estações ferroviárias, serviços de resposta em telecomunicações e leitura de e-mail são alguns exemplos. Conforme a tecnologia evolui, uma fala mais natural e personalizada pode ser produzida. Avanços recentes em algoritmos de aprendizado de máquinas, como por exemplo redes neurais de aprendizado profundo, têm mostrado resultados impressionantes no domínio da síntese da fala. No entanto, a qualidade da conversão de texto em voz nos sistemas TTS convencionais ainda está longe de ser difundida em diferentes línguas com qualidade excepcional.

Esta tese testa o que há de mais moderno em tecnologia de síntese texto-fala para a língua Portuguesa do Brasil (PT-BR). Analisando a literatura sobre sistemas TTS ponta a ponta, alguns frameworks disponíveis publicamente foram escolhidos para testar seu desempenho na linguagem PT-BR. Nos experimentos, sistemas texto para fala foram construídos usando uma rede de predição de espectrograma na escala mel chamado Tacotron 2, que é capaz de aprender e produzir espectrogramas apenas a partir da entrada de texto. Foram testados diferentes codificadores de voz, que são os que geram a fala a partir dos espectrogramas gerados. Neste trabalho, os sintetizadores neurais de última geração testados foram WaveNet e Waveglow. Além deles, o Griffin-Lim também foi selecionados para comparação.

Chapter 1 Introduction

Communication has always been a crucial part of people's daily lives. As a collective of individuals, we survive through communication. Technology has developed up to a point in which we no longer communicate just between ourselves but also with machines. Another case is when advanced devices help us communicate, such as the well-known theoretical physicist Stephen Hawking who due to a neurodegenerative disease was no longer able to speak and used a speech synthesizer.

Text-To-Speech (TTS) is in the process of becoming one of the main forms of communication between humans and computers. As the acronym states, it is the translation of written text into sound, or more specifically a waveform signal. It has numerous uses such as: helping in accessibility for blind or dyslexic people; home assistants, like Alexa from Amazon [1], Siri from Apple [2], or Google Home [3]; Announcements in public spaces, for instance public transportation stations and stadiums.

TTS history starts in the 20th century with the Russian professor C. G. Kratzenstein [4]. An electrical device manipulated by keys and levers was invented with the capability of generating synthetic speech. In the following years, the field of TTS research became more popular, consequently many other devices and implementations were developed. An example of mechanical vocoder can be seen in Figure 1.1 below.



Figure 1.1: Image of mechanical vocoder the Voder by Homer W. Dudley [5].

One of the methods created to enhance TTS was using Machine Learning. A technique that was developed in the mid-50s that has been having great success with state-of-the-art speech synthesis. More specifically, Deep learning, which is a field of Machine learning, has made good progress in providing natural results. Deep learning, which consists of learning from the data itself, has made the machines independent. Its great quality is also a flaw, because of its dependency on the data it learns from, Deep Learning models require a large amount of consistent and clean information. In the case of TTS, noisy and poor-quality speech examples can lead to defective models which in turn will generate distorted sounds.

In this work, state-of-the-art deep learning frameworks for TTS are used to synthesize a Brazilian-Portuguese (BR-PT) model, and consequently generate audio samples. Existing tools were used to adapt to the language, there are no records of a BR-PT TTS system [6]. This is the main inspiration of this thesis work, to implement and modify existing TTS systems to BR-PT and compare their performances.

The Portuguese language is spoken in many countries. Originating from the Iberian Peninsula of Europe, the language is spoken on three different continents. As time the time has gone by, the language has changed. European and Brazilian Portuguese have their differences [7]. For example, European speakers have a higher speech rate in terms of words per minute [7], there are differences in word pronunciation and vocabulary. These differences can have a great impact when building a TTS system dedicated to the BR-PT language.

The implementation of the BR-PT TTS system was separated into two well defined stages. The first stage, the framework used is Tacotron 2 [8], a neural network architecture for speech synthesis directly from text. This framework will be used to generate a subproduct, which are the mel spectrograms that will be used as input in the next stage. The second stage uses the subproduct of Tacotron 2 and generate the speech. This is done in 3 different ways: the first is using WaveNet [9], secondly WaveGlow [10] and lastly the Griffin-Lim algorithm [11]. The training of all systems is based on high-quality dataset of speech examples. For BR-PT, the publicly available dataset created by Cassanova [6] was used. It was created with the purpose of providing a solid speech for TTS use.

This work is divided into chapters in which the following will be explained. In Chapter 2, there will be a brief introduction about how TTS works. Then, Chapter 3

explains the Deep Learning Neural Networks that are used in the models. Chapter 4 is about methodology used in this work. The following frameworks used to produce the BR-PT model will be presented: Tacotron 2 [8], the framework responsible for taking the first step towards TTS; WaveNet [9], WaveGlow [10] and Griffin-Lim [11] which are the final piece responsible for synthesizing the speech. Chapter 5 is the description of the experiments conducted and their results. Lastly, Chapter 6 is the conclusion of the work and annotations for future work.

1.1 Related Work

The first initiative to explore TTS in PT-BR started with the publicly available Brazilian Portuguese Database [6] created by Casanova. As stated in his work, there is no public dataset with a large amount of speech and quality available for speech synthesis.

About other TTS end-to-end systems, there are plenty of different system being studied. Other models based on RNNs, which is the case of Tacotron [8], [12], like Deep Voice 1 [13] and Deep Voice 2 [14]. Deep Voice 1 is inspired by traditional text-to-speech pipelines and adopts the same structure, while replacing all components with neural networks. It is relatively simple, and its inference was faster than real-time meaning audio could be produced within a fraction of seconds. Deep Voice 2 is an improvement over the original work and an increase of scope by creating a multi-speaker TTS system. Multi speaker TTS systems had been studied before, they studied the possibility of adapting different linguistic features [15]. The multi speaker scenario is interesting for cases where there is not much data available from one source only. In the case of this work, the dataset of a single speaker [6] was used so Deep Voice did not have any advantage over the chosen frame works.

Attention based text-to-speech has been accomplished in Char2Wav as well as in Tacotron [12]. Char2Wav is an extension of the work developed at SampleRNN [16]. SampleRNN investigates the use of recurrent neural networks to model dependencies in audio data. Char2Wav uses the attention-based RNNs from Chorowski work [17] as a frontend processor and the SampleRNN as the vocoder reconstructing the speech. SampleRNN's architecture is based on WaveNet [9] with certain limitations cited in their

paper. Consequently, the choice of WaveNet seemed a good way to test the BR-PT corpus since the other architectures available are based on WaveNet itself.

As suggested in Casanova's [6] work, the future investigations with the BR-PT dataset should be using flow based architectures [18] such as FlowTron [19], Flow-TTS [20] and WaveGlow [10]. Flow-TTS is the first flow-based spectrogram generation network giving the same output as Tacotron [12]. It also uses Glow [21], just like WaveGlow [10], in order to model the images. In this thesis work, the mel spectrogram generation will be solely done by the Tacotron framework.

Chapter 2 Text-To-Speech

Before the implementation of any TTS system, there are two important things that need to be understood: text and speech. Firstly, it is necessary to understand how we produce speech. To produce a simple sound, we use many parts of our body: the lungs, diaphragm, the larynx, lips, and tongue. The lungs and diaphragm are responsible for pumping the air of the adequate pressure to the larynx. The larynx is responsible for adjusting the pitch and tone. Lastly, the lips and tongue articulate the sound to the desired degree.

At this point, the noise produced is going to be recorded and saved on a computer. Audio can be understood as samples of air pressure over a period of time. The sampling frequency in which these measurements are taken can vary from 22kHz to 44kHz, this sampling frequency is important because can impact the final result [22]. This range is explained by the Nyquist Theorem [23]. It states that the sampling frequency during data acquisition should be at least twice the highest frequency contained in the signal. For example, in the case of choosing a sampling frequency of 22kHz means each second of speech has 22k amplitudes saved. The information in this format is suitable for the binary computational environment and will be used as input to train TTS models.

Another model of voice used that can be mathematically computed is with Fourier Transform, which will decompose the signal into the frequency domain [11]. This representation after some filtering and adjustments to the scales of frequency and amplitude will generated the spectrograms. They are the visual representation of the sound in the frequency domain and will be used to synthesize the speech in the TTS model.

Secondly, another material for the speech synthetization is written text. In the finalized model, it will be the only input necessary to produce the synthetic speech. Phrases are groups of words which act together as a grammatical unit. They are a good representation of what will be said, but they do not explicitly portray the sound that needs to be produced from that word. The primary concern in TTS models from text is not how grammatically correct or well-constructed the sentence is, but rather how each word is pronounced. Consequently, a better representation of the words needs to be explored.

Following this, phonemes are introduced in the TTS system to help the machine to understand how to pronounce the words correctly. Phonemes are units of sound that distinguish one word from another in any language [24]. The learning curve is similar to when children are learning how to speak. At first, grammar or complexity of words is not a concern, but how the words are going to be said. In the case of TTS systems, that is the main concern.

TTS systems are composed of many building blocks before the synthetization of the final audio. It could be simplified to 4 main blocks: Preprocessor, Encoder, Decoder, and the Vocoder as can be seen in Figure 2.1. These blocks can be split in two categories: Front-end and Back-end. Preprocessor and encoder are part of Front-end, they will be the ones interpreting and extracting features from the text. Consequently, the Decoder and Vocoder are the ones synthesizing the final audio.

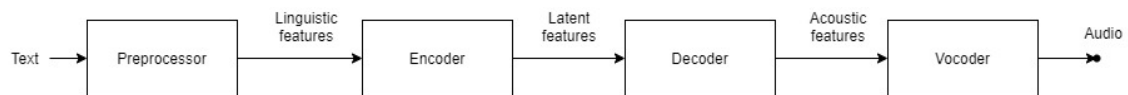


Figure 2.1: General architecture of TTS system.

Deep Learning (DL) models are composed by these blocks, some models such as Tacotron 2 [8] have an end-to-end architecture, meaning that it has all 4 blocks implemented in its code. On the other hand, there are models solely focused on the last process such as WaveNet [9] and WaveGlow [10]. In this chapter, these 4 blocks will be explained separately.

2.1 Preprocessor

Preprocessors are responsible for interpreting the text and transforming it into a more valuable format for machines to use. Tools like Ossian and Festival [25] are open source tools available for this processing. Festival implementation is based on Hidden Markov Models [26]. The goal of front-end tools is to transform into a more symbolic description. Preprocessing usually starts at text cleaning, which removes every piece of information in the string of text that will not be synthesized. Following that, is text

normalization, which is the conversion of dates, numbers, addresses, currency, abbreviations into normal orthographic form. After this, there is Phonetization which transforms the normalized text string into the phonetic version. Lastly, the syllabification of phonetic text divides it into syllables.

This analysis is independent from the back-end. The final output is the Linguistic features described in Figure 2.1.

2.2 Encoder-Decoder

The encoder-decoder block in a TTS system comes from how the TTS problem itself is approached by machine learning research. One difficulty of TTS is how the model is going to map each Linguistic feature to the input audio. This association can be done with a Sequence-to-sequence (seq2seq) model. It maps fixed-length input and output with different sizes. This model comes from Natural Language Processing models introduced in translations [27].

Apart from that, the text needs to be understood by the computer in a binary form. The preprocessing of the text only transforms it into a more valuable resource. The encoder is responsible for performing the embedding. Embedding is the term used in Natural Language Processing for the representation of words into vectors [28]. Computers do not understand language as humans do, everything needs to be translated to numbers. Depending on the embedding, it can represent a single word or even a unique character. This encoded form of the speech will go through the encoder layers in order for the model to learn how to predict the text. The prediction is made on how each character and each word influences the next character or word.

