Implementation and evaluation of a Spanish TTS based on FastPitch

Author: Anne Manero Alvarez

Advisors: Prof. Inma Hernaez Rioja

Prof. Eva Navas Cordón

University of the Basque Country

hap/lap

Hizkuntzaren Azterketa eta Prozesamendua

Language Analysis and Processing

Final Thesis

September 2022

Departments: Computer Systems and Languages, Computational Architectures and Technologies, Computational Science and Artificial Intelligence, Basque Language and Communication, Communications Engineer.

Abstract

Text-to-speech (TTS) generates speech from text. This tool helps improve people's quality of life. However, when extending these models to support languages like Spanish, we find scarce databases, data processing tools, and model training resources.
In this thesis, I implemented and evaluated a Spanish TTS model on FastPitch with a 10 hour database. FastPitch is a neural network-based end-to-end TTS system that allows for prosody transformations. I first researched state-of-art TTS and preprocessed the dataset, then implemented and evaluated the model. As a result, several resources are provided: tools for raw database processing, methods for linguistic module adaptation, a clean dataset and a quality TTS system in Spanish.
This model's quality is compared with two vocoders (WaveGlow/HiFiGan) and two other state-of-art acoustic models (FastSpeech2/Tacotron2). The FastPitch model synthesized with HiFiGan vocoder obtained the highest quality results. To conclude, prosody transformation experiments at inference resulted successful with this FastPitch Spanish TTS.
Keywords: Text-To-Speech, Spanish, acoustic models, data preprocessing, Deep Neural Networks

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Acronym/ Abbreviation	**Description**
ARPA	Advanced Research Projects Agency
BFCC	Bark-Frequency Cepstral Coefficients
CER	Character Error Rate
CNN	Convolutional Neural Network
DNN	Deep Neural Network
FFTr	Feed-Forward Transformer
G2P	Grapheme-to-Phoneme
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HCI	Human Computer Interaction
HMM	Hidden Markov Model
IPA	The International Phonetic Alphabet
LSTM	Long-Short Term Memory
MFCC	Mel-frequency cepstrum coefficients
MOS	Mean Opinion Score
MSE	Mean-Squared Error
NISQA	Non-Intrusive Speech Quality Assessment
NLP	Natural Language Processing
NN	Neural Networks
OOV	Out-of-Vocabulary
PER	Phoneme Error Rate
POS	Part-of-Speech
RAM	Random Access Memory
RNN	Recurrent Neural Network
SAMPA	The Speech Assesment Methods Phonetic Alphabet
Seq2Seq	Sequence-to-Sequence
SPSS	Statistical Parametric Speech Synthesis
ToBI	Tone and Break Indices
TTS	Text-To-Speech

Acronym/ Abbreviation	**Description**
VOCODER	VOice CODER
WER	Word Error Rate
WFST	Word Finite State Transducer

Table 1: Table of acronyms and abbreviations

1 Introduction

Speech is the most natural communication tool that humans use to be easily understood. The area of human computer interaction (HCI) aims to help computers communicate and interact with humans, for instance, through speech. This is possible thanks to the computer's ability to process voice signals. Text-to-speech (TTS) is the technology used to generate speech from a given input text in real time by a computer. In other words, it helps a computer learn how to read text or symbols and pronounce them by producing speech sound waves automatically. Nowadays, the purpose of building TTS is to produce synthesized speech that can be easy to understand by people, as well as indistinguishable from real human speech (Azizah et al., 2020).

With recent developments in technology, increasing computing power and development of deep learning techniques, the delivery methods and speech quality have advanced to a great extent over the last years, deriving into a rapid expansion of the text-to-speech technology (O'Malley, 1995). Nowadays, TTS is an essential component for many technological applications such as navigation announcements on smartphones and in cars, or interactive interfaces of smart assistant systems (Hayashi et al., 2019). They are also present in the media and entertainment industry, providing audio narration for media subtitles or video and mobile gaming. TTS systems also have multiple applications in the fields of health care and education. They can serve as assistants for people without the physical ability to speak and visually impaired people (Edward, 2018), convert sign language to speech in real (Tiku et al., 2020), reduce mind wandering in students with dyslexia (Bonifacci et al., 2022), facilitate processing of written materials for people with reading impairments (Wallace et al., 2021), or as a tool for language education (Dutoit and Dutoit, 1997), among many other applications.

1.1 TTS systems

Text to speech is the conversion of text into voice output. It is therefore key for a TTS model to learn the relation between text and audio. In order to do so, the first step in a TTS system is to process text and audio data as input, and extract information from the two sources. As represented in Figure 1, the information features extracted from text are called linguistic features. These features are typically phonemes. A phoneme is a symbol representing the smallest unit of sound that distinguishes one word from another (Britannica). To get these phonemes from text, the preprocessor tokenizes a sentence

into words. Then, it breaks the words into phonemes based on their pronunciation. For instance, “hello”, would be converted into “hɜˈləʊ”.

