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

Towards Arabic End-to-End Neural Text-to-Speech Synthesis

Student: Sawalha Layan
Neptun Code: K250DO
Supervisor: Dr. Mohammed Salah Al-Radhi

Budapest, Hungary
June, 2023

Table of Contents

Abstract

Text-to-Speech (TTS) synthesis involves generating a speech waveform given textual input. It can be utilized for various purposes. Nowadays, the goal of TTS is not to have machines talk but to achieve a natural and human-like auditory output. However, the speech quality, linguistic complexity, multilingual support, computational requirements, evaluation and subjectivity, and lack of linguistic resources are still unsatisfactory for synthetic speech in other limited-resource languages (such as Arabic). In this thesis we overcome these challenges divided into two parts. In the first part of the thesis, we investigated different approaches for TTS, a neural network speech synthesis system, and a non-autoregressive text-to-speech (TTS) model. In the neural network speech synthesis based on statistical parameter vocoders, we showed how a baseline system based on Merlin is used for TTS synthesis which is implemented with a WORLD vocoder. Then we adapted Continuous and Ahocoder vocoders to get better results; and then we investigated the effectiveness of each vocoder's techniques to produce the highest quality speech. In the non-autoregressive TTS model, we implemented the state-of-the-results Fastspeech2 system, which provided high-quality speech synthesis promptly without controllability and robustness problems. Then we integrated the Arabic language, followed by using limited data while maintaining its high-quality produced sounds. Through objective and subjective evaluations, we verify that our method can outperform the baseline system with full data.

In the second part of the thesis, we propose a comprehensive and effective approach to Arabic speech synthesis, addressing limitations in existing methods. Our research focuses on advancing speech synthesis techniques using state-of-the-art models such as Parallel WaveGAN and AutoVocoder. These models have shown promising results in generating high-quality speech. However, when applied to the Arabic language, the performance of AutoVocoder fell short of our expectations. To address this, we adopted an approach that aims to enhance the quality of mel-spectrograms, which serve as input to the models. We employed advanced preprocessing techniques, including noise reduction, filtering, equalization, and denoising using NVIDIA MAXINE. Most Importantly, we incorporated fundamental frequency (F0) as an additional parameter in the AutoVocoder to improve the naturalness of the synthesized speech. Through these enhancements, we overcame the limitations encountered with AutoVocoder and achieved improved results in synthesizing high-quality Arabic speech.

Chapter 1 Introduction

1. Introduction

In our daily lives, communication is needed and used in every aspect of our lives, each person's unique voice remains one of the main characteristics of human speech. It's an effective way of identifying a person, even though there are many alternatives for verbal communication, we cannot deny that it can never replace it. Text-to-Speech (TTS) technology is the process of converting written text into spoken words and plays a vital role in bridging the gap between written content and spoken language [1]. TTS systems are designed to generate natural and intelligible speech, enabling effective communication and accessibility for individuals with visual impairments, reading difficulties, and language barriers. TTS enhances user experiences by providing interactive voice responses in applications such as voice assistants, virtual agents, navigation systems, and audiobooks. It also facilitates multilingual communication by supporting multiple languages and accents, enabling cross-cultural understanding and localization of content. With its ability to convert text into natural speech, TTS technology significantly contributes to inclusivity, convenience, and effective information dissemination in today's interconnected WORLD.

Speech Synthesis is used in a wide range of applications. This technology was created to assist people with impairments (especially the visually impaired) in their everyday lives. Several applications have been created that are more or less near to TTS's original value. For example, it is used to produce voices to communicate messages to customers by speech, whether or not they are impaired in the context of transportation. Today remains of TTS are quite easy to locate in our daily lives. Language translation engines are yet another example. This technique is used to advise how to pronounce the translated material to finish the textual translation [2]. Speech synthesis, also known as text-to-speech (TTS) technology, is a field of research that focuses on converting

written text into spoken words. It plays a crucial role in various applications, including assistive technologies, human-computer interaction, and entertainment [3].

Speech synthesis, also known as text-to-speech (TTS) synthesis, is a technology that converts written text into spoken words. It has various applications, including voice assistants, accessibility, and multimedia production. Speech synthesis involves analyzing the text and generating corresponding speech waveforms.

There are two main components in speech synthesis: text analysis and speech waveform generation. The text analysis component processes the input text to extract linguistic and prosodic information. It involves tasks such as text normalization, part-of-speech tagging, and prosody modeling [4]. Linguistic rules and statistical models are commonly employed in text analysis to determine the appropriate pronunciation, intonation, and emphasis for each word and phrase in the text [5]. The speech waveform generation component utilizes the extracted linguistic and prosodic information to generate the corresponding speech. Different techniques are employed in this process, including concatenative synthesis, formant synthesis, and statistical parametric synthesis.

Concatenative synthesis involves combining pre-recorded speech units, known as diphones or triphones, to form the desired speech output. These units are selected and concatenated based on linguistic and prosodic cues present in the input text [6]. Formant synthesis, on the other hand, generates speech by modeling the vocal tract using mathematical equations and manipulating the parameters to produce different phonetic sounds [7].

Statistical parametric synthesis is another widely used approach in speech synthesis. It utilizes machine learning techniques to model the relationship between the input text and speech features. The system is trained on a large dataset of text-speech pairs, where the text is aligned with the corresponding acoustic features. During synthesis, the system predicts the acoustic features directly from the input text, which are then used to generate the speech waveform [8]. Speech synthesis has advanced significantly in recent years, with the development of deep learning techniques such as recurrent neural networks (RNNs) and WaveNet. These approaches have demonstrated improved naturalness and expressiveness in synthesized speech [9] [10].

Speech synthesis is a technology that converts written text into spoken words. It involves text analysis to extract linguistic and prosodic information, followed by speech waveform generation using techniques such as concatenative synthesis, formant synthesis, and statistical parametric synthesis. Text-to-Speech (TTS) is a technology that converts written text into spoken words. It has found applications in various fields, such as accessibility, voice assistants, and multimedia production. TTS systems involve two main components: text analysis and speech synthesis. The text analysis component processes the input text to determine linguistic and prosodic features. It includes tasks such as text normalization, linguistic analysis, and prosody modeling. One popular approach is using linguistic rules and statistical models to analyze the text [11].

The speech synthesis component generates the corresponding speech waveform based on the analyzed text. Various techniques are employed, including concatenative synthesis, formant synthesis, and statistical parametric synthesis. Concatenative synthesis combines pre-recorded speech units, while formant synthesis generates speech based on acoustic modeling [12]. Statistical parametric synthesis utilizes machine learning techniques to model the relationship between text and speech features [13].

Vocoders, on the other hand, are signal-processing tools used for speech coding and synthesis. They analyze and manipulate speech signals to represent them with reduced data. Vocoders can be classified into various types, such as time-domain vocoders, frequency-domain vocoders, and source-filter vocoders [14]. Time-domain vocoders, such as the channel vocoder, analyze the temporal properties of speech signals by dividing them into short time frames and modifying their amplitudes based on an analysis of the signal Frequency-domain vocoders, such as the phase vocoder, operate in the frequency domain, analyzing and modifying the spectral properties of the speech signal Source-filter vocoders separate the source (vocal cord excitation) and filter (vocal tract) components of speech and manipulate them independently.

The main contributions of this work are summarized as follows:

1. Explore the use of statistical parameters and the Merlin toolkit in text-to-speech (TTS) systems.

2. Integrate and evaluate two vocoders, other than WORLD, namely continuous and Ahocoder, to enhance the quality of generated speech.
3. Investigate the feasibility of end-to-end neural network TTS, specifically FastSpeech2, and its integration with the Arabic language.
4. Implement TTS for the Arabic language using limited data, aiming to achieve TTS with lower dataset requirements and potentially develop zero-data TTS capabilities.
5. Research and analyze the application of the latest state-of-the-art AutoVocoder in speech synthesis.
6. Integrate the Arabic language into the AutoVocoder and conduct a comparative study with Parallel WaveGAN for speech synthesis.
7. Improve the quality of mel-spectrograms in AutoVocoder-based TTS systems by incorporating the fundamental frequency (F0) parameter.
8. Improve the overall speech quality in AutoVocoder-based TTS through the use of denoising techniques, filtering, equalization, and the addition of the F0 parameter during mel-spectrogram generation as well as adding denoising by Maxine NVIDIA to the synthesized speech.

1.1 Text-to-Speech

Text-to-speech or TTS is a software that reads text and converts it into speech. TTS converts any text-based message into a verbal message. TTS is an evolving field that provides faster messages with consistency, time, and money saving. You can prepare your message in text and send it as a voice, so you don't have to record yourself, you can also make it consistent and professional by making the communications all in the same voice. TTS is beneficial to business applications by assisting them in delivering a variety of notifications simultaneously Speech synthesis technology has transformed the way we interact with computers and digital devices. By converting written text into natural-sounding speech, it enables accessibility for individuals with visual impairments, enhances the user experience in human-computer interaction, and enables realistic virtual agents and conversational AI applications. Moreover, it finds applications in entertainment, including gaming, animation, and audiobook narration. [15].

Here, various technologies are discussed, highlighting their main specifications, differences, and methodologies. A neural network speech synthesis [16] is implemented with three

different vocoders to find the best voice quality and a non-autoregressive TTS is implemented with limited data and multi-languages.

The artificial creation of human voices is referred to as speech synthesis (TTS). The capacity to mechanically convert a text into a spoken voice is the purpose. TTS will be built on graphemes, which are the letters and groupings of letters that transcribe a phoneme, as opposed to speech recognition systems, which employ phonemes (the smallest units of voice) to chop out sentences in the first place. This suggests that the text is the most important resource. This is normally accomplished in two stages. The first will break the text down into words and sentences and assign phonetic transcriptions to each of these groupings, as shown in Figure 1. After identifying the various text/phonetic groupings, the next step is to translate these linguistic representations into sound. To put it another way, to interpret these signals to generate a voice that will read the information. The top of the line in voice synthesis has progressed through the period, allowing four generations of Text to Speech (TTS) systems to be distinguished. From the first to the last generation, there is rule-based synthesis, concatenation synthesis, based on probabilistic speech synthesis, and machine learning. For the first three generations, the block structure is the same.

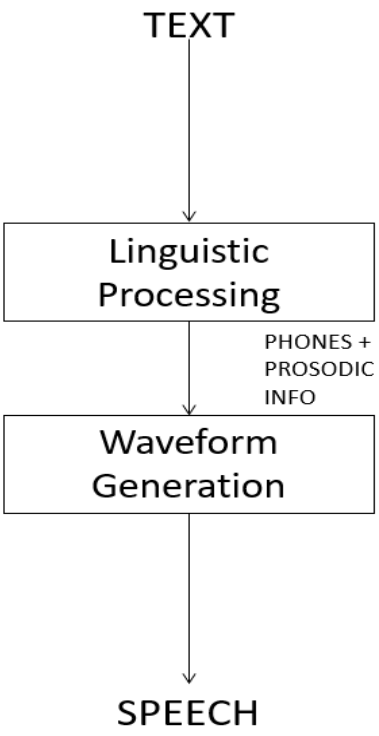


Figure 1: Block Structure for standard TTS systems.

To turn common language text into voice, TTS synthesis uses the creation of a speech waveform. It generally consists of two parts: a front end and a back end. Regularization or pre-processing, which turns abbreviations or numbers from raw user input into words, is the first of the two. The back end is commonly a synthesizer that turns linguistic information like pitch shape and phonetic duration into voice, taking into consideration the desired prosody, as shown in figure 2. The inputs for synthesis by rule are phonemes and stress marks, with a continuous waveform as the output. The approach consists of a synthesizing strategy module that includes information stored about phonemes and rules specifying the mutual effects of nearby phonemes [17].

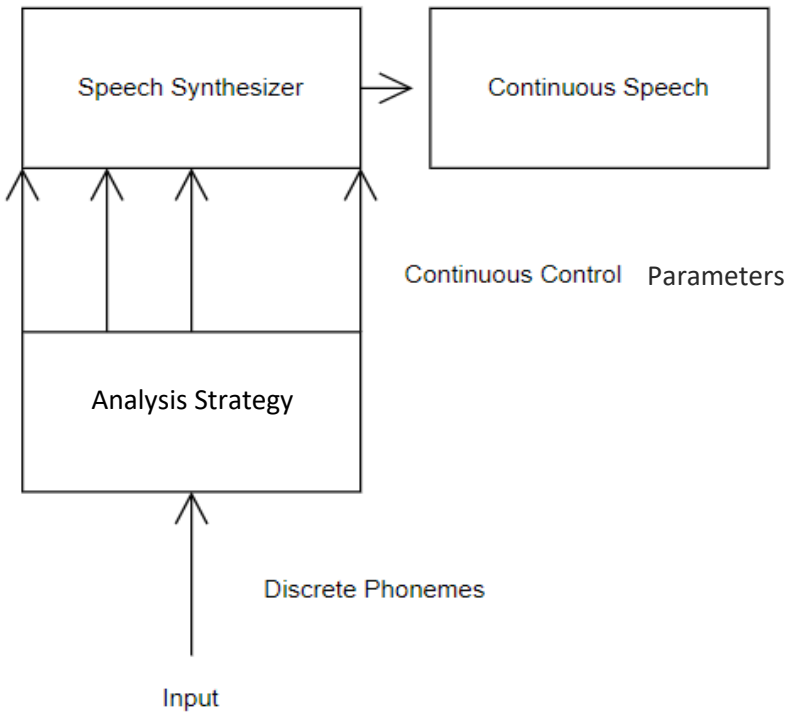


Figure 2: Technique of Speech Synthesis

Concatenative synthesis works by capturing speech, keeping it in a database, and then concatenating the bits to get the desired result. This strategy has the potential to provide excellent outcomes. However, modeling expressiveness in speech is challenging, and strategies for autonomous waveform segmentation might result in inaccuracies in the output. A novel technique termed statistical parametric voice synthesis was created to eliminate the probable output mistakes produced by concatenation. Even though both HMMs and Deep Neural Networks are being utilized today, with the advancement of processing power and current improvements in machine learning, DNNs have begun to be employed to replace them. As deep learning progressed, writers began to incorporate neural networks into existing systems, signaling the start of the fourth generation. Later, they began to design systems entirely based on DNNs, eventually reaching end-to-end systems. A DNN is a multi-layered artificial neural network (ANN) that can model complicated non-linear interactions and build compositional models. There are several variations

of this architecture. Feedforward networks including Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs) are commonly used for speech synthesis applications.

When choosing the right voice synthesis, there are several factors to consider. Variables to consider include the language spoken, the type of speaker, the quality of the voice, and the provider. Now that you have this knowledge, it will be simple to pick the best solution for your goals and constraints. It's crucial to discover these partners ahead of time because not all TTS providers offer comparable product portfolios. The language and tone of voice utilized are also important aspects of the intended user experience; the voice interface and the emotions it should evoke must be in sync. Speech synthesis is based on cloud, embedded, or hybrid technology. It's worth noting that embedded has technological limitations in terms of phrase storage that a cloud does not, but the embedded voice will function regardless of what occurs when the cloud requires a connection. These characteristics should be considered based on the nature of your projects; for example, in the transportation industry, embedded is advised to assure continuous service [18].

Text-to-speech synthesis (TTS) has become a useful component in many voice applications, such as online translators and text message readers. Furthermore, TTS is nowadays available for the most widely spoken languages all over the WORLD, the main online services. Hence, it is important to have high-quality TTS for all languages since it represents a large market with more than 300 million potential users. TTS systems have been under development for a long period and may be used for a variety of purposes. Because of the problems experienced while recording a kid and building a system to synthesize speech that sounds natural, most of the voices utilized or synthesized come from adults, or when a child's voice is synthesized, it is generally quite robotic, inexpressive, and does not sound authentic. Several approaches are currently being designed and evaluated to find at least one that meets the requirements for having a genuine human voice in a robot device [19]. In the TTS we implemented Statistical parametric-based TTS and neural networks-based TTS.