The Encoder-Decoder are the main processors of the model. The Encoder will receive the Linguistic feature and learn its characteristics. It will collect the information and propagate forward as an intermediate vector to the Decoder.

The input of the Decoder contains all the information needed for it to make the prediction. It will read the entire intermediate vector and output Acoustic feature. The Acoustic feature outputted by the decoder is commonly mel spectrogram. It is a modification of the spectrograms explained before. A mel spectrogram is a spectrogram

converted into the mel scale [29]. The mel scale was created to better suit human hearing. It takes tone and pitch into consideration and therefore produces a more accurate representation of the sound for humans.

2.3 Vocoder

The last block of the diagram is the one responsible for synthesizing the speech. Vocoderes are old implementations and once they existed before computers. The VODER [4] invented was purely mechanic. Nowadays, more complex and accurate models exist. Algorithms based on the Griffin-Lim algorithm [11] can make a good prediction of the final waveform from spectrograms. WaveGlow [10] and WaveNet [9] use other types of vocoderes that will be explained later.



Figure 2.2: Modern day vocoder. An instrument used to produce human speech electronically [30].

A modern-day vocoder is shown in Figure 2.2. This equipment is used by musicians in order to reproduce human speech. It can be understood as a physical interpretation of the machine learning models studied in this work. These are all the pieces needed to understand a general TTS architecture. In the next chapter, each part of the systems used in this thesis work are going to be explained in detail. Also, the frameworks used take different parts of the whole system.

Chapter 3 Deep Neural Networks

Machine learning and Deep learning have been used to describe the mathematical algorithms that can learn from its results. Both are under the same category of Artificial Intelligence. According to the Cambridge Dictionary it is “*the study of how to produce machines that have some of the qualities that the human mind has, such as the ability to understand language, recognize pictures, solve problems, and learn*” [31]. However, there is a clear difference. While machine learning operates with a manual extraction of features, meaning that the features have been previously extracted such as someone creating a table of information about the housing market. Deep learning is an independent learning algorithm, in the housing market context, it would mean giving a picture of a house to the model and it would interpret and evaluate the features of the house on its own. In that case, the features in TTS are the linguistic and acoustic features that are automatically extracted from the audio and spectrograms. Deep learning relies on high-performance computing and large amounts of data.

3.1 Artificial Neural Networks

Artificial neural networks are inspired by the most powerful processor known, the human brain. The basic structure is the neuron, which is a processing node connected to each other by weight functions. Weight functions modify the input to be processed by the node [32]. Each node receives weighted information via these connections and produces an output by passing this information through an activation function. It will define how the weighted sum of the inputs is transformed into an output from the node. Equation 3.1 simplifies the output of a node of the Artificial Neural Network in mathematical form.

$$Output = Activation\left[\sum (weight * input)\right] \quad (3.1)$$

There are several activation functions. The choice of an activation function will depend on the problem and architecture of the ANN. The most common functions used are: Sigmoid, Tanh and ReLu [33]. The functions can be seen respectively in Equations 3.2, 3.3 and 3.4 below:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (3.2)$$

$$\text{Tanh}(x) = \frac{2}{(1 + e^{-2x})} - 1 \quad (3.3)$$

$$\text{ReLU}(x) = \max(0, x) \quad (3.4)$$

The elementary architecture of an artificial neural network is the Feed Forward. As the information comes from the input, it passes through the weight functions and neurons, it then moves forward to produce the last output. In the Feed Forward architecture, there is no feedback from the output of the neurons to other neurons, that information only moves forward to the output [32]. These can be single or multi-layered depending on the number of hidden layers between the input and output layer. Single layer architecture means the input is passed through the weight functions and then computed by the output layer. On the other hand, for multi-layer, there are layers of neurons between input and output, these are called the hidden layers which contain neurons. A multi-layer architecture is shown in Figure 3.1. The input layer with 3 nodes represents the inputs. The Hidden layer contains 4 nodes. Lastly the output contains two nodes.

For example, in the case of predicting the price of a house based on its features. The input nodes would receive information regarding the house, for instance the number of rooms, size of the house in square meters and its distance from the city centre. Based on historical data, these features would be computed, and the output would be a price based on that. The learning process is to change the weights to achieve an output desired. Below it is demonstrated in Figure 3.1 the architecture of a multi-layer Feed Forward ANN.

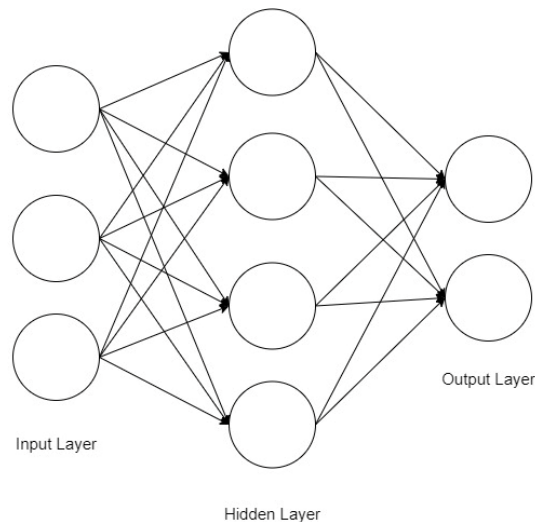


Figure 3.1: Artificial Neural Network architecture.

This training is made by the “back-propagation algorithm” [34]. It consists of updating the weights by back propagating a gradient vector in which each element is defined as the derivative of an error measured with respect to a parameter [32]. The loss function is responsible for measuring how far from the target value the output is. This later influences the adjustment of weights to get closer to the correct value. An epoch or step indicates the number of passes of the entire training dataset has completed [32].

Continuing the example used before, the algorithm would correct the weighted arrows with back-propagation giving the right importance to each input. In the housing market, a bigger house impacts the price stronger than the number of windows. On the other hand, the number of windows may impact the final price more than the distance from the city centre. The weights that connect each node in the architecture are what influences the final output. They are the spine of the algorithm. After the training of the model, the resulting loss should be the closest as possible to zero, i.e., the predicted value is similar to the real one.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are commonly used in image deep learning problems [35]. CNNs are important in TTS deep learning because the audio signals are synthesized from mel spectrograms, and they are interpreted as images.

Algorithms have shown that CNNs work well in speech recognition by preprocessing audio signals [36].

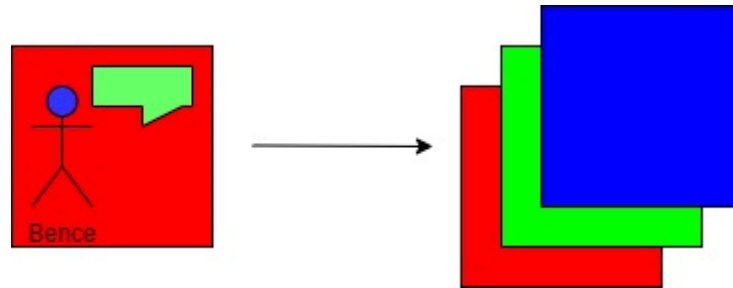


Figure 3.2: Destructuring of the image in the three primary colors. Each layer based on the primary color will be summed up to give the final pixel color.

To begin with, a brief introduction on how computers understand images. Each pixel in an image has a color. A diagram representing the RGB system can be seen in Figure 3.1. The higher the quality of the image, the more pixels it has. The most popular high resolution commercial television today is 4k, it has around 8 million pixels. Each of these pixels contain the information of the color they are. The system followed is the RGB. It is an additive color model in which red, green and blue colors are added together generating all the different colors and shades. The three layers of colors are added up giving the final pixel color [37].

A practical example of this system can be seen in Figure 3.3. The color of the Budapest University of Technology and Economics (BME) logo in their website can be represented in the RGB system as (147, 5, 47), or in hexadecimal representation 93052F. This is the addition of different shades of Red, Blue and Green [37]. The first color in Figure 3.3 is the shade of Red, which is very close to the actual final color. Then, it is followed by the shade of Green. because of its low value the color resembles black. And finally, the shade of Blue, which is a dark blue.



Figure 3.3: Representation of color addition with the BME logo color.

In Figure 3.4, a basic structure for a CNN is shown. The input image is modeled as 3 different images in each primary color of the RGB representation. The next layers are called Convolutional Layers. In each convolutional layer a Kernel convolution is applied to local regions of the input through the calculation of the scalar product between the kernel convolution weights and the region connected to the input [38].

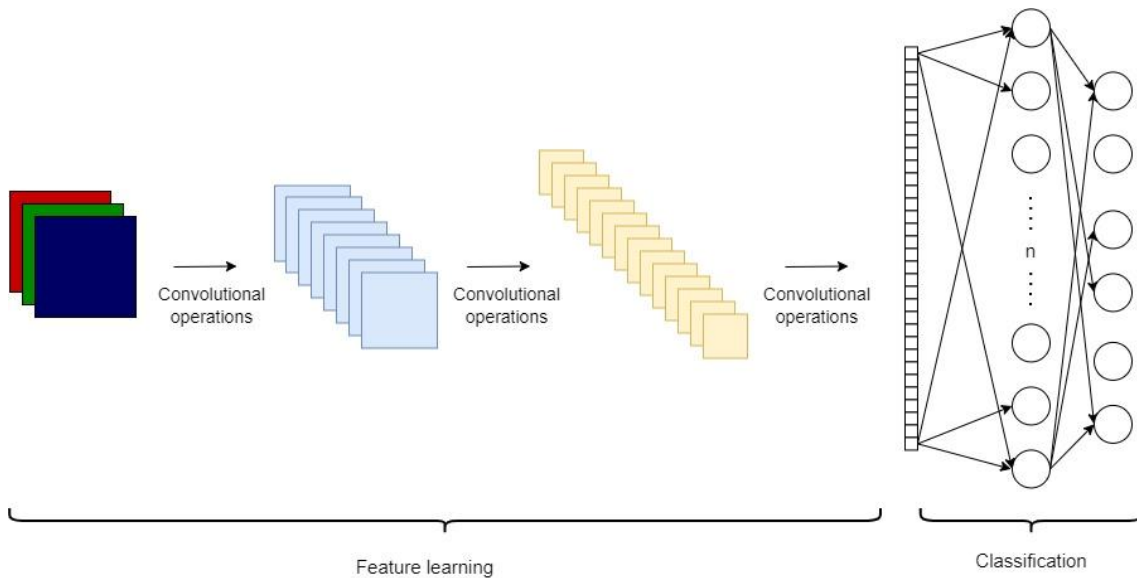


Figure 3.4: Architecture of Convolutional neural network. First layers being convolutional and pooling layers until the image is reduced to one dimension. The output of the convolutional layer is the input of a Feed Forward Artificial Neural Network.

The Kernel convolution process is shown in Figure 3.5. A kernel is a function that calculates dot product of the image according to the kernel mapping. The kernels are usually smaller than the input image but spread along the entirety of the depth of the input [38]. A new image is generated from the convolution between the input and the kernel. This produced image represents a possible feature of the original image.

