People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for Degree of

MASTER

In **Electrical and Electronic Engineerng**

Option: **Computer Engineering**

Title:

Deep Learning Methods for Speech Synthesis

Presented by: **Bechetella Abderraouf**

Supervisor: **Dr Tabet Youcef**

October 30, 2023

Contents

List of Abriviation

TTS Text to Speech

DSP Digital Signal Processing

RNN Recurrent Neural Network

RNN Convolutional Neural Network

SPSS Statstical Parametric Speech Synthesis

HMM Hidden Markov Model

HTK Hidden Markov Model Toolkit

HTS Hidden-Markov-model based text-to-speech

ASR Automatic Speech Recognition

MFCC Mel Frequency Cepstral coefficients

HSMM Hidden semi-Markov Model

ANN Artificial Neural Network

MSE Mean Squared error

SGD Stochastic gradient descent

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

STFT Short-Time Fourier Transform

DFT Discrete Fourier Transform

GAU Gated Activation Unit

SNR Signal to Noise Ratio

Dedication

I would like to dedicate this work to my extraordinary mother, whose unconditional love, countless sacrifices, and support have not only provided me with the strength to face any challenge that comes my way but have also been the guiding light throughout my entire academic journey. Your graceful prayers have been my source of inspiration and comfort. I am forever grateful for your presence in my life, as your faith in me has been the driving force behind my determination and the reason I stand at this point.

To my remarkable father, your support, opportunities, and belief in me have been the driving force behind my journey. Your guidance has provided me with the opportunities to continue this path with determination and passion. I dedicate this work to you, expressing my deepest gratitude for the invaluable role you have played in my life.

Abdeljalil , to my dear big brother who has always been by my side, your support, and companionship have been a constant source of strength and comfort throughout my life. You have been there to share both the joys and challenges that life brings.

Abdelali, Mohammed El-Amine, I am truly honored to have you by my side. I dedicate this project to you with profound love and gratitude. Wishing you nothing but the best as you embark on your journey.

To my dearest friends , Abdelwadoud , Aya , Wissem , your presence in my journey has filled it with cherished memories and unforgettable moments and I am forever grateful for the memories we have created together.

Finally, I want to dedicate this work to my beloved family. To my dear friends your support and love have touched my heart deeply. This dedication is a heartfelt expression of my deepest appreciation and love for each and every one of you.

Acknowledgemnt

In the name of Allah, the Most Merciful and the Most Gracious,

We would like to begin by expressing our gratitude to Allah for His blessings, guidance, and unwavering support throughout this research journey. His mercy and wisdom have been a source of strength and inspiration.

We would also like to extend our heartfelt appreciation to our supervisor, **Dr.Tabet**, for his invaluable guidance, mentor-ship, and continuous encouragement. His expertise and wisdom have been instrumental in shaping the direction of this project.

We would like to acknowledge and appreciate the teachers and staff of **The Institute of Electrical and Electronics Engineering** for their knowledge, guidance, and support throughout our academic journey. Their dedication and expertise have greatly contributed to our growth and learning.

Lastly, We would like to express our sincere gratitude to our family, friends, and loved ones for their unwavering belief in us, their love, and their constant support. Their encouragement and presence have been a source of motivation and strength.

Abstract

This master's project presents a comprehensive exploration of advanced techniques in the field of speech synthesis. The study focuses on two state-of-the-art neural waveform generation models: Gaussian WaveNet and Logistic WaveNet, comparing their performance and applications. Additionally, the project delves into the realm of Text-to-Speech (TTS) systems by investigating two variations of Tacotron 2—a pioneering end-to-end TTS model. The first Tacotron 2 model adheres to the original architecture, utilizing the high-quality WaveNet vocoder. The second model, a modified version, employs the Griffin-Lim vocoder as an alternative waveform generation method, addressing computational constraints. Furthermore, this project tries to adapt Tacotron 2 to Arabic, highlighting the challenges of working with non-Latin scripts. Results showcase the distinct strengths and limitations of each approach, providing valuable insights into the synthesis of natural and expressive speech across various languages and contexts, ultimately contributing to the broader field of speech technology.

CHAPTER 1

Introduction

Speech synthesis, also known as text-to-speech (TTS) synthesis, is the artificial production of human speech from written text. It involves generating artificial human-like speech from text input. Speech synthesis systems use various methods and technologies to create audible speech that can be easily understood by listeners [1]. Speech synthesis systems first convert the input text into its corresponding linguistic or phonetic representations and then produce the sounds corresponding to those representations. With the input being a plain text, the generated phonetic representations need to be augmented with information about the intonation and rhythm that the synthesized speech should have, this task is done by a Natural Language Processing (text analysis) module in most speech synthesizers. The phonetic transcription and prosodic information obtained from the text analysis module, is then given to a Digital Signal Processing (DSP) module that produces synthetic speech [2].

Deep Learning is a special part of Computer Learning, like teaching computers to be really smart by using something called neural networks with many layers. It's great in the world of Artificial Intelligence because it helps computers learn and make good decisions, much like the functioning of our brains.It has a wide range of applications, like making characters in animations move realistically and making computers talk to us in a way that sounds like a real person. It's super handy because it helps computers understand and do all sorts of stuff really well [3].

Neural Network-based Speech Synthesis, uses deep learning architectures to model complex relationships between linguistic and acoustic features in speech. By analyzing

input text and extracting linguistic nuances, these systems predict prosody, including pitch, rhythm, and intonation, generating remarkably natural speech. Acoustic models like WaveNet, alongside advanced synthesis techniques like Tacotron, Tacotron 2, FastSpeech and DeepVoice, generate high-quality audio waveforms, making them indispensable in applications such as virtual assistants, audio-books, and accessibility tools [4][5][6][7]. Deep learning, particularly with Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), excels in this context, often utilizing mel spectrograms to capture essential acoustic characteristics and enhance speech quality. Despite these advancements, modeling raw audio signals remains challenging due to complex sample correlations [4].

This thesis addresses the challenge of improving and adapting speech synthesis technology for the Arabic language. It focuses on leveraging deep learning methods and neural networks to enhance the quality of Arabic text-to-speech (TTS) systems. By comparing various models like Tacotron and Wavenet and adapting them for Arabic, the research aims to contribute to more natural and understandable synthetic speech in Arabic, addressing the unique phonological characteristics of the language and advancing Arabic language processing and accessibility.

Our project titled "Deep Learning Methods for Speech Synthesis", follows a well-structured organization comprising five chapters. Starting with an introduction as the first chapter. Chapter 2 provides an introduction to traditional methods for speech synthesis, offering historical context. In Chapter 3, a background on deep learning methods is presented, laying the groundwork for subsequent chapters. Chapter 4 introduces the deep learning methods chosen for experimentation. Chapter 5 constitutes the core of the project, focusing on the implementation of three crucial experiments: comparing Gaussian WaveNet and Logistic WaveNet, evaluating two versions of Tacotron 2, and implementing an Arabic Text-to-Speech system based on Tacotron 2. Finally, the conclusion synthesizes the project's findings, offering a cohesive summary and reflecting on the broader implications of the research in the realm of speech synthesis and deep learning.

CHAPTER 2

Traditional Methods for Speech Synthesis

Traditional methods for speech synthesis encompass a range of techniques predating deep learning. These methods, including concatenative, formant, and articulatory synthesis, were instrumental in early speech technology development. While they have been largely surpassed by deep learning, their historical importance and foundational principles continue to influence modern speech synthesis research.

2.1 Formant synthesis

Formant synthesis is a technique used in speech synthesis to generate artificial speech by precisely controlling the resonant frequencies, known as formants, in the human vocal tract. These formants are crucial in shaping the distinct qualities of different speech sounds, including vowels and consonants. In formant synthesis, mathematical models and algorithms are employed to recreate these formants and their variations, allowing for the generation of intelligible speech. While formant synthesis provides a high degree of control over the articulation and quality of synthetic speech, it may require careful tuning and algorithmic sophistication to achieve naturalness in prosody and voice quality[8]. This method has found applications in early text-to-speech systems, voice response systems, and assistive technology for speech generation, contributing to the development of synthetic voices for various purposes[9].

2.2 Articulatory Synthesis

Articulatory synthesis, in its essence, offers a compelling approach to generate speech by directly modeling the intricate behaviors of human articulators, making it theoretically the most satisfying method for achieving high-quality speech synthesis. However, it also presents some of the most formidable implementation challenges in practice. The articulatory control parameters encompass critical aspects such as lip aperture, lip protrusion, tongue tip position, tongue tip height, tongue position, and tongue height [10]. Two major hurdles confront articulatory synthesis. Firstly, acquiring the necessary data for articulatory modeling poses a substantial challenge, often relying on X-ray photography. Unfortunately, X-ray data alone does not provide comprehensive information about the masses or degrees of freedom of the articulators [9]. Secondly, striking the right balance between an intricately accurate model and one that remains manageable in terms of design and control proves to be another significant challenge. Generally, the results achieved through articulatory synthesis do not consistently match the quality attained in formant synthesis or concatenative synthesis, emphasizing the intricate nature of this method.

2.3 Concatenative Synthesis

The primary challenge faced by both formant synthesis and articulatory synthesis lies not so much in the generation of speech from parametric representations, but rather in the complex task of deriving these essential parameters from the input specifications initially created through text analysis. To address this limitation, concatenative synthesis adopts a data-driven approach. This synthesis method generates speech by seamlessly connecting pre-recorded, natural speech units. These units can vary in granularity, encompassing words, syllables, half-syllables, phonemes, diphones, or even triphones, each with its unique impact on the synthesized speech's quality. The choice of unit length plays a pivotal role: longer units enhance naturalness while requiring more memory and resulting in a voluminous database. Conversely, shorter units reduce memory requirements but introduce complexities in sample collection and labeling [11]. Among the widely employed units in concatenative synthesis, diphones stand out.

Diphones are speech segments that begin in the middle of one phoneme and extend to the middle of the following one. Their distinct advantage lies in their ability to model coarticulation by incorporating the transition to the next phoneme within the diphone itself. The comprehensive list of diphones is termed the ”diphone inventory.” Building this inventory involves recording natural speech in a manner that encompasses all possible phonemes within various contexts (allophones). Subsequently, diligent labeling and segmentation efforts are undertaken to identify individual diphones. Once the diphone inventory is established, adjustments to pitch and duration are necessary to align with the prosodic aspects specified in the input. This meticulous process enables concatenative synthesis to produce speech that effectively captures the nuances of natural speech, while also offering flexibility in adapting to diverse prosodic requirements.

2.4 Unit Selection Synthesis

In the realm of concatenative synthesis, the modification of diphones to achieve the desired prosody can introduce artifacts that undermine the naturalness of synthesized speech. To address this issue, unit selection synthesis, also known as corpus-based concatenative synthesis, offers a solution by populating the unit inventory with multiple instances of each unit, each exhibiting varying prosodic characteristics [11]. This strategic approach ensures that the selected unit closely aligns with the target prosody, reducing or even eliminating the need for substantial prosodic modifications. Given the multiple instances stored in the inventory, the selection process relies on a unit selection algorithm designed to minimize two distinct cost functions: the target cost and the join cost. In the context of automatic unit selection, the scope of coarticulatory influence extends beyond solely the last phoneme. The database employed in this approach is substantially larger, typically spanning 1 to 10 hours, and encompasses numerous occurrences of each acoustic unit, capturing a wide array of contextual factors such as neighboring phonemes, pitch, duration, and syllable position. Consequently, when synthesizing a sequence of phonemes, a lattice of acoustic units is formed, with the selected units aiming to match expected contexts while minimizing spectral and prosodic discontinuities. This approach notably requires fewer modifications of the

speech units, yielding synthesized speech characterized by a significantly more natural quality than diphone-based synthesis. However, unit selection techniques come with certain drawbacks. They hinge on extensive databases, necessitating considerable time and cost for data collection and labeling, as well as substantial memory resources for data storage. Furthermore, they are susceptible to incorrect labeling and the occurrence of unseen target contexts, resulting in segments of synthesized speech of notably low quality. This challenge of handling unseen contexts may persist as a limitation of concatenative synthesis, as rare linguistic events are inherently part of language dynamics, as suggested by [12].

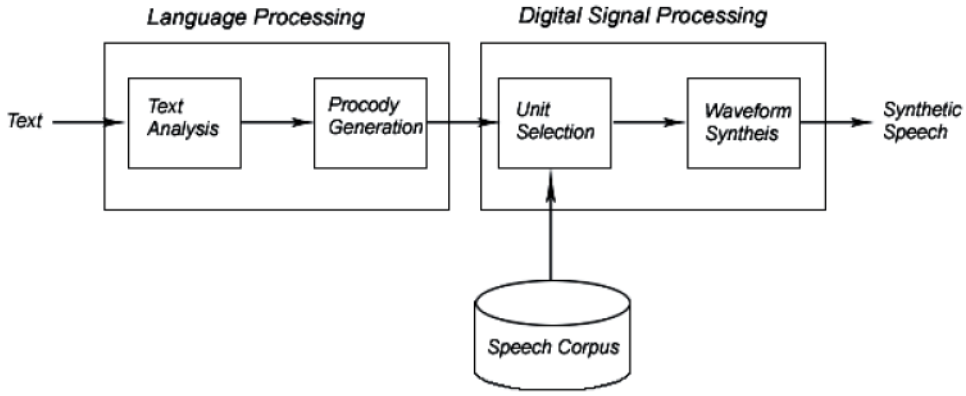


Figure 2.1: Block Diagram of Unit Selection TTS System [13]

2.5 Statistical parametric speech synthesis

Statistical parametric speech synthesis (SPSS) represents a significant milestone in the field of text-to-speech technology. It harnesses the power of statistical modeling to bridge the gap between linguistic information and the acoustic properties of natural speech and offers a wide range of techniques to improve spoken output [14]. By analyzing vast datasets of recorded human speech and extracting linguistic and acoustic features, SPSS systems are capable of generating remarkably natural and intelligible synthetic speech. These systems have found extensive applications in voice assistants, navigation systems, audiobook narration, and accessibility tools for individuals with speech impairments. SPSS not only enables the customization of voice characteristics but also allows for the expression of emotions and speaking styles, making it an indispensable technology in the realm of human-computer interaction and digital voice

communication. As research in the field continues to advance, SPSS promises even more realistic and expressive synthetic speech, pushing the boundaries of naturalness in synthetic voices.

The historical foundation of statistical parametric speech synthesis can be traced back to the success of Hidden Markov Models (HMMs) in automatic speech recognition [15]. Although the HMM may not be a perfect representation of speech, its effectiveness stems from the availability of robust and efficient learning algorithms like Expectation-Maximization, automatic methods to manage model complexity through parameter tying, and computationally efficient search algorithms such as Viterbi search. The performance of this model, measured by criteria like word error rates in speech recognition and perceptual quality in speech synthesis, heavily relies on selecting an appropriate configuration.