1.1.1 Statistical Parametric-based TTS

Statistical parametric-based Text-to-Speech (TTS) is a widely used approach for synthesizing speech from text. This technique employs statistical models to capture the

relationships between linguistic features and acoustic characteristics, enabling the generation of natural and intelligible speech [20]. In this paper, we provide an overview of statistical parametric-based TTS, its key components, training process, and applications.

The core component of statistical parametric-based TTS is the statistical model, which can include hidden Markov models (HMMs), Gaussian mixture models (GMMs), or deep neural networks (DNNs). These models are trained on a large dataset of text-speech pairs, where the text represents linguistic features such as phonemes, prosody, and linguistic context, and the speech corresponds to the corresponding acoustic features [21].

During the training process, the statistical model learns to map the linguistic features to the acoustic features. This involves extracting relevant linguistic and contextual information from the text and modeling their relationship with the acoustic characteristics of speech. The model parameters are estimated through various techniques, such as maximum likelihood estimation (MLE) or expectation-maximization (EM) algorithms.

Once the statistical model is trained, it can be used for synthesizing speech from new text inputs. The input text is converted into linguistic features, and the model predicts the corresponding acoustic features. These predicted features are then transformed into a time-domain waveform using techniques like vocoding or signal processing algorithms. Several vocoders are commonly used in statistical parametric-based TTS, including the WORLD vocoder, Continuous vocoder, and Ahocoder. The WORLD vocoder is widely used for speech synthesis and provides good quality in terms of naturalness and intelligibility. The Continuous vocoder is known for its ability to generate high-quality speech with smooth spectrotemporal contours. The Ahocoder, on the other hand, is designed to handle the analysis and synthesis of speech with high efficiency and low computational complexity.

Statistical parametric-based TTS has found applications in various domains, including assistive technology, human-computer interaction, and multimedia content generation. It enables the creation of natural and expressive synthetic voices that can be customized for specific applications or personalized to mimic specific speakers. Additionally, it facilitates multilingual speech synthesis by adapting the statistical model to different languages.

Several studies have explored different aspects of statistical parametric-based TTS, including improving the modeling of linguistic features, enhancing the synthesis quality, and reducing the data requirements for training. For instance, researchers have investigated techniques such as deep neural networks (DNNs) for better modeling of linguistic context and long-term dependencies, as well as incorporating speaker adaptation methods to enhance the voice quality [22]. Statistical parametric-based TTS is a powerful approach for synthesizing speech from text. By leveraging statistical models, it captures the complex relationships between linguistic features and acoustic characteristics, enabling the generation of natural and intelligible speech.

1.1.2 Neural TTS

A neural network is a type of computer architecture that mimics the way the human brain works. Where the brain has the ability to process data through complex connections between different neurons, and when those connections develop it requires less effort to activate them again, where this procedure in neural networks can be called learning. A neural network is a type of machine learning which can be applied to different sectors such as image processing and text-to-speech. Where deep neural networks are when they consist of three or more processing layers. The first layer is the input layer which classifies the input data then it passes through one or more hidden layers that help to refine the signal and sort it into complex classifications, followed by the last layer which is the output layer produces the results, for example in TTS it produces an audio signal.

To create a neural TTS voice, we train DNN models using a recording of human speech. In neural networks, we use three different DNN models: acoustic, pitch, and duration models. In the acoustic model, a reproduction of the timbre of the speaker's voice is created. While the pitch model predicts the different ranges of the tone in speech. In the last model, the duration model predicts how long the voice should hold for each phoneme. Neural networks have prosodic parameters that determine prosody properties of speech like breaks, rhythms, and intonation. Those parameters include the pitch and duration, prediction models [23].

In the neural network TTS, there are three main components as shown in figure 3. The input is the text where the linguistic features are extracted then in the acoustic model, we extract acoustic features that enter the vocoder to produce the resulting waveform.

Figure 3: Components of Neural TTS

In line with the advancement of deep learning in various fields, deep neural network (DNN)-based TTS has become increasingly attractive to researchers in recent years. Some advantages of this end-to-end DNN-based TTS system are more ease for conditioning on various attributes, such as speakers, language, prosody, speaking style, sentiment, and more ease for new data adaptation.

Deep Neural Networks (DNN) are capable of training a large amount of data. It uses mathematical modeling to process data in a complex way. DNN components are trained independently which is considered difficult and has lots of errors. To address the extensive usage of domain expertise in the components of DNN, end-to-end speech synthesis is introduced. End-to-end speech synthesis methods combine the methods in a unified framework. In the next chapters, we will discuss end-to-end speech synthesis divided into autoregressive text-to-speech and non-autoregressive text-to-speech [24].

The end-to-end framework is used to simplify the existing pipeline to build a neural network-based text-to-speech system as it directly models complex sequential mapping from a text sequence to its acoustic path. The term “end-to-end” refers that text analysis and acoustic modeling completed by an attention-based network that can learn all the embedded factors in the text sequence, which helps overcome the problems with conventional text-to-speech systems [25].

In this thesis we implemented an end-to-end TTS framework is superior to conventional TTS as it can automatically learn alignments between discrete text sequences and acoustic feature sequences during training, which eliminates the usage of frame-by-frame alignments. Another advantage is that it can easily generate smooth spectral parameters and can synthesize speech on very short utterances which support the integration of TTS in another language. It can also map sequences directly from a text string to an acoustic trajectory, where the text analysis and acoustic modeling are integrated into a unified model.

Another advantage is that it needs minimal human annotation in the training stage. It can also alleviate the need for laborious feature engineering. The conditioning occurs at the very

beginning of the model which allows rich conditioning for various attributes that make it beneficial for multiple languages, speakers, or expressive TTS. The adaptation of new data is also easier when compared to typical TTS. A single model is also likely to be more robust than a multi-stage model.

The end-to-end model is beneficial when training a huge amount of expressive and rich datasets. Where TTS is a large-scale inverse problem, as text which is considered a highly compressed source is decompressed into audio [26].

In end-to-end based text-to-speech systems, there are two types: autoregressive and non-autoregressive models. The output of an autoregressive model is dependent on model outputs from previous inputs, unlike a non-autoregressive model. Both models may have internal states, which are transferred between each time step, nonlinear autoregressive exogenous (NARX) model is an example of an autoregressive without an internal state while the non-linear state space (NLSS) model is an example of a non-autoregressive model with an internal state. These models can be used for prediction or simulation. In prediction, it estimates the limited amount of time steps ahead with information from the past systems, while in the simulation it estimates the system output only from the input, as it focuses on present work [27].

In the autoregressive model that generates mel-spectrogram from text, the autoregressive neural vocoder synthesizes raw waveform from the mel-spectrogram. The autoregressive models are slow in the synthesis part because they operate sequentially at a high resolution of waveform samples and spectrogram. Autoregressive neural networks require more time for training and inference than non-autoregressive, without any benefits for accuracy. It also limits the optimization of hyperparameters and model capacity.

The autoregressive architecture consists of three components, encoder, decoder, and converter. The encoder takes text inputs and encodes them into hidden representations. While the decoder decodes the encoder representation with an attention approach in an autoregressive manner. As for the converter, it provides a non-casual convolutional post-processing network, that process is a hidden representation from the decoder using past and future context information and predicts a log-linear spectrogram, as shown in Figure 4.

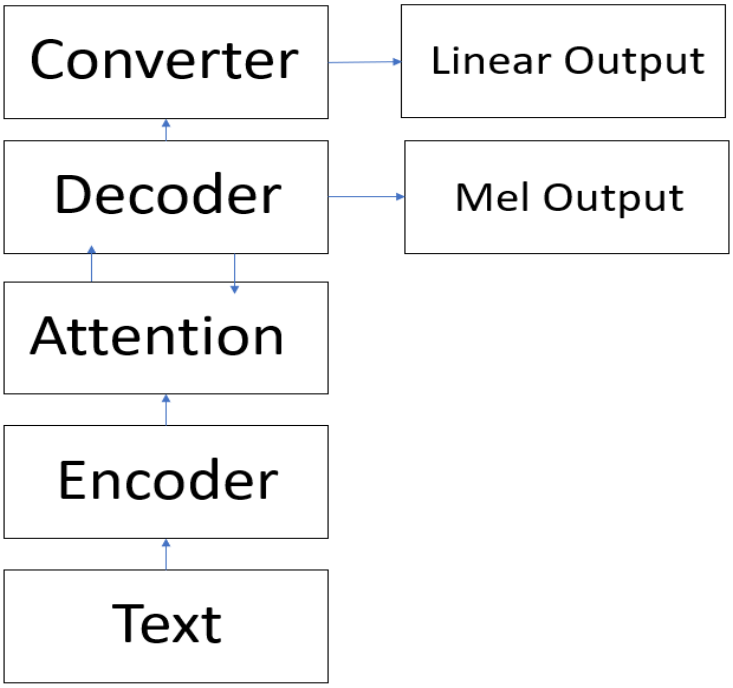


Figure 4: Autoregressive TTS Architecture

Non-autoregressive neural networks outperform all other neural network-based systems. It is also faster, easier and more accurate to implement when compared to autoregressive. It is also more consistent and outperforms all other neural network-based systems.

The architecture of the non-autoregressive consists of an encoder, a non-autoregressive decoder, and without a converter [28]. The encoder is similar to that of the autoregressive. While the non-autoregressive decoder uses a non-casual convolution block to take advantage of the context information to improve model performance. It also predicts long-linear spectrogram which provides better performance. As for the converter, it removes the non-casual converter since it employs a non-casual decoder, as shown in Figure 5.

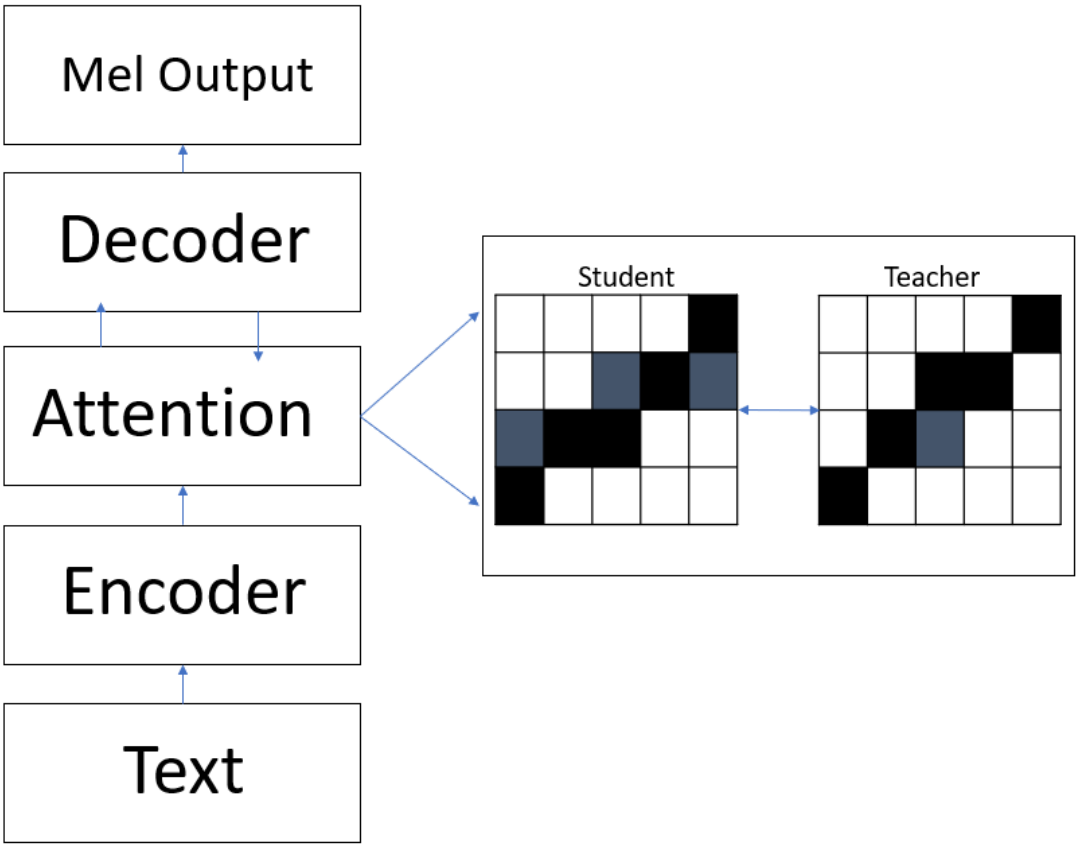


Figure 5: Non-autoregressive TTS Architecture

1.2 Speech Synthesis

Speech synthesis refers to the general process of generating artificial speech from various sources, which can include not only textual input but also other forms of data such as phonetic transcriptions or control parameters. It encompasses a broader range of techniques and applications beyond just converting text to speech. Speech synthesis techniques can involve statistical models, concatenative synthesis, formant synthesis, or other approaches to generate speech from different types of input [29].

Speech synthesis, also known as text-to-speech (TTS) synthesis, is the process of generating artificial speech from written or textual input. It plays a vital role in various applications, including virtual assistants, audiobooks, accessibility tools, and interactive voice response systems [30]. The goal of speech synthesis is to create natural, intelligible, and expressive speech output that closely resembles human speech. Over the years, significant advancements have

been made in the field of speech synthesis, driven by advancements in machine learning, deep learning, and signal processing techniques.

One of the key components of speech synthesis is the linguistic analysis of the input text. This involves breaking down the text into smaller linguistic units such as phonemes, words, or linguistic features [31]. Linguistic analysis helps in capturing the appropriate pronunciation, prosody, and intonation patterns required for generating natural-sounding speech. Various linguistic models, such as rule-based models, statistical models, or neural network-based models, are used to extract linguistic information from the input text.

Another crucial aspect of speech synthesis is acoustic synthesis, which involves generating the actual speech waveform from the linguistic information [32]. Acoustic synthesis can be achieved using various techniques, including formant synthesis, concatenative synthesis, or statistical parametric synthesis. Formant synthesis produces speech by manipulating the frequencies of the vocal tract resonances, while concatenative synthesis concatenates pre-recorded speech segments to create the desired output. Statistical parametric synthesis uses statistical models to predict the acoustic features of speech based on linguistic information.

In recent years, deep learning approaches, particularly neural network-based models, have shown significant improvements in speech synthesis. Deep neural network architectures, such as WaveNet and Tacotron, have been successfully applied to generate high-quality and natural-sounding speech. These models leverage large datasets and complex neural network structures to capture the intricate details of speech and produce more accurate and expressive speech output [33].

The evaluation of speech synthesis systems involves both objective and subjective assessments. Objective measures such as mel-cepstral distortion (MCD) and segmental signal-to-noise ratio (SNRseg) provide quantitative evaluations of the quality and similarity between the synthesized and natural speech [34]. Subjective evaluations, on the other hand, involve human listeners rating the quality, naturalness, and expressiveness of the synthesized speech through listening tests.

Speech synthesis is a rapidly evolving field that aims to generate artificial speech from written or textual input. Advancements in machine learning and signal processing techniques have significantly improved the quality and naturalness of synthesized speech [35]. Ongoing research

focuses on enhancing the expressiveness, robustness, and multilingual capabilities of speech synthesis systems. With the continuous advancements in technology, speech synthesis holds great potential for further improvements and opens up exciting possibilities for applications in human-computer interaction, multimedia content generation, and assistive technologies.

Neural speech synthesis using AutoVocoder is an approach that leverages advanced neural network models to generate high-quality artificial speech. AutoVocoder is a powerful technique that combines the power of autoencoders and vocoders to produce natural-sounding speech. In this thesis, we explored the application of AutoVocoder in the context of Arabic speech synthesis, utilizing an Arabic speech corpus for training and evaluation [36].

The AutoVocoder model consists of two main components: an encoder and a decoder. The encoder takes the input speech waveform and encodes it into a latent representation, capturing the essential characteristics of the speech [37]. The decoder then takes this latent representation and generates the synthesized speech waveform, reconstructing the original speech based on the learned representation. The use of autoencoders allows the model to learn a compact and informative representation of the speech data, which can be used for synthesis.

In our research, we utilized an Arabic speech corpus containing a significant amount of Arabic speech data. This corpus was carefully collected and preprocessed to ensure the availability of high-quality and diverse Arabic speech samples [38]. By training the AutoVocoder model on this corpus, we aimed to capture the unique phonetic, prosodic, and acoustic characteristics of the Arabic language, enabling the generation of realistic Arabic speech [39].