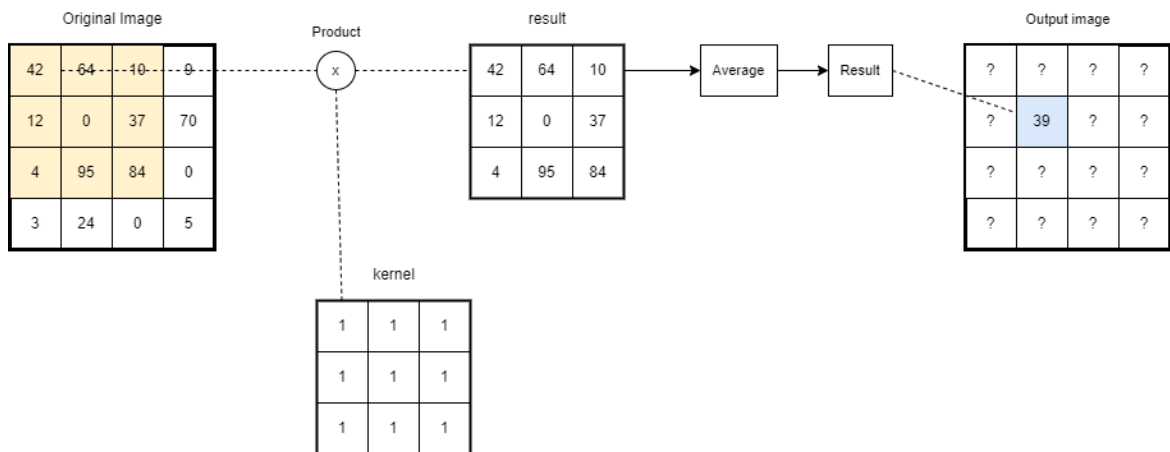


Figure 3.5: Kernel Convolution of input image. The activation map is an average. The output of the operation is placed in a new image.

Since these are the hidden layers of the ANN, they may not make much sense for humans, but they are useful for machine computations. These will generate a stack of output images increasing the dimension as can be seen in Figure 3.4 [38]. Since working with large images can become very costly due to the size that the ANN gets, every convolutional layer down-samples the size of the images reducing the amount of pixels in the convoluted image.

After a certain number of Convolutional Layers, the image will be reduced to the size of one pixel. This effect be seen in the Figure 3.4. They image will be flattened and only the features generated remain. The last set of layers are regular fully-connected layers that will do the work just like in a regular ANNs [38]. These flattened features will be learned and generate the final output.

3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are also implemented in TTS systems [9], [16], [39]. RNNs are called like this because the output of neurons are used horizontally as input of another neurons [40]. It is useful in the case of acoustic modelling because they can process sequences of inputs, which is completely different from a static image for example and produce sequences of outputs. RNNs are able to map and remember features learned from past processed nodes and enhance the accuracy of the output. The RNN's architecture is described in Figure 3.6.

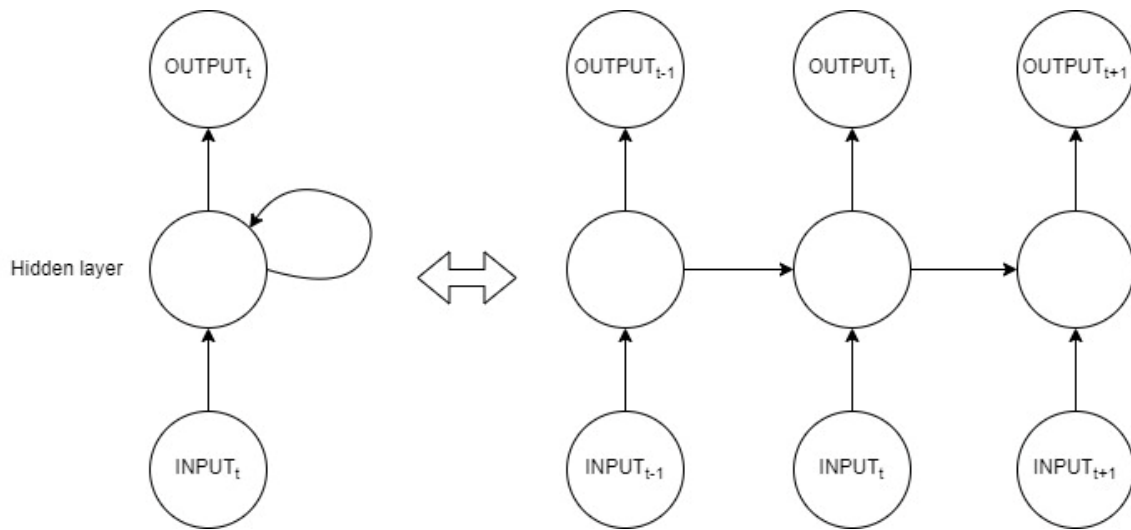


Figure 3.6: Representation of Recurrent Neural Network. On the left, the rolled visual representation of the whole network. On the right the unrolled representation of individual layers.

As a result of its unique architecture, the training of this kind of network is different from regular Back propagation [34]. The training for RNNs is called Back propagation through time (BPTT) [41], it is more advanced and specific for sequence data types. It works similarly to the regular version, where the model trains itself by calculating error from the output to the input. BPTT differs from the traditional version because it sums errors at each timestep because it shares the parameters across the layers. Because of how the weights are adjusted through the layers, it is easy for an error to be propagated. The wrong operation on the weights can result in its sudden decrease at the point it gets too close to zero. This issue is called the vanishing gradient. The opposite effect is also possible. A weight's importance can be wrongly interpreted by the algorithm causing it to increase too much. This effect is called exploding gradient.

There are other architectures of RNNs that implement new features. Bidirectional Recurrent Neural Networks (BRNN) and Long Short-Term Memory (LSTM). Moreover, the fusion of the two is an existing and used model called the Bidirectional Long Short-term Memory Recurrent Neural Networks (BLSTM). They try different approaches to better process sequential data and have a memory of the processed inputs.

3.3.1 Bidirectional Recurrent Neural Networks (BRNN)

A BRNN considers all available input sequence in both the past and future for estimation of the output vector [42]. To do so, one RNN processes the sequence from start to end in a forward time direction, this is seen as the blue arrows in Figure 3.7. Another RNN processes the sequence backwards from end to start in a negative time direction as demonstrated in Figure 3.7 with the red arrows. Outputs from forward states are not connected to inputs of backwards states, and vice versa.

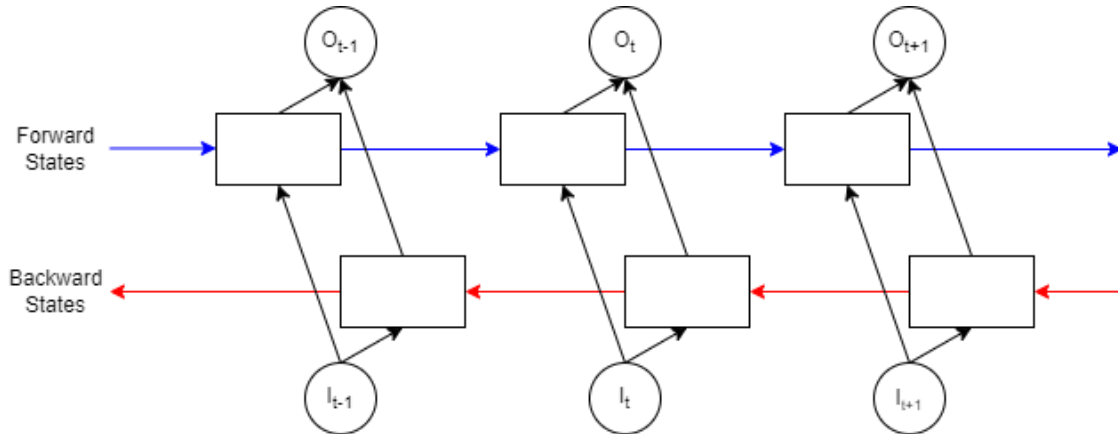


Figure 3.7: Architecture of Bidirectional Recurrent Neural Network. Forward states are represented by blue lines. Backward states are represented by red lines.

This is an advantage and a constraint since BRNNs will only work if the start and end of input sequences are known in advance [42].

3.3.2 Long short-term memory RNN (LSTM RNN)

Another architecture created to solve the vanishing gradient problem is the Long Short-Term Memory (LSTM). LSTMs are designed to remember the information for longer and therefore be more efficient [42]. In their flow, some pieces of information run down the entire chain of layers with minimal linear interactions. LSTMs have a special memory cell with self-connections in the recurrent hidden layer to maintain its states over time, and three gating units (input, forget, and output gates) which are used to control the information flows in and out of the layer as well as when to forget and recollect previous states.

A general overview of the LSTM cell can be seen in Figure 3.8. The core information runs through all cells horizontally. This is called the Cell State. It is

highlighted in Figure 3.9. Also, it is composed by several operations that represent the gate units which will control the flow of the cell state.

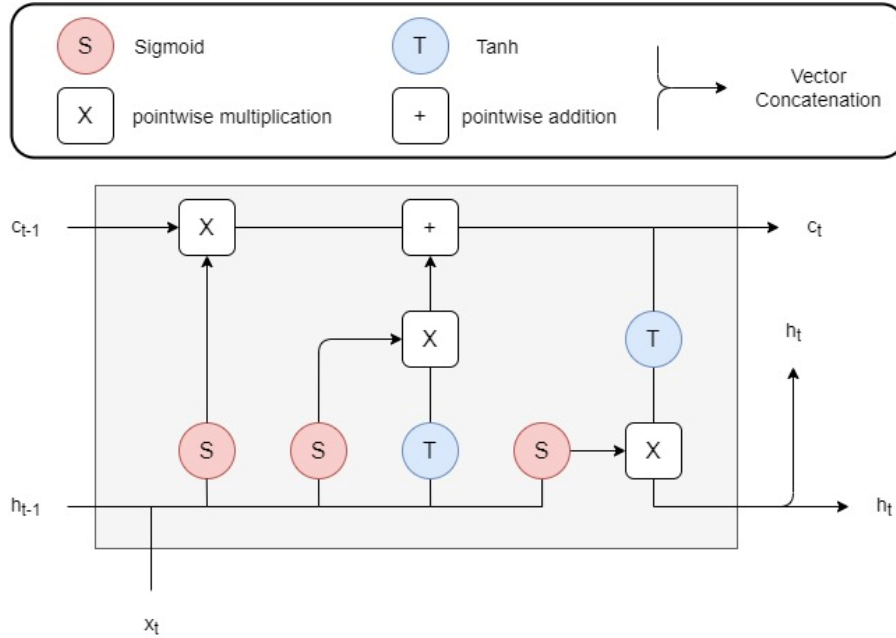


Figure 3.8: General overview of LSTM cell.

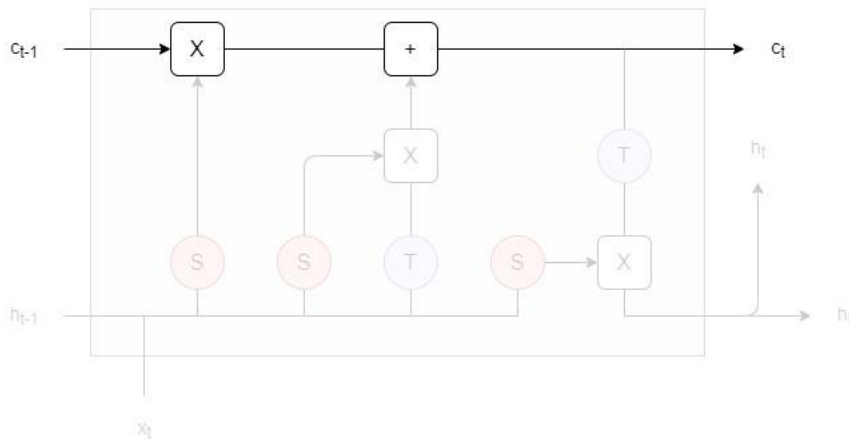


Figure 3.9: Cell state highlighted in the High-level diagram of a LSTM cell.