2.5.1 Hidden Markov model:

Hidden Markov Models (HMMs) are versatile models originally developed for speech recognition [15] but now widely used in various fields. One of the popular tools for working with HMMs in speech recognition is the HTK speech recognition toolkit of Young et al [16] . HTK is not only extensively used in research labs but also serves as the foundation for some highly successful Automatic Speech Recognition (ASR) engines in recent years. Additionally, HTK has given rise to Hidden Markov Model Toolkit (HTS) [17].

HMMs provide a framework for describing both observable events, such as words in speech input, and hidden events, such as underlying factors like part-of-speech tags, which play a causal role in our probabilistic model. HMMs essentially combine two stochastic processes. Firstly, there's a Markov process governing state transitions, where transitions are determined by probabilities. Secondly, there's another stochastic process responsible for emitting symbols associated with each state.

Consider the representation of a left-to-right HMM shown in Figure 2.2. In this configuration, there's an initial state on the left, from which transitions can occur either to the same state or to the next one on the right. Importantly, transitions do not occur in the reverse direction.

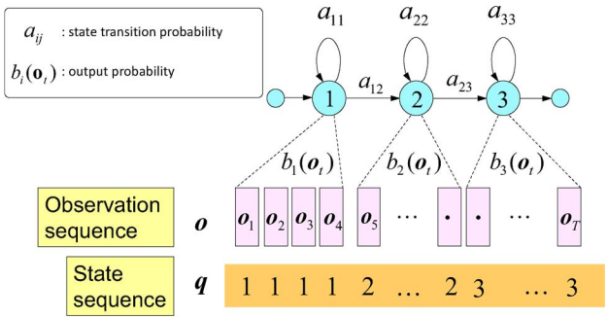


Figure 2.2: Left to right example of an HMM with three states [18]

2.5.2 HMM- Based speech synthesis:

In unit selection synthesis, multiple instances of each phone in different contexts are stored in the database. To build such a database is a time-consuming task and the database size increases enormously. Another limitation of the concatenative approach is that it limits us to recreate what we have recorded [18]. An alternative is to use statistical parametric synthesis techniques to infer specification for parametric mapping from data. These techniques have two advantages: firstly, less memory is needed to store the model parameters than to store the data itself. Secondly, more variations are possible for example; the original voice can be converted into another voice. One of the most usable statistical parametric synthesis techniques is the hidden Markov model (HMM) synthesis. It consists of two main phases, the training phase and the synthesis phase. At the training phase, it should be decided which features the models should be trained for. Mel frequency cepstral coefficients (MFCC) and their first and second derivatives are the most common types of features used. The features are extracted per frame and put in a feature vector. The Baum-Welch algorithm is used with the feature vectors to produce models for each phone. A model usually consists of three states that represent the beginning, the middle and the end of the phone. The synthesis phase consists of two steps: firstly, the feature vectors for a given phone sequence have to be estimated. Secondly, a filter is implemented to transform those feature vectors into audio signals. The quality of the HMM generated speech is not as good as the quality of the speech

generated from unit selection synthesis. The modeling accuracy can be improved by using hidden semi-Markov models (HSMMs)[19], trajectory HMMs [20], and stochastic Markov graphs [21].

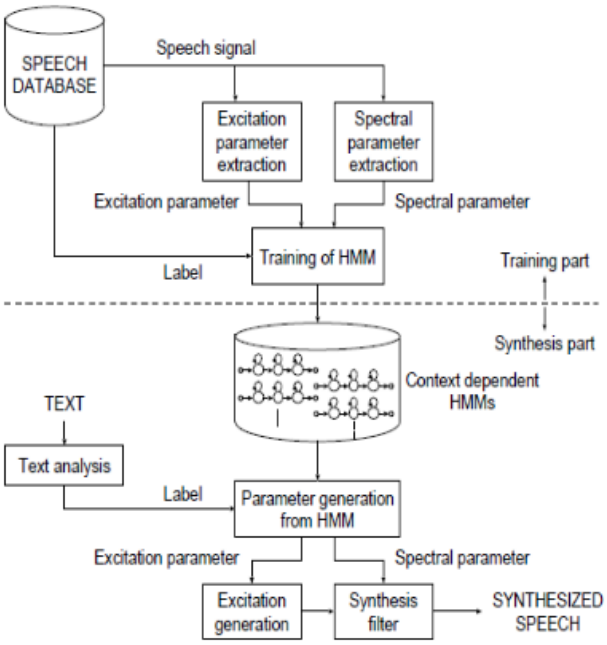


Figure 2.3: Typical architecture of HMM based speech synthesis system [22]

CHAPTER 3

Background on Deep Learning Methods

Deep learning, a subfield of machine learning, has ushered in a revolution in artificial intelligence. It harnesses the power of artificial neural networks to tackle intricate tasks through data-driven learning and adaptation. With its capacity to process extensive datasets, deep learning has been a driving force behind advancements in areas like image recognition, natural language understanding, and speech synthesis. This introduction highlights the transformative role of deep learning in modern AI applications.

3.1 Neural Networks (ANN)

A neural network is a computational model inspired by the human brain's structure and functioning. It consists of interconnected nodes or neurons organized into layers. These networks are used in machine learning to process data, recognize patterns, and make predictions. The information flows through the network, and during training, the connections between neurons are adjusted to minimize errors, allowing the network to learn and generalize from data. Neural networks have found applications in diverse domains, including healthcare, finance, and robotics, and have played a significant role in advancing artificial intelligence[23].

3.1.1 Neuron structure and functionality

The structure of an Artificial Neural Network (ANN) serves as its architectural foundation, describing how data is processed and knowledge is acquired[23]. At its core, ANNs encompass three fundamental components: neurons, layers, and connections.

- Neurons: Neurons in neural networks process input signals with assigned weights and bias terms. They use activation functions to introduce non-linearity, producing outputs that enable the network to perform various tasks in machine learning.

- Layers: In neural networks, layers are essential components that organize data processing. These networks consist of three primary types of layers: the Input Layer, Hidden Layers, and the Output Layer. The Input Layer receives external data attributes, while Hidden Layers perform complex computations and pattern extraction using activation functions. The Output Layer generates the network's final predictions tailored to the specific task.

- Connections: represented as synapses, facilitate the flow of information within ANNs. Each connection between neurons carries a weight that signifies its strength or importance in transmitting information. Additionally, bias terms are incorporated to introduce flexibility into the model.

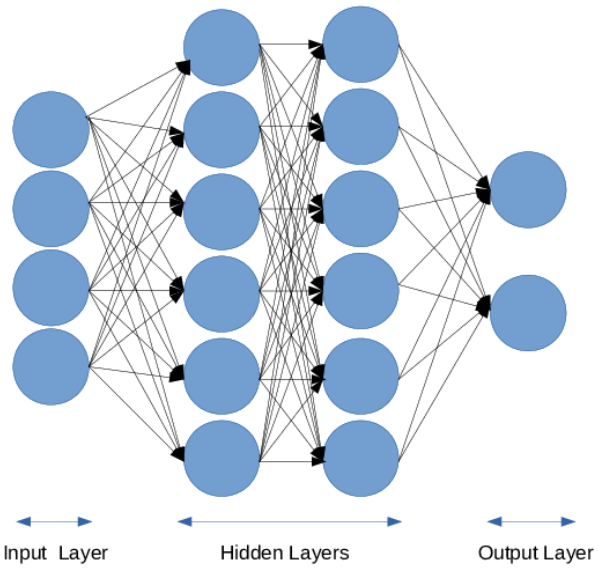


Figure 3.1: A typical ANN structure[24]

3.1.2 Activation functions

Activation functions are essential mathematical transformations applied to the output of individual neurons in neural networks. Their primary role is to introduce non-linear properties to the network, allowing it to effectively capture intricate patterns and handle various tasks[25].Activation functions determine whether a neuron should be activated (i.e., transmit information to the next layer) or not, based on a threshlod.

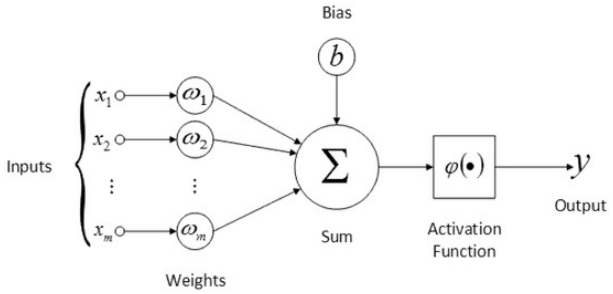


Figure 3.2: Schematic representation of artificial neural network[26]

Sigmoid Function (σ)

The sigmoid activation function is historically used for binary classification tasks, converting inputs into probabilities. However, it suffers from the vanishing gradient problem as input values move away from zero, leading to slow convergence and difficulty in training deep neural networks[27]. The sigmoid activation function is mathematically represented as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

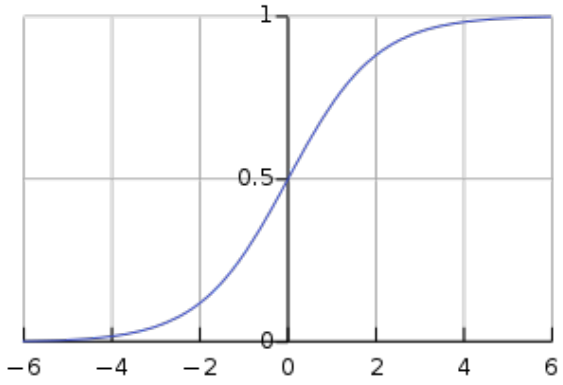


Figure 3.3: Sigmoid Activation Function Graph.

Hyperbolic Tangent

The tanh activation function maps input values to a range between -1 and 1, introducing non-linearity in neural networks. It's useful for tasks like binary classification, where values near -1 represent one class, values near 1 represent another class, and values near 0 indicate an intermediate state[28].

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.2}$$

Rectified Linear Unit (ReLU)

ReLU (Rectified Linear Unit) is a widely used activation function in modern neural networks for its computational efficiency and ability to address the vanishing gradient problem, which was a challenge with previous activation functions like sigmoid or tanh. ReLU's simplicity accelerates computation and speeds up training by allowing gradients to flow freely through positive values[29]. In mathematical terms, the ReLU activation function can be defined as:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

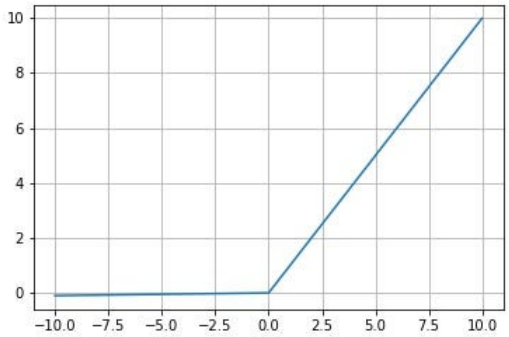


Figure 3.4: Rectified Linear Unit Graph .

Softmax function

The Softmax activation function is a commonly employed technique in the output layer of neural networks for tasks involving multi-class classification. Its purpose is to process a real-number vector and convert it into a probability distribution[30]. In

essence, the softmax function ensures that the resulting output values are non-negative and collectively sum to unity. This fundamental characteristic allows the model to generate probabilistic predictions for multi-class classification endeavors[31]. From a mathematical perspective, the softmax activation of an input vector x, which possesses a dimensionality of n, can be expressed as follows:

$$\text{softmax}(x[i]) = \frac{e^{x[i]}}{\sum_{j=1}^{n} e^{x[j]}}, \quad \text{for } i = 1 \text{ to } n, \tag{3.4}$$

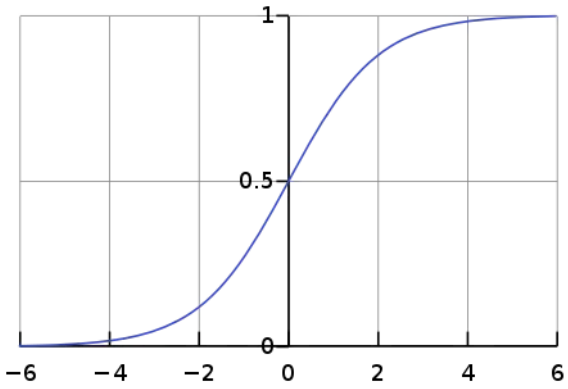


Figure 3.5: Softmax Function Graph.

3.1.3 Training of Neural Network

Training a neural network is the process of teaching it to learn patterns and make predictions from data[32]. This involves fine-tuning the network's internal parameters, such as weights and biases, using a dataset with known outcomes. Through iterative adjustments, the network aims to minimize the difference between its predictions and the actual data.Here's a simplified overview of the neural network training process:

1. Data Preparation: This involves gathering a dataset that includes input data and corresponding target outputs. The collected data is then cleaned and prepared, which may involve tasks like normalization and scaling.

2. Initialization: Initialize the neural network's parameters, such as weights and biases. Common initialization methods include random initialization

3. Forward Propagation: Pass the training data through the neural network in the forward direction. The input data is transformed as it propagates through the layers, ultimately generating predictions.

4. Loss Calculation: Compute a loss function that quantifies the difference between the network's predictions and the actual target values. Common loss functions include mean squared error (MSE) for regression and cross-entropy for classification.

5. Backpropagation: Calculate gradients of the loss with respect to the network's parameters using the chain rule of calculus. These gradients indicate how much each parameter should be adjusted to minimize the loss.

6. Gradient Descent: Update the network's parameters in the opposite direction of the computed gradients to minimize the loss. The learning rate determines the step size for each parameter update. Popular optimization algorithms include stochastic gradient descent (SGD), Adam, and RMSprop[33].

7. Epochs: Repeat steps 3 to 6 for multiple epochs (training cycles) to allow the network to learn from the data. With each epoch, the network refines its parameter estimates and improves its performance.

3.2 Recurrent Neural Networks (RNNs)

Recurrent neural networks have been an important focus of research and development during the 1990s. They are designed to learn sequential or time-varying patterns[34],making them invaluable in fields like speech synthesis and time series analysis. They stand out for their unique ability to retain and utilize information from past time steps, enabling them to predict future outputs based on historical context.

However, RNNs have a significant limitation—they can struggle to capture long-range dependencies in sequences due to the vanishing gradient problem. This limitation arises when gradients diminish exponentially as they propagate backward through time, hindering the model's ability to effectively learn from distant past inputs. Despite this challenge, RNNs remain a fundamental tool in sequential data analysis, including speech synthesis.

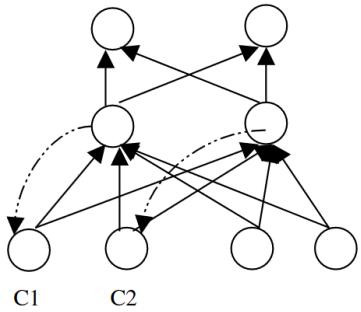


Figure 3.6: An example of a simple recurrent network[34]

3.2.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks were developed to address the shortcomings of traditional Recurrent Neural Networks (RNNs) in learning long-term dependencies. They introduce extended memory to RNNs, composed of linear and logistic units with multiplicative interactions, enabling them to retain input information over extended periods—a significant enhancement compared to standard RNNs.