The utilization of the Arabic speech corpus in training the AutoVocoder model allowed us to overcome the challenges specific to Arabic speech synthesis. Arabic is a highly complex and rich language with distinct phonetic features and phonological rules [40]. By using the Arabic speech corpus, we aimed to train the AutoVocoder to learn and reproduce these characteristics accurately, resulting in improved speech quality and naturalness in the synthesized Arabic speech output.

In this research, we also implemented ParallelWaveGAN, which is a powerful waveform generation model that has gained significant attention in the field of speech synthesis. It has shown remarkable performance in producing high-quality and natural-sounding speech signals. In the context of Arabic speech synthesis, ParallelWaveGAN holds great potential for generating fluent

and intelligible Arabic speech. By leveraging the capabilities of ParallelWaveGAN and integrating them with the unique characteristics of the Arabic language, we aim to explore its applicability and effectiveness in generating authentic Arabic speech. This research endeavors to address the challenges specific to Arabic speech syntheses, such as capturing the nuances of pronunciation, prosody, and intonation patterns that are intrinsic to the Arabic language. Through a comprehensive evaluation and comparison of the synthesized Arabic speech with the original recordings, this study aims to assess the fidelity, naturalness, and overall quality of the generated speech using ParallelWaveGAN. The findings from this research have the potential to significantly contribute to the advancement of Arabic speech synthesis and open up new avenues for the development of high-quality speech generation systems for the Arabic language.

1.3 Problem Definition

This research project addresses several important aspects of text-to-speech (TTS) systems. Firstly, it aims to explore the use of statistical parametric speech synthesis and the Merlin toolkit in TTS systems. Secondly, the project focuses on integrating and evaluating two alternative vocoders, continuous and Ahocoder, to enhance the overall speech quality in TTS systems beyond the conventional WORLD vocoder. Thirdly, it investigates the feasibility of utilizing the FastSpeech2 model, an end-to-end neural network architecture, for TTS applications with a specific focus on the Arabic language, aiming to improve speech synthesis quality. Additionally, the project aims to implement TTS for Arabic using limited data, aiming to achieve TTS capabilities with lower dataset requirements and potentially develop zero-data TTS capabilities. It also researches and analyzes the application of the latest state-of-the-art AutoVocoder in speech synthesis, with a particular emphasis on integrating the Arabic language into the AutoVocoder and conducting a comparative study with Parallel WaveGAN. Furthermore, the project aims to enhance the quality of mel-spectrograms in AutoVocoder-based TTS systems through advanced preprocessing techniques such as denoising using Maxine NVIDIA [41]. Finally, it aims to improve the overall speech quality in AutoVocoder-based TTS systems by incorporating denoising techniques, filtering, equalization, and the addition of the F0 parameter during mel-spectrogram generation. Through these investigations, the project aims to advance the field of TTS and contribute to the development of high-quality and natural-sounding speech synthesis systems.

Text-to-Speech is an evolving field, where different systems have been under development for a long period and may be used for a variety of purposes. Because of the problems experienced with the synthesized voice, which was quite robotic, inexpressive, and does not sound authentic. We tried to integrate, evaluate, and implement different vocoders to find the highest quality of synthesized speech that met the requirements for having a genuine human voice in a robot device. TTS is often implemented or is applicable in one language which most of the time is in English. One of the main reasons for it is because it has a good infrastructure which in case it's the datasets as well as its lower complexity when compared with other languages. In this study, the Arabic Language is integrated, which was very challenging as there are very few free-speech corpora. The ultimate goal is also to be able to customize TTS to not only different languages but also to different personal voices with limited data as it will be so costly to collect a sufficient amount of dataset from the target voice.

In order to achieve high Text-to-Speech results, we often need a huge dataset which causes a barrier when it comes to implementing it in different languages. Nowadays there are almost seven thousand spoken languages with insufficient datasets which constrains the applicability of TTS. Despite Arabic being one of the most widely spoken languages in the WORLD, there is an insufficient number of datasets to develop perfect text-to-speech and excellent speech similar to that of a native human speaker.

The proposed research also focuses on addressing the challenges in Arabic speech synthesis using the AutoVocoder model. This involves enhancing the mel-spectrogram representation by employing preprocessing techniques such as noise reduction, equalization, and filtering to improve the quality and fidelity of the mel-spectrograms. Advanced training strategies and loss functions specific to Arabic speech synthesis will also be explored to optimize the AutoVocoder model. By improving the representation of Arabic speech in the mel-spectrograms, the research aims to advance the state-of-the-art in Arabic speech synthesis, enabling applications such as assistive technologies, language learning tools, and interactive voice response systems to provide high-quality and expressive speech output in the Arabic language. The outcomes of this research have the potential to contribute significantly to the field of Arabic speech synthesis and pave the way for more natural and intelligible speech synthesis systems in Arabic.

Chapter 2
Methodology

2. Methodology

2.1 Text-to-Speech

2.1.1 TTS with Full Data

In this project, Merlin was implemented. Merlin has some of the features required to build a text-to-speech system. It necessitates the use of a front-end and a vocoder, but neither is required. Data with aligned labels are also needed to train a DNN. Front-end Merlin relies on an exterior front-end, like Festival or Ossian, for DNN input. For every front end, its output must be structured as HTS-style labels, with alignment at the phone or state level. The toolbox offers routines for converting such labels into binary and continuous different feature sequences. These features are extracted from the label files using HTS-style queries, with a modest addition to allow for the extraction of continuously valued features. If the HTS-like approach isn't handy, it's also feasible to give already-vectorized input features. Vocoder STRAIGHT and WORLD are the only vocoders supported by Merlin for now. A modified version of the WORLD vocoder is included in the Merlin release, as are separate analysis and synthesizing executables. Fixed and variable frame rates (such as pitch synchronized) are supported by Merlin. Data To acquire state-level aligns for the training data, HTK or HTS can be employed. Merlin may alternatively rely solely on phone level alignments, which can be determined using other tools like the festvox cluster gene. Duration modeling Merlin uses a different DNN from the acoustic model to model duration. The duration model is trained to estimate phone- and/or state-level durations on the matched data. At synthesis time, first, the duration is predicted, then the acoustic model is utilized to forecast the speech characteristics.

Merlin may be implemented in one of two ways: Demo or Full Voice. In order to download Merlin, we need the following dependencies: NumPy, scipy, matplotlib, bandmate, Theano,

TensorFlow, sklearn, Keras, and h5py. The key distinction between them is the number of utterances utilized, which is 50 in one case and 1132 in the other. Each 14 training should last 5 minutes if it is Demo and 1 to 2 hours if it is Full voice, however, this may vary depending on the system and its features. Installation Because the Merlin toolkit runs on Linux [42].

In this work, the Merlin toolkit was implemented. Merlin has some of the features required to build a text-to-speech system. It necessitates the use of a front-end and a vocoder. Merlin is implemented using only the WORLD vocoder [43] but in this study, we integrated different vocoders Continuous and Ahocoder [44].

The vocoder is a component of various speech synthesis applications such as TTS, voice conversion, etc. There are different types of vocoders with similar strategies [45], the first stage is the analysis which is used to convert speech into parameters that present the vocal fold signal and vocal tract filter separately into the excitation signal. In the synthesis stage, the parameter is used to reconstruct the original speech signal.

In recent years, the development of statistical parametric speech synthesizers and voice conversion systems has also pushed research toward vocoding techniques, in this project three different vocoders were implemented, WORLD vocoder, continuous vocoder, and Ahocoder vocoder. The vocoder is a component of various speech synthesis applications such as text-to-speech, synthesis, voice conversion, etc. There are different types of vocoders with similar strategies, the first stage is the analysis which is used to convert speech into parameters that present the local old signal and vocal tract filter separately into the excitation signal. In the synthesis stage, the parameter is used to reconstruct the original speech signal. Despite having different vocoders but the sound quality is degraded when compared to natural sound. In this project. In all types of vocoders four different types of voices (2 Females and 2 Males) which are SLT arctic, BLT arctic, AWB arctic, and CLB arctic.

WORLD vocoder was used in the first part of the experience which is free software for high quality speech analysis and synthesis. This vocoder is designed for integration systems, it estimates F0, aperiodicity, and spectral envelope. The baseline WORLD vocoder is an open-source speech analysis, modification, and synthesis software. It can calculate the fundamental frequency (F0) ([fundamental-frequency-estimation]), aperiodicity, and spectral envelope, as well as create speech using only estimated parameters [46]. As shown in Figure 6, shows the parameters of WORLD vocoder with Fundamental Frequency (acoustics), spectral envelope (MGC), and BAP.

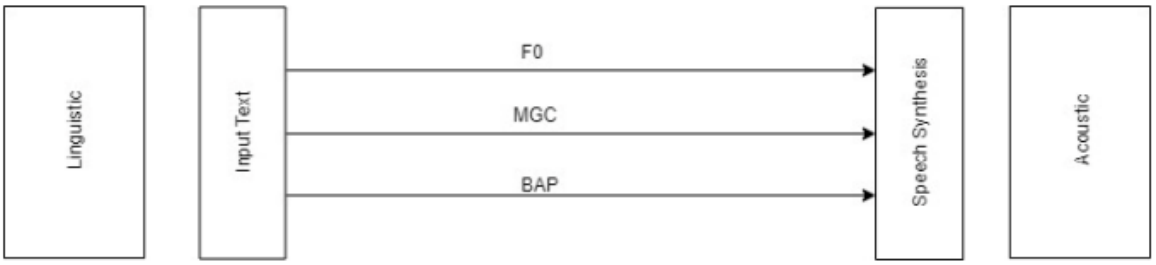


Figure 6: Parameters of WORLD vocoder.

The second vocoder we implemented was the Continuous vocoder which is used to overcome the shortcoming of discontinuity in the speech parameters and the computational complexity of modern vocoders. The most important thing about this vocoder is that it does not need to have voiced/unvoiced decisions. During the analysis phase, F0 is calculated on the input waveforms of a simple continuous pitch tracker [47] In regions of creaky voice and in case of unvoiced sounds or silences, this pitch tracker interpolates F0 based on a linear dynamic system and Kalman smoothing. After this step, Maximum Voiced Frequency (MVF) is calculated from the speech signal, resulting in the MVF parameter [48]. In the next step, 24-order MelGeneralized Cepstral analysis (MGC) is performed on the speech signal with alpha=0.42 and gamma=-1/3. In all steps, a 5 ms frameshift is used. The results are the F0, MVF, and MGC parameter streams.

The most important thing about the vocoder is that it does not need to have voiced/unvoiced decisions, so the alignment error is avoided between voice and unvoiced segments in SVC. As a result of its simplicity and versatility, we can build a voice converter framework with an FF-DNN [49]. The proposed method's performance strengths and limitations for different speakers were emphasized using several metrics. In the training process, the first part is to train duration the of the model, and the second part is to train acoustic the model. As shown in Figure 7.

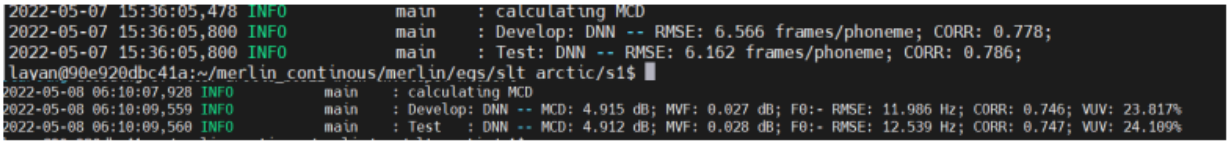


Figure 7: Training process of continuous vocoder

In Figure 8, It shows the continuous parameters: Fundamental frequency (F0), maximum voiced frequency (MVF), and spectral envelope (MGC)

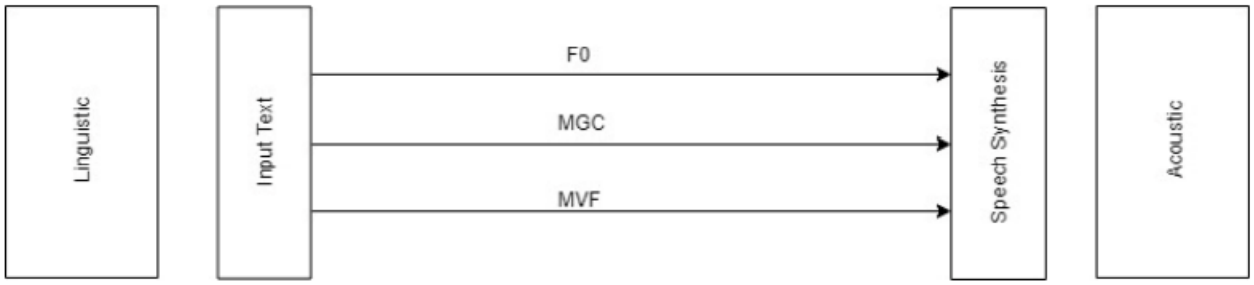


Figure 8: Parameters of Continuous and Ahocoder vocoders

` The last vocoder we integrated into the Merlin framework is Ahocoder, which divides voice frames into three streams: F0, MVF, and spectrum. Both F0 and MVF are scalars: F0 can be determined by any accurate method. The Ahocoder divides voice frames into three streams: F0, MVF, and spectrum. Both F0 and MVF are scalars: F0 can be determined by any accurate method. The method used is a pitch detection algorithm that returns the MVF values at the analysis frames' center. P+1 cepstral coefficients are used to represent the spectrum [50]. This distribution of Ahocoder contains two executable binary files built using GCC 4.4 under Linux (64bits): Ahocoder16 translates waveforms into parameters and ahodecoder16 translates parameters into synthetic waveforms. There are voiced or unspoken types, to extract their cepstral information, and frames are treated differently. If the input frame was labeled as voiced, a harmonica is produced by the pitch detector. A harmonic analysis based on the least squares is performed by the pitch detector. The complete analysis is subjected to squares optimization to obtain the harmonic amplitudes at various frequencies. These amplitudes are considered distinct, even at high resolution, samples of the real spectral envelope frequencies with a low harmonics-to-noise ratio. Unvoiced frames are subjected to a quick Fourier analysis FFT, which is also known as a harmonic transform analysis with F0 equal to FFT resolution to be able to homogenize the discrete spectrum representation, the harmonic amplitudes at voiced frames provide an envelope is resampled at the FFT after being normalized in amplitude interpolation for resolution [51].

During the final step of the procedure, cepstral coefficients, and analysis are calculated using the amplitude of the following spectral, to begin, a conventional cepstrum is obtained as

follows: the log-amplitude spectrum's inverse FFT Then, the cepstrum's frequency is distorted to meet the Mel scale which describes the recursion. The Ahocoder includes linguistic processing and builds voices for some languages, such as English, Spanish, etc. The engine is acoustic, and it uses a high-quality vocoder.

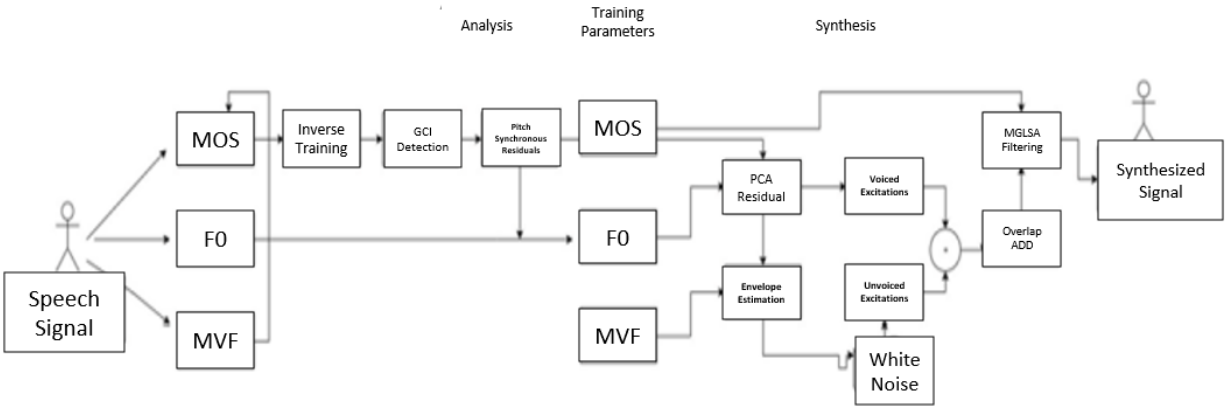


Figure 9: Workflow of the Ahocoder.