The first gate is the Forget Gate. It is the one that decides if the cell state should be remembered or not. Its result is a value between zero and one. The closer to zero means it forgets and the closer to one means it remembers the information [42]. The Forget Gate result is shown in Figure 3.10 below.

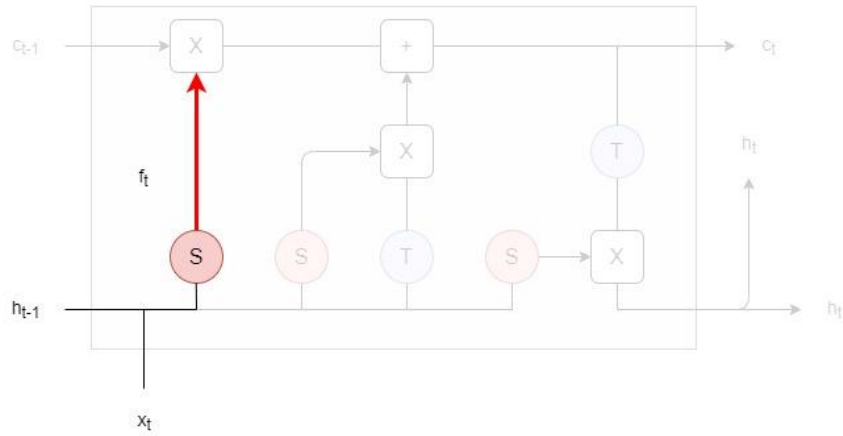


Figure 3.10: Forget Gate highlighted. Red arrow represents the output of the Forget Gate.

The mathematical representation of this operation is the following:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.5)$$

The next step is to decide if the information received should be stored in the cell state. This is done by the Input Gate. The Input Gate is shown in Figure 3.11. Another operation performed by the input gate is the generation of the updated parameters of the Cell state. There will be two outputs of this gate: i_t which is the decision of how much should the Cell state be updated, and c'_t which will be added to the Cell state.

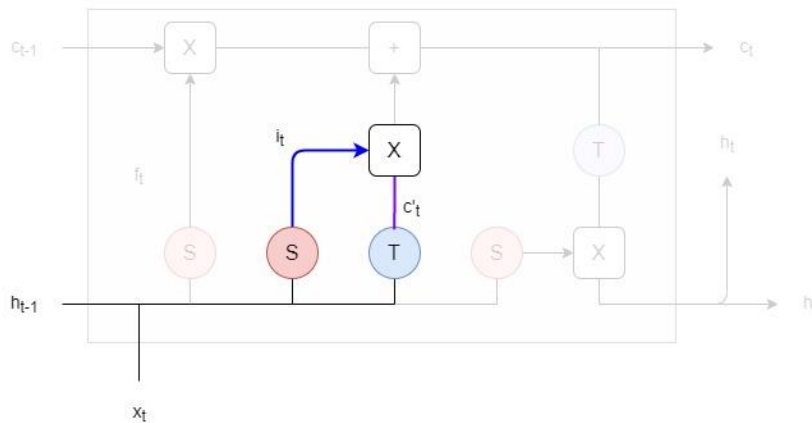


Figure 3.11: Input Gate highlighted. The i_t represents the decision to update the Cell State or not. The purple arrow represents c'_t which is the candidate to update the Cell state.

Mathematically, this is how the Cell state is represented:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.6)$$

$$c'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3.7)$$

The next operation on the chain is to update the Cell state finally. This updated state will be shared through other Cell units. This operation is described in Figure 3.12.

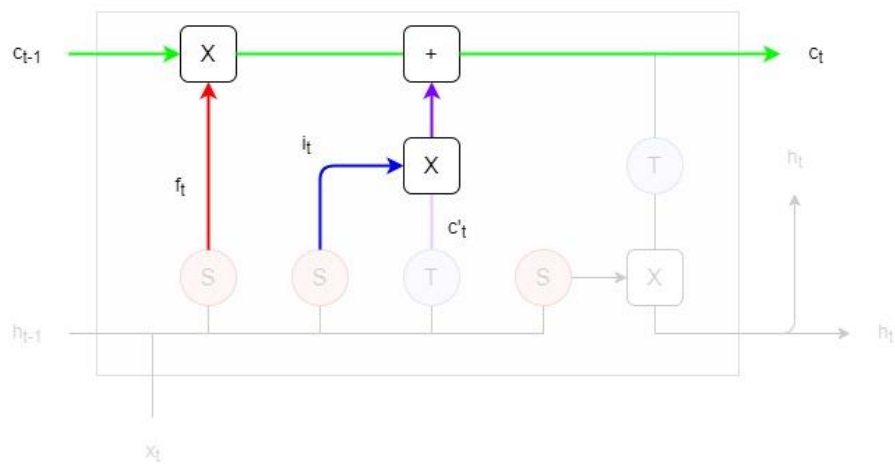


Figure 3.12: The Cell state is updated based on the Forget Gate and the Input Gate. New values to be updated are also used as inputs.

This update is translated mathematically to: $c_t = f_t * c_{t-1} + i_t * c'_t$. The Forget gate output seen in Equation 3.5 is used to update the state cell. As well as the newly calculated cell state c'_t seen in Equation 3.7 and the deciding factor which is i_t described in Equation 3.6.

Lastly, The Output Gate. The output of this gate is based on the cell state. This operation is shown in Figure 3.13.

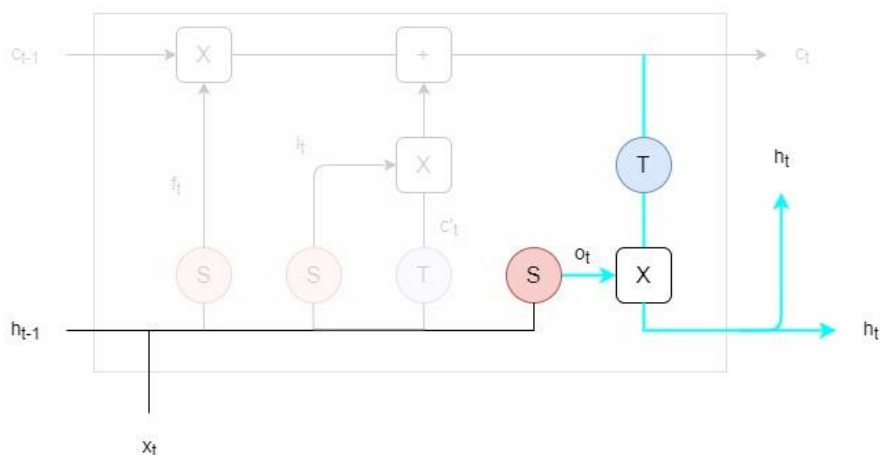


Figure 3.13: Output Gate highlighted in light blue. The last gate that gives the parallel information for other Cell units as well as the output of the node itself.

These final operations are the following:

$$o_t = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3.8)$$

$$h_t = o_t * \tanh(c_t) \quad (3.9)$$

The updated cell state c_t goes through the Tanh transformation to be multiplied with o_t from the Output Gate. This operation is displayed in Equation 3.8 above. Equation 3.9 gives the Hidden state which will be carried to other LSTM cells. It is also the final output of the layer going forward giving the final result of the network.

The previous architectures shown before have the vanishing gradient problem, resulting in the network failing to learn long term sequential dependencies in data. LSTM RNNs are popular for tackling this problem specifically [39], [42]. There are many other architectures of LSTMs, but this standard version presented is necessary to understand the frameworks used for TTSs systems.

3.3.2 Bidirectional Long short-term memory RNN (BLSTM)

BLSTMs are the fusion of the two previously presented architectures. Instead of regular ANN nodes in the RNN, BLSTMs have LSTM memory cells in their architecture [43]. In this manner, they are capable of learning in both directions, as well as keeping record of it. They have shown to outperform unidirectional LSTMs and bidirectional RNNs [44].

Chapter 4 Methodology

This chapter describes the Brazilian Portuguese TTS system implementations based on Tacotron 2 [8] as the mel spectrogram generator. Also, using WaveNet [9], Waveglow [10] and the Griffin-Lim Algorithm [11] as synthesizers. For Portuguese, there is not any resource on TTS dedicated implementations.

4.1 Data

Firstly, it is necessary to point out that Brazilian and European Portuguese (EU-PT) are quite different. Some pronunciations may differ a lot from one to another [45]. Some words are completely different in both languages. While “train” in BR-PT is translated to “Trem”, in EU-PT is “Comboio”. Brazilians may condense expressions into one verb. For instance, the act of wishing happy birthday to someone in can be said in EU-PT as “dar os parabens” while in BR-PT it can be simplified to “parabenizar”. Other examples can be seen in Table 4.1 below.

Table 4.1 Differences between European and Brazilian Portuguese language and its translation to English

English	EU-PT	BR-PT
sport	desporto	esporte
goalkeeper	guarda-redes	goleiro
queue	bicha	fila
fridge	frigorífico	geladeira
pedestrian crossing	passadeira	faixa de pedestre
girl	rapariga	menina
espresso	bica	cafézinho
stapler	agrafador	grampeador

Portuguese is a language with few publicly available resources for speech synthesis. In BR-PT, the database created by Casanova [6] was the only one found with a decent size for deep learning. Although there are some public speech datasets for European Portuguese, for example, the work has a small amount of speech, approximately 100 minutes, which normally is not optimal for training deep-learning models [46]. Other more extensive databases can be found such as the CETUC dataset. It was constructed

for Speech Recognition studies; it has 145 hours of 100 speakers. On the other hand, each speaker produced 1.45h on average. The amount of speech per speaker makes it difficult for creating a single-speaker voice synthesis [6].

The recording was made by a native BR-PT speaker man. The final dataset consists of a total of around 71 thousand words spoken by the speaker, 13 thousand unique words, resulting in 3632 audio files and totaling 10 hours and 28 minutes of speech. Audio files range in length from 0.67 to 50.08 seconds [6]. Since the audios were not recorded in an acoustic studio, there is background noise present in the audio files, the authors also made a trained and filtered version. For this purpose, the library RNNoise was used to produce a 22kHz sampling rate [43]. It is based on Recurrent Neural Networks, more specifically Gated Recurrent Unit (GRU) and demonstrated good performance for noise suppression.

In the TTS-Portuguese Corpus paper, the authors conclude that this dataset could be further investigated with other TTS system such as flow-based ones. In this thesis work, this dataset is used not only with a flow-based TTS system, which is WaveGlow [10], but also the probabilistic and autoregressive WaveNet [9].

4.2 Tacotron 2

Tacotron 2 [8] is an end-to-end TTS system that can be trained on <text, audio> pairs generating a good quality speech. It is a revised and modified version of the original Tacotron [12] to better fit WaveNet's [9] architecture. Both versions are end-to-end models, which means they contain both front and backend. The main difference between the two is the synthesizer used. In the first Tacotron version, the Griffin-Lim Algorithm while Tacotron 2 uses mel-spectrograms to synthesize speech using the WaveNet vocoder [9]. Its part on this work is generating mel spectrograms for the analyzed vocodes: WaveNet [9] and WaveGlow [10]. Tacotron 2 is responsible for the first 3 blocks of the TTS system shown in Figure 2.1 in Chapter 2: Preprocessing, Encoding and Decoding. It will receive <text, audio> pairs from training and output mel spectrograms. This architecture is pictured in Figure 4.1 below. The three blocks are highlighted with different colors. Each of them will be explained separately.