LSTM memory cells, equipped with gated decision-making, include three pivotal gates: the input gate, forget gate, and output gate. These gates autonomously learn the importance of information over time, leveraging learned weights to prioritize its significance. Based on these priorities, the gates decide whether to store or discard

information, empowering LSTMs to efficiently manage memory and excel in learning and modeling long-term dependencies[35].

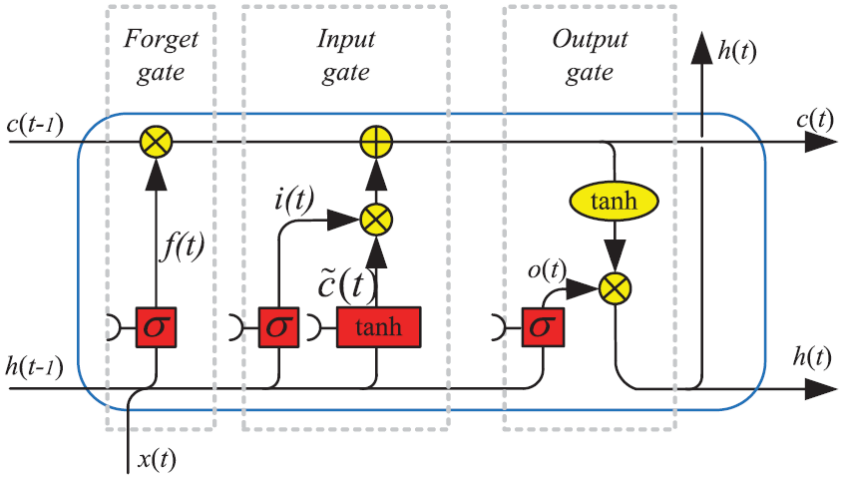


Figure 3.7: Architecture of LSTM [35]

3.2.2 Bidirectional LSTM (Bi-LSTM)

Bidirectional LSTM (Bi-LSTM) is a specialized variant of Long Short-Term Memory (LSTM) networks that consists of two LSTM layers, one processing input in the forward direction (from left to right), and the other in the reverse direction (from right to left). The outputs of these two layers are then passed through the activation function. This bidirectional approach is particularly advantageous for models requiring consideration of both past and future data dependencies. Unlike unidirectional LSTMs, which only take input from the past, Bi-LSTM effectively addresses this limitation by incorporating information from both directions, allowing it to handle future dependencies by leveraging knowledge from both past and future contexts.

3.2.3 The Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) was proposed by [36].IT is a modified version of the LSTM, but it's simpler. Instead of three gates, it has just two. The first one called the update gate: decides how much of the old information to keep. The other gate called the reset gate,wich determines how to blend the old information with the new input. Unlike LSTMs, GRU cells don't have an output gate, so they share their entire state with the network at each step. This simpler GRU design helps with a common problem in LSTM training called the "exploding gradient problem," which can make models unstable and affect their weights. It also speeds up the network, although LSTMs often produce more detailed results, albeit more slowly.

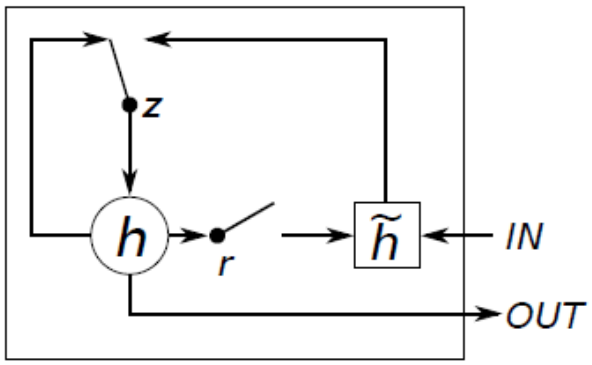


Figure 3.8: Gated Recurrent Unit[37].

3.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs), sometimes called ConvNets, are specialized computer programs that are really good at looking at pictures and videos. They're like computerized detectives that can find important things in visual information, much like how our eyes and brain work together. These networks are made up of special layers that help them process images and videos. The cool thing about CNNs is that they can learn to recognize different parts of pictures, slowly figuring out complex stuff while keeping everything in the right place.

The key to CNNs is the convolutional Layers and Filters. These are like tools that help them find important details in the pictures and videos they're looking at. They work a bit like filters in photography, highlighting what's important and ignoring what's

not. The great part is that these tools can learn and get better at their job over time. In this exploration, we'll dive into these tools, known as Convolution filters, which help extract specific details from images. Think of them as special lenses for the CNNs, each having its own unique way of looking at things, like blurring or sharpening, to help us understand the math and computer part of this research [38].

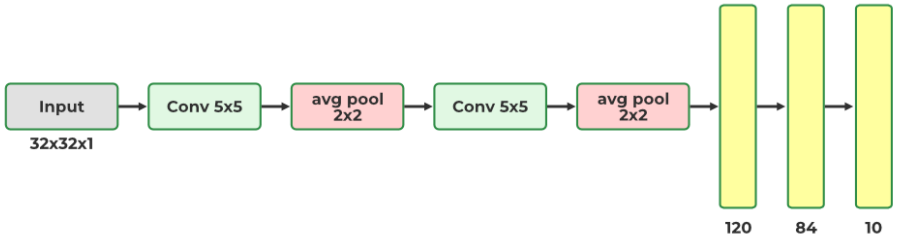


Figure 3.9: Basic Architecture of LeNET-5 most widely known CNN architecture[39].

3.3.1 Pooling layers

In Convolutional Neural Networks (CNNs), the pooling layer is a frequently employed layer that typically follows convolutional layers. Its primary purpose is to decrease the spatial dimensions (namely, the width and height) of the feature maps, all while retaining the depth (i.e., the number of channels). The pooling layer operates by dividing the input feature map into discrete, non-overlapping regions known as pooling regions. Each of these regions is then processed to produce a single output value, which serves as a representation of the features within that region[40].

3.3.2 1D Convolutional Neural Networks

CNN which are designed for images and two-dimensional data are called 2-D CNNs are the most common.On the other hand there is 1D convolutional neural networks (CNNs) are specialized neural networks used to process one-dimensional data, such as time series, audio signals, or text sequences.1D CNNs operate on sequences where data points are arranged linearly. These networks use convolutional layers and filters to

automatically detect relevant patterns and features within the sequential data. They are particularly useful in tasks like speech recognition, natural language processing, and analyzing time-dependent information. The application of 1D CNNs has significantly improved the performance of various machine learning models in these domains by effectively capturing sequential dependencies and extracting meaningful information from the data[41].

3.3.3 Residual Network

Residual Network (ResNet),commonly known as ResNet, is a type of deep neural network architecture that's specifically designed to tackle the challenges of training very deep neural networks. The key innovation in ResNet is the use of residual blocks, which enable the network to learn residual functions. These residual blocks contain shortcut connections, also called skip connections, that allow the network to skip one or more layers during training The concept behind skip connections is to add the input signal, denoted as x, to the output of a layer situated higher up in the network stack. This addition operation transforms the network's objective from directly modeling the target function h(x) to capturing the residual function f(x) = h(x) - x. This approach is known as residual learning and plays a vital role in improving the training and convergence of deep neural networks like ResNet .

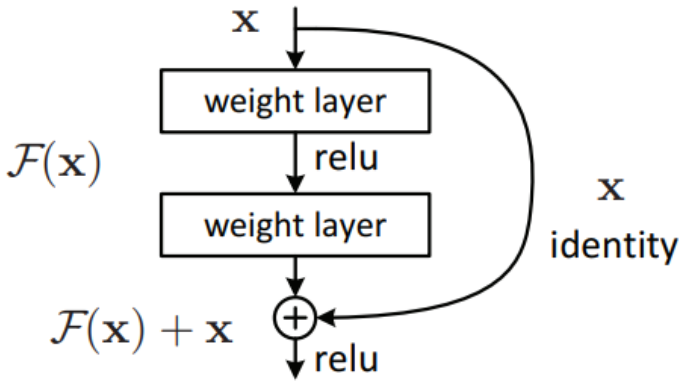


Figure 3.10: Residual learning: a building block[42].

When initializing a conventional neural network, its weights typically start close to zero, causing the network to produce outputs that are also close to zero. However, when we introduce skip connections, the resulting network begins to produce outputs

that resemble a copy of its inputs, essentially modeling what's known as the identity function. This becomes advantageous when the target function closely resembles the identity function, as it significantly expedites the training process. Moreover, if multiple skip connections are incorporated, the network can show progress even when certain layers have not yet started learning[38].

3.3.4 Dilated Causal Convolution

Dilated convolution, also known as "a trous in French" convolution, is a clever technique applied in convolutional neural networks (CNNs) to increase the receptive field of a filter without introducing more parameters. The core idea involves introducing gaps or holes between the values of a convolutional filter, allowing it to effectively skip pixels during the operation. This expansion of the receptive field is essential for maintaining causality in sequential data analysis, ensuring that the network considers information from past time steps. Instead of altering the filter size or stride, dilated convolution achieves this by adjusting the dilation rate, which governs the size of the gaps between filter values. Importantly, this technique doesn't increase the number of parameters, as the filter size remains the same, only operating with these gaps. By introducing an additional parameter (dilation factor), represented as d, the input is expanded, skipping $d-1$ pixels in the kernel. This approach allows dilated convolution to capture more contextual information while preserving causality. Unlike pooling methods, dilated convolution enhances the coverage of the input data during convolution operations, effectively widening the field of view without significantly increasing computational complexity [43].

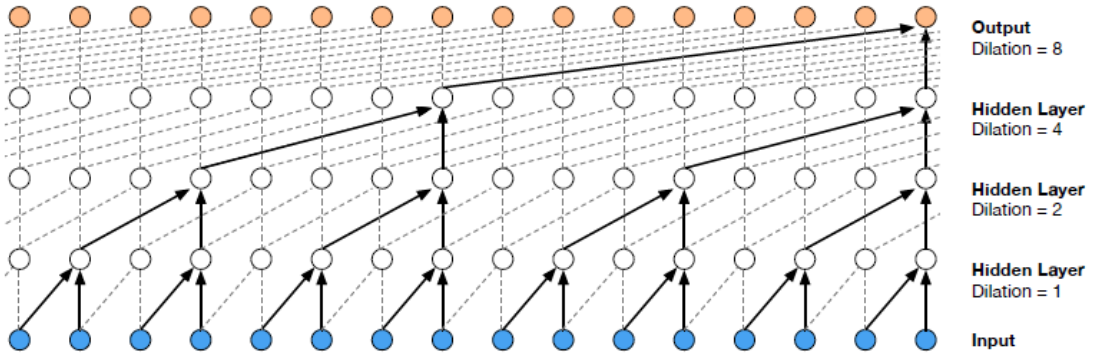


Figure 3.11: Visualization of a stack of dilated causal convolutional layers [4].

CHAPTER 4

Deep Learning Methods selected for experiments

The selected deep learning methods for experiments constitute a vital component of this study's research framework. These methods, including Wavenet, Tacotron, and their variants, have been chosen for their capabilities in advanced speech synthesis and natural language processing. This section provides an in-depth exploration of these cutting-edge techniques and their applications in our research context.

4.1 Spectrograms

Spectrograms, which are time-frequency representations of audio signals, form a cornerstone of speech synthesis, providing invaluable insights into the spectral content of sound over time. These visual and mathematical representations serve as a fundamental tools for understanding the nuances of speech and play a pivotal role in various aspects of speech synthesis. Spectrograms, denoted as $S(t, f)$, are generated through the Short-Time Fourier Transform (STFT), dividing an audio signal $x(t)$ into short time frames and quantifying the magnitude of frequency components for each frame. They offer a time-frequency trade-off, where shorter time frames provide finer temporal details, making them essential for feature extraction and prosody modeling in the synthesis of natural-sounding speech[44].

- Generation: Spectrograms are computed by segmenting the audio signal into short time frames using a window function $w(t)$, and then applying the STFT, which is essentially a continuous Fourier Transform with respect to time.

$$X(t, f) = \int_{-\infty}^{\infty} x(\tau) w(\tau - t) e^{-j2\pi f \tau} d\tau \tag{4.1}$$

- Role in Speech Synthesis: Spectrograms serve as essential tools in speech synthesis, performing two critical roles. Firstly, they excel in feature extraction by capturing the dynamic distribution of energy across diverse frequency bands over time. This results in the creation of the spectrogram matrix $S(t, f)$, a pivotal feature representation employed in speech synthesis models. Secondly, spectrograms are indispensable for prosody modeling, offering valuable insights into pitch, formants, and prosody variations throughout the speech signal's duration. This information plays a vital role in generating speech that sounds natural, characterized by the appropriate rhythm and intonation for effective communication.

4.2 Mel Spectrograms

The Mel scale, denoted as $M(f)$, offers a mathematical approximation that closely mirrors the nonlinear frequency perception of the human auditory system[45]. This scale plays a pivotal role in the field of audio and speech processing by mapping linear frequencies (f) to Mel frequencies $M(f)$. This mapping is achieved through mathematical formulas, such as the Mel filterbank equations. One of the most commonly used equations for this transformation is:

$$M(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{4.2}$$

In essence, the Mel scale provides a vital framework for understanding how humans perceive and process frequencies, which is indispensable in various audio-related applications, including speech synthesis and analysis.

- The generation of Mel spectrograms involves a series of essential steps. Initially, the continuous audio signal is divided into shorter time frames, typically spanning 20-30 milliseconds, allowing for focused analysis of spectral content over brief intervals. These frames are then refined using window functions, such as Hamming

or Hanning, to reduce spectral leakage and enhance separation between adjacent frames. The subsequent step entails computing the Discrete Fourier Transform (DFT) for each windowed frame, transforming the signal from the time domain to the frequency domain, and revealing frequency components' amplitude and phase. The power spectrum is then derived by squaring the magnitude of each complex DFT coefficient, providing insights into energy distribution across different frequencies in the frame. Following this, Mel filtering comes into play, as the power spectrum passes through a bank of filters designed to mimic human auditory perception, emphasizing perceptually significant frequencies spaced according to the Mel scale. For each frame, energy within each Mel filter's pass-band is summed, and a logarithmic transformation is applied to align the representation with human auditory perception. The result is a Mel spectrogram, a matrix with time frames on the x-axis and Mel frequency bins on the y-axis, where values represent the logarithmic energy content within each Mel frequency bin at each time frame. This meticulously orchestrated process yields a compact and perceptually meaningful representation of an audio signal's spectral content over time, highly valuable in speech and audio processing tasks for capturing essential acoustic features while reducing computational complexity.Figure 4.1 illustrate the process of mel spectrograms generation.