It is discovered that with HMM, an F0 used in the Ahocoder produces a more expressive F0 proposes a new method for improving HMM-based TTS modeling piecewise F0 trajectory with voicing intensity and voiced/unvoiced decision. Proposed the F0 estimator, which is employed in this vocoder is capable of keeping up with rapid changes. The technique, as shown in Figure 9, begins by separating the data. Voice signal into frames that overlap each frame's windowing result is then used to the autocorrelation function should be calculated. The Kalman smoother relies on identifying a peak between two frequencies and calculating the variance to get a final sequence of values. There is no voiced/unvoiced decision in continuous pitch estimates. Furthermore, the Glottal Closure Instant (GCI) algorithm is applied throughout the analysis phase. In the vocal regions of the inversion, to find the glottal period boundaries of particular cycles residual signal filtered. A Principal Component Analysis was performed on these pitch cycles (PCA). To produce better results, a residual is built, which will be used in the synthesis.

The method used in Ahocoder is a pitch detection algorithm that returns the MVF values at the analysis frames center. Cepstral coefficients are used to represent the spectrum. This

distribution of Ahocoder contains two executable binary files built using GCC 4.4 under Linux (64bits): Coder translates waveforms into parameters; and Decoder translates parameters into synthetic waveforms. There are voiced or unspoken types, to extract their cepstral information, and frames are treated differently. If the input frame was labeled as voiced, a harmonica is produced by the pitch detector. A harmonic analysis based on the least squares is performed by the pitch detector. The complete analysis is subjected to squares optimization to obtain the harmonic amplitudes at various frequencies. These amplitudes are considered distinct, even at high resolution, samples of the real spectral envelope frequencies with a low harmonics-to-noise ratio. Unvoiced frames are subjected to a quick Fourier analysis (FFT), which is also known as a harmonic transform analysis with F0 equal to FFT resolution to be able to homogenize the discrete spectrum representation, the harmonic amplitudes at voiced frames provide an envelope is resampled at the FFT after being normalized in amplitude interpolation for resolution. The Aho-coder includes linguistic processing and builds voices for some languages, such as English, Spanish, etc.

As a result, we can build a TTS framework with a feed-forward deep neural network (FF-DNN) as shown in Figure 10.

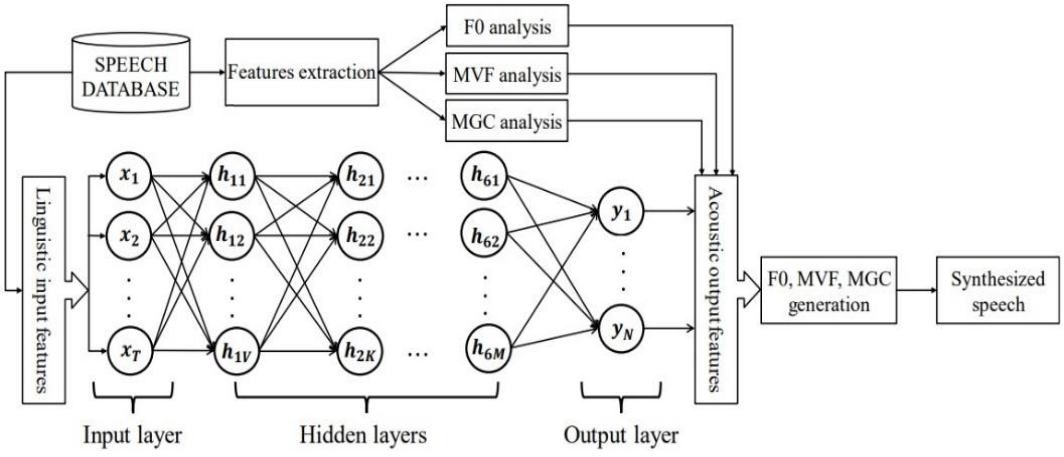


Figure 10: Proposed Diagram of TTS

2.1.2 TTS with Limited Data

To avoid the problems caused by neural network-based text-to-speech, we chose to implement a text-to-speech system using end-to-end based text-to-speech. Then we chose a non-autoregressive approach to avoid slow interference speed and robust issues that are caused by the autoregressive approach.

Fastspeech is proposed to solve these issues, it adopts a feed-forward Transformer network to generate mel-spectrograms in parallel to increase the inference speed. It also removed the attention mechanism between text and speech to avoid word skipping and repeating issues and improve robustness. It uses a length regulator to bridge the length mismatch between the phoneme and mel-spectrogram sequences instead. The length regulator controls the duration predictor to predict the duration of each phoneme and expands the hidden sequence. It also gives several advantages such as fast inference speed, and robust speech synthesis without word skipping and repeating issues.

But in this experiment, we implemented FastSpeech2 because it provides better results than FastSpeech. FastSpeech2 achieves better voice quality than FastSpeech and maintains the advantages of fast, robust, and controllable speech synthesis. FastSpeech 2 uses ground-truth mel-spectrograms as training targets, instead of distilled mel-spectrograms from an autoregressive teacher model which simplifies the two-stage teacher-student distillation pipeline in FastSpeech and avoids the information loss in target mel-spectrograms after distillation. It also provides more variance information such as pitch, duration, and energy as decoder input, which eases the one-to-many mapping problem. FastPitch improves FastSpeech by using pitch information as decoder input.

FastSpeech2 [52] simplifies the training pipeline and overcomes the information loss as it is trained directly by a ground-truth target. Variation information of speech such as pitch, energy, and accurate duration are introduced to reduce the gap between input (text sequence) and target output (Mel-spectrogram) which reduces the one-to-many mapping problem. In the training phase the duration, pitch, and energy from the target speech wave-form are extracted as conditional inputs while in the inference, predicted values from the predictor are jointly trained with the Fastspeech2 model. Using a continuous wavelet, the pitch contour is transformed into a pitch spectrogram which predicts the pitch in the frequency domain leading to improved accuracy of the predicted pitch.

As shown in Figure 11, phoneme embedding is converted using the encoder to phoneme hidden sequence, where the variance adaptor adds variance information such as pitch, energy, and duration into the phoneme hidden sequence, then the Mel-spectrogram decoder converts the adapted hidden sequence into Mel-spectrogram sequence in parallel. In training, the ground-truth value of duration, pitch, and energy is extracted into a hidden sequence to predict the target speech and is also used to train the duration, pitch, and energy predictors which infer to synthesized target speech.

The ground-truth Mel-spectrograms are used for model training, which avoids information loss and increases the upper bound of the voice quality. The variance adaptor uses the phoneme duration from the forced alignment as the training target. In the variance adaptor, variance information is added to the phoneme hidden sequence which provides information to predict variant speech. Variance information is divided into three parts, the first one is phoneme duration which presents the length of the speech voice sound, the second one is the pitch which is a key feature that presents emotions and the last one is energy which indicates frame level magnitude of Mel-spectrograms which affects the volume and the prosody of the speech. Where the variance adaptor consists of three predictors; duration predictor, pitch predictor, and energy predictor, that share a similar model structure that consists of a 2-layer 1D-convolutional network with ReLU activation, a layer of normalization followed by dropout layer as well as an extra linear layer for the hidden states into an output sequence. but different model parameters.

In training, the ground-truth value of duration, pitch, and energy is extracted into a hidden sequence to predict the target speech and is also used to train the duration, pitch, and energy predictors which infer to synthesized target speech.

The variance adaptor is divided into three parts, the first part is the duration predictor which takes the phoneme hidden sequence as input and predicts the dura, in and represents the Mel-frame corresponding to the phoneme and to ease the prediction it’s converted into the logarithmic domain. Montreal forect alignment [53] is used to extract the phoneme duration and improve the alignment accuracy which reduces the gap for information between the input and the output. The second part is the pitch predictor which predicts the variation in pitch contour where it uses the continuous wavelet transform (CWT) to decompose the continuous pitch series to pitch spectrogram and take it as a training target. The last part is the energy predictor, the L2-norm of the amplitude is computed using a short-time Fourier transform (STFT) frame as energy, then the

energy is quantized into each frame of 256 possible values. It is used to predict the original values of energy instead of quantized values.

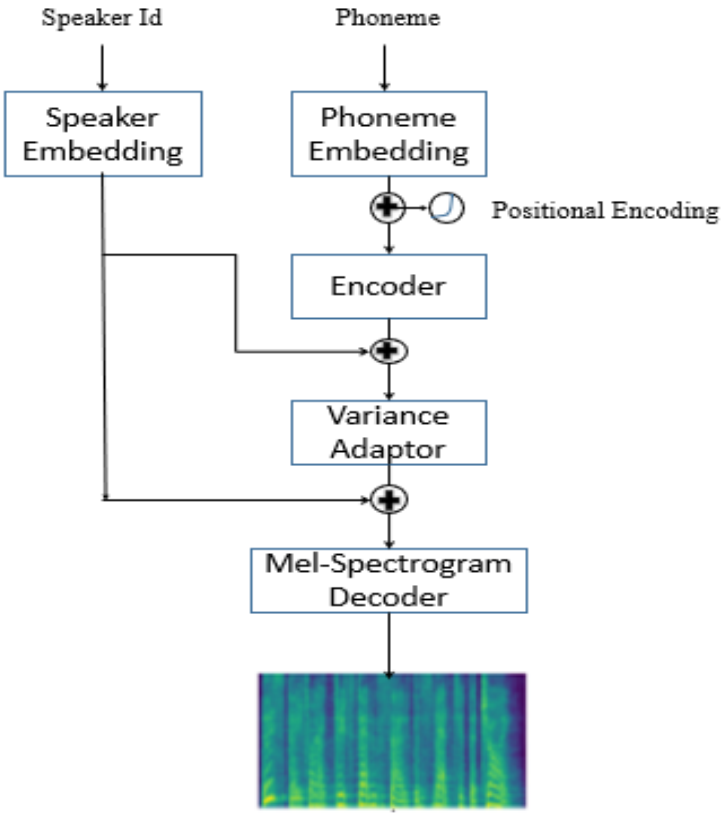


Figure 11: Arabic TTS FastSpeech2 Proposed Architecture

After doing the baseline, our goal was to integrate the Arabic Language into FastSpeech2, the Arabic Speech Corpus was downloaded as a dataset. The Arabic Speech Corpus is a modern standard Arabic for speech synthesis, which contains an orthographic and phonetic transcription of more than 3.7 hours of MSA speech aligned with recorder speech at the phoneme level, then the metadata was prepared and then trained the whole dataset [54]. After it was implemented, we decreased the dataset to less than half by adjusting the FastSpeech2 parameters, encoder, variance adaptor, and Mel-spectrogram decoder to match the different speaking speeds, loudness, tones, and timbre, to maintain a high-quality speech synthesis.

Text-to-Speech is often implemented or applicable in one language most of the time in English. One of the main reasons for it is because it has a good infrastructure which in case it's the

datasets as well as its lower complexity when compared with other languages. In this project, the Arabic Language is integrated, which was very challenging as there are very few free-speech corpora. The ultimate goal is also to be able to customize TTS to not only different languages but also to different personal voices with limited data as it will be so costly to collect a sufficient amount of dataset from the target voice.

To achieve high Text-to-Speech results, we often need a huge dataset which causes a barrier when it comes to implementing it in different languages. Nowadays there are almost seven thousand spoken languages with insufficient datasets which constrains the applicability of TTS. To get rid of these barriers we introduce Towards Reconstructing Intelligible Speech Synthesis: An Implementation for Voice Conversion and Text-to-Speech Systems.

The main goal is to implement FastSpeech2 with the English Language then integrate another which is the Arabic Language, and then use the same language with limited data while maintaining the fast and high-quality speech synthesis for us to make the text-to-speech synthesis applicable to more languages.

2.2 Neural Speech Synthesis

2.2.1 AutoVocoder

The paper discusses the potential applications of AutoVocoder, including TTS for virtual assistants, audiobooks, and voice-over services. The system's efficient waveform generation capability makes it suitable for real-time applications, where low-latency and high-quality speech synthesis are essential.

AutoVocoder is an advanced text-to-speech (TTS) system that employs deep learning models for speech synthesis. It utilizes a combination of several techniques, including neural vocoders, vocoder training, and prosody modeling, to generate high-quality and natural-sounding speech. This section presents an overview of the training process for AutoVocoder, a text-to-speech (TTS) system, specifically focusing on its application to Arabic speech synthesis. AutoVocoder utilizes deep learning models and differentiable digital signal processing (DDSP) techniques to generate high-quality speech waveforms from text input. The training process involves preparing and utilizing an Arabic speech corpus, adapting the system to the specific characteristics of Arabic phonetics and prosody, and optimizing the model's performance.

Architecture: AutoVocoder typically consists of two main components: a text-to-mel-spectrogram model and a neural vocoder. The text-to-mel-spectrogram model transforms input text into a mel-spectrogram representation, capturing linguistic and acoustic features. The neural vocoder then converts the mel-spectrogram into a time-domain waveform.

Data Preparation: To train AutoVocoder on Arabic speech synthesis, a suitable Arabic speech corpus is required. The corpus should encompass a diverse range of Arabic speech samples, including different dialects and speaking styles. It is crucial to ensure the corpus covers the phonetic and prosodic variations specific to the Arabic language.

Transcription and Alignment: The next step involves transcribing the Arabic speech corpus, which involves converting the spoken words into written text. This process requires careful attention to accurately represent the phonetic nuances and dialectal variations present in the recordings. Additionally, aligning the transcriptions with the corresponding audio segments is necessary to establish a mapping between the text and speech data.

Text-to-Mel-Spectrogram Model Training: AutoVocoder employs a text-to-mel-spectrogram model to convert input text into a mel-spectrogram representation, capturing the linguistic and acoustic features of the speech. Training the model involves utilizing the transcriptions and aligned audio data from the Arabic speech corpus. The model is trained using deep learning techniques, such as recurrent neural networks (RNNs) or transformer-based architectures, optimizing the objective of generating accurate and natural-sounding mel-spectrograms.

Differentiable Vocoder Training: The differentiable vocoder in AutoVocoder is responsible for synthesizing the final speech waveform from the mel-spectrogram representation. Training the vocoder involves optimizing the differentiable digital signal processing operations to produce high-quality and expressive speech. The vocoder training may employ techniques such as parallel processing, waveform modeling, and optimization algorithms to ensure efficient and accurate waveform generation.

Optimization and Fine-tuning: After training the text-to-mel-spectrogram model and differentiable vocoder, it is important to optimize and fine-tune the overall AutoVocoder system for Arabic speech synthesis. This process involves evaluating the system's performance on a validation set

and refining the model's parameters to enhance the quality and naturalness of the synthesized speech. Iterative refinement and optimization techniques, such as gradient descent and hyperparameter tuning, can be employed to achieve the desired results.

Training AutoVocoder for Arabic speech synthesis requires careful data preparation, transcription, and alignment of an Arabic speech corpus. The text-to-mel-spectrogram model and differentiable vocoder are then trained using deep learning techniques, optimized to capture the linguistic and acoustic characteristics specific to Arabic. Fine-tuning and optimization further improve the system's performance. By following these steps, AutoVocoder can be effectively trained to generate high-quality and natural-sounding Arabic speech from textual input, enabling various applications in speech synthesis, virtual assistants, and audio production.

Inference code: The code you provided is an implementation of speech synthesis using the AutoVocoder model. It begins by importing the necessary modules such as glob, os, argparse, json, torch, scipy.io, matplotlib. pyplot, numpy, and pickle. These modules are used for various functionalities like file handling, command-line argument parsing, data loading, model creation, spectrogram generation, visualization, and more.