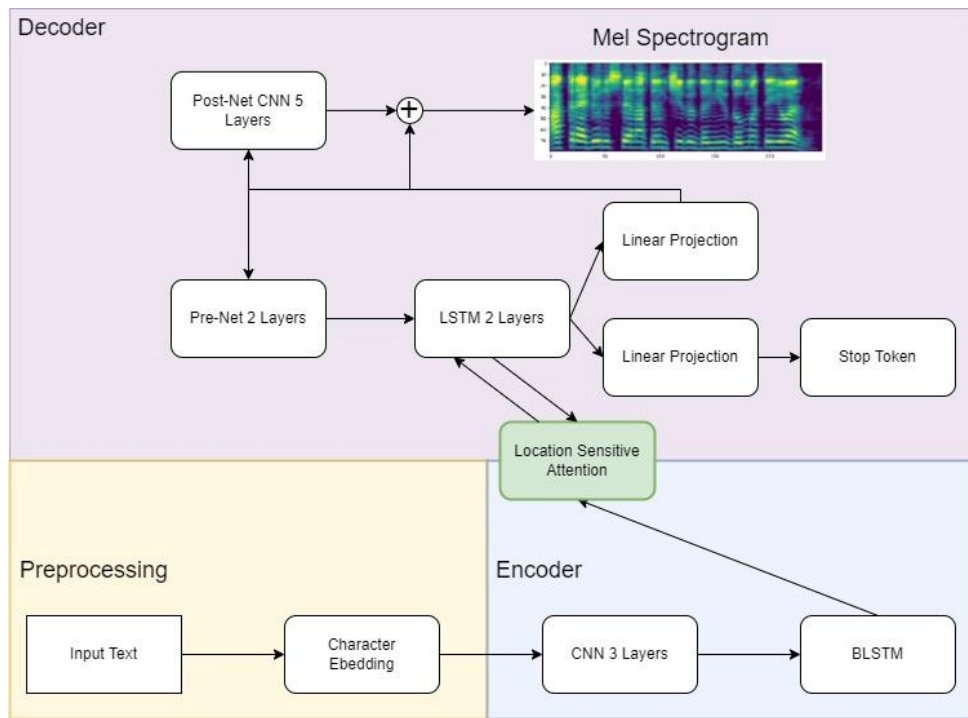


Figure 4.1: Tacotron 2 architecture. It is divided in three sections according to traditional TTS systems models.

4.2.1 Preprocessing

Firstly, the process is initiated with the preprocessing of the Data. In Figure 4.1, the Preprocessing blocks are highlighted in yellow. The preprocessor of Tacotron 2 [8] is an improvement of the one introduced in Tacotron [12]. It is a 512-dimension character embedding. Character embedding is very common in Natural Language Processing. In the case of TTS, it is used to compute continuous vector representations of the words [28]. Recurrent Neural Networks have shown to be efficient in language modeling [47]. Because of the architecture of the RNNs, the model is capable of learning on how to associate each word with its neighbors, not only the successor words, or characters, but its predecessor.

4.2.2 Encoding stage

The character embedding is processed by the encoder. This step helps with convergence and improves generalization [12]. It consists of three Convolutional Neural Network layers with Rectified Linear Unit (ReLU) [48]. CNNs have shown to be very popular in the Text-To-Speech field getting excellent results through different types of architecture [9], [12], [16]. It is important in this context because it will learn features from the character embeddings that will better connect to the final mel spectrogram

output. The last step of preprocessing is the bidirectional LSTM layer generating the encoded features. LSTM are designed to remember information for longer and therefore the encoded features are more precise [49].

4.2.3 Attention network

The attention mechanism is what connects the output of the encoder to the input of the decoder. A potential issue with this encoder–decoder architecture is that an ANN would need to compress the input into a fixed-length vector to be used by the decoder [50]. This may make it difficult for ANNs to cope with long sequences, such as long sentences.

It is a component responsible for managing the interdependence between the input and the output of the decoder. This mechanism is depicted in Figure 4.1 by the arrows in both directions connecting the encoder to the decoder. In the case of Tacotron 2, it is connecting the encoded features to the generated mel spectrograms [51]. The output of the attention is called the attention context, which is the mapping of the input with the desired output. The Attention model used in Tacotron 2 is the Location-Sensitive proposed by Chorowski [17]. This model is an addition to the additive attention mechanism [50].

The additive attention mechanism works each time the proposed model generates a decoder timestep, it does a soft search in the sources where the most relevant information is concentrated [8]. The location sensitiveness is computed for the alignment between the generated state and the previous alignment [17]. Different attention mechanisms have been proposed to Tacotron 2 which could improve its results [51].

4.2.4 Decoding stage

The decoder is an autoregressive RNN which predicts the mel spectrogram from the encoded input sequence one frame at a time [8]. The predicted previous timestep passes through the pre-net block in Figure 4.1 and is concatenated with the attention context. Then, the result of this concatenation is passed through a linear transformation predicting the spectrogram frame. Finally, this prediction is passed through a CNN of 5 layers improving its overall reconstruction [8].

In parallel with the prediction, the output of the concatenation of the decoder, which are the two layers LSTM, and the attention context is further processed by a linear projection to predict the probability that the output sequence has been completed [8]. This

process is the “Stop Token” depicted in Figure 4.1. The final output is the mel spectrogram which will be used in this work by WaveNet and WaveGlow to synthesize the audio.

4.3 WaveNet

WaveNet is fully probabilistic and autoregressive, with the predictive distribution for each audio sample conditioned on all previous ones [9]. The main concept behind WaveNet is Causal Convolutions. They make sure the model follows the ordering of the data, the prediction of a certain timestep is not conditioned on any future timestep. The implementation used in WaveNet is the Dilated Causal Convolution, it is a convolution where the filter is applied over an area larger than its length by skipping input values by a given length [9]. This architecture helps to receive the biggest amount of input possible maintaining the input’s resolution as well as computational efficiency [9]. The architecture of dilated causal convolutions is depicted in Figure 4.2.

Other components of WaveNet architecture are based on the PixelCNN architecture and will not be explored in this thesis work [52]. The system will be used as a vocoder, trained with mel spectrograms generated from the Tacotron 2.

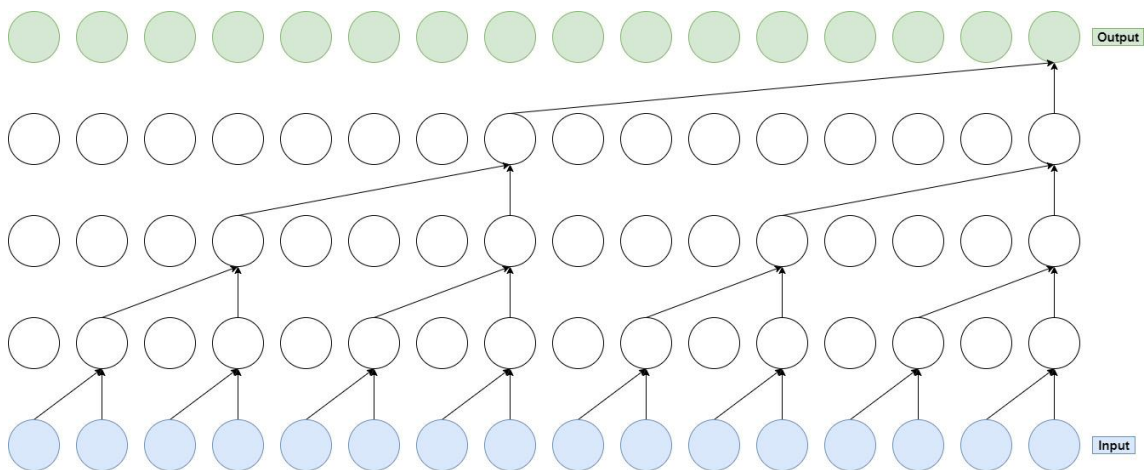


Figure 4.2: Architecture of the Dilated Causal Convolution network.

4.4 WaveGlow

WaveGlow is a flow based network capable of generating state of the art audio files from mel spectrograms [10]. It combines ideas from WaveNet [9] and Glow [21].

The input goes through the network as groups of 8 audio samples in vector form, which is depicted in Figure 4.3 as the “squeeze to vectors” block in the schema. Then, it processes these vectors through “steps of flow” which are represented by the highlighted number twelve in the blue box in Figure 4.3. A step of flow here consists of an invertible 1x1 convolution followed by an affine coupling layer, described below [10].

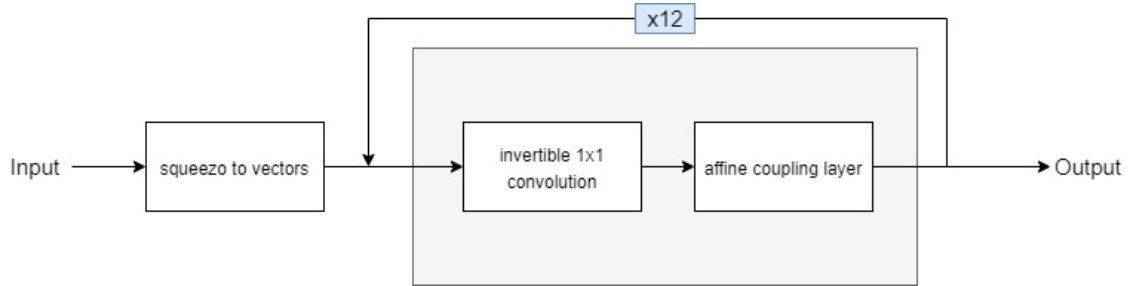


Figure 4.3: WaveGlow architecture.

The affine coupling layer is a way to stack a sequence of invertible bijective transformation functions [21]. The functions used were inspired in the WaveNet [9] dilated convolutions and skip connections, with the exception that they are not causal. The affine coupling layer is also where the upsampled mel spectrograms are included in order to condition the generated result on the input [10]. Up sampling is bringing back the initial resolution that the image had originally. On the case of CNNs, the images are down sampled, and its feature extracted in the process. This helps with computational power, it is easier to work with smaller vectors and matrices [53]. In WaveGlow, the mel spectrograms are upsampled to be transformed into the standard resolution.

4.4 Griffin-Lim

The Griffin-Lim algorithm [11] is the only synthesizer used in this work that is not based on ANNs. It is an algorithm that synthesizes waveforms from predicted features, a different target for seq2seq decoding [27] and waveform synthesis. The seq2seq target can be highly compressed if it provides sufficient intelligibility and information for an inversion process, which could be fixed or trained. It was firstly used in the original Tacotron [12], the post-net processing is responsible for taking the seq2seq output and feeding it to the Griffin-Lim that synthesizes the waveform. It showed strong results and it has a simple implementation.

Chapter 5 Experiments and Evaluation

The experiments were divided into two categories: the mel spectrograms generation and the speech synthesis. Two publicly available versions of the Tacotron 2 were used in this work. One of them is the NVIDIA [54] codebase which is compatible with WaveGlow [10] based on PyTorch [55]. Additionally, the TensorFlow implementation [56] compatible with WaveNet [57]. These two versions will be tested with the BR-PT [6] speech corpus. Following this, the speech is synthesized by the three vocoders presented in Chapter 4: WaveGlow [10], WaveNet [9] and Griffin-Lim [11]. Each one of them will generate the audio files which will be evaluated through Objective measurements.

Objective evaluations were carried out with the finalized audio samples. Two metrics were taken into consideration to perform the analysis: Frequency-weighted segmental SNR (fwSNRseg) and Extended Short-Time Objective Intelligibility (ESTOI). fwSNRseg relates the SNR of the signal to its intelligibility. Lastly, ESTOI was introduced to measure speech performance by calculating correlation between temporal envelopes of natural and synthesized speech [58], [59].

5.1 Setup

The training of the models was conducted in the SmartLab, which is the Speech Technology and Smart Interactions Laboratory. The laboratory has a server provided by the Department of Telecommunications and Media Informatics. The server is equipped with a NVIDIA GeForce GTX TITAN Xp GPU. The GPU has 8 cores and 32 GB of Memory. This kind of equipment is necessary to make the extensive calculations and matrix operations that are performed in the training of a DNN.