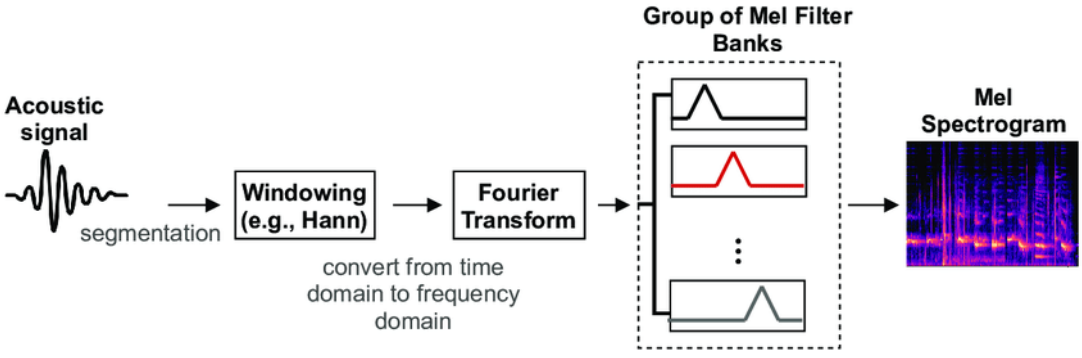


Figure 4.1: Mel Spectrograms Generation[45].

4.3 Character Embedding

Character embedding is a method for understanding text in different ways. It's unlike word embedding because it can handle any word as long as it's made up of characters

the model knows. This is super useful because it can handle tricky words, such as those that are spelled incorrectly, slang words, and out-of-vocabulary (OOV) words. Character embedding works by looking at how words are put together with their letters, and it uses 1D Convolutional Neural Networks (CNNs) to turn words into numbers. These numbers capture both what words mean and how they're used in sentences. Character embedding is great for getting information from short parts of long texts, and it's especially useful when dealing with challenging words that other methods might struggle to handle. It's a useful tool in natural language processing, especially when dealing with words that aren't in a fixed list[46].

4.4 Sequence to Sequence Model

A Sequence-to-Sequence (Seq2Seq) model is a neural network architecture widely used in natural language processing[47], including translation, summarization, and speech synthesis. It's designed to handle variable-length input and output sequences. The Seq2Seq model includes:

- Encoder: processes the input sequence, creating a fixed-size context vector and appending a STOP token at the end.
- Decoder: generates the output sequence step by step, with a STOP token indicating completion based on a calculated probability threshold

This framework is valuable for tasks like speech synthesis, which involve predicting sentence durations, challenging due to pronunciation variations. In contrast to fixed-size input tasks like image recognition, speech synthesis benefits from Seq2Seq's ability to handle variable-length inputs and outputs effectively.

4.5 Attention Network

The integration of attention networks within Seq2Seq architectures has proven highly beneficial across various domains. When applied to Seq2Seq models, attention mechanisms excel in effectively managing sequences of variable lengths. They empower Seq2Seq models to dynamically prioritize specific portions of input data while generating corresponding output sequences. This adaptability is particularly valuable in

scenarios like machine translation, where aligning words between source and target languages can pose challenges. By enabling precise alignments between input and output elements, attention networks enhance the accuracy and contextual appropriateness of generated sequences. This dynamic and interpretable approach has broadened the utility of Seq2Seq models, extending their relevance to diverse applications outside of specific language-related tasks[48].

4.6 Griffin-Lim Vocoder

The Griffin-Lim Vocoder plays a pivotal role in audio signal processing, particularly in the realm of audio reconstruction from short-time Fourier transforms (STFT). Named after its creators, Donald S. Griffin and Lim JaeSung, this vocoder addresses the intricate challenge of phase reconstruction, an essential aspect of audio fidelity. By iteratively updating the phase while preserving the magnitude information, the Griffin-Lim algorithm gradually reconstructs the audio waveform from its STFT representation, facilitating various applications in audio denoising, source separation, and music generation. While it is a valuable tool, it's important to acknowledge its limitations, especially when dealing with complex audio signals or highly modified STFTs, inspiring ongoing research to enhance its capabilities in audio engineering and creative expression[49] .

Algorithm 1 Griffin-Lim Algorithm

1: Initialize an initial estimate of the complex-valued spectrogram $X^{(0)}$.
2: Set the maximum number of iterations T.
3: Set the desired phase angle $\theta^{(0)}$ for $X^{(0)}$.
4: **while** $t < T$ **do**
5: Compute the complex spectrogram phase using the current magnitude and the previous phase: $\theta^{(t)} = \arg(X^{(t-1)})$.
6: Inverse STFT: $x^{(t)} = \text{ISTFT}(X^{(t-1)}, \theta^{(t)})$.
7: Re-compute the complex-valued spectrogram: $X^{(t)} = \text{STFT}(x^{(t)})$.
8: Set the magnitude of $X^{(t)}$ to match the observed magnitude: $|X^{(t)}| = |X|$.
9: Increment t.
10: **end while**
11: Output the final estimate of the signal: $x^{(T)}$.

When working with linear spectrograms Griffin-Lim steps in as a valuable tool for the reconstruction of the corresponding time-domain audio waveform. This process is essential in tasks like speech synthesis, where the goal is to convert spectrogram representations back into natural-sounding speech. Griffin-Lim employs an iterative approach to estimate the phase information that is lost during the transformation from the time domain to the frequency domain, which is necessary to reconstruct the original waveform accurately. This technique, although simplistic in nature, proves effective in achieving high-quality audio reconstruction from linear spectrograms and is commonly employed in Tacotron-based text-to-speech systems and various audio processing applications.

4.7 Wavenet

The WaveNet Vocoder is an advanced technology used in speech and audio processing. It's designed to create realistic and high-quality speech and audio synthesis. What makes WaveNet Vocoder stand out is its ability to generate audio waveforms sample by sample, allowing for precise control over the generated sound. WaveNet Vocoder has greatly improved the quality of synthetic audio, making it a valuable tool in industries that rely on realistic audio production [4].
Wavenet is a generative model operating directly on the raw audio waveform. The joint probability of a waveform $x = \{x_1, \ldots, x_T\}$ is factorised as a product of conditional probabilities as follows:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}) \tag{4.3}$$

4.7.1 Architecture of Wavenet

The architecture of WaveNet is characterized by its deep and autoregressive structure, which means it predicts one audio sample at a time, conditioning each prediction on previous samples[4]. The core of WaveNet is built on two concept. The first is dilated causal convolution and the reason behind this is to ensure that each prediction at time step t only depends on past and current samples $p(x_{t+1}|x_1, \ldots, x_t)$, not future ones. This ensures that the network maintains a realistic temporal flow. In addition to that the dilation is used to capture long-range dependencies and maintain efficiency in modeling

audio data. Traditional convolutional layers have a limited receptive field, which means they can only capture local patterns in the data. In audio, where long-term dependencies are crucial for generating realistic waveforms, this limitation can be problematic.The second concept which is crucial is residual block and skip connection.

Residual and Skip connections

In WaveNet's architecture, Residual and Skip Connections represent a crucial innovation that addresses the challenge of training deep neural networks effectively. In WaveNet, the use of residual connections allows for the efficient modeling of complex dependencies within the audio data, facilitating the synthesis of more natural and expressive speech. Skip connections, on the other hand, enable the model to capture information at different levels of abstraction simultaneously. By skipping over certain layers and connecting directly to later layers, WaveNet can effectively capture both fine-grained details and high-level features in the audio signal. This contributes to the model's ability to generate audio samples that exhibit both local intricacies and global coherence.

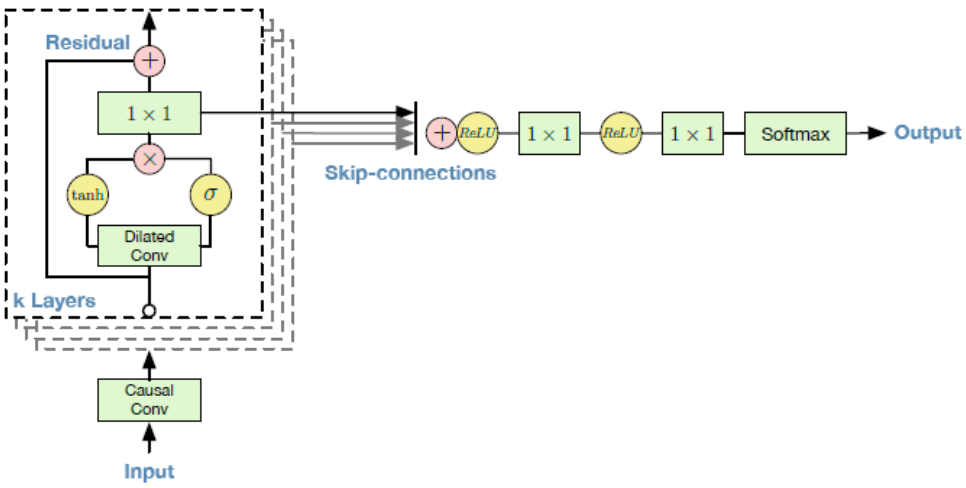


Figure 4.2: Overview of the residual block and the entire architecture[4].

Together, these residual and skip connections in WaveNet create a powerful architecture that not only overcomes training challenges but also excels at capturing the rich structure of audio data. They play a pivotal role in elevating the quality of synthesized speech, making WaveNet a state-of-the-art solution in the realm of natural and expressive speech synthesis.

Gated Activation Units

Within the architecture of WaveNet, Gated Activation Units (GAUs) play a pivotal role in enhancing the model's capacity to capture complex dependencies and generate high-quality audio. GAUs are inspired by the Long Short-Term Memory (LSTM) architecture and consist of a pair of parallel neural networks: one for capturing the signal and another for modeling the gate. The signal network processes the input data and computes an intermediate representation, while the gate network modulates this representation by determining which information to pass through. This gating mechanism allows WaveNet to selectively retain relevant information while discarding less crucial aspects, effectively mitigating the vanishing gradient problem commonly encountered in deep neural networks. By introducing non-linearity and adaptability into the model's computations, GAUs enable WaveNet to model long-range dependencies in audio data, resulting in improved speech synthesis quality and capturing nuances that contribute to a more natural and expressive audio output. Here the gate being used is the same used in the gated PixelCNN [50]:

$$z = \tanh(W_{f,k} * x) \cdot \sigma(W_{g,k} * x) \tag{4.4}$$

where $*$ denotes a convolution operator, $\odot$ denotes an element-wise multiplication operator, $\sigma(\cdot)$ is a sigmoid function, k is the layer index, f and g denote filter and gate, respectively, and W is a learnable convolution filter. In our initial experiments, we observed that this non-linearity worked significantly better than the rectified linear activation function (Nair & Hinton, 2010) for modeling audio signals[4].

4.7.2 Output Distribution

In the context of the WaveNet vocoder, the concept of output distribution plays a pivotal role in shaping the quality and expressiveness of the synthesized audio. WaveNet, known for its innovation in speech synthesis, leverages various output distributions to capture the intricate nuances of sound.

- **Single Gaussian Distribution**: Within the realm of probability theory, the single Gaussian distribution, denoted as $\mathcal{N}(\mu, \sigma^2)$, stands as a fundamental and widely applied probability distribution. This univariate Gaussian distribution is defined

by two key parameters: the mean (μ) and the variance (σ^2). It's characterized by its probability density function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.5}$$

Here, x represents the continuous variable, while μ represents the mean, and σ^2 represents the variance. The shape of the distribution is determined by these parameters, with the mean (μ) serving as the central location, and the variance (σ^2) controlling the spread or dispersion of the data points. In practical applications, particularly within the context of audio synthesis in the WaveNet vocoder, the single Gaussian distribution is employed to model the amplitude values of audio samples. By leveraging this distribution, WaveNet can probabilistically generate audio waveforms that exhibit a natural and continuous progression of amplitude values, ultimately contributing to the high-fidelity synthesis of realistic audio signals.

- **Logistic Distribution**: The single logistic distribution is a probabilistic model that finds utility in various machine learning applications, including binary classification tasks. In this distribution, the probability of an event occurring, often denoted as $p(x)$, is modeled as a sigmoid function of a linear combination of input features, typically represented as x. Mathematically, it can be expressed as:

$$p(x) = \frac{1}{1 + e^{-(wx+b)}} \tag{4.6}$$

Here, w represents the weights associated with the input features, b is the bias term, and e is the base of the natural logarithm. The logistic distribution output lies in the range [0, 1], making it particularly well-suited for problems where the goal is to estimate the probability of a binary outcome, such as spam detection in emails or disease diagnosis in healthcare. In machine learning, this distribution is often used as the basis for logistic regression, a simple yet powerful classification algorithm. Logistic regression models the probability that a given input instance belongs to one of two classes and is a fundamental building block in many more complex models, illustrating its importance and versatility in the field of data analysis and predictive modeling.

- **Softmax Distribution:** The Single Softmax distribution is a fundamental component in machine learning, particularly in the context of probabilistic modeling

and classification tasks. It is used to model categorical data, where an observation can belong to one of several discrete classes. The essence of the Single Softmax distribution lies in its ability to assign probabilities to each class, expressing the likelihood of an observation falling into a specific category.[51] suggests that, even when dealing with implicitly continuous data—such as image pixel intensities or audio sample values a Softmax distribution often proves to be more effective. This effectiveness stems from the categorical distribution's remarkable flexibility, allowing it to model diverse distributions without presuming any specific shape. Given that raw audio data is typically represented as a sequence of 16-bit integer values, employing a Softmax layer directly would necessitate generating a staggering 65,536 probabilities per time-step to account for all possible values. To make this computationally more manageable, a μ-law companding transformation is initially applied to the data. Subsequently, the data is quantized into 256 possible values using the following formula:

$$f(x_t) = \text{sign}(x_t) \cdot \frac{\ln(1 + \mu \cdot |x_t|)}{\ln(1 + \mu)} \tag{4.7}$$

Here, where $-1 < x_t < 1$ and $\mu = 255$. This non-linear quantization approach not only significantly enhances reconstruction but also retains the fidelity of the audio signal post-quantization. Particularly in the context of speech synthesis, it results in a reconstructed signal that closely resembles the original, demonstrating the efficacy of this approach.

4.8 Tacotron

4.8.1 Model defintion

Tacotron has emerged as a revolutionary development in text-to-speech synthesis, introducing an end-to-end system capable of training with minimal human annotation, allowing for the direct conversion of text characters into speech[5]. This eliminates the need for complex, multi-component systems used traditionally, streamlining the TTS process and enhancing adaptability. Tacotron's remarkable ability to handle real-world data variations and potential to transform speech synthesis is particularly noteworthy. Traditional text-to-speech systems involve intricate pipelines with multiple components,

demanding substantial engineering efforts during development. In contrast, Tacotron's integrated approach simplifies the process, reduces the necessity for intricate feature engineering, and offers exceptional flexibility. In essence, Tacotron represents a transformative approach in text-to-speech synthesis, simplifying the process, enhancing flexibility, and achieving remarkable results in natural-sounding speech synthesis.

4.8.2 Model architecture

The Tacotron architecture is a sequence-to-sequence model designed for end-to-end text-to-speech (TTS) synthesis see figure . It directly converts input text into a corresponding spectrogram, which can then be converted into speech through a vocoder in this case Griffin-Lim. Here are the basics component of Tacotron architecture.