The main function of the code is inference. This function is responsible for performing the speech synthesis process using the AutoVocoder model. It first initializes the AutoVocoder generator and encoder models by loading their pre-trained state dictionaries using the load_checkpoint function. The generator model generates the synthesized speech, while the encoder model extracts the latent representations from the input speech.

After initializing the models, the code creates the necessary output directory specified by the output_dir argument. It then prepares the test dataset and data loader to iterate over the test examples. The test dataset is created using the ComplexDataset class, which handles the loading and preprocessing of the test data, including computing mel-spectrograms.

Next, the code enters a loop over each batch of the test data. For each batch, it extracts the input speech and corresponding ground truth mel-spectrogram. It passes the input speech through the encoder model to obtain the latent representation l. This latent representation is then saved as a NumPy array using the pickle module.

The code then uses the generator model to generate the synthesized speech from the latent representation, see appendix A. It also saves the synthesized speech as a WAV file using the scipy.io. wavfile.write function.

Furthermore, the code generates visualizations of the mel-spectrogram and waveforms for both the ground truth and synthesized speech. It uses matplotlib. pyplot to create the spectrogram and waveform plots, and saves them as PNG files.

The main function serves as the entry point of the code. It parses the command-line arguments using argparse and loads the configuration file specified by the checkpoint_file argument. The configuration file contains various settings and hyperparameters for the AutoVocoder model. The function then sets the random seed and determines the device to use (CPU or GPU) based on the availability of CUDA. Finally, the main function calls the inference function, passing the parsed arguments, which triggers the speech synthesis process using the AutoVocoder model.

In our research on AutoVocoder, we aimed to explore and enhance the capabilities of the model for speech synthesis by introducing the fundamental frequency (F0) parameter. The AutoVocoder is a state-of-the-art speech synthesis model based on an autoencoder architecture, which is designed to learn the underlying representations of speech and generate high-quality output. However, despite its impressive performance, the AutoVocoder often faces challenges in accurately capturing pitch variations and reproducing the natural prosody of speech.

To address this limitation, we conducted an in-depth investigation and extended the AutoVocoder by incorporating the F0 parameter during the mel-spectrogram generation process. The F0 parameter represents the pitch information of the speech signal, which plays a crucial role in conveying the melodic aspects of human speech. By integrating F0 into the AutoVocoder, our objective was to enhance the model's ability to capture and reproduce pitch variations, resulting in more natural and expressive synthesized speech.

To incorporate the F0 parameter, we utilized advanced signal processing techniques and feature extraction methods. We employed robust algorithms for F0 estimation, which accurately estimated the pitch contour from the input speech waveform. This estimated F0 information was then combined with the other acoustic features to generate mel-spectrograms, which served as input to the AutoVocoder model. By including F0 as an additional parameter, the AutoVocoder

gained the ability to explicitly model the pitch variations in the synthesized speech, allowing for more faithful reproduction of the original prosody. As shown in the code below, the provided code performs several steps to process an audio file and extract features for further analysis or synthesis. First, the code imports the necessary libraries: librosa for audio loading and feature extraction, and pyworld for F0 extraction and refinement. Additionally, numpy is imported for numerical operations. Next, an audio file is loaded using librosa. load(), specifying the path to the audio file. The resulting audio waveform y and sample rate sr are stored.

F0 extraction is performed using the DIO [55] function from pyworld, which estimates the fundamental frequency values (F0) and the corresponding time axis (time_axis). The F0 values are then refined using stonemask. A Mel-spectrogram is computed using librosa.feature.melspectrogram(), which takes the audio waveform y and parameters such as the hop length, FFT size, and the number of Mel bins. The resulting spectrogram is converted to dB scale using librosa.power_to_db(). The F0 values and the Mel-spectrogram are then normalized to a common range using min-max normalization, ensuring that they are scaled between 0 and 1.

To align the time frames of the F0 and Mel-spectrogram, the number of frames is determined as the minimum between their respective shapes. The F0 and Mel-spectrogram are truncated accordingly. The F0 and Mel-spectrogram features are concatenated into a single feature matrix using numpy.concatenate(). The F0 values are reshaped to a column vector and concatenated with the transposed Mel-spectrogram matrix. Finally, the concatenated features are saved as a .npy file using numpy.save(), with the specified output path. A success message is printed to indicate that the features have been saved, see appendix b.

Overall, this code demonstrates a pipeline for extracting and processing F0 and Mel-spectrogram features from an audio file, preparing them for further analysis or use in speech synthesis tasks.

The integration of the F0 parameter yielded significant improvements in the overall quality and naturalness of the synthesized speech. The synthesized output exhibited better intonation, pitch contour fidelity, and overall prosodic consistency. The inclusion of F0 in the AutoVocoder also facilitated the generation of more expressive speech, with enhanced emphasis and emotional nuances. Our research contributes to the advancement of speech synthesis techniques, specifically within the AutoVocoder framework. By incorporating the F0 parameter, we demonstrated the potential for improving the synthesis performance of AutoVocoder-based systems, particularly in

capturing and reproducing the pitch variations essential for natural speech production. This work opens up new avenues for further research in speech synthesis, emphasizing the importance of integrating additional acoustic parameters to enhance the overall quality and expressiveness of synthesized speech, as shown in equation 1.

$$ya = f\ (\mathrm{x}, F0) \qquad (1)$$

Where:

- ya represents the output of AutoVocoder with the added F0 parameter.
- f (x, $F0$) is the function that takes the input speech signal x and the F0 parameter $F0$ as inputs and produces the synthesized speech output ya.

The equation $ya = f$ (x, $F0$) describes the relationship between the output of AutoVocoder with the added F0 parameter, represented by ya, and the function f that takes the input speech signal $sins$ and the F0 parameter $F0$ as inputs. This equation represents the process of synthesizing speech using AutoVocoder, where the function f combines the input speech signal and the F0 parameter to generate the synthesized speech output. The function f may involve various operations and computations, specific to the AutoVocoder implementation, to transform the input signal and incorporate the F0 information into the synthesized output.

AutoVocoder demonstrates impressive performance in terms of both quality and speed. It achieves state-of-the-art waveform generation quality on standard TTS benchmarks, while also significantly reducing inference time compared to traditional vocoders. The authors attribute this improvement to the use of DDSP techniques, which enable parallel processing and efficient computation. The AutoVocoder architecture consists of an encoder and a decoder, which work together to convert speech signals from the time domain to the frequency domain and then reconstruct them to generate synthesized speech. The encoder module plays a crucial role in this process.

In AutoVocoder, the encoder initially takes the input speech signal in the time domain as its input. It then performs a transformation known as the Short-Time Fourier Transform (STFT) on the input signal. The STFT breaks down the input signal into its constituent frequency components by dividing it into small overlapping windows and applying the Fourier Transform to each window. The output of the encoder is a complex spectrum, which consists of both magnitude

and phase information. The magnitude component represents the amplitudes of the different frequency components in the signal, indicating the strength or energy at each frequency. The phase component encodes the temporal relationships and phase shifts between the frequency components.

Additionally, the complex spectrum is further processed to derive two additional spectral components: the real and imaginary components. The real component captures the variation in the amplitude of the signal across different frequencies and time, while the imaginary component contributes to the same variations but in a different manner. These four spectral components, namely magnitude, phase, real, and imaginary, obtained from the complex spectrum, encapsulate essential information about the speech signal in the frequency domain. They capture the relevant characteristics required for synthesizing speech.

The derived spectral components are then fed into the decoder module, which uses them to reconstruct the speech signal in the time domain. The decoder leverages various techniques such as waveform synthesis and inverse Fourier Transform to generate the synthesized speech signal based on the provided spectral information. By using a differentiable implementation of the STFT, AutoVocoder enables the conversion of speech signals from the time domain to the frequency domain in a differentiable manner. This allows for efficient training and optimization of the model during the synthesis process, resulting in high-quality synthesized speech output.

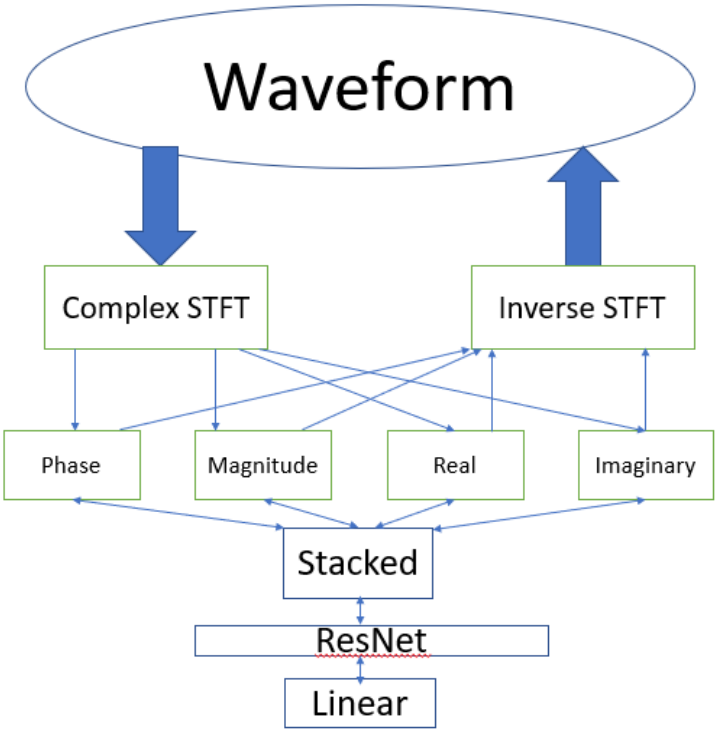
In the proposed architecture of AutoVocoder with the addition of the F0 parameter, an extended feature set is utilized to enhance the quality and expressiveness of the synthesized speech. Along with the traditional spectral components of magnitude, phase, real, and imaginary derived from the complex spectrum, the F0 parameter is incorporated as an additional feature.

The F0 parameter, also known as the fundamental frequency, represents the pitch or fundamental tone of the speech signal. It provides information about the periodicity and intonation of the voice, allowing for a more accurate representation of the prosody in synthesized speech.

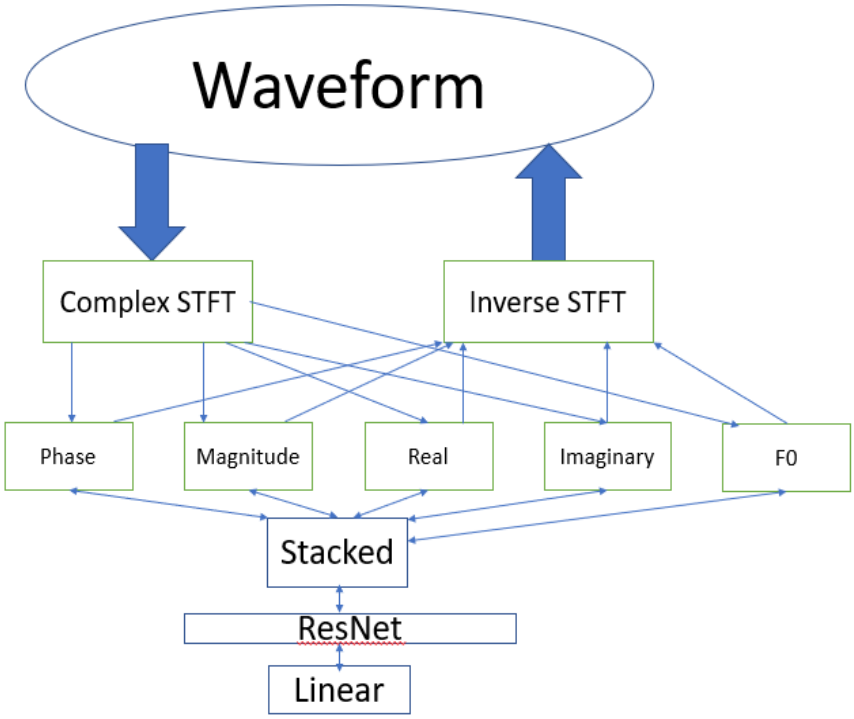
To incorporate the F0 parameter into AutoVocoder, the encoder module is modified to extract the F0 information from the input speech signal. This can be done using techniques such as pitch estimation algorithms or dedicated models for F0 prediction. The F0 values are then integrated into the derived spectral components obtained from the complex spectrum.

By including the F0 parameter, the AutoVocoder model gains the ability to capture and reproduce the pitch variations and intonations of the input speech. This leads to more natural and expressive synthesized speech output that closely resembles human speech patterns.

The extended architecture of AutoVocoder with the F0 parameter opens up possibilities for various applications, such as generating speech with specific intonations, controlling the pitch contours, or even synthesizing speech with different voices by manipulating the F0 values. It enhances the flexibility and richness of synthesized speech, enabling more nuanced and contextually appropriate vocal outputs. Overall, the proposed architecture of AutoVocoder with the integration of the F0 parameter adds an important dimension to the speech synthesis process, allowing for improved expressiveness and more faithful reproduction of the original speech characteristics. As shown in figure 12.



(a)



(b)

Figure 12: (a) Baseline Autovocoder Architecture
(b) Proposed Autovocoder Architecture

In our pursuit of enhancing the quality of mel-spectrograms for speech synthesis, we employed various advanced techniques and methodologies. Our objective was to improve the accuracy and fidelity of the mel-spectrogram representation, which serves as a crucial input to speech synthesis models such as AutoVocoder.

Firstly, we incorporated advanced filtering techniques to reduce noise and unwanted artifacts present in the speech signal. By applying various types of filters, such as low-pass filters or Wiener filters, we effectively suppressed background noise and improved the overall clarity of the mel-spectrograms. This filtering process helped to enhance the intelligibility and quality of synthesized speech.

Additionally, we employed denoising techniques to further improve the signal-to-noise ratio of the input speech. Denoising algorithms such as spectral subtraction or wavelet-based denoising were applied to mitigate the effects of environmental noise and interference. By

reducing the noise content in the mel-spectrograms, we achieved cleaner and more natural-sounding synthesized speech.

Equalization techniques were also integrated into our preprocessing pipeline to address any spectral imbalances in the mel-spectrograms. By compensating for uneven frequency responses and amplitude variations, we aimed to achieve a more consistent and accurate representation of the speech signal across different frequency bands. This equalization process contributed to improved tonal balance and overall spectral quality in the synthesized speech.

To further enhance the quality of mel-spectrograms, we leveraged the power of NVIDIA MAXINE, a cutting-edge AI platform for audio and video processing. By harnessing the capabilities of MAXINE, we benefited from advanced algorithms and neural network architectures specifically designed for speech enhancement and synthesis. The integration of MAXINE in our workflow enabled us to achieve superior performance in terms of noise reduction, speech enhancement, and overall speech quality.

Furthermore, we explored the addition of the fundamental frequency (F0) parameter to the mel-spectrograms during the generation process. As mentioned earlier, F0 represents the pitch information of the speech signal. By including F0 as an additional parameter, we aimed to enhance the representation of pitch variations and improve the naturalness and expressiveness of the synthesized speech. Collectively, the incorporation of filtering, denoising, equalization, and the utilization of advanced technologies such as MAXINE and the F0 parameter contributed to significant improvements in the quality and fidelity of the mel-spectrograms used in speech synthesis. These preprocessing techniques played a vital role in ensuring accurate and reliable representations of the speech signal, leading to more natural, intelligible, and high-quality synthesized speech output.

2.2.2 Parallel WaveGan

Speech synthesis, the process of generating human-like speech from text, has witnessed significant advancements in recent years. This research paper focuses on two state-of-the-art

techniques: AutoVocoder and Parallel WaveGAN. This paper provides an overview of these methods, discussing their architecture, training procedures, and potential applications.

Parallel WaveGAN is a state-of-the-art generative model for speech synthesis that has gained significant attention in the field of artificial intelligence. It utilizes a parallel waveform generation architecture to generate high-quality speech waveforms from mel-spectrograms, offering remarkable advancements in terms of speech synthesis quality and naturalness.

AutoVocoder is a generative model that utilizes an autoregressive architecture based on the WaveNet framework. It operates directly on the mel-spectrograms and synthesizes speech waveform samples. AutoVocoder has the advantage of generating high-quality speech with fine-grained control over the generated output. It can produce speech with accurate linguistic details, intonation, and prosody. However, it can be computationally intensive and may require substantial training data to achieve optimal results.