5.2 Mel spectrogram prediction

Compatibility between the mel spectrogram prediction and the vocoder to synthesize the input is crucial. Since images can be vectorized differently, incompatible frameworks would not generate good quality speech or would not work due to the

mathematical impossibility of some matrix operations. The two versions used of Tacotron, and their output are separated.

5.2.1 Tacotron 2 TensorFlow version

The first Tacotron version experimented was the publicly available model based on TensorFlow [57], [56]. It is an implementation of the Tacotron 2 [8], the version compatible with WaveNet [9]. The network was trained up to 100 thousand steps. The evaluation of the model is shown in Figure 5.1. The enhanced model used for this work was decreasing the learning rate of the training. The learning rate proposed in the available model [57] lead to exploding loss, when the loss between the predicted output and the ground-truth was too big. This caused problems during the training because it would stop every time the loss exploded, needing it to be resumed manually.

Due to the problem with the learning of Tacotron ANN, the training time was not exactly measured since it would only be resumed when the server was checked. In total, a rough estimation of the training time of Tacotron was around 12 days.

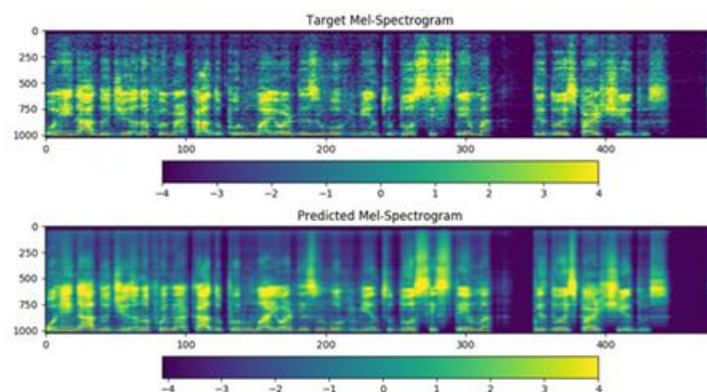


Figure 5.1: Comparison between target and predicted mel spectrogram on training set made by Tacotron.

Two custom sentences were used to produce mel spectrograms. These sentences are the only input used to produce the mel spectrograms differently from the mel spectrogram in Figure 5.1 which had the <text, audio> pair as input.

The two custom sentences and their translations were:

- “O futebol para mim era feito de gols, muitos gols.” Which translates to “Soccer for me was made of goals, lots of goals.” (1)

- “Ocorrem em grande quantidade, especialmente nas encostas das montanhas.” Which translate to “[They] Happen in big quantities, especially in the mountainside.” (2)

These sentences will be used as input for the vocoders in the next section of the Experiments chapter. The mel spectrograms generated are displayed below in Figure 5.2 and Figure 5.3.

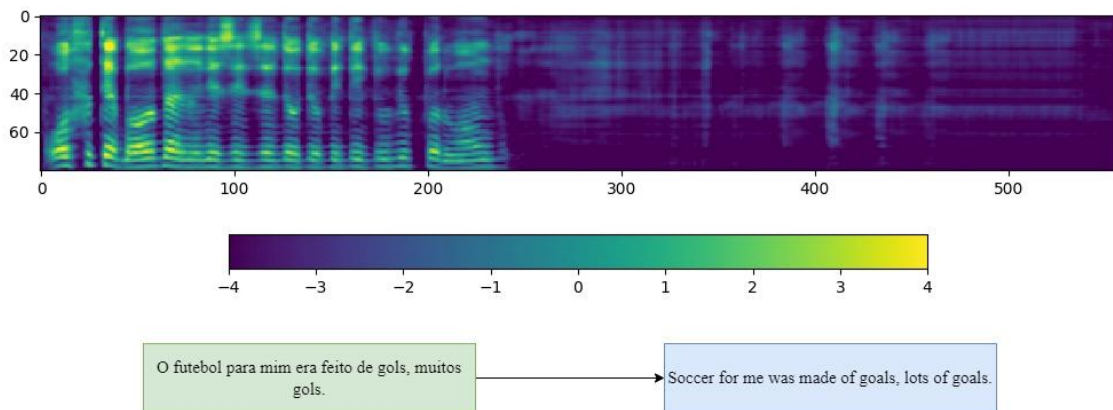


Figure 5.2: Mel spectrogram generated from custom sentence (1).

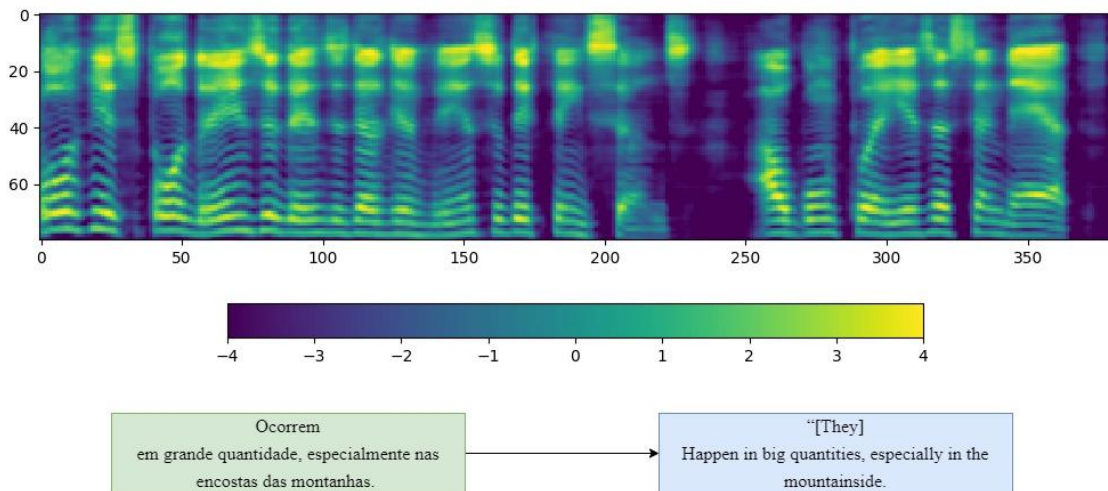


Figure 5.3: Mel spectrogram generated from custom sentence (2).

It can be concluded that when predicting the mel spectrogram with a baseline, the model seems to be more successful. The <text, audio> pair seemed to give more information for the network to predict the mel spectrogram.

5.2.2 Tacotron 2 PyTorch version

The Tacotron 2 PyTorch version was made by NVIDIA [54]. The mel spectrograms generated by the NVIDIA's Tacotron performed the worst. It is possible to see in Figure 5.4 and 5.5 that the network was not capable of learning any feature generation. The training of the model was stopped after 240k steps for 7 days, this does not represent the full training suggested by the framework. The mel spectrogram displayed for both sentences have basically nothing expressed.

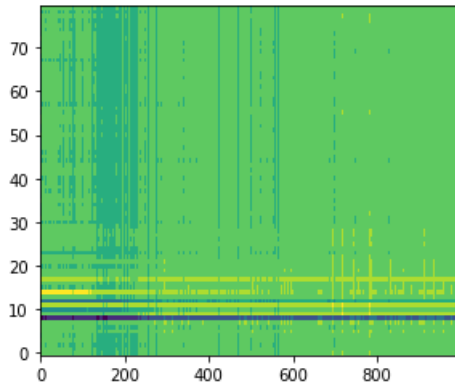


Figure 5.4: Mel spectrogram generated from NVIDIA's Tacotron 2 implementation for sentence (1).

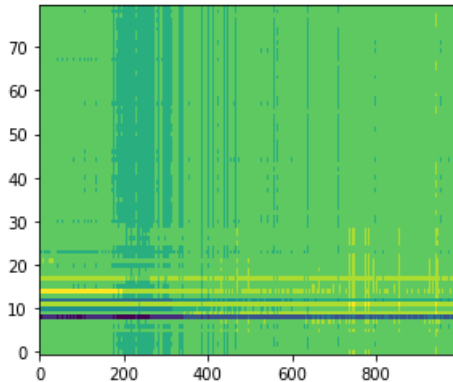


Figure 5.5: Mel spectrogram generated from NVIDIA's Tacotron 2 implementation for sentence (2).

The predominant green color in the mel spectrograms represents a weak strength of the signal. Meaning, it is basically zero.

5.3 Speech Synthesis

The speech synthesis chapter will be divided into 3 sub chapters. Each of these will describe the experiments conducted in each of the vocoders studied in this thesis

work. Linear-frequency spectrograms are produced from the final audio files generated by the synthesizers and compared to the ground truth.

5.3.1 Griffin-Lim synthesis

The Griffin-Lim synthetization was made with linear frequency spectrograms from the TensorFlow Tacotron implementation [57]. The post-processing net of the framework can produce the linear frequency spectrograms as well as mel scale spectrograms. Apart from the objective measurements taken, the linear frequency spectrograms from the audio generated give more detail about the synthesized audio. The linear frequency spectrograms comparison was made with MATLAB. The original and the synthesized audio from Sentence 1 can be seen below in Figure 5.6 and 5.7 respectively.

It is possible to notice that the Griffin-Lim algorithm was not capable of interpreting fully the mel-spectrogram generated. At the beginning of the audio, which is the left side of the figure, the Griffin-Lim had a better accuracy. At the right side of Figure 5.6 and 5.7, they both differ, meaning the speech produced had some mismatches.

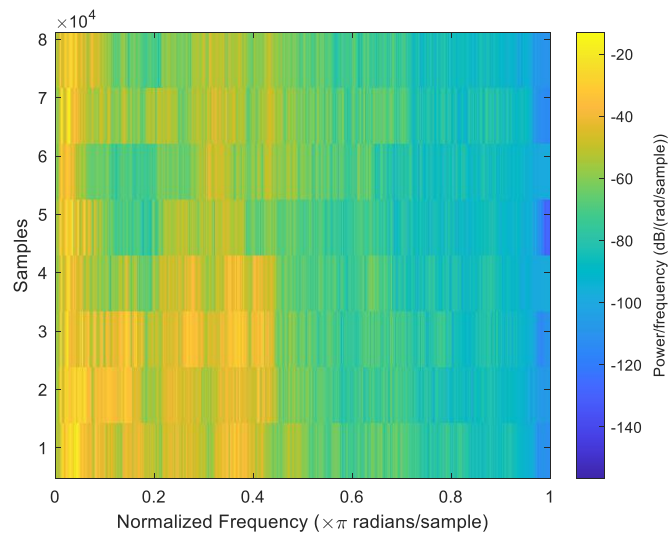


Figure 5.6: Sentence (1) ground truth audio file spectrogram.

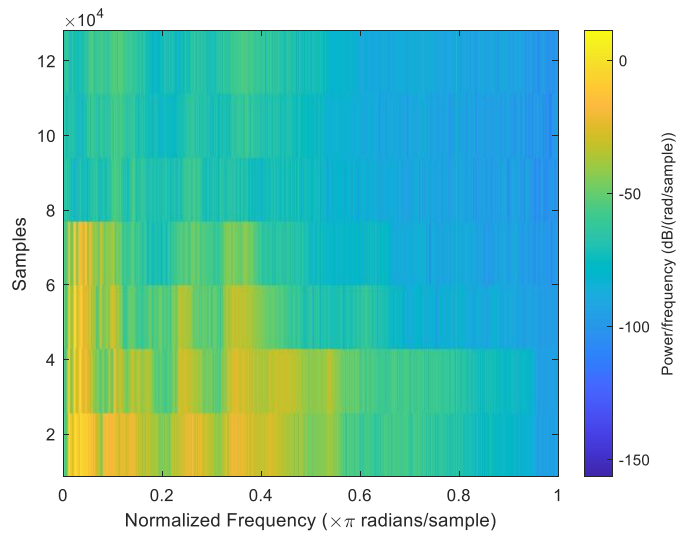


Figure 5.7: Sentence (1) Griffin-Lim generated audio file spectrogram.