1. Character Embedding: The input text is first passed through an embedding layer. This layer converts discrete symbols into continuous vector representations,"text embedding".

2. Encoder: The character embeddings are passed to the encoder, responsible for deriving robust sequential representations. The "pre-net" module processes the character embeddings, applying necessary transformations. At the core of the encoder is the CBHG module, named for its combination of a 1D Convolution Bank, Highway network, and Bidirectional GRU. This module's purpose is to extract improved character representations while mitigating overfitting. To predict spectrograms accurately, the model must learn the relationships between text and speech signals. For this task, the authors employ a stack of GRUs with vertical residual connections in the decoder phase. This decoder generates a series of linear-scale spectrograms, one for each time step, ultimately forming the synthesized speech output.

3. Decoder with Attention:To predict the spectrograms, the model needs to learn the alignments between text and speech signals. The authors use a stack of GRUs with vertical residual connections for the decoder, to produce a sequence of linear-scale spectrograms, one per time-step of the decoder.

4. Griffin-Lim vocoder: which convert the spectrograms generated by seq2seq model to speech.

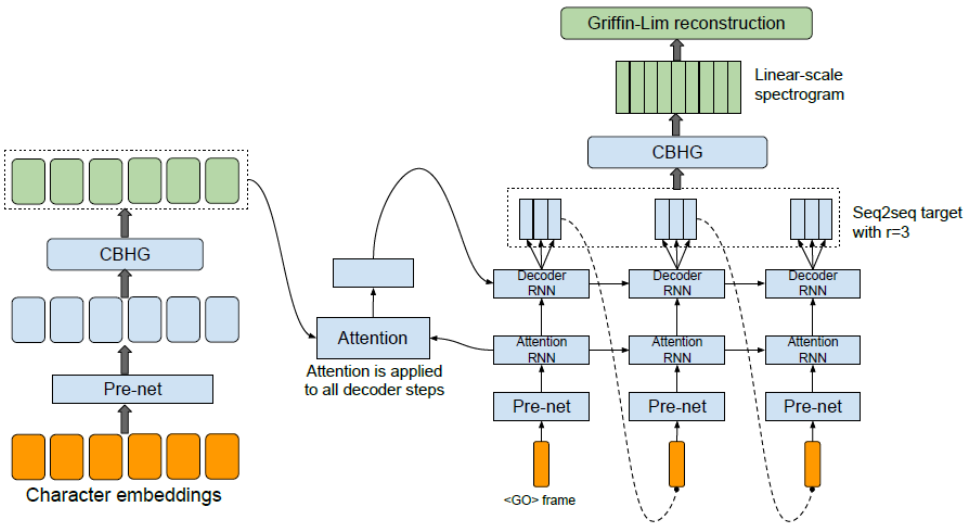


Figure 4.3: Tacotron architecture. The model takes characters as input and outputs the corresponding linear-scale spectrogram, which is then fed to the Griffin- Lim reconstruction algorithm to synthesize speech[5]

4.9 Tacotron 2

4.9.1 Model definition

Tacotron 2 integrates cutting-edge deep learning techniques and addresses some of the limitations of earlier TTS systems, offering improvements in both naturalness and intelligibility of the synthesized speech [52]. By combining a text-to-spectrogram model with a high-quality waveform synthesis model, often based on a variant of WaveNet, Tacotron 2 achieves impressive results in generating human-like speech that closely approximates the nuances of human vocalization. This technology has found applications in various domains, underscoring its significance in making human-computer interactions more engaging and inclusive.

4.9.2 Model Architecture

Tacotron 2 has the same main structure as Tacotron which is seq2seq with a vocoder. The difference is that in Tacotron 2 the seq2seq models tends to predict the Mel-spectograms

for each input text.the use of mel-spectograms leads us to use a new vocoder which is in this case a variant of Wavenet.

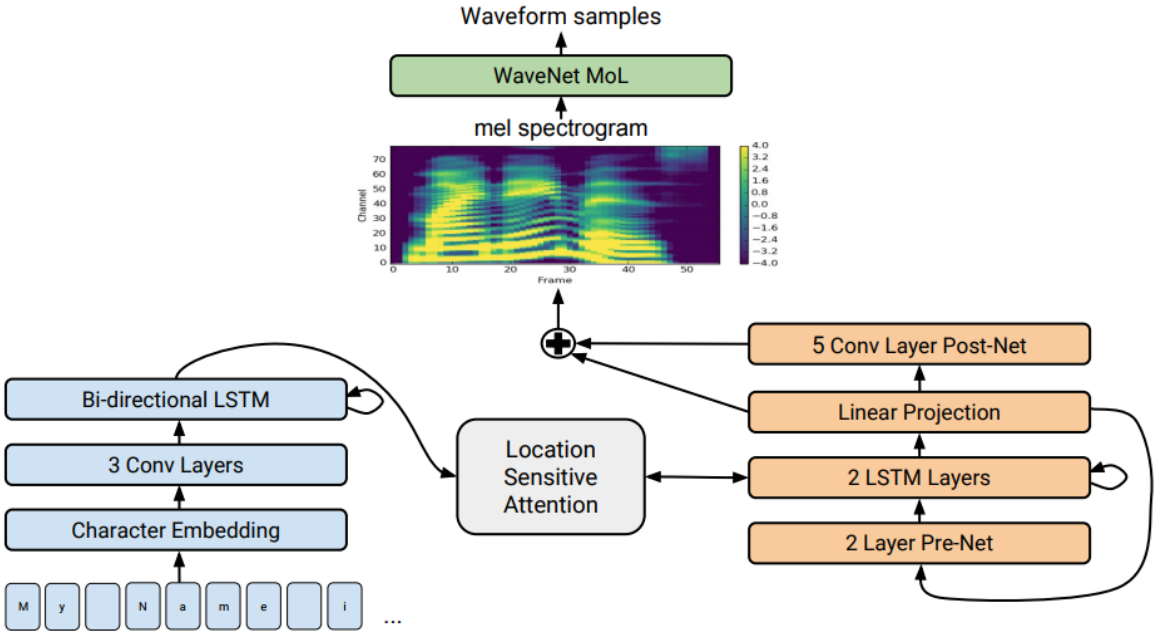


Figure 4.4: Block diagram of the Tacotron 2 system architecture [52].

1. Encoder: The encoder takes text consisting of characters and transforms it into word embedding vectors by following a series of steps. Initially, the input text is processed through a character embedding layer, where each character is converted into a numerical representation. Subsequently, the text goes through a 1-dimensional convolutional neural network (CNN) comprising three layers. Finally, it passes through a Bidirectional Long Short-Term Memory (Bi-LSTM) network. These sequential transformations ultimately result in feature vectors that capture the essence of the input text.

2. Attention:The attention mechanism facilitates communication between the encoder and decoder by extracting essential information from the encoder's output. It does so by considering both the features generated by the encoder LSTM and the previous decoder time-step information. This process ensures that at each time step, the decoder receives the relevant encoder details needed to produce the next output element.

3. Decoder: It is the final step in the system, responsible for creating a mel spec-

trogram and deciding when to stop. It's composed of key elements: a pre-net, an LSTM, a projection layer, and a Post-Net. The LSTM combines information from the attention layer and pre-net to calculate termination probabilities and produce the mel spectrogram. It generates the mel spectrogram until the stopping condition is met.

4. Wavenet as a vocoder:A modified version of the WaveNet architecture from[4] is used to invert the mel spectrogram feature representation into time-domain waveform samples.

4.10 Adapting Tacotron on Arabic language

Adapting Tacotron to Arabic languages presents a unique set of challenges and opportunities in the field of text-to-speech synthesis. Arabic is a Semitic language known for its rich phonological and morphological features, including complex consonant clusters, vowel variations, and a unique script from right to left. To make Tacotron effectively generate natural and expressive Arabic speech, researchers and engineers must address issues related to phoneme recognition, prosody modeling, and script-specific nuances. Additionally, accommodating for the diverse dialects and accents across the Arabic-speaking world is crucial. Adapting Tacotron to Arabic involves training the model on large and diverse Arabic speech datasets, fine-tuning its architecture to handle Arabic phonetics and prosody, and ensuring compatibility with the Arabic script. Successfully adapting Tacotron to Arabic languages can open doors to a wide range of applications, from Arabic language learning tools to assistive technologies and voice assistants, ultimately enhancing accessibility and communication for Arabic speakers worldwide[53].

CHAPTER 5

Implementation and Results

In this chapter, we will delve into the process of transforming theoretical speech synthesis concepts into functional systems. We will elucidate two variations of Wavenet models: one referred to as Gaussian WaveNet and the other as Mixture of Logistics WaveNet. Furthermore, we will examine Tacotron 2 and its modified iteration. Additionally, we will showcase our attempts to adapt a version of Tacotron 2 for the Arabic language.

5.1 Signal to Noise Ratio (SNR)

Signal-to-Noise Ratio (SNR) stands as a fundamental metric employed across a range of domains, encompassing electronics, telecommunications, and audio engineering. It serves to quantify the relative potency of a desired signal in comparison to undesired background noise or interference. Expressed in decibels (dB), a heightened SNR signifies a more robust and clearer signal when contrasted with the noise. SNR plays an indispensable role in ascertaining the excellence and dependability of communication systems, audio recordings, and data transmission. The pursuit of a higher SNR frequently constitutes a primary objective in these applications to assure the precise reception of signals and the preservation of optimal data integrity.

In its essence, SNR serves as a pivotal gauge of signal quality and the capacity to differentiate the desired information from ambient noise[47].The formula to calculate SNR is as follows:

$$SNR = 10 \cdot \log_{10} \left(\frac{\text{Signal Power}}{\text{Noise Power}} \right) \tag{5.1}$$

5.2 Data

1. **The English dataset**: employed in both the first and second experiments is sourced from the LJSpeech dataset [54]. This publicly accessible speech dataset encompasses a total of 13,100 concise audio segments. Within these segments, a solitary speaker recites excerpts from seven non-fiction books. Each audio clip is accompanied by a transcript of its content. These clips exhibit varying durations, spanning from 1 to 10 seconds, ultimately amassing to an aggregated duration of approximately 24 hours. The source texts used for these recordings were originally published between 1884 and 1964 and now reside in the public domain, devoid of copyright restrictions. The actual audio recordings themselves were captured during the years 2016 and 2017 as part of the LibriVox project, also available freely within the public domain. Metadata pertaining to these recordings is conveniently provided in the 'metadata.csv' file. Each entry in this file is structured as a single record, delineated by the pipe character (0x7c), and encompasses the following fields:

   (a) ID: this is the name of the corresponding .wav file.

   (b) Transcription: words spoken by the reader (UTF-8).

   (c) Normalized Transcription: transcription with numbers, ordinals, and monetary units expanded into full words (UTF-8).

   Each audio file is a single-channel 16-bit PCM WAV with a sample rate of 22050 Hz.

2. **Arabic Dataset**: Arabic Speech corpus has been developed as part of PhD work carried out by Nawar Halabi at the University of Southampton[55]. The corpus was recorded in south Levantine Arabic (Damascian accent) using a professional

studio. Synthesized speech as an output using this corpus has produced a high quality, natural voice.The dataset contains about 2.41 hours of Arabic speech, a total of 906 utterances, and 694556 frames. The dataset consists of htext, audioi pairs. The input text is diacritic Arabic characters, while the output is a 16-bit 48 kHz PCM audio clip is with a bit-rate of 768 kbps. We use Unicode character symbols instead of the diacritic Arabic in the transcript of the audio files to achieve phonetic accuracy, capturing diacritic marks, representing contextual variations, and addressing unique linguistic features inherent in the Arabic language. This standardized approach ensures consistency, accessibility, and internationalization while facilitating the creation of high-quality, natural-sounding Arabic speech synthesis models.We do this transformation from diacritic Arabic to Unicode character symbols using an open source phonitization algorithm.

5.3 Comparing Gaussian Wavenet and Logistic Wavenet

WaveNet has emerged as a robust deep learning architecture renowned for its ability to generate high-fidelity audio waveforms. This architecture presents a probabilistic generative model capable of capturing the intricacies and subtleties found in natural audio. Initially, WaveNet predominantly relied on a Gaussian distribution to model the conditional probabilities governing audio waveform samples.

In recent years, researchers have delved into alternative probability distributions to enrich the expressive capabilities and adaptability of WaveNet-based models. One noteworthy alternative is the Logistic distribution, which markedly diverges from the Gaussian distribution in terms of its shape and inherent characteristics. The Logistic distribution exhibits a distinctive non-Gaussian behavior.

The objective of this experiment is to conduct a comparative analysis between two variants of the WaveNet architecture: the Single Gaussian WaveNet and the Logistic Distributed WaveNet.

5.3.1 Training setup

Both the Gaussian WaveNet and Logistic WaveNet models were trained on the following configurations:

1. Hardware Configuration:
    - Processor: Intel® Core™ i5-4460
    - Cache: 6 MB
    - Clock Speed: Up to 3.40 GHz
2. Dataset: The training dataset is a 216 audio files from the LJSpeech data set, equivalent to approximately 20 minutes of audio content.
    - Dataset Split:
        - Training Data: 186 audio files were allocated for training,equivalent to approximately 16 minutes of audio content.
        - Development (Dev) Data: 10 audio files were reserved for model fine-tuning and optimization during training.
        - Evaluation Data: 20 audio files were set aside for bench marking and assessing the generated audio waveforms.
    - Preprocessing encompassed the application of pre-emphasis, a technique used to boost the high-frequency components within the audio data. This enhancement served to improve the models' capability to capture intricate acoustic nuances.
    - Postprocessing involved the application of inverse pre-emphasis to the generated audio waveforms following synthesis. This step was crucial in guaranteeing that the final output maintained its natural sound characteristics.

3. Model Configuration: We used the TensorFlow WaveNet vocoder [56] which is an open-source implementation of the WaveNet architecture, designed for high-quality waveform synthesis, particularly in text-to-speech (TTS) applications. The Hyper parameters for both models are shown in table 5.1.

Table 5.1: Hyperparameters for Gaussian WaveNet and Logistic WaveNet

Parameter	**Gaussian WaveNet**	**Logistic WaveNet**
Quantize Channels	65,536	65,536
Sample Rate	22,050 Hz	22,050 Hz
Number of Mel Filters	80	80
Minimum Frequency (fmin)	125 Hz	125 Hz
Maximum Frequency (fmax)	7,600 Hz	7,600 Hz
FFT Size	1,024	1,024
Hop Size	256	256
Window Function	hann	hann
Output Distribution	Normal (Gaussian)	Logistic
Output Channels	2	30
Number of Layers	24	24
Number of Stacks	4	4
Residual Channels	128	128
Gate Channels	256	256
Skip Output Channels	128	128
Kernel Size	3	3
Batch Size	2	2
Optimizer	Adam	Adam

As shown in table5.1 we use the Adam optimizer in both Gaussian and logistic Wavenet ,the parameters of the optimizer are shown in table 5.2.