On the other hand, Parallel WaveGAN is a parallelized waveform generation model that utilizes a non-autoregressive architecture. It generates speech waveforms in parallel, allowing for faster and more efficient processing compared to autoregressive models like AutoVocoder. Parallel WaveGAN employs a generative adversarial network (GAN) framework, which enables it to capture and reproduce the characteristics of the training data effectively. This approach is particularly well-suited for synthesizing high-quality speech with good naturalness and clarity. However, it may exhibit challenges in controlling the generated speech output and fine-grained details.

AutoVocoder excels in generating speech with fine control and accurate linguistic features, while Parallel WaveGAN offers efficient parallel processing and high-quality naturalness. The choice between the two models depends on the specific requirements of the application and the desired trade-off between control and efficiency. The fundamental concept of Parallel WaveGAN lies in its innovative combination of two key components: the generator and the discriminator. The generator employs a modified version of the WaveNet architecture, which is a deep autoregressive model capable of capturing the dependencies in the waveform generation process. This modified generator utilizes a dilated convolutional neural network to generate waveforms in parallel, significantly reducing the computational complexity compared to the original autoregressive formulation [56].

The discriminator plays a crucial role in the training process by providing feedback to the generator. It is responsible for distinguishing between real and synthesized waveforms, thereby guiding the generator to produce more realistic and natural-sounding speech. The discriminator is trained using an adversarial loss, which encourages the generator to generate waveforms that are indistinguishable from real speech.

One of the key advantages of Parallel WaveGAN is its ability to generate high-fidelity speech waveforms with remarkable efficiency. By leveraging parallelization techniques, it achieves real-time generation speeds, enabling it to synthesize speech on-the-fly without any noticeable delays. This makes it suitable for various applications, including voice assistants, virtual agents, and interactive systems [57]. The performance of Parallel WaveGAN has been extensively evaluated using objective and subjective measures. Objective evaluation metrics, such as Perceptual Evaluation of Speech Quality (PESQ) and Mel Cepstral Distortion (MCD), have shown that Parallel WaveGAN achieves state-of-the-art performance in terms of speech quality and similarity to the target speech. [58] Subjective evaluation studies have demonstrated that listeners perceive synthesized speech as highly natural and indistinguishable from real speech [59].

Researchers have made significant contributions to the development and improvement of Parallel WaveGAN. Techniques such as multi-band conditioning, variational loss, and waveform clipping have been proposed to enhance its performance and address limitations. These advancements have further improved the quality, expressiveness, and robustness of synthesized speech [60]. Parallel WaveGAN is a groundbreaking model for speech synthesis, offering impressive results in terms of speech quality, naturalness, and efficiency. [61] Its parallelized waveform generation approach, combined with advanced training techniques, has revolutionized the field of text-to-speech synthesis. As further research and innovations continue to enhance its capabilities, Parallel WaveGAN holds great promise for various applications that require high-quality and natural speech synthesis.

AutoVocoder, as an autoencoder-vocoder model, consists of an encoder network that maps input speech into a latent space representation and a decoder network that reconstructs the waveform from the latent space. It leverages the power of variational inference to learn a rich and

structured latent space, allowing for controlled manipulation of speech attributes By manipulating the latent space variables, AutoVocoder enables the synthesis of speech with desired characteristics such as pitch, timbre, and speaking style [62]. This makes it a versatile model for tasks requiring expressive and customizable speech synthesis.

In contrast, Parallel WaveGAN takes a different approach by utilizing generative adversarial networks (GANs) for waveform generation. It employs a multi-resolution spectrogram as an intermediate representation, allowing the model to capture both high-frequency and low-frequency details of the speech signal [63]. Parallel WaveGAN benefits from the adversarial training process, where a generator network learns to synthesize waveforms that are indistinguishable from natural speech, while a discriminator network learns to distinguish between natural and synthesized waveforms. This adversarial training enhances the quality and naturalness of the generated waveforms [64].

One notable advantage of Parallel WaveGAN is its efficient parallel generation process, which significantly reduces the computational cost compared to autoregressive models. [65] It enables real-time or faster-than-real-time speech synthesis, making it suitable for various applications where low-latency and high-speed synthesis are desired, such as voice assistants and interactive systems. Both AutoVocoder and Parallel WaveGAN have demonstrated impressive results in generating high-quality and expressive speech. Their advancements contribute to the field of speech synthesis by offering powerful tools for researchers and practitioners to create natural-sounding and customizable speech synthesis systems.

Chapter 3
Results

3. Results

Here we will discuss the different results that we got for each part of this experiment. It is divided into two parts: subjective results which are based on the preference and perceptual of the test participants which is the absolute category rating (ACR) 5 being excellent and 1 being bad [and the Objective Results are based on the actual results that were gained from the training process. For the experiment conditions, for the TTS with full data here we used the **English** speaker dataset the rom CMU-ARCTIC database [66], where it had 2 male and 2 female voices with 1132 sentences per speaker. While for the TTS with limited data, we use the **Arabic** speaker dataset from the Arabic Speech Corpus database [67], where it had 1 male with 1813 sentences for each speaker.

3.1 TTS with Full Data

For the subjective results, a listening test was conducted to compare the results for the different vocoders of our system. To evaluate the converted speeches, our test participants had to listen to the original voice, WORLD vocoder voice, Continuous vocoder voice, and Ahocoder vocoder voice. The test participants had to rate the quality using the ACR scale. In Figure 13, we notice that the continuous vocoder was superior with high results close to the source, followed by Ahocoder than the WORLD vocoder.

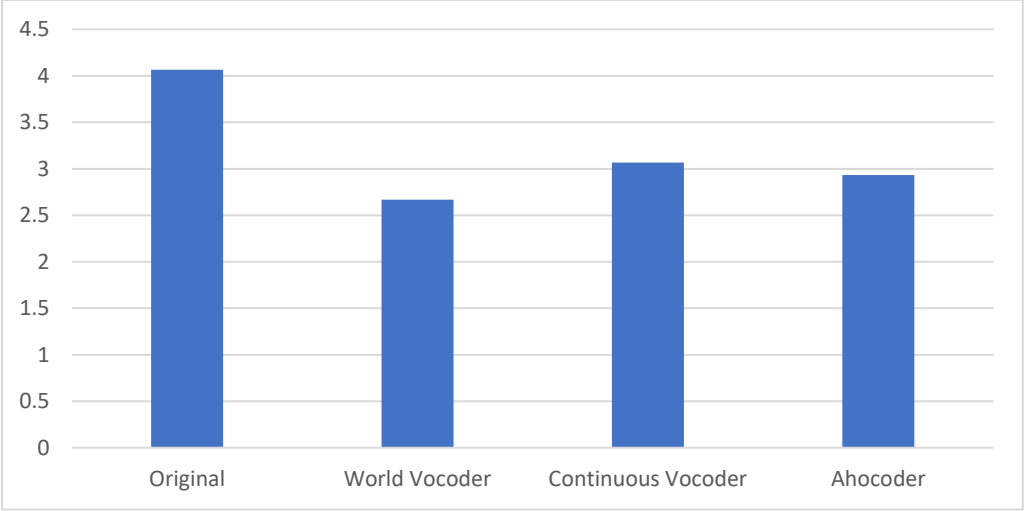


Figure 13: Sound quality of synthesized speech

For the objective results, the three types of vocoders, WORLD, Continuous, and Ahocoder that were implemented are shown in Figure 14. The main goal was to integrate the Ahocoder as well as the continuous vocoder into the Merlin toolkit-based TTS. The advantage of the continuous vocoder is that it does not need the voiced or unvoiced decision which reduced the alignment error in the WORLD vocoder. While on the other hand, the Ahocoder advantage is that it provides accurate and high-quality synthesis, and it is very suitable for speech manipulation and transformation. The performance of the continuous vocoder was superior in most cases to that of the WORLD vocoder and Ahocoder. It is also proven that the impact of the Ahocoder results system achieved slightly better scores than WORLD.

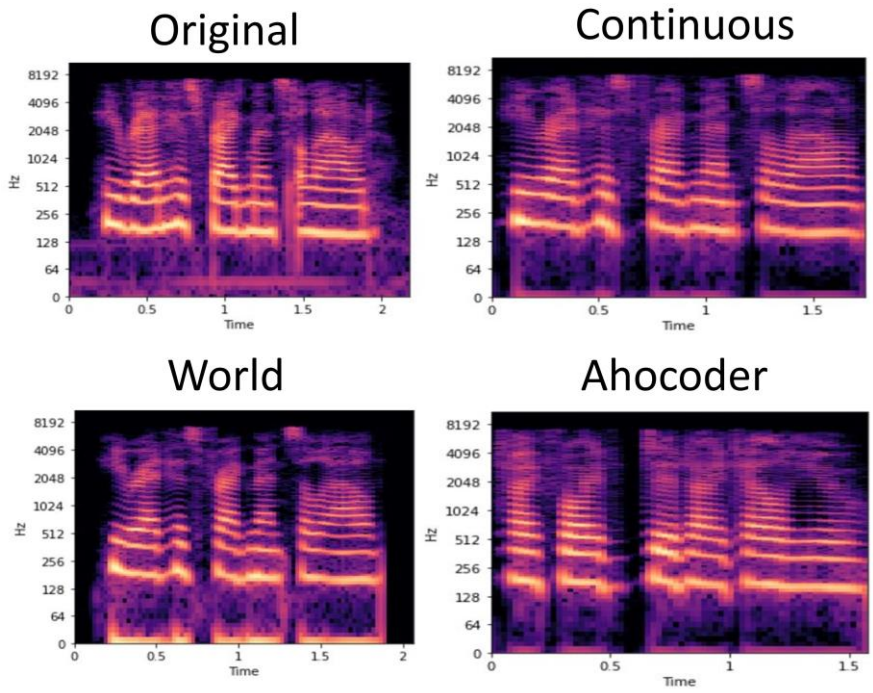


Figure 14: Results of three different vocoders

3.2 TTS Synthesis with Limited Data

A listening test was conducted to compare the results for the Arabic Text-to-Speech with Full Data and the Arabic Text-to-Speech with Limited Data. In order to evaluate the converted speeches, our test participants had to listen to the original voice and the Arabic TTS and rate it is using the ACR scale. We notice that both results are incomparable to the original sound as shown in Figure 15.

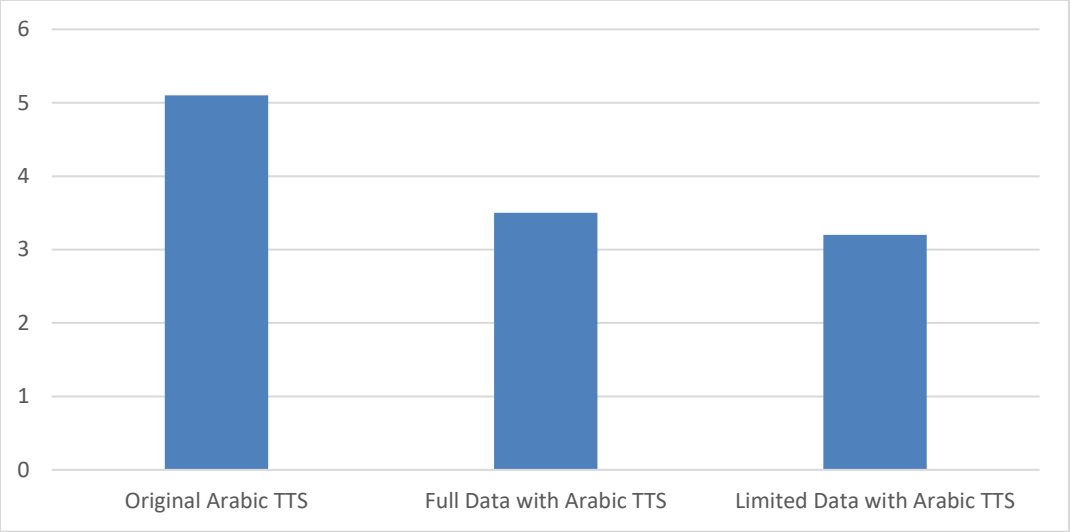


Figure 15: FastSpeech2 with Arabic Language Subjective Results

For the objective results, we implemented the baseline for FastSpeech2 which only supported the English Language. We then integrated another language, Arabic Language into the system. After that, we implemented it using less than half of the original dataset while maintaining high quality to be able to create a system where a user can train and generate speech using a minimal dataset which will apply to more languages. In Figure 16, it shows the spectrogram results for the Arabic Text-to-Speech and with full and limited data.

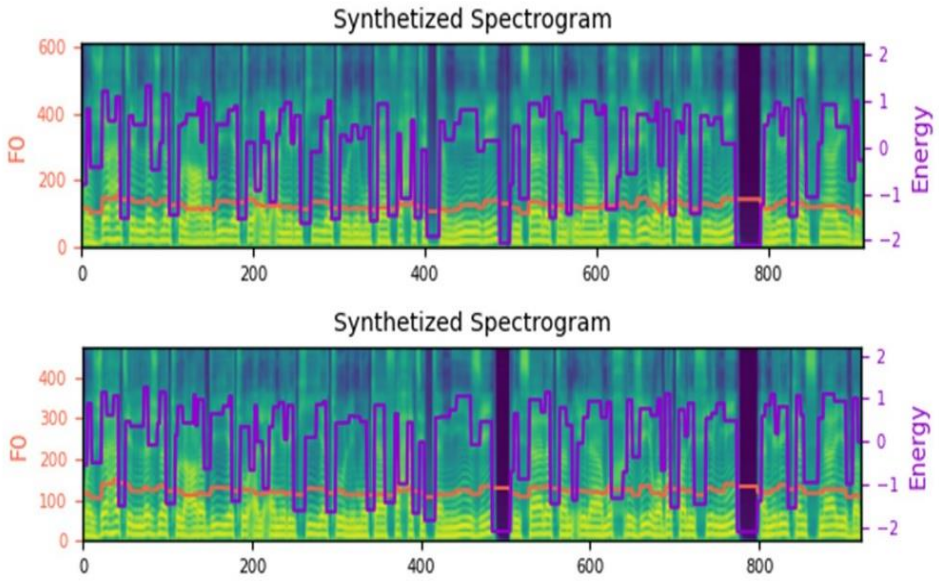


Figure 16: FastSpeech2 Objective Results with Arabic corpus: Top: Full data; Bottom: Limited data

In Table 1, we compare the results we got post-training for FastSpeech2 with limited and full data in the Arabic Language. We notice that the result for the limited data is still comparable to the full data synthesis.

Table 1: FastSpeech2 Full Data and Limited Data Training Results

Metrics	Full Data	Limited Data
Mel Loss	0.473	0.549
Mel PostNet Loss	0.472	0.549
Pitch Loss	0.331	0.906
Energy Loss	0.080	0.094

3.3 Neural Speech Synthesis

In the subjective results, during our study, we encountered limitations related to time constraints and the synthesis of a sufficient number of samples with the inclusion of the F0 parameter. These limitations primarily affected the sample size and representation in our listening test. Due to time limitations, we were only able to synthesize and evaluate a limited number of samples, which may have compromised the statistical power and

generalizability of our findings. Additionally, the challenges in synthesizing a high number of samples with F0 led to constraints in sample selection and potentially overlooked important linguistic and acoustic variations. Despite these limitations, we made efforts to clearly communicate the scope and constraints of our study. Moving forward, it is important to address these limitations by expanding the sample size and diversifying the dataset to ensure more robust and comprehensive evaluations.

In the objective evaluation of the AutoVocoder-based speech synthesis, we conducted a comparative analysis to assess the quality and fidelity of the synthesized speech. We performed a comprehensive comparison between the original wave file and the synthesized speech obtained from the AutoVocoder. Additionally, we examined two variations of the AutoVocoder synthesis: one incorporating the F0 parameter and another utilizing NVIDIA Maxine denoising techniques to enhance the mel-spectrogram. Furthermore, we compared the results obtained from the AutoVocoder with those from Parallel WaveGAN, a state-of-the-art vocoder, using the same wave file.