5.3.2 WaveNet synthesis

The WaveNet synthesis was also made with the TensorFlow Tacotron 2 implementation [57]. The training consisted of 500k steps over 15 days. The learning rate of the model was changed during the training to find a better fit for the Brazilian Portuguese corpus. WaveNet displayed a noisier signal than the Griffin-Lim version of Sentence (1). In Figure 5.8, it is possible to see a lot more yellow in areas where the original version, seen in Figure 5.6, is blue.

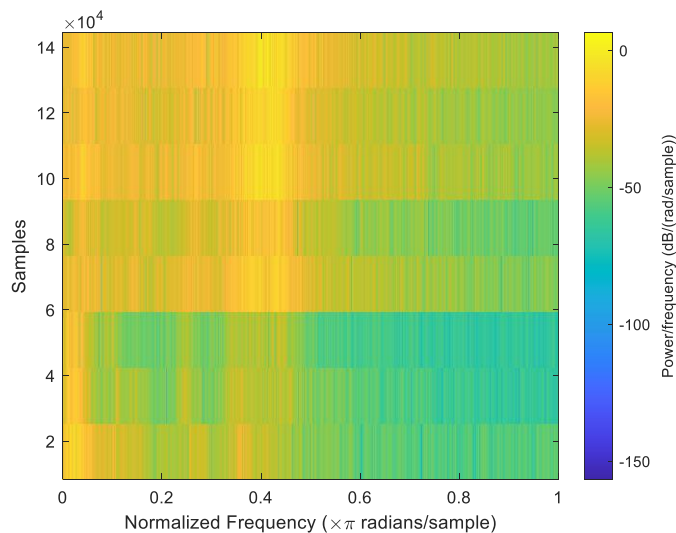


Figure 5.8 Sentence (1) WaveNet generated audio file spectrogram.

5.3.3 WaveGlow synthesis

WaveGlow experiments had a poor performance. The trained model was the NVIDIA Tacotron 2 PyTorch implementation [54]. This training was made until the 147k step but that is not equivalent to the full completion of the training. WaveGlow architecture was made to support distributed training, meaning the workload can be processed by several GPUs therefore speeding up the process. Due to the nature of its architecture, the training needed for the model to reach a good quality is very long. In the department's available server, NVIDIA's Tacotron was trained in 7 days with one GPU. The changed version had accents added to the symbols table, so letters such as “à”, “ã” and “â” could also be decoded and translated to the vectorized version of sentences.

Due to the bad quality mel spectrograms generated from the Tacotron 2 PyTorch version, the audio generated was blank. The vocoder output matched the bad quality mel spectrogram seen in Figure 5.4 and 5.5 above. The lack of training for Tacotron 2 affected directly the wave generated. We can observe that a good quality mel spectrogram generation can have a greater impact than a good quality synthesizer.

The results obtained can be explained by the difference in size of databases used in the English and Mandarin Chinese versions [9]. The authors used two different databases one containing 44h of recorded speech and another containing 24h in English. For training the network in Mandarin Chinese, the database contained 34.8h of audio files [9]. The first publicly available BR-PT database used in this thesis work is a starting point in the Portuguese TTS field but requires more data.

The smaller learning rate is something that directly affected the training of the models and therefore the output. Even though it was necessary to reduce the learning rate in order to run the training smoothly. Choosing a proper learning rate can be difficult. A too small learning rate may lead to slow convergence, while a too large learning rate can deter convergence and cause the loss function to fluctuate and get stuck in a local minimum or even to diverge [60]. In the case of the experiments conducted, it seems that WaveNet was not capable of reaching the desired output. The WaveNet training was finished in 497 thousand steps. It had the same issue with training of Tacotron 2, where the loss would be exploding. Due to the faced issues, the training lasted longer and had periods of inactivity. Additionally, because of the smaller learning rate, it can be concluded that WaveNet required more training to result in better generated speech.

The Griffin-Lim algorithm displayed a better performance. The training for the algorithm is much faster compared to the others. According to the implemented version [57], 60 iterations is enough for it to converge. Producing the speech based on linear-frequency mel-spectrograms seemed to work more efficiently. The results obtained in this work match the work done by Casanova in the creation of the PT-BR Portuguese database [6]. A different codebase for the Tacotron 2 was used, the Mozilla TTS [6] and produced speech with good quality.

5.4 Objective Measurements

Table 1 summarizes the objectives measurements that were analyzed. Each generated speech was compared to the original audio file to produce the scores. The ESTOI [59] metric introduced by Jensen and Taal are used to calculate the correlation between the temporal envelopes of natural and synthesized speech. Because of the bad performance of WaveGlow, it shows a small number, basically zero. WaveNet and Griffin-Lim algorithm display similar performance. For the fwSNRseg [61], WaveNet showed decent performance compared to the other, but the signals seem to have a lot of noise as it can be seen in Table 5.1.

Table 5.1 - Objective Measurements for produced speech with different synthesizers

Models	ESTOI	fwSNRseg
Griffin-Lim	0.0052925	0.20784
WaveNet	0.0057028	0.31812
WaveGlow	5.74E-06	-0.12038

Chapter 6 Conclusion

This Thesis work and its results are useful for future of BR-PT Text-To-Speech technology. I have analyzed and tested the first publicly available BR-PT speech dataset made for TTS systems. The data extracted from the trainings and generated mel spectrograms as well as the audio files will be important for future research developing a dedicated TTS architecture solely based on BR-PT.

The practical applications of the results are a shortcut to start developing a BR-PT dedicated TTS system. WaveNet [9] displayed a bright future to be investigated. The architecture showcased the best objective results among the synthesizers tested. With more time for training and a bigger BR-PT speech corpus, I believe there's a great chance to achieve state-of-the-art quality such as was achieved in their English and Chinese Mandarin version. Tacotron [12] combined with Griffin-Lim algorithm [11] is a good alternative for simpler implementations. The need to work only on Tacotron neural network may ease the progress towards a cost-efficient system. WaveGlow showcased that it requires way more time of training in order to display exciting results. Its architecture optimized for parallel training may affect its performance when working only on one GPU.

One of the challenges faced during the experiments was the amount of computational power needed. The server provided by the department was efficient in its work. But the number of experiments to be done combined with the complexity of the implementation on a new language required more time to train the models for longer. In future work, I intent to investigate further other architectures such as the HiFi-GAN [62], which is based on generative adversarial networks [63] and FastPitch [64] a fully parallel TTS model conditioned on fundamental frequency contours.

List of Figures

1.1	Image of mechanical vocoder the Voder by Homer W. Dudley.	7
2.1	General architecture of TTS system	12
2.2	Modern day vocoder. An instrument used to produce human speech electronically	14
3.1	Artificial Neural Network architecture.....	17
3.2	Destructuring of the image in the three primary colors. Each layer based on the primary color will be summed up to give the final pixel color.	18
3.3	Representation of color addition with the BME logo color.....	18
3.4	Architecture of Convolutional neural network. First layers being convolutional and pooling layers until the image is reduced to one dimension. The output of the convolutional layer is the input of a Feed Forward Artificial Neural Network.	19
3.5	Kernel Convolution of input image. The activation map is an average. The output of the operation is placed in a new image.....	20
3.6	Representation of Recurrent Neural Network. On the left, the rolled visual representation of the whole network. On the right the unrolled representation of individual layers. It is possible to visualize how the information is passed through layers	21
3.7	Architecture of Bidirectional Recurrent Neural Network. Forward states are represented by blue lines. Backward states are represented by red lines	22
3.8	General overview of LSTM cell.....	23
3.9	Cell state highlighted in the High-level diagram of a LSTM cell	23
3.10	Forget Gate highlighted. Red arrow represents the output of the Forget Gate.....	24
3.11	Input Gate highlighted. The i_t represents the decision to update or not the Cell State. The purple arrow represents c'_t which is the candidate to update the Cell state	24
3.12	Cell state update based on Forget Gate and Input Gate. New values to be updated are also used as input.....	25
3.13	Output Gate highlighted in light blue. The last gate that gives the parallel information for other Cell units as well as the output of the node itself.	25
4.1	Tacotron 2 architecture. It is divided in three sections according to traditional TTS systems models	29
4.2	Architecture of dilated causal Convolution network.....	31
4.3	WaveGlow architecture	32
5.1	Comparison between target and predicted mel spectrogram on training set made by Tacotron.....	34
5.2	Mel spectrogram generated from custom sentence (1).....	35
5.3	Mel spectrogram generated from custom sentence (2).....	35
5.4	Mel spectrogram generated from NVIDIA's Tacotron 2 implementation for sentence (1)	36
5.5	Mel spectrogram generated from NVIDIA's Tacotron 2 implementation for sentence (2)	36
5.6	Sentence (1) ground truth audio file spectrogram	37
5.7	Sentence (1) Griffin-Lim generated audio file spectrogram	38
5.8	Sentence (1) WaveNet generated audio file spectrogram	38

References

- [1] A. Purington, J. G. Taft, S. Sannon, N. N. Bazarova, and S. H. Taylor, “‘Alexa is my new BFF’: Social Roles, User Satisfaction, and Personification of the Amazon Echo,” in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, Denver Colorado USA, May 2017, pp. 2853–2859. doi: 10.1145/3027063.3053246.
- [2] T. Gruber, “Siri: A Virtual Personal Assistant,” p. 21.
- [3] P. Dempsey, “The Teardown: Google Home personal assistant,” *Engineering & Technology*, vol. 12, no. 3, pp. 80–81, Apr. 2017, doi: 10.1049/et.2017.0330.
- [4] H. Dudley, R. R. Riesz, and S. S. A. Watkins, “A synthetic speaker,” *Journal of the Franklin Institute*, vol. 227, no. 6, pp. 739–764, Jun. 1939, doi: 10.1016/S0016-0032(39)90816-1.
- [5] C. Trilnick, “Voder,” Jan. 21, 1938. <https://proyectoidis.org/voder/> (accessed Aug. 12, 2021).
- [6] E. Casanova *et al.*, “TTS-Portuguese Corpus: a corpus for speech synthesis in Brazilian Portuguese,” *arXiv:2005.05144 [cs, eess]*, Jun. 2021, Accessed: Nov. 08, 2021. [Online]. Available: <http://arxiv.org/abs/2005.05144>
- [7] B. S. A. e Castro, V. de O. Martins-Reis, A. C. Baptista, and L. C. Celeste, “Fluency profile: comparison between Brazilian and European Portuguese speakers,” *CoDAS*, vol. 26, no. 6, pp. 444–446, Dec. 2014, doi: 10.1590/2317-1782/20142014184.
- [8] J. Shen *et al.*, “Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions,” *arXiv:1712.05884 [cs]*, Feb. 2018, Accessed: Nov. 08, 2021. [Online]. Available: <http://arxiv.org/abs/1712.05884>
- [9] A. van den Oord *et al.*, “WaveNet: A Generative Model for Raw Audio,” *arXiv:1609.03499 [cs]*, Sep. 2016, Accessed: Nov. 08, 2021. [Online]. Available: <http://arxiv.org/abs/1609.03499>
- [10] R. Prenger, R. Valle, and B. Catanzaro, “WaveGlow: A Flow-based Generative Network for Speech Synthesis,” *arXiv:1811.00002 [cs, eess, stat]*, Oct. 2018, Accessed: Nov. 08, 2021. [Online]. Available: <http://arxiv.org/abs/1811.00002>
- [11] D. W. Griffin and J. S. Lim, “SIGNAL ESTIMATION FROM MODIFIED SHORT-TIME FOURIER TRANSFORM,” p. 4.
- [12] Y. Wang *et al.*, “Tacotron: Towards End-to-End Speech Synthesis,” *arXiv:1703.10135 [cs]*, Apr. 2017, Accessed: Nov. 18, 2021. [Online]. Available: <http://arxiv.org/abs/1703.10135>
- [13] S. O. Arik *et al.*, “Deep Voice: Real-time Neural Text-to-Speech,” *arXiv:1702.07825 [cs]*, Mar. 2017, Accessed: Dec. 09, 2021. [Online]. Available: <http://arxiv.org/abs/1702.07825>
- [14] S. Arik *et al.*, “Deep Voice 2: Multi-Speaker Neural Text-to-Speech,” *arXiv:1705.08947 [cs]*, Sep. 2017, Accessed: Dec. 09, 2021. [Online]. Available: <http://arxiv.org/abs/1705.08947>
- [15] Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King, “A study of speaker adaptation for DNN-based speech synthesis,” in *Interspeech 2015*, Sep. 2015, pp. 879–883. doi: 10.21437/Interspeech.2015-270.
- [16] S. Mehri *et al.*, “SampleRNN: An Unconditional End-to-End Neural Audio Generation Model,” *arXiv:1612.07837 [cs]*, Feb. 2017, Accessed: Nov. 18, 2021. [Online]. Available: <http://arxiv.org/abs/1612.07837>
- [17] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-Based Models for Speech Recognition,” *arXiv:1506.07503 [cs, stat]*, Jun. 2015, Accessed: Nov. 27, 2021. [Online]. Available: <http://arxiv.org/abs/1506.07503>