Table 5.2: Optimizer Parameters for both WaveNet Vocoder variants

Optimizer Parameter	**Value**
Learning Rate (lr)	0.001
Epsilon (eps)	1e-08
Weight Decay	0.0

4. Training: Owing to hardware constraints, the training duration for both models was restricted to around 800 steps, which corresponds to roughly 9 epochs. This limitation stemmed from our decision to reduce the batch size to 2. In an attempt to circumvent these limitations, we explored the possibility of utilizing Google Colab. However, this endeavor was unsuccessful due to the lack of support for TensorFlow 1.x versions. The training process was executed sequentially for both models, Tacotron2 and Wavenet.

5.3.2 Results and Evaluation

In Figure 5.1 and 5.2 we see the loss for the two model.

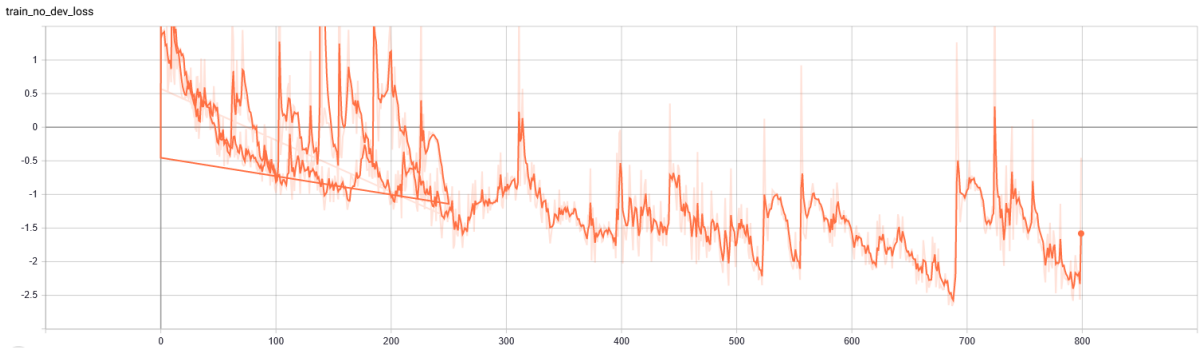


Figure 5.1: The loss validation for the Gaussian Wavenet.

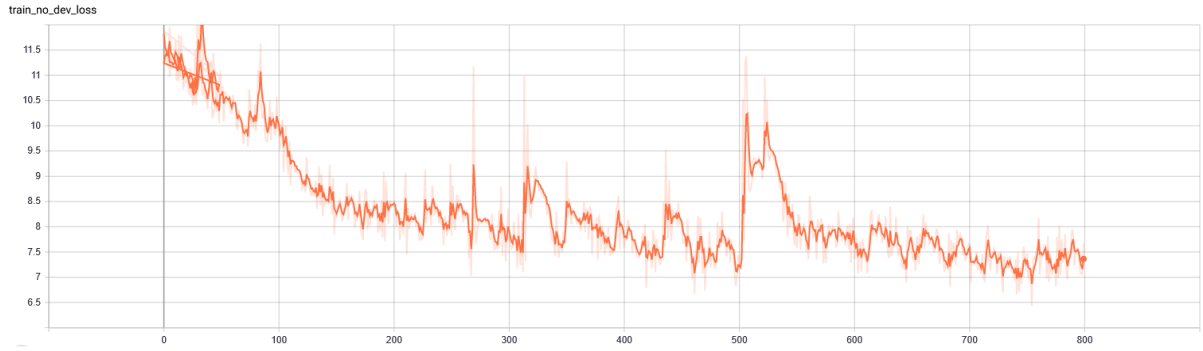


Figure 5.2: The loss validation for the Logistic Wavenet.

We observe that the validation loss for the Logistic Wavenet is steadily decreasing, indicating successful training. Conversely, in the case of the Gaussian Wavenet, it's worth noting that the loss occasionally exhibits negative values. In the graph in figure 5.1 that in the beginning the first 200 steps there is an overlaping of two graphs this is due to starting two training at

In Figure 5.3, we have the original mel spectrograms for the audio sample from the evaluation set. The mel spectrograms of the generated audio samples from both models are presented in Figure 5.4 and Figure 5.5. we can see that the mel spectograms genereted are quite similar in differences in some frequencies but comparing to the original one they are not .

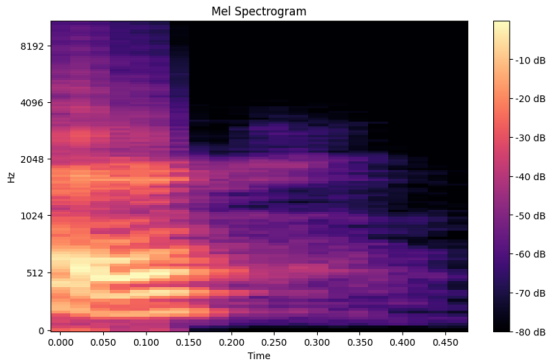


Figure 5.3: The mel spectogram representation of the reference signal.

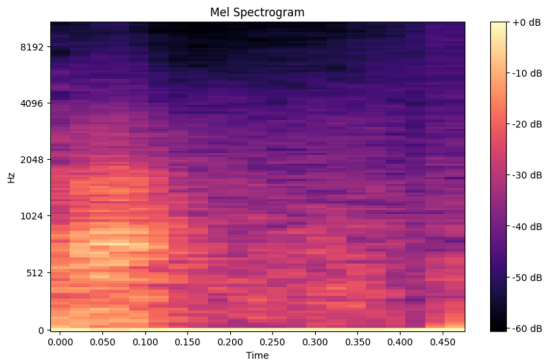


Figure 5.4: The mel spectogram representation of the generated signal by Gaussian Wavenet.

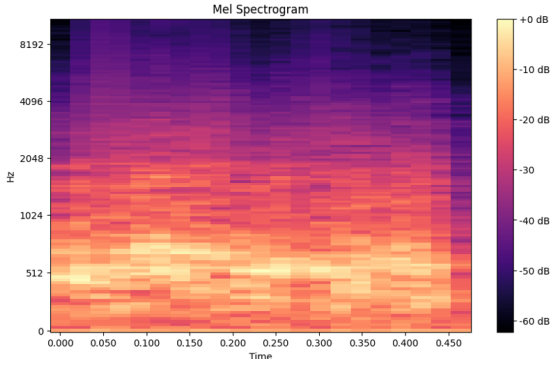


Figure 5.5: The mel spectogram representation of the generated signal by Logistic Wavenet.

Evaluation by Signal to Noise Ratio

Assessing the audio signal quality generated by the Gaussian and Logistic WaveNet models through Signal-to-Noise Ratio (SNR) is an essential procedure for evaluating their performance. SNR offers a quantitative assessment of how closely the generated signals align with the reference signals.

This evaluation is carried out on pairs of generated signals from the evaluation set, after 800 training steps, alongside their respective signal references. The outcomes of this evaluation are presented in Figure 5.6 and Figure 5.7.

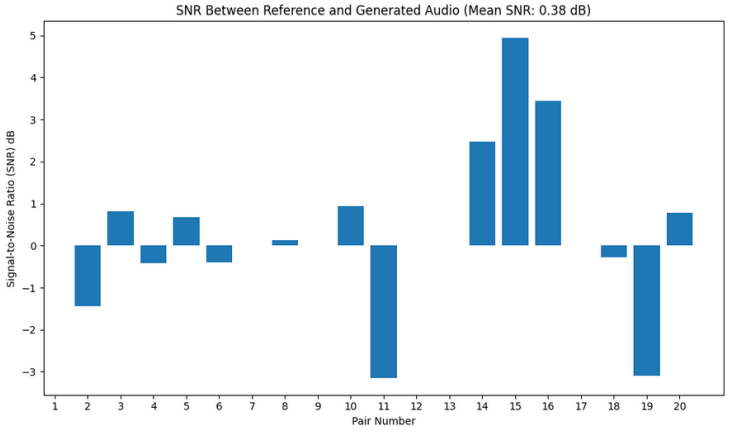


Figure 5.6: The SNR evaluation for Gaussian Wavenet.

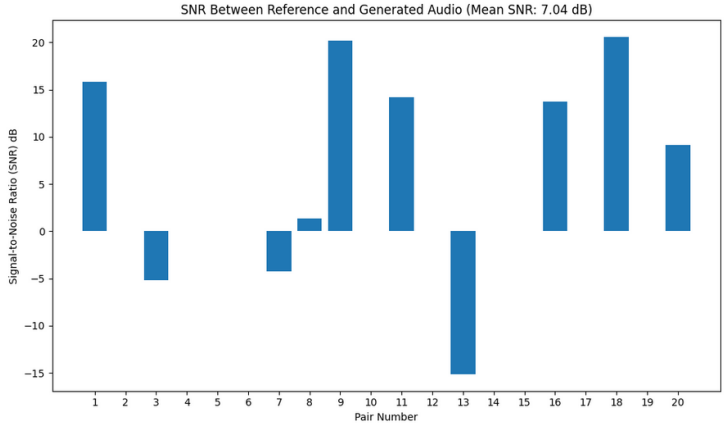


Figure 5.7: The SNR evaluation fro logistic Wavenet.

5.3.3 Discussion

In Table 5.1, we can observe that the hyperparameters of both models are mostly identical, with the exception of the output distribution and the number of output channels. This discrepancy can be explained as follows:

The Gaussian WaveNet models the probability distribution of audio waveform samples using a mixture of Gaussian distributions. Each output channel typically corresponds to a Gaussian component within this mixture. Two output channels suffice for Gaussian WaveNet because it assumes that each Gaussian component is characterized by two parameters: mean (μ) and standard deviation (σ). Therefore, the two output channels represent these parameters for a single Gaussian component. Essentially, for each time step, two values are needed to describe the mean and standard deviation of one Gaussian component.

In contrast, the Logistic WaveNet models the probability distribution of audio waveform samples using a mixture of logistic distributions. Unlike Gaussian distributions, logistic distributions have a sigmoid-shaped probability density function, which makes them suitable for modeling discrete and bounded data.

Logistic WaveNet often employs more output channels because it assumes that each output channel corresponds to a logistic component. This design choice allows the model to capture various aspects of the probability density function associated with logistic distributions.

In figures 5.1 and 5.2, we can observe the validation loss for the two models during training. It's important to note that the loss validation for Gaussian WaveNet exhibits negative values due to the utilization of two different loss functions. Gaussian WaveNet uses a mixed Gaussian loss, whereas Logistic WaveNet employs a discretized mixed logistic loss.

In figures 5.6 and 5.7, we can see that, overall, Logistic WaveNet outperforms Gaussian WaveNet with a mean SNR improvement of 7.04 dB, while Gaussian WaveNet only achieved a 0.38 dB improvement.

Gaussian WaveNet and Logistic WaveNet represent two distinct variations of the WaveNet generative model, each with its own set of unique characteristics. Gaussian WaveNet directly models the waveform's probability distribution as a Gaussian distribution, enabling it to generate more natural-sounding audio with subtle variations in

amplitude. Conversely, Logistic WaveNet employs a logistic mixture model, which can efficiently model the waveform using a mixture of logistic functions, capturing complex data dependencies.

5.4 Comparing the two versions of Tacotron

Tacotron 2 is an end-to-end neural Text-to-Speech (TTS) system that combines text-to-spectrogram synthesis with a vocoder to produce natural-sounding speech waveforms. Its default waveform generation component employs the Wavenet vocoder, setting a high standard for TTS quality. However, the computational demands of the Wavenet vocoder can be quite significant, posing challenges in real-time applications and resource-constrained environments.

In order to address these concerns and explore alternative approaches to TTS synthesis, this experiment seeks to compare the original Tacotron 2 configuration, which uses the Wavenet vocoder, with a modified version of Tacotron 2. This modified version can be seen as a hybrid approach, combining elements of Tacotron 2 and Tacotron, and it utilizes the Griffin-Lim vocoder as an alternative waveform generation method.

5.4.1 Training setup

1. Hardware Configuration:
    - Processor: Intel® Core™ i5-4460
    - Cache: 6 MB
    - Clock Speed: Up to 3.40 GHz
2. Dataset: The training dataset is a 216 audio files from the LJSpeech data set, equivalent to approximately 20 minutes of audio content.
    - Dataset Split:
        - Training Data: 186 audio files were allocated for training,equivalent to approximately 16 minutes of audio content.
        - Development (Dev) Data: 10 audio files were reserved for model fine-tuning and optimization during training.

  - Evaluation Data: 20 audio files were set aside for bench marking and assessing the generated audio waveforms.

- Data Preprocessing:The dataset underwent:
  - preprocessing: Which included , preemphasis to enhance high-frequency components in the audio data, improving the models' ability to capture subtle acoustic details.
  - postprocessing: After synthesis, inverse preemphasis was applied to the generated audio waveforms to ensure the final output retained natural sound characteristics.

3. Model Configuration: The implementation represents a tailored variation of Tacotron 2. In this modified version, two distinct vocoders are incorporated: the first being the WaveNet vocoder, as initially introduced in the original implementation [52], and the second is the Griffin-Lim algorithm, originally utilized in Tacotron [5]. A visual representation of this modified model can be found in Figure 5.8.

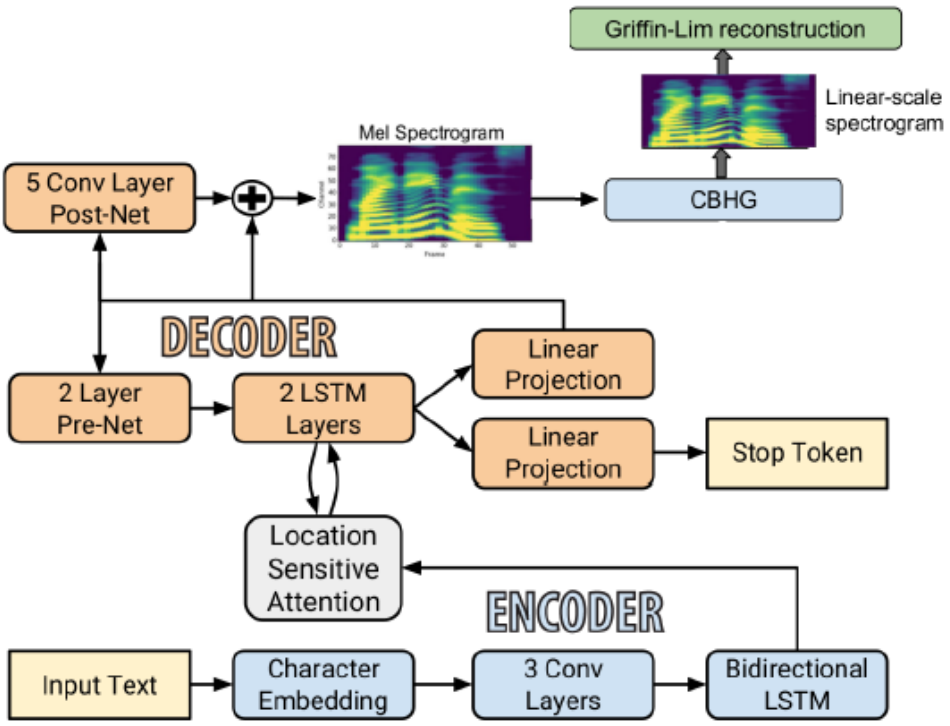


Figure 5.8: The modified version of Tacotron 2[57].