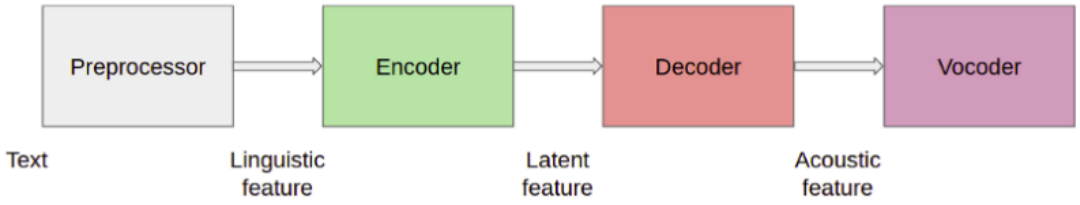


Figure 1: Basic processes involved in TTS (Maheshwari, 2021)

The preprocessor step calculates the duration of the phonemes, but also procesess prosodic features. Prosody refers to acoustic features of speech, such as rhythm, stress and intonation. Inside prosody we can define different perceptual measurements of sound, including pitch and energy. Pitch is a key feature to convey emotions using our voices, and it greatly affects the speech prosody. Energy indicates frame-level magnitude of mel-spectrograms and directly affects the volume and prosody of speech. In general terms, the mel-spectrogram is a signal processing method for capturing and measuring acoustic features, or more specifically, frequencies, of an audio signal.

The linguistic features (phonemes) are the input of the encoder, which outputs an embedding known as latent feature. The complete encoding process has been represented in Figure 3. A latent feature encapsulates the information extracted from the pre-processing step. Latent features are also used for the prediction of energy, pitch, and duration, as seen in 3a.

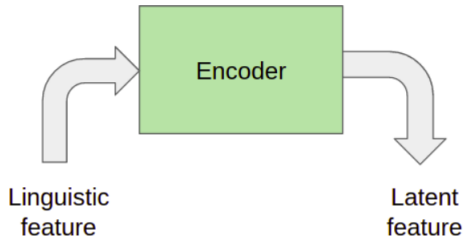
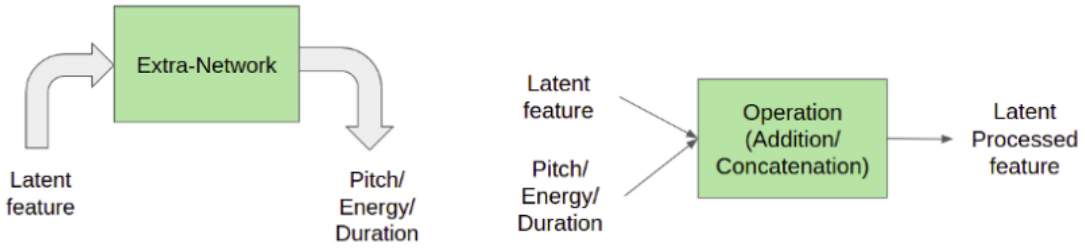


Figure 2: Encoding process of TTS (Maheshwari, 2021)

Energy, pitch, and the duration of the phonemes are actually used to train the energy predictor, the pitch predictor, and the duration predictor respectively, which are used by the model to get a more natural output (Maheshwari, 2021). The last operation in the encoding process adds and concatenates both latent features and pitch, energy and duration



(a) Extra network in the encoder (b) Operation of feature addition/concatenation

Figure 3: Process inside the encoder (Maheshwari, 2021)

predictions to output a new latent feature containing all this processed information. We can see this process in 3b.

The processed latent features are the input of the decoder. As represented in Figure 4, the decoder converts the information embedded in the latent features into acoustic features, that is, mel-spectrograms. These output acoustic features are sent to the vocoder as input. The reason why there is an extra step of obtaining mel-spectrogram as output, instead of directly producing the speech audio directly from the vocoder, is that audio contains more variance information (e.g., phase) than mel-spectrograms. This means that there is some information missing between the input and the output for text-to-audio, leading to potential problems in the output, compared to text-to-spectrogram generation (Maheshwari, 2021).

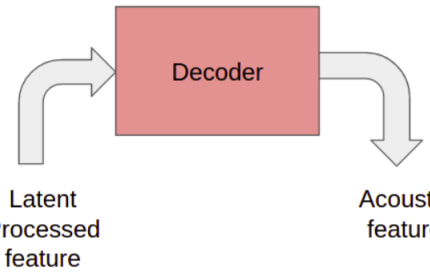


Figure 4: Decoding process of TTS (Maheshwari, 2021)

To conclude, in Figure 5 we can observe how the vocoder converts the acoustic features (mel-spectrograms) into the final waveform (audio) (Maheshwari, 2021). A vocoder is an audio processor that captures the characteristic elements of an an audio signal, in this case, the output mel-spectrograms, and uses this characteristic signal to affect other audio signals. This technology behind the vocoder is used to synthesize the speech waveform,

the final output of a TTS system (TechTarget, 2005).

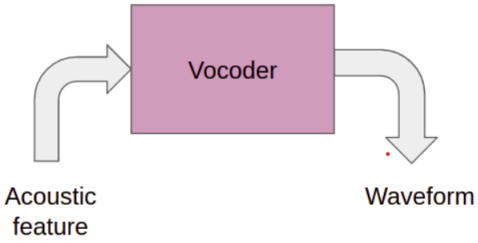


Figure 5: Vocoder for wave synthesizing process (Maheshwari, 2021)

1.2 Motivation and scope of the thesis

Text-to-speech is technological tool that can assist in improving people's lives. As such, intelligent speech applications have undergone unprecedented breakthroughs and growth, and recent research on end-to-end TTS has gained success in terms of human-like and high-quality generated speech. For instance, TTS systems have demonstrated a powerful capability with regard to reproducing prosody style or speaker characteristics (Chen et al., 2019).

Nevertheless, obtaining good quality TTS requires a great investment of effort and resources, such as many