- [18] E. Hoogeboom, R. van den Berg, and M. Welling, “Emerging Convolutions for Generative Normalizing Flows,” *arXiv:1901.11137 [cs, stat]*, May 2019, Accessed: Dec. 09, 2021. [Online]. Available: <http://arxiv.org/abs/1901.11137>
- [19] R. Valle, K. Shih, R. Prenger, and B. Catanzaro, “Flowtron: an Autoregressive Flow-based Generative Network for Text-to-Speech Synthesis,” *arXiv:2005.05957 [cs, eess]*, Jul. 2020, Accessed: Dec. 09, 2021. [Online]. Available: <http://arxiv.org/abs/2005.05957>
- [20] C. Miao, S. Liang, M. Chen, J. Ma, S. Wang, and J. Xiao, “Flow-TTS: A Non-Autoregressive Network for Text to Speech Based on Flow,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020, pp. 7209–7213. doi: 10.1109/ICASSP40776.2020.9054484.
- [21] D. P. Kingma and P. Dhariwal, “Glow: Generative Flow with Invertible 1x1 Convolutions,” *arXiv:1807.03039 [cs, stat]*, Jul. 2018, Accessed: Nov. 27, 2021. [Online]. Available: <http://arxiv.org/abs/1807.03039>
- [22] H. G. Hirsch, K. Hellwig, and S. Dobler, “Speech Recognition at Multiple Sampling Rates,” p. 5, 2001.
- [23] L. Lévesque, “Nyquist sampling theorem: understanding the illusion of a spinning wheel captured with a video camera,” *Phys. Educ.*, vol. 49, no. 6, pp. 697–705, Nov. 2014, doi: 10.1088/0031-9120/49/6/697.
- [24] V. Vasilevski, “Phonologic Patterns of Brazilian Portuguese: a grapheme to phoneme converter based study,” p. 10.
- [25] M. Toman and M. Pucher, “An Open Source Speech Synthesis Frontend for HTS,” in *Text, Speech, and Dialogue*, vol. 9302, P. Král and V. Matoušek, Eds. Cham: Springer International Publishing, 2015, pp. 291–298. doi: 10.1007/978-3-319-24033-6_33.
- [26] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, “Speech Synthesis Based on Hidden Markov Models,” *Proc. IEEE*, vol. 101, no. 5, pp. 1234–1252, May 2013, doi: 10.1109/JPROC.2013.2251852.
- [27] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” *arXiv:1409.3215 [cs]*, Dec. 2014, Accessed: Nov. 15, 2021. [Online]. Available: <http://arxiv.org/abs/1409.3215>
- [28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *arXiv:1301.3781 [cs]*, Sep. 2013, Accessed: Nov. 18, 2021. [Online]. Available: <http://arxiv.org/abs/1301.3781>
- [29] J. Volkmann, S. S. Stevens, and E. B. Newman, “A Scale for the Measurement of the Psychological Magnitude Pitch,” p. 7.
- [30] “MicroFreak Vocoder Edition FLUENT IN CHAOS,” *Arturia*. <https://www.arturia.com/products/hardware-synths/microfreak-vocoder/overview> (accessed Aug. 12, 2021).
- [31] “Artificial-Intelligence.” <https://dictionary.cambridge.org/pt/dicionario/ingles/artificial-intelligence>
- [32] M. H. Sazli, “A brief review of feed-forward neural networks,” *Communications, Faculty Of Science, University of Ankara*, pp. 11–17, 2006, doi: 10.1501/0003168.
- [33] J. Feng and S. Lu, “Performance Analysis of Various Activation Functions in Artificial Neural Networks,” *J. Phys.: Conf. Ser.*, vol. 1237, no. 2, p. 022030, Jun. 2019, doi: 10.1088/1742-6596/1237/2/022030.
- [34] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, “Backpropagation: The Basic Theory,” p. 34.

- [35] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun. 2014, pp. 1701–1708. doi: 10.1109/CVPR.2014.220.
- [36] O. Abdel-Hamid, L. Deng, and D. Yu, “Exploring convolutional neural network structures and optimization techniques for speech recognition,” in *Interspeech 2013*, Aug. 2013, pp. 3366–3370. doi: 10.21437/Interspeech.2013-744.
- [37] N. A. Ibraheem, M. M. Hasan, R. Z. Khan, and P. K. Mishra, “Understanding Color Models: A Review,” vol. 2, no. 3, p. 12, 2012.
- [38] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” *arXiv:1511.08458 [cs]*, Dec. 2015, Accessed: Nov. 20, 2021. [Online]. Available: <http://arxiv.org/abs/1511.08458>
- [39] M. S. Al-Radhi, T. G. Csapó, and G. Németh, “Deep Recurrent Neural Networks in Speech Synthesis Using a Continuous Vocoder,” in *Speech and Computer*, vol. 10458, A. Karpov, R. Potapova, and I. Mporas, Eds. Cham: Springer International Publishing, 2017, pp. 282–291. doi: 10.1007/978-3-319-66429-3_27.
- [40] P. J. Werbos, “Generalization of backpropagation with application to a recurrent gas market model,” *Neural Networks*, vol. 1, no. 4, pp. 339–356, Jan. 1988, doi: 10.1016/0893-6080(88)90007-X.
- [41] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990, doi: 10.1109/5.58337.
- [42] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks,” *arXiv:1801.01078 [cs]*, Feb. 2018, Accessed: Nov. 21, 2021. [Online]. Available: <http://arxiv.org/abs/1801.01078>
- [43] J.-M. Valin, “A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement,” *arXiv:1709.08243 [cs, eess]*, May 2018, Accessed: Nov. 27, 2021. [Online]. Available: <http://arxiv.org/abs/1709.08243>
- [44] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Networks*, vol. 18, no. 5–6, pp. 602–610, Jul. 2005, doi: 10.1016/j.neunet.2005.06.042.
- [45] R. McMunn, “Key Differences Between Brazilian and European Portuguese,” *mondly*. <https://www.mondly.com/blog/2019/01/01/differences-brazilian-european-portuguese/> (accessed Nov. 21, 2021).
- [46] J. P. Teixeira, D. Freitas, D. Braga, M. J. Barros, and V. Latsch, “Phonetic Events from the Labeling the European Portuguese DataBase for Speech Synthesis, FEUP/IPB-DB,” p. 5, 2001.
- [47] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, “Exploring the Limits of Language Modeling,” *arXiv:1602.02410 [cs]*, Feb. 2016, Accessed: Nov. 18, 2021. [Online]. Available: <http://arxiv.org/abs/1602.02410>
- [48] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” p. 8.
- [49] 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.
- [50] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” *arXiv:1409.0473 [cs, stat]*, May 2016, Accessed: Nov. 27, 2021. [Online]. Available: <http://arxiv.org/abs/1409.0473>
- [51] S. Choi, S. Han, D. Kim, and S. Ha, “Attentron: Few-Shot Text-to-Speech Utilizing Attention-Based Variable-Length Embedding,” *arXiv:2005.08484 [cs]*,

- eess*], Aug. 2020, Accessed: Nov. 27, 2021. [Online]. Available: <http://arxiv.org/abs/2005.08484>
- [52] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel Recurrent Neural Networks,” *arXiv:1601.06759 [cs]*, Aug. 2016, Accessed: Nov. 28, 2021. [Online]. Available: <http://arxiv.org/abs/1601.06759>
- [53] J. Pons, S. Pascual, G. Cengarle, and J. Serrà, “Upsampling artifacts in neural audio synthesis,” *arXiv:2010.14356 [cs, eess]*, Feb. 2021, Accessed: Dec. 09, 2021. [Online]. Available: <http://arxiv.org/abs/2010.14356>
- [54] NVIDIA, “Tacotron 2 - PyTorch implementation with faster-than-realtime inference.” <https://github.com/NVIDIA/tacotron2>
- [55] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” *arXiv:1912.01703 [cs, stat]*, Dec. 2019, Accessed: Dec. 05, 2021. [Online]. Available: <http://arxiv.org/abs/1912.01703>
- [56] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” p. 21.
- [57] R. Mama, “Tensorflow implementation of DeepMind’s Tacotron 2.” <https://github.com/Rayhane-mamah/Tacotron-2>
- [58] M. S. H. Al-Radhi, “High-Quality Vocoding Design with Signal Processing for Speech Synthesis and Voice Conversion,” p. 123.
- [59] J. Jensen and C. H. Taal, “An Algorithm for Predicting the Intelligibility of Speech Masked by Modulated Noise Maskers,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 24, no. 11, pp. 2009–2022, Nov. 2016, doi: 10.1109/TASLP.2016.2585878.
- [60] Y. Wu *et al.*, “Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks,” *arXiv:1908.06477 [cs, stat]*, Oct. 2019, Accessed: Nov. 29, 2021. [Online]. Available: <http://arxiv.org/abs/1908.06477>
- [61] J. Ma, Y. Hu, and P. C. Loizou, “Objective measures for predicting speech intelligibility in noisy conditions based on new band-importance functions,” *J. Acoust. Soc. Am.*, vol. 125, no. 5, p. 3387, 2009, doi: 10.1121/1.3097493.
- [62] J. Kong, J. Kim, and J. Bae, “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis,” *arXiv:2010.05646 [cs, eess]*, Oct. 2020, Accessed: Dec. 09, 2021. [Online]. Available: <http://arxiv.org/abs/2010.05646>
- [63] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, “Generative Adversarial Networks: An Overview,” *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018, doi: 10.1109/MSP.2017.2765202.
- [64] A. Łańcucki, “FastPitch: Parallel Text-to-speech with Pitch Prediction,” *arXiv:2006.06873 [cs, eess]*, Feb. 2021, Accessed: Dec. 09, 2021. [Online]. Available: <http://arxiv.org/abs/2006.06873>