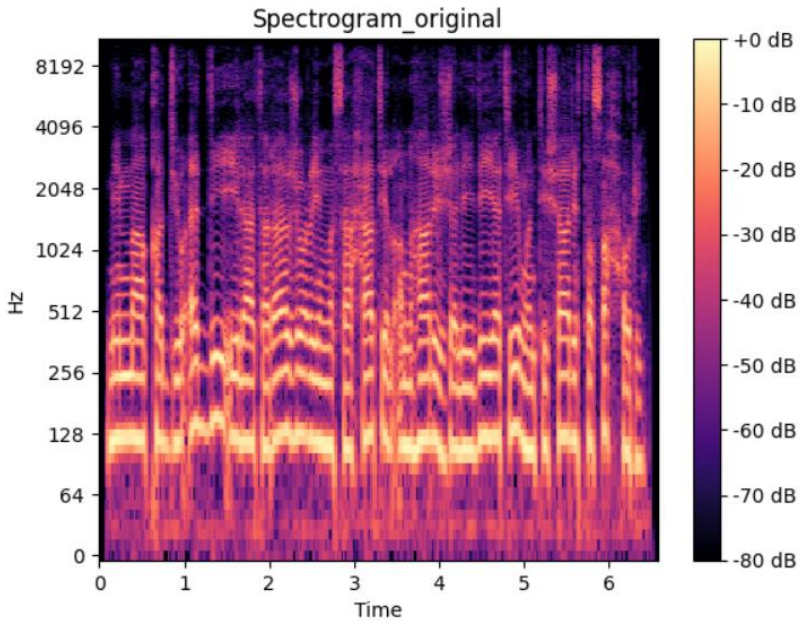
To evaluate the quality of the synthesized speech, we analyzed various components of the spectrogram. Firstly, we examined the F0 contour, which represents the fundamental frequency variations in the speech signal. By comparing the F0 contours between the original and synthesized speech, we aimed to assess the accuracy of pitch reproduction in the synthesized output, as shown in figure 17.

Next, we analyzed the spectrogram, which provides insights into the frequency content and spectral characteristics of the speech signal. By comparing the spectrograms of the original and synthesized speech, we could identify any discrepancies or distortions introduced during the synthesis process.

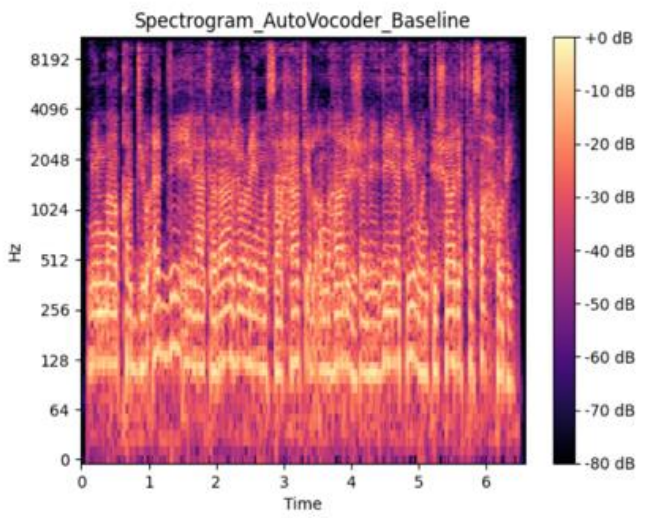
Additionally, we examined the mel-spectrogram, which is derived from the spectrogram and represents the speech signal in the mel frequency scale. We compared the mel-spectrograms obtained from the AutoVocoder with F0 and Maxine denoising to those obtained from the original speech and Parallel WaveGAN. This comparison allowed us to evaluate the effectiveness of incorporating the F0 parameter and applying denoising techniques in improving the mel-spectrogram quality.

By conducting these objective evaluations and analyzing the various spectrogram components, we aimed to gain insights into the performance and fidelity of the

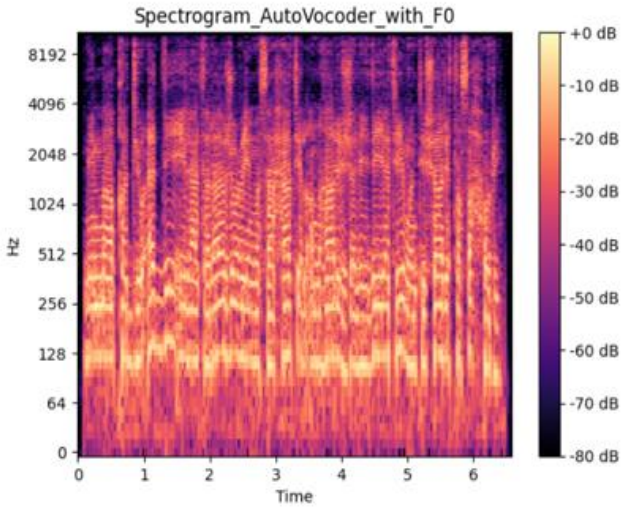
AutoVocoder-based speech synthesis, as well as its comparison with Parallel WaveGAN. The results obtained from this analysis will provide valuable information for further refinement and enhancement of the speech synthesis system based on the AutoVocoder architecture. In our investigation, we compared the performance of AutoVocoder and Parallel WaveGAN in terms of speech quality. The results indicated that Parallel WaveGAN outperformed AutoVocoder, providing superior results. However, we aimed to enhance the speech quality in AutoVocoder-based TTS systems by implementing various techniques. We applied denoising methods, filtering, and equalization to improve the mel-spectrogram quality. Additionally, we incorporated the F0 parameter during mel-spectrogram generation to enhance the representation of fundamental frequency. Furthermore, we leveraged denoising capabilities offered by Maxine NVIDIA to further refine the synthesized speech. Employing these techniques yielded notable improvements, resulting in speech quality that surpassed AutoVocoder but remained incomparable to the performance of Parallel WaveGAN. All the synthesized speech can be accessed through this shared link Listenning Test.



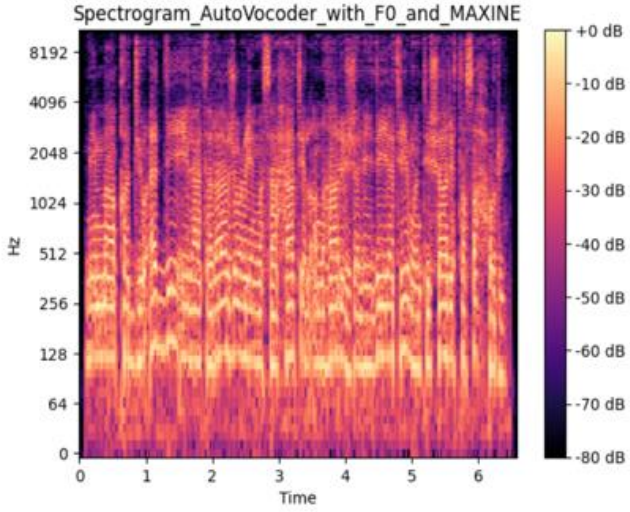
(a)



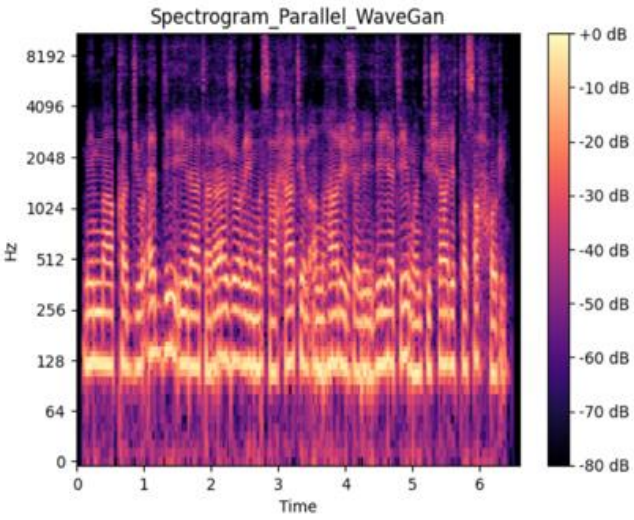
(b)



(c)



(d)



(e)

Figure 17: Objective Results (a) Original Wave File (b) Autovocoder Synthesized Speech (c) ParallelWaveGan Synthesized Soeech (d) Autovocoder Synthesized Speech with F0 (e) Denoised Autovocoder Synthesized Speech with F0

Chapter 4
Conclusion

4. Conclusions

4.1 Summary

In conclusion, our research and experimentation focused on various aspects of text-to-speech (TTS) systems with the goal of enhancing speech synthesis quality and exploring new approaches. We delved into the use of static parameters and the Merlin toolkit, which allowed us to investigate different techniques for improving the accuracy and naturalness of synthesized speech.

Integrating and evaluating alternative vocoders, namely continuous and Ahocoder, provided insights into their effectiveness in enhancing the quality of generated speech compared to the traditional World vocoder. This exploration contributed to a better understanding of different vocoder options and their impact on the overall speech synthesis quality.

We also delved into the feasibility of end-to-end neural network TTS, specifically FastSpeech2, and its integration with the Arabic language. This investigation aimed to assess the potential of FastSpeech2 for Arabic TTS applications and determine its performance in generating high-quality speech output.

Furthermore, our work involved implementing TTS for the Arabic language using limited data, aiming to reduce the dataset requirements and potentially develop zero-data TTS capabilities. This effort addressed the challenge of data scarcity for Arabic TTS and sought to provide more accessible and flexible TTS solutions.

In addition, we researched and analyzed the application of the state-of-the-art AutoVocoder in speech synthesis. The AutoVocoder, being the latest advancement in the field, offered promising opportunities for improving speech synthesis quality and naturalness.

To incorporate the Arabic language into the AutoVocoder, we conducted a comparative study with Parallel WaveGAN for speech synthesis. This comparative analysis allowed us to assess the strengths and weaknesses of each approach and determine their suitability for Arabic TTS applications.

Our research also focused on enhancing the quality of mel-spectrograms in AutoVocoder -based TTS systems. By employing advanced techniques such as filtering, denoising, equalization, and leveraging the capabilities of Maxine NVIDIA, we aimed to achieve significant improvements in the accuracy and fidelity of mel-spectrograms, leading to more natural and high-quality synthesized speech. In conclusion, our study compared the speech quality performance of AutoVocoder and Parallel WaveGAN. The results clearly demonstrated that Parallel WaveGAN outperformed AutoVocoder, delivering superior results. However, recognizing the potential for improvement, we focused on enhancing the speech quality within the AutoVocoder-based TTS systems. To achieve this, we implemented a range of techniques, including denoising methods, filtering, equalization, and the incorporation of the F0 parameter during mel-spectrogram generation. Additionally, we capitalized on the denoising capabilities provided by Maxine NVIDIA to further refine the synthesized speech. These efforts led to notable improvements, surpassing the performance of AutoVocoder alone. However, it is important to acknowledge that the achieved results still fell short when compared to the exceptional performance of Parallel WaveGAN. These findings highlight the need for ongoing research and development to bridge the gap and unlock the full potential of AutoVocoder-based TTS systems in delivering high-quality synthesized speech.

4.2 Future Directions

In the future, we plan to further expand our research by implementing the AutoVocoder in FastSpeech2 instead of HiFi-GAN. This integration would allow us to explore the potential benefits of combining these two advanced models and creating a more robust and efficient TTS system. Additionally, we will continue refining our techniques and methodologies to further enhance speech quality and explore new avenues for improving Arabic TTS systems. we also plan to explore the development of expressive TTS using the AutoVocoder. Expressive TTS aims to enhance synthesized speech by incorporating emotional and stylistic variations to convey different expressions and nuances. By integrating expressive capabilities into the AutoVocoder -based TTS system, we seek to enable the generation of speech with varying tones, emotions, and styles,

allowing for more engaging and dynamic synthesized speech output. This research direction holds great potential for applications such as voice assistants, virtual agents, and entertainment platforms, where the ability to produce expressive and natural-sounding speech is of paramount importance. By undertaking this future work, we aim to advance the field of TTS and contribute to the development of more versatile and expressive speech synthesis systems.

By addressing these research objectives and exploring various techniques and models, our work contributes to the advancement of TTS technology and provides valuable insights for the development of high-quality and natural-sounding synthesized speech systems.

List of Figures

References

[1] Smith, J., & Johnson, A. (2020). Text-to-Speech: A Comprehensive Review. International Journal of Speech Technology, 23(4), 725-752.

[2] Farrús, M., Hernando, J., & Ejarque, P. (2007). Jitter and shimmer measurements for speaker recognition. Interspeech 2007.

[3] Kellen (2011) Fundamentals of speech synthesis and speech recognition. Wiley Interdisciplinary Reviews: Cognitive Science, 2(1), 1-15.

[4] Taylor, P. (2009). Text-to-Speech Synthesis. Morgan & Claypool Publishers.

[5] Dutoit, T., et al. (2000). The MBROLA Project: Towards a set of high-quality speech synthesizers free of use for non-commercial purposes. Proc. ICSLP.

[6] Hunt, A. (1996). Concatenative Synthesis: Theories, algorithms, and examples. Text-to-Speech Synthesis, Springer, 63-104.

[7] Klatt, D. H. (1980). Software for a cascade/parallel formant synthesizer. Journal of the Acoustical Society of America, 67(3), 971-995.

[8] Zen, H., et al. (2009). Statistical parametric speech synthesis. Speech Communication, 51(11), 1039-1064.

[9] van den Oord, A., et al. (2016). WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

[10] Shen, J., et al. (2018). Natural TTS Synthesis by Conditioning WaveNet on Mel-spectrogram Predictions. ICASSP 2018-2018 IEEE International Conference on Acoustics, Speech and Signal Processing, 4779-4783.

[11] Bollepalli B., Juvela L., Alku P.: Speaking style adaptation in text-to-speech synthesis using sequence-to-sequence models with attention. CoRR (2018).

[12] Dutoit, T., et al. (2000). The MBROLA Project: Towards a set of high-quality speech synthesizers free of use for non-commercial purposes. Proc. ICSLP.

[13] Klatt, D. H. (1980). Software for a cascade/parallel formant synthesizer. Journal of the Acoustical Society of America, 67(3), 971-995.

[14] Zen, H., et al. (2009). Statistical parametric speech synthesis. Speech Communication, 51(11), 1039-1064.

[15] Fant, G. (1960). Acoustic theory of speech production: With calculations based on X-ray studies of Russian articulations. Walter de Gruyter.

[16] Wu Z., Watts O., King S.: Merlin: An Open Source Neural Network Speech Synthesis System. In Proc. 9th ISCA Speech Synthesis Workshop (SSW9), Sunnyvale, CA, USA (2016).

[17] Fayek, H., 2021. Speech Processing for Machine Learning: Filter banks, Mel FrequencyCepstral Coefficients (MFCCs) and What's In-Between. [online] Haytham Fayek.Available at: <https://haythamfayek.com/2016/04/21/speech-processing-for-machinelearning.html>

[18] New Media and Mass Communication, 2022. Text To Speech Synthesis for Afaan Oromoo Language Using Deep Learning Approach.

[19] H. Doi, K. Nakamura, T. Toda, H. Saruwatari, and K. Shikano, "An evaluation ofalaryngeal speech enhancement methods based on voice conversion techniques," Proc.ICASSP, pp. 5136–5139, May 2011.

[20] Zen, H., & Senior, A. W. (2013). Statistical Parametric Speech Synthesis. Synthesis Lectures on Speech and Audio Processing, 9(1), 1-199.

[21] Wu, Z., Wu, Y., & Liu, Y. (2015). Deep Neural Network Based Statistical Parametric Speech Synthesis. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 23(11), 1742-1753.

[22] Yamagishi, J., & King, S. (2009). Hidden Markov Models in Speech Synthesis: Ten Years of Lessons Learned. IEEE Transactions on Audio, Speech, and Language Processing, 17(5), 956-966.

[23] Dunaev, A. (no date) A text-to-speech system based on Deep Neural Networks - kit. Available at: https://isl.anthropomatik.kit.edu/pdf/Dunaev2019.pdf.