The underlying code for this adaptation is derived from Rayhane-mamah's Tacotron 2 implementation [58]. The hyperparameters for this implementation are detailed in Table 5.3.

Table 5.3: The hyper parameter of the Tacotron 2 and the vocoders.

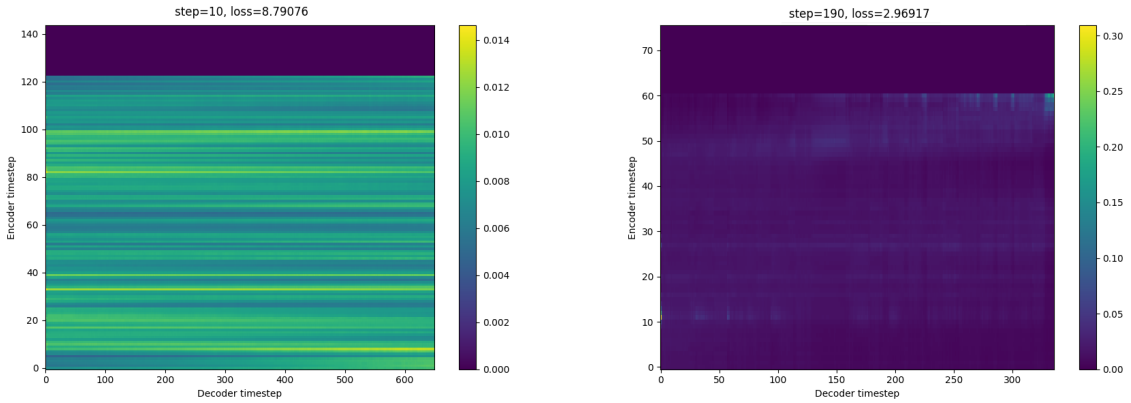
	Parameter	Value
Audio Parameters	sample rate	22050
	hop size	257
	win size	1100
Optimization parameters	tacotron adam beta1	0.9
	tacotron adam beta2	0.99
	tacotron adam epsilon	e^{-6}
Tacotron	embedding dim	512
	enc conv num layers	3
	attention dim	128
	prenet layers	[256,256]
	decoder layers	2
Wavenet	out channels	30
	layers	20
	stack	2
	residual channels	128
	gate channels	256
Griffin-Lim	power	1.5
	griffin lim iters	60

4. Training: Due to hardware limitations, the training duration for both models was restricted to 190 steps, which is equivalent to about 31 epochs. This constraint arose from the decision to use a batch size of 32, coupled with the dataset size of 186. In an attempt to overcome these limitations, we explored the possibility of using Google Colab. However, this option was rendered impractical since Google Colab no longer supports TensorFlow 1.x versions. The training process was carried out sequentially for both models, Tacotron 2 and WaveNet. The spectrogram

prediction network was trained on pairs consisting of :audio, transcript, while the WaveNet took audio files, preprocessed them, and generated their corresponding mel spectrograms for training.

5.4.2 Results and Evaluation

This implementation of Tacotron 2 has been adapted to predict both linear spectrograms and mel spectrograms. This modification grants the model the versatility to function with two distinct types of vocoders, namely Griffin-Lim and WaveNet. Following 190 training steps for both the WaveNet and the Spectrogram Feature Prediction network, we proceeded to test the system. The alignment between the encoder and decoder of Tacotron 2 at various steps is depicted in Figure 5.9. These visualizations illustrate the model's learning progress as the number of iterations increases. A more desirable alignment would resemble a diagonal line, indicating improved learning and alignment between the encoder and decoder.



(a) 10 steps

(b) 190 steps

Figure 5.9: The Alignment between the encoder and the decoder in different steps

In Figures 5.10 and 5.11, we can observe the similarities between the predicted mel spectrograms and the target spectrograms. These figures reveal that the overall shape of the mel spectrograms closely resembles each other, although there are some variations in the frequencies.

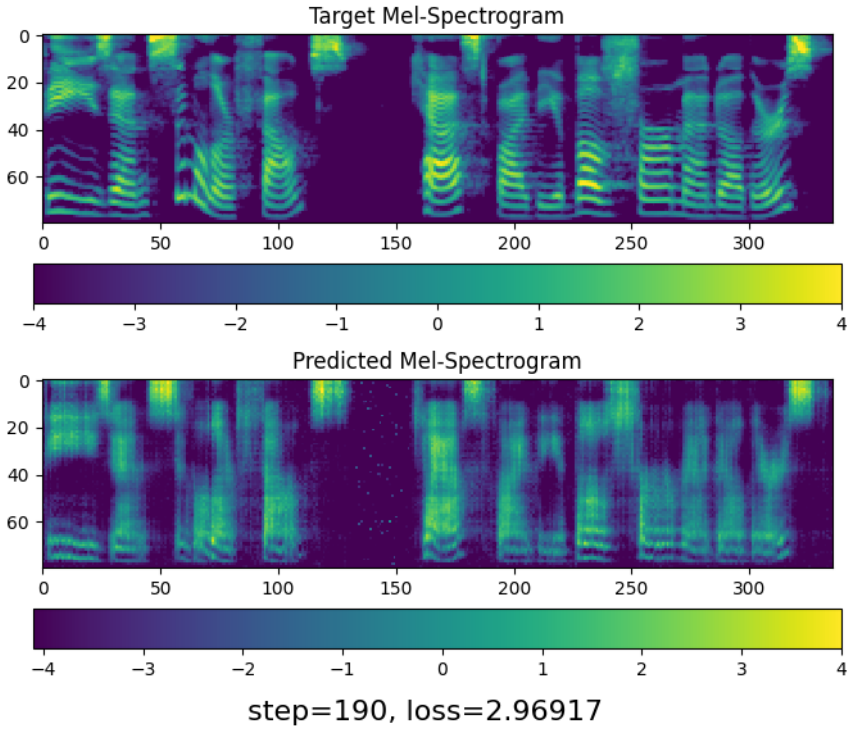


Figure 5.10: Comparison of Predicted vs Target Mel Spectrograms

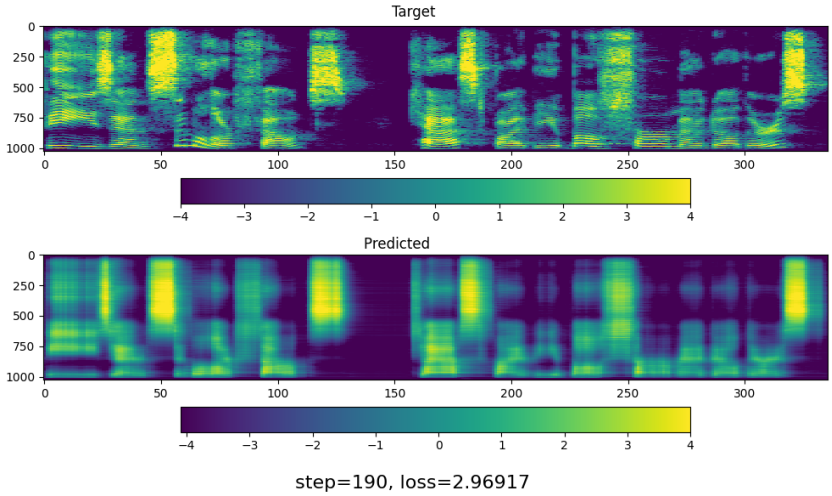


Figure 5.11: Comparison of Predicted vs. Target linear Spectrograms

The generated signal samples can be seen in Figures 5.12 and 5.13.The signals are practically the same but there are differences in the amplitude of the signals.

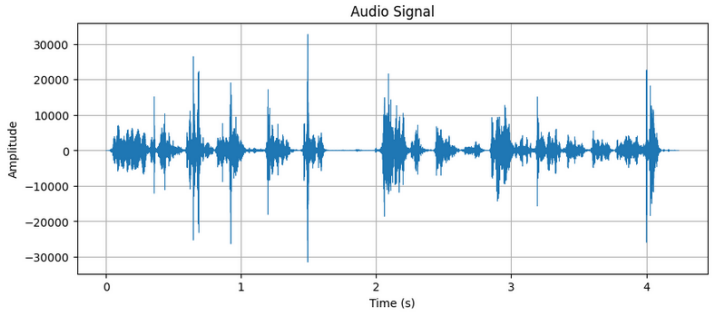


Figure 5.12: The signal generated by wavenet.

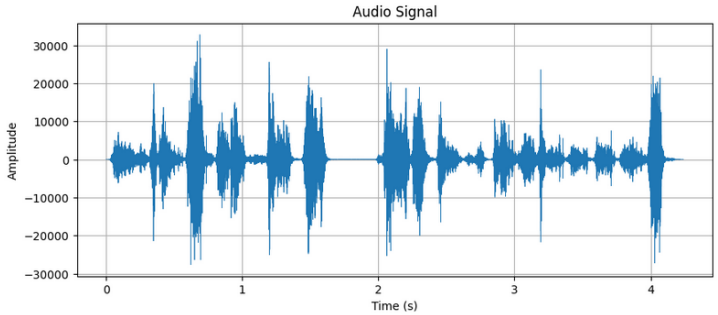


Figure 5.13: The signal generated by Griffin Lim.

5.4.3 Discussion

The experiment demonstrated that the modified Tacotron 2, with Griffin-Lim as a vocoder, can produce audio waveforms comparable to those generated by the original Tacotron 2 with the computationally intensive WaveNet vocoder. The Griffin-lim vocoder showed good results in reconstructing the audio samples but the biggest flop of it that it always generate a robotic sound with is way far from the the natural sound generated by the wavenet. Tacotron 2, with its default WaveNet vocoder, sets a high standard for TTS quality but demands significant computational resources. The modified version offers an alternative by integrating the Griffin-Lim vocoder, which, while less computationally intensive, sacrifices some audio quality. The alignment between the encoder and decoder in the showed learning progress. Spectrogram predictions closely matched target spectrograms in both models, but some frequency variations were present. The generated signals from WaveNet and Griffin-Lim were notably simi-

lar, with slight amplitude differences. Ultimately, the choice between Tacotron 2 and its modified version depends on the specific requirements of the application, balancing computational resources with audio quality. This experiment provides valuable insights into the adaptability of TTS systems to different vocoders, enabling more versatile and resource-efficient solutions in various contexts.

5.5 Implementing an Arabic TTS System Based on Tacotron2

Text-to-Speech (TTS) systems play a crucial role in converting written text into natural and understandable spoken language. While TTS technology has advanced considerably, there remains a notable deficiency in the availability of high-quality TTS systems for languages characterized by complex phonology and script, such as Arabic. Arabic, being one of the world's most widely spoken languages, is renowned for its rich linguistic heritage and distinctive script. Creating an efficient TTS system for Arabic presents substantial challenges and considerable prospects. The primary objective of this experiment is to tailor a version of Tacotron 2 to the Arabic language.

5.5.1 Training Setup

1. Hardware Configuration:

    - Processor: Intel® Core™ i5-4460
    - Cache: 6 MB
    - Clock Speed: Up to 3.40 GHz

2. Dataset: The training dataset is a 130 audio files from the arabic speech corpus data set, equivalent to approximately 22 minutes of audio content.We changed the transcript of the dataset has to accommodate the preprocess.The metadate.csv has to be in the format of the LJSpeech dataset.

    - Dataset Split:
        - Training Data: 100 audio files were allocated for training,equivalent to approximately 19 minutes of audio content.

  - Development (Dev) Data: 10 audio files were reserved for model fine-tuning and optimization during training.
  - Evaluation Data: 20 audio files were set aside for bench marking and assessing the generated audio waveforms.

- Data Preprocessing:The dataset underwent:
  - preprocessing: Whic included , preemphasis to enhance high-frequency components in the audio data, improving the models' ability to capture subtle acoustic details.
  - postprocessing: After synthesis, inverse preemphasis was applied to the generated audio waveforms to ensure the final output retained natural sound characteristics.

3. Model Configuration: The implementation represents a customized adaptation of Tacotron 2 on the Arabic language .The underlying code is derived from Rayhane-mamah's implementation of Tacotron 2 [58].The hyper-parameters of this implementation are basically the same as the one's in table 5.3 .altrhout there are some changes in the audio parameters. the changed parameters are shown in table5.4.

Table 5.4: The Hyper parameter for the Arabic Tacotron 2

	Parameter	Value
Audio Parameters	sample rate	48000
	hop size	600
	win size	2400

4. Training :Due to hardware limitations, the training duration for both models was constrained to only 100 steps approximately 32 epoch since the batch size to 32 and the size of the dataset is 100.To overcome these limitation we tried to use Google Colab but this was not possible since it does not support Tensorflow 1.x versions anymore.

5.5.2 Results

After 100 steps of training the loss validation is shown in figure5.14.

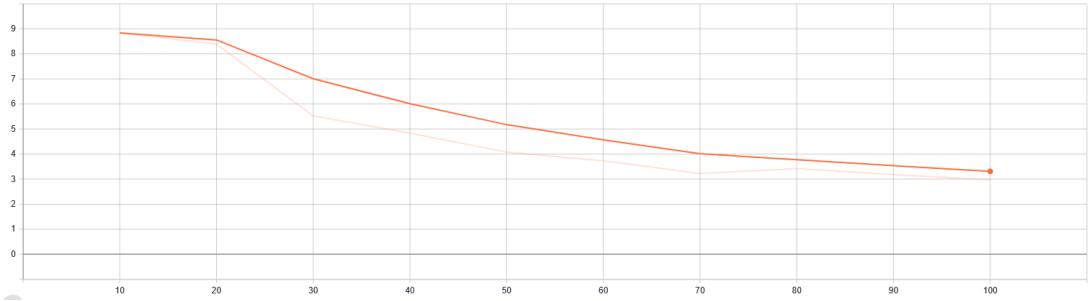
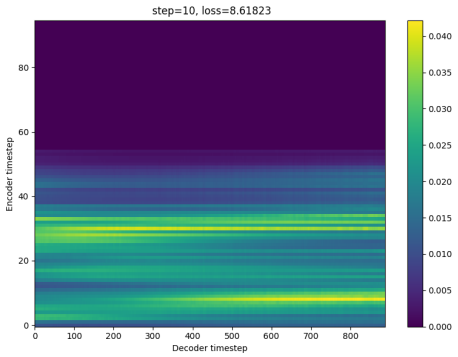


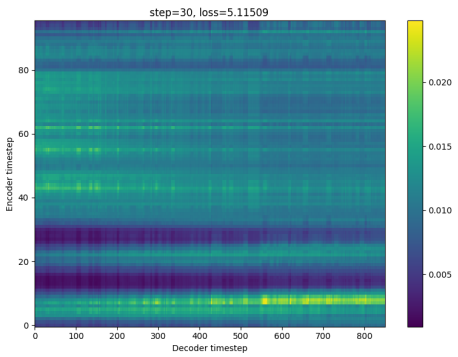
Figure 5.14: Validation Loss for Arabic Tacotron2.