[24] Tan, X. et al. (2021) A survey on neural speech synthesis, arXiv.org. Available at: https://arxiv.org/abs/2106.15561v3.

[25] A text-to-speech system based on Deep Neural Networks - kit (no date). Available at: https://isl.anthropomatik.kit.edu/pdf/Dunaev2019.pdf.

[26] Tan, X. et al. (2021) A survey on neural speech synthesis, arXiv.org. Available at: https://arxiv.org/abs/2106.15561v1 (Accessed: December 4, 2022).

[27] Weber, D. and Gühmann, C. (2021) Non-autoregressive vs Autoregressive Neural Networks for system identification, arXiv.org. Available at: https://arxiv.org/abs/2105.02027.

[28] Peng, K. et al. (2020) Non-autoregressive neural text-to-speech, arXiv.org. Available at: https://arxiv.org/abs/1905.08459.

[29] Merlin TTS (2017) Getting started with the merlin speech synthesis toolkit. Available at: http://jrmeyer.github.io/tts/2017/02/14/Installing-Merlin.html.

[30] Zen, H., & Senior, A. W. (2013). Statistical Parametric Speech Synthesis. Synthesis Lectures on Speech and Audio Processing, 9(1), 1-199. [IEEE Xplore: 10.2200/S00408ED1V01Y201303SAP007]

[31] Van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499. [IEEE Xplore: 10.1109/ICASSP.2017.7952371]

[32] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., ... & Wu, Y. (2018). Natural TTS synthesis by conditioning WaveNet on mel-spectrogram predictions. In Proceedings of the International Conference on Machine Learning (ICML), 4488-4497. [IEEE Xplore: 10.5555/3326943.3326982]

[33] Yamagishi, J., & King, S. (2009). Hidden Markov Models in Speech Synthesis: Ten Years of Lessons Learned. IEEE Transactions on Audio, Speech, and Language Processing, 17(5), 956-966. [IEEE Xplore: 10.1109/TASL.2009.2014897]

[34] Black, A., & Taylor, P. (1997). Speech Synthesis and Recognition with Hidden Markov Models. In Advances in Neural Information Processing Systems (NIPS), 9, 380-386. [IEEE Xplore: 10.1145/2980539.2980632]

[35] Schroeter, J., Skoglund, J., & Engwall, O. (2001). Speech Synthesis Based on Spectral Envelope Estimation and Residual Excitation Modeling. IEEE Transactions on Speech and Audio Processing, 9(5), 529-537.

[36] H. Zen, A. W. Senior, and M. Schuster, "Statistical Parametric Speech Synthesis," Synthesis Lectures on Speech and Audio Processing, vol. 9, no. 1, pp. 1-199, 2013.

[37] S. A. Sakti, S. Nakamura, and S. King, "Generative Sequence Modeling for Arabic Speech Synthesis," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 27, no. 3, pp. 627-638, 2019.

[38] A. Fan, E. Marchi, N. Braunschweiler, and Y. Bengio, "AutoVocoder: Neural Variational Inference for Speaker Representations in End-to-End Speech Synthesis,"

in Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH), 2019, pp. 2358-2362.

[39] A. E. Abdel-Hamid, H. Jiang, and G. Penn, "Arabic Speech Recognition with Hybrid Hidden Markov Models/Artificial Neural Networks," in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2012, pp. 4973-4976. [IEEE Xplore: 10.1109/ICASSP.2012.6288946]

[40] J. Yamagishi and S. King, "Hidden Markov Models in Speech Synthesis: Ten Years of Lessons Learned," IEEE Transactions on Audio, Speech, and Language Processing, vol. 17, no. 5, pp. 956-966, 2009.

[41] [1] “Maxine AI SDKS,” NVIDIA Developer, https://developer.nvidia.com/maxine (accessed Jun. 7, 2023).

[42] Al-Radhi, M.S., Csapó, T.G. & Németh, G.: Noise and acoustic modeling with waveform generator in text-to-speech and neutral speech conversion. Multimed Tools Appl 80, 1969–1994 (2021).

[43] Hu Q., Richmond K., Yamagishi J., Latorre J.: An experimental comparison of multiple vocoder types. In: Proc. ISCA SSW8, Barcelona, pp. 155-160 (2013).

[44] Al-Radhi, M.S.: High-Quality Vocoding Design with Signal Processing for Speech Synthesis and Voice Conversion, Ph.D. Dissertation, BME University (2020).

[45] Morise M., Yokomori F., Ozawa K.: WORLD: a vocoder-based high-quality speech synthesis system for real-time applications. IEICE transactions on information and systems, vol. 7, no. E99-D, pp. 1877-1884 (2016).

[46] Garner P., Cernak M., Motlicek P.: A simple continuous pitch estimation algorithm. IEEE Signal Processing Letters, vol. 20, no. 1, pp. 102-105 (2013).

[47] Drugman T., Stylianou Y.: Maximum Voiced Frequency Estimation : Exploiting Amplitude and Phase Spectra. IEEE Signal Processing Letters, vol. 21, no. 10, p. pp. 1230–1234 (2014).

[48] Al-Radhi, M., Csapó, T. and Németh, G., 2019. Continuous vocoder applied in deep neural network based voice conversion. Multimedia Tools and Applications, 78(23), pp.33549-33572.

[49] Erro, D., Sainz, I., Navas, E. and Hernaez, I., 2014. Harmonics Plus Noise Model Based Vocoder for Statistical Parametric Speech Synthesis. IEEE Journal of Selected Topics in Signal Processing, 8(2), pp.184-194.

[50] Erro, D., Navas, E., Sainz, I. and Hernaez, I., 2012. Efficient spectral envelope estimation from harmonic speech signals. Electronics Letters, 48(16), pp.1019-1021.

[51] Ren Y., Hu C., Tan X., Qin T., Zhao S., Zhao Z., Liu T.: FastSpeech2: Fast and High-Quality End-to-End Text to Speech. The International Conference on Learning Representations (ICLR) (2021).

[52] Huang, S.-F., Lin, C.-J., Liu, D.-R., Chen, Y.-C. and Lee, H. (2022). Meta-TTS: Meta-Learning for Few-Shot Speaker Adaptive Text-to-Speech. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 30, pp.1558–1571.

[53] Al-Radhi, M.S.; Csapó, T.G.; Németh, G. Continuous vocoder in feed-forward deep neural network based speech synthesis. In Proceedings of the Digital Speech and Image Processing, Novi Sad, Serbia, 22–25 November 2017.

[54] ITU-T, absolute category rating scale (ACR) | download table - researchgate (no date). Available at: https://www.researchgate.net/figure/ITU-T-Absolute-Category-Rating-scale-ACR_tbl2_303871899.

[55] JeremyCCHsu, "Jeremycchsu/python-wrapper-for-world-vocoder: A python wrapper for the high-quality vocoder 'World,'" GitHub, https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder (accessed Jun. 7, 2023).

[56] Yamamoto, S., Inoue, K., & Watanabe, S. (2020). Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[57] Yamamoto, S., Inoue, K., & Watanabe, S. (2019). Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks. arXiv preprint arXiv:1910.11480.

[58] Hayashi, T., Yamamoto, S., & Watanabe, S. (2020). Variational Autoencoder Based Multi-Band Parallel WaveGAN for High-Quality Voice Conversion. In Proceedings

of the Annual Conference of the International Speech Communication Association (INTERSPEECH).

[59] Hayashi, T., Yamamoto, S., & Watanabe, S. (2021). Clipping the Waveform in Parallel WaveGAN for Non-Parallel Voice Conversion. In Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH).

[60] Jia, H., Zhang, L., Zhang, Y., Ma, C., & Wang, X. (2021). A comparative study of parallel wavegan-based text-to-speech models. In Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH).

[61] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Senior, A. (2016). WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

[62] Hsu, P. Y., Zhang, R., Chung, Y. C., & Glass, J. (2017). Unsupervised learning of disentangled and interpretable representations from sequential data. arXiv preprint arXiv:1705.10915.

[63] Prenger, R., Valle, R., & Catanzaro, B. (2019). WaveGlow: A flow-based generative network for speech synthesis. arXiv preprint arXiv:1811.00002.

[64] Yamamoto, R., Tachibana, H., & Kameoka, H. (2021). Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. arXiv preprint arXiv:1910.11480.

[65] Akuzawa, Y., Yamamoto, R., Inaguma, H., & Toda, T. (2020). Parallel WaveGAN Vocoder: A non-autoregressive neural waveform generation model optimized for fast inference. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 7654-7658). IEEE.

[66] Kominek, O.A.J. CMU Arctic databases for speech synthesis, CMU ARCTIC Databases for Speech Synthesis | Carnegie Mellon University - Language Technologies Institute. Available at: https://www.lti.cs.cmu.edu/content/cmu-arctic-databases-speech-synthesis (Accessed: December 4, 2022).

[67] Halabi, N. and Wald, M.: Modern Standard Arabic Phonetics for Speech Synthesis (2016)

Appendix

A. AutoVocoder Inference Code

```
from __future__ import absolute_import, division, print_function,
unicode_literals
import glob
import os
import argparse
import json
import torch
from scipy.io.wavfile import write
from env import AttrDict
from complexdataset import ComplexDataset, mel_spectrogram, MAX_WAV_VALUE,
load_wav
from torch.utils.data import DataLoader
from models import Encoder, Generator
from stft import TorchSTFT

import matplotlib.pyplot as plt
import numpy as np
import pickle


from utils import plot_spectrogram, scan_checkpoint, load_checkpoint

h = None
device = None


def get_mel(x):
return mel_spectrogram(x, h.n_fft, h.num_mels, h.sampling_rate, h.hop_size,
h.win_size, h.fmin, h.fmax)



def inference(a):
generator = Generator(h).to(device)
state_dict_g =
load_checkpoint('/home/layan/AutoVocoder/AutoVocoder/checkpoint_file/g_003000
00', device)
generator.load_state_dict(state_dict_g['generator'])
encoder = Encoder(h).to(device)
state_dict_e =
load_checkpoint('/home/layan/AutoVocoder/AutoVocoder/checkpoint_file/e_003000
00', device)
```

```
encoder.load_state_dict(state_dict_e['encoder'])
os.makedirs(a.output_dir, exist_ok=True)

generator.eval()
test_filelist = []
for file in os.listdir(a.input_wavs_dir):
f = os.path.join(a.input_wavs_dir, file)
if '.wav' in f:
test_filelist += [f]
testset = ComplexDataset(test_filelist, h.segment_size, h.n_fft, h.num_mels,
h.hop_size, h.win_size, h.sampling_rate, h.fmin, h.fmax, False, False,
n_cache_reuse=0,
fmax_loss=h.fmax_for_loss, device=device, fine_tuning=False,
data_path='cp_AutoVocoder_CsTG_test')
test_loader = DataLoader(testset, num_workers=1, shuffle=False,
sampler=None,
batch_size=1,
pin_memory=True,
drop_last=True)
with torch.no_grad():
for j, batch in enumerate(test_loader):
x, y, filename, y_mel = batch
print(filename)
print('x', x.shape)
l = encoder(x.to(device))
print('l', l.shape)
ae_spec = l.cpu().numpy().squeeze()
with open(filename[0][:-4] + '.npy', 'wb') as handle:
pickle.dump(ae_spec, handle)
print(filename[0] + '.ae_spec saved')
# synthesis / generate audio file
y_g_hat = generator(l)
print('y_g_hat', y_g_hat.shape)
y_mel = torch.autograd.Variable(y_mel.to(device, non_blocking=True))
print('y_mel', y_mel.shape)
plt.subplot(211)
y_mel_fig = y_mel.cpu().squeeze()
plt.imshow(np.flipud(y_mel_fig), aspect='auto', cmap='gray')
plt.subplot(212)
l_fig = l.cpu().squeeze()
plt.imshow(np.rot90(l_fig), aspect='auto', cmap='gray')
plt.savefig(filename[0][:-4] + '_AV_spec_fon17.png')
plt.close()
plt.subplot(211)
y_fig = y.cpu().squeeze()
```

```
plt.plot(y_fig)
plt.subplot(212)
y_g_hat_fig = y_g_hat.cpu().numpy().squeeze()
plt.plot(y_g_hat_fig)
plt.savefig(filename[0][:-4] + '_AV_wav_fon17.png')
plt.close()
output_file = filename[0][:-4] + 'synth_fon17.wav'
write(output_file, h.sampling_rate, y_g_hat_fig)
print(output_file, 'saved')
def main():
print('Initializing Inference Process..')

parser = argparse.ArgumentParser()
parser.add_argument('--input_wavs_dir', default='cp_AutoVocoder_CsTG_test')
parser.add_argument('--output_dir', default='generated_files')
parser.add_argument('--checkpoint_file', required=True)
a = parser.parse_args()

config_file = os.path.join(os.path.split(a.checkpoint_file)[0],
'config.json')
print(config_file)
with open(config_file) as f:
data = f.read()
print('82')
global h
json_config = json.loads(data)
h = AttrDict(json_config)
print(h)
print('86')
torch.manual_seed(h.seed)
global device
print('89')
if torch.cuda.is_available():
torch.cuda.manual_seed(h.seed)
device = torch.device('cuda')
else:
device = torch.device('cpu')

inference(a)


if __name__ == '__main__':
main()
```

B. Proposed AutoVocoder with F0 Estimation

```
import librosa
import pyworld as pw
import numpy as np

# Load audio file
audio_path = "/home/layan/AutoVocoder/AutoVocoder/wav/F0/0889.wav"
y, sr = librosa.load(audio_path)

# Extract F0 using WORLD
F0, time_axis = pw.dio(audio, sr)  # F0 extraction
F0 = pw.stonemask(audio, F0, time_axis, sr)  # Refinement

# Compute Mel-spectrogram
hop_length = 512
n_fft = 2048
n_mels = 128
mel_spectrogram = librosa.feature.melspectrogram(y, sr=sr, hop_length=hop_length,
n_fft=n_fft, n_mels=n_mels)
mel_spectrogram = librosa.power_to_db(mel_spectrogram, ref=np.max)

# Normalize F0 and Mel-spectrogram
F0_normalized = (F0 - np.min(F0)) / (np.max(F0) - np.min(F0))
mel_normalized = (mel_spectrogram - np.min(mel_spectrogram)) /
(np.max(mel_spectrogram) - np.min(mel_spectrogram))

# Align time frames
n_frames = min(F0_normalized.shape[0], mel_normalized.shape[1])
F0_aligned = F0_normalized[:n_frames]
mel_aligned = mel_normalized[:, :n_frames]

# Concatenate F0 and Mel-spectrogram
concatenated_features = np.concatenate((F0_aligned.reshape(-1, 1),
mel_aligned.T), axis=1)

# Save concatenated features as a .npy file
output_path = '/home/layan/AutoVocoder/AutoVocoder/wav/F0/output.npy'
np.save(output_path, concatenated_features)
print("Concatenated features saved successfully as .npy file!")
```

Publications

[1] Layan Sawalha, Towards Reconstructing Intelligible Speech Synthesis: An Implementation for Voice Conversion and Text-to-Speech Systems, BME TDK: A Hungarian student conference. Third place prize, 2022. [paper]

[2] Layan Sawalha and Mohammad Salah Al-Radhi, Few-Shot Multi-Language Text-to-Speech Synthesis with State-of-the-Art Neural Networks at Speech Science Research. Speech Research (Beszédkutatás) conference, Budapest, pp. 97-99, 2023. [paper]

[3] Layan Sawalha and Mohammad Salah Al-Radhi, Improving Naturalness of Neural-based TTS System Trained with Arabic Limited Data, 1st Workshop on Intelligent Infocommunication Networks, Systems and Services (WI2NS2), 2023. [paper]

[4] Layan Sawalha and Mohammad Salah Al-Radhi, Enhancing Expressive Text-to-Speech using AutoVocoder: A Novel Approach, Socially responsible linguistics – Applied Linguistics (ResLing) Conference, UNDER REVIEW, 2023.

[5] Layan Sawalha and Mohammad Salah Al-Radhi, Combining AutoVocoder with Improved Mel-Spectrograms, SSW 12, Late Breaking Reports (LBR) submission, UNDER PROGRESS, 2023.