From the figure above, it's evident that the training progressed smoothly. The loss decreased from 9 to 2.9 in just 100 steps, which is considered quite favorable given the duration of training the system underwent.

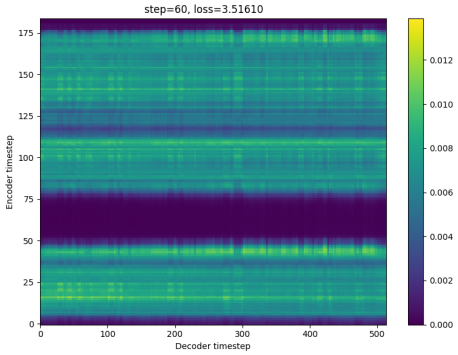
Figure 5.15 illustrates the alignment between the encoder and the decoder at various



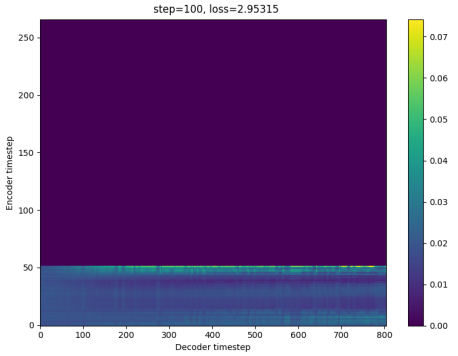
(a) 10 steps



(b) 30 steps



(c) 60 steps



(d) 100 steps

Figure 5.15: Alignment Graphs at different steps of training

time steps. This visualization demonstrates a notable improvement in alignment as the training progresses. Figure 5.16 shows the mel spectograms generated by our Arabic TTs system . We can see how well the system has started to learn as in sub-figure 5.16 the predicted mel spectogram does look anything like the targeted one but in sub figure5.17 the resemblance between the two spectograms started to appear.

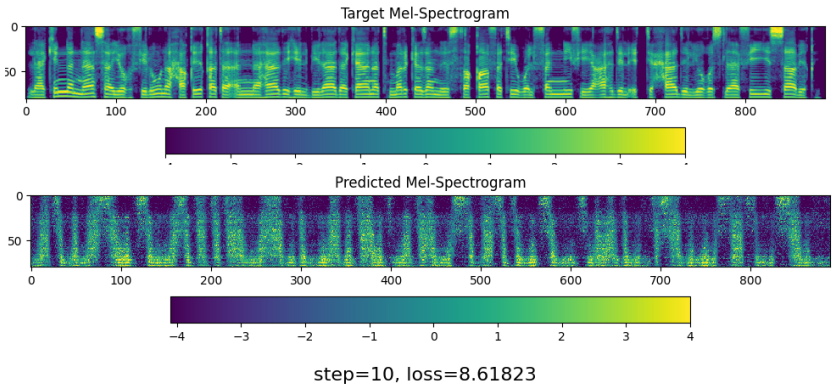


Figure 5.16: The predicted spectogram after 10 steps.

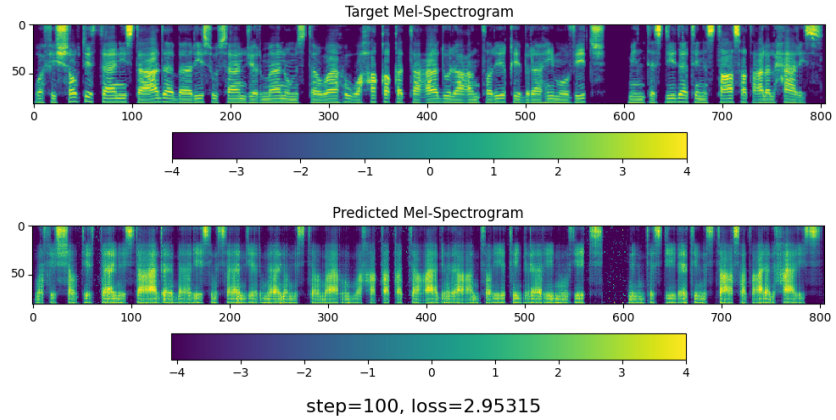


Figure 5.17: The predicted spectogram after 100 steps.

To see how our system is working,Figure 5.18 and 5.19 shows the original signal and the generated one.

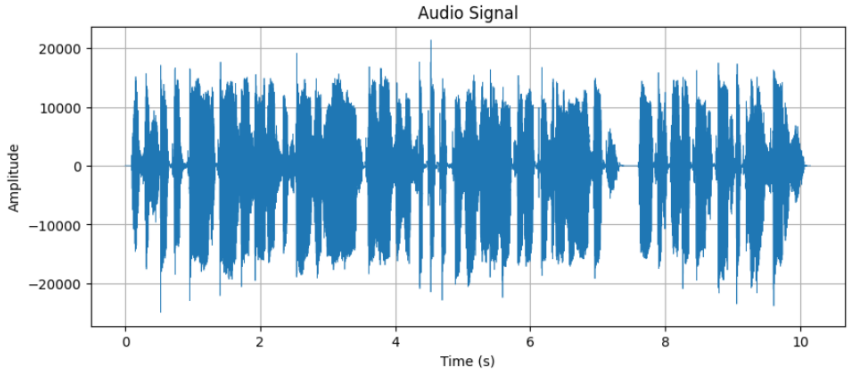


Figure 5.18: The original signal.

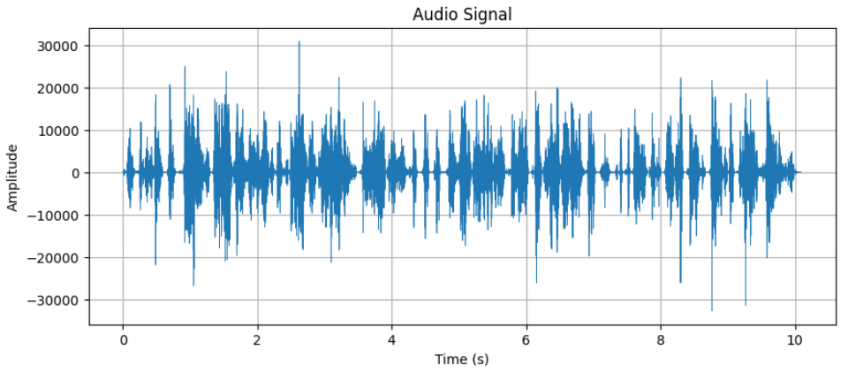


Figure 5.19: The generated signal.

The two figures above shows how the system is doing,we can see that it generates audio samples from its evaluation set and even for the sentences that was trained on.

5.5.3 Disscusion

The adaptation of Tacotron 2 for Arabic Text-to-Speech (TTS) synthesis presents intriguing results and challenges. One notable observation is the striking resemblance between the generated signals and the predicted spectrograms in terms of overall audio quality and spectral characteristics. This outcome underscores the potential of Tacotron 2 in capturing the complex phonetics and acoustic nuances of the Arabic language. It suggests that the model can effectively generate natural-sounding Arabic speech, a crucial factor for user acceptance and engagement.

However, the challenge of alignment in the adaptation process is a noteworthy concern. The alignment graphs reveal sub-optimal alignments between the encoder and decoder, which can lead to mispronunciations and unnatural prosody in the synthesized speech. This misalignment issue may arise due to the intricacies of Arabic script, which features diacritics, ligatures, and variations not encountered in English. Addressing alignment challenges is crucial for enhancing the intelligibility and naturalness of the synthesized Arabic speech.

Another notable challenge is the character embedding and text preprocessing. The use of Unicode character symbols instead of diacritic Arabic and English cleaners can impact the model's ability to accurately interpret and synthesize Arabic text. The choice of text preprocessing techniques greatly influences the quality of the synthesized speech, and in the context of Arabic, it's imperative to develop tailored preprocessing methods that consider the unique characteristics of the language.

CHAPTER 6

Conclusions

In this project we have explored neural speech synthesis and different methods that are the most popular in this domain.

We have explored the wavenet vocoder and theirs different variants as the Gaussian wavenet and logistic wavenet .This two models showed that are really useful and practical.Gaussian WaveNet excels in probabilistic waveform generation, providing diverse and expressive audio samples, making it ideal for applications like high-quality text-to-speech synthesis. Logistic WaveNet is designed for discrete audio synthesis. The choice depends on the need for naturalness and expressiveness (Gaussian WaveNet) or discrete signal modeling (Logistic WaveNet) in audio generation tasks.

We have studied Tacotron 2 and the modified version and concluded that the choice between Tacotron 2 and its modified version with the Griffin-Lim vocoder depends on specific application requirements. While Tacotron 2 excels in delivering high TTS quality, the modified version offers a computationally lighter alternative without significantly compromising speech quality.

The results demonstrated in adapting Tacotron 2 on Arabic language are very promising. The challenges of alignment and text preprocessing cannot be overlooked. Future work in adapting Tacotron 2 for Arabic TTS should focus on overcoming these challenges. It will be crucial for realizing the full potential of Tacotron 2 in providing natural and expressive Arabic TTS, thereby benefiting various applications, including assistive technology and multimedia content generation.

Bibliography

[1] Paul Taylor. *Text-to-speech synthesis*. Cambridge university press, 2009.

[2] Thierry Dutoit. *An introduction to text-to-speech synthesis*, volume 3. Springer Science & Business Media, 1997.

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

[4] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[5] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[6] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32, 2019.

[7] Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. Deep voice: Real-time neural text-to-speech. In *International conference on machine learning*, pages 195–204. PMLR, 2017.

[8] T. Styger and E. Keller. Formant synthesis. In E. Keller, editor, *Fundamentals of Speech Synthesis and Speech Recognition: Basic Concepts, State of the Art, and Future Challenges*, pages 109–128. John Wiley, Chichester, 1994.

[9] D.H. Klatt. Review of text-to-speech conversion for english. *Journal of the Acoustical Society of America*, 82(3), 1987.

[10] B. Kroger. Minimal rules for articulatory speech synthesis. In *Proceedings of EUSIPCO92*, pages 331–334, 1992.

[11] T. Dutoit. High-quality text-to-speech synthesis: an overview. *Journal of Electrical & Electronics Engineering*, 17:25–37, 1999. Special Issue on Speech Recognition and Synthesis.

[12] B. Möbius. Rare events and closed domains: Two delicate concepts in speech synthesis. In *Proceedings of the 4th ESCA Workshop on Speech Synthesis*, Perthshire, Scotland, 2001.

[13] Yannis Pantazis and Yannis Stylianou. On the detection of discontinuities in concatenative speech synthesis. pages 89–100, 01 2005.

[14] Heiga Zen, Keiichi Tokuda, and Alan W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064, 2009.

[15] L. E. Baum, T. Petrie, G. Souled, and N. A. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chain. *The Annals of Mathematical Statistics*, 41(1):249–336, 1970.

[16] S. J. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, A. Povey, V. Calchev, and P. Woodland. *The HTK Book*. 1995–2006.

[17] HTS Website. Hts - hidden markov model toolkit. https://hts.sp.nitech.ac.jp/, Accessed on 2023-09-22.

[18] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech synthesis based on hidden markov models. *Proceedings of the IEEE*, 101(5):1234–1252, 2013.

[19] K. Tokuda et al. Hidden semi-markov model based speech synthesis. In *InterSpeech*, page 1185–1180, 2004.

[20] K. Tokuda et al. An introduction of trajectory model into hmm-based speech synthesis. In *ISCA SSW5*, 2004.

[21] M. Eichner et al. Speech synthesis using stochastic markov graphs. In *ICASSP*, page 829–832, 2001.

[22] K. Tokuda, H. Zen, and A. Black. An hmm-based speech synthesis system applied to english. In *IEEE Speech Synthesis Workshop*, 2002.

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

[24] Chayan Chatterjee, Linqing Wen, Kevin Vinsen, Manoj Kovalam, and Amitava Datta. Using deep learning to localize gravitational wave sources, 09 2019.

[25] Roumiana Ilieva and Yoto Nikolov. Artificial neural networks with java implementation. In *2019 II International Conference on High Technology for Sustainable Development (HiTech)*, pages 1–5. IEEE, 2019.

[26] John Smith and Mary Johnson. A comprehensive study of food science, 2022. Accessed on august 25, 2023.

[27] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer, 1995.

[28] Eric W Weisstein. Hyperbolic functions. *https://mathworld. wolfram. com/*, 2003.

[29] Jason Brownlee. A gentle introduction to the rectified linear unit (relu). *Machine learning mastery*, 6, 2019.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 6.2. 2.3 softmax units for multinoulli output distributions. *Deep learning*, 180, 2016.

[31] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[32] Frauke Günther and Stefan Fritsch. Neuralnet: training of neural networks. *R J.*, 2(1):30, 2010.

[33] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, pages 1–2. Ieee, 2018.

[34] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[35] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[36] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[37] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[38] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[39] GeeksforGeeks. Convolutional neural network (cnn) architectures. https://www.geeksforgeeks.org/convolutional-neural-network-cnn-architectures/, Accessed on 2023-09-22.

[40] Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. Fundamental concepts of convolutional neural network. *Recent trends and advances in artificial intelligence and Internet of Things*, pages 519–567, 2020.

[41] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

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

[43] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1091–1100, 2018.

[44] Henry Pfister. Discrete-time signal processing. *Lecture Note, pfister. ee. duke. edu/courses/ece485/dtsp. pdf*, 2017.

[45] Maria Habib, Mohammad Faris, Raneem Qaddoura, Manal Alomari, Alaa Alomari, and Hossam Faris. Toward an automatic quality assessment of voice-based telemedicine consultations: a deep learning approach. *Sensors*, 21(9):3279, 2021.

[46] Daniel Watson, Nasser Zalmout, and Nizar Habash. Utilizing character and word embeddings for text normalization with sequence-to-sequence models. *arXiv preprint arXiv:1809.01534*, 2018.

[47] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[48] Denny Britz. Attention and memory in deep learning and nlp. http://www.wildml.com/2016/01/attentionand-memory-in-deep-learning-and-nlp/, January 2016.

[49] Daniel W. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, April 1984.

[50] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016b.

[51] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016a.

[52] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[53] Fady K Fahmy, Mahmoud I Khalil, and Hazem M Abbas. A transfer learning end-to-end arabic text-to-speech (tts) deep architecture. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 266–277. Springer, 2020.

[54] Keith Ito and Linda Johnson. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

[55] Nawar Halabi. *Modern standard Arabic phonetics for speech synthesis*. PhD thesis, University of Southampton, 2016.

[56] Ryuichi Yamamoto. Tensorflow wavenet vocoder. https://github.com/r9y9/wavenet_vocoder, Year accessed.

[57] Per Näslund. Artificial neural networks in swedish speech synthesis, 2018.

[58] Rayhane Mamah. Tacotron-2 github repository. https://github.com/Rayhane-mamah/Tacotron-2.git, 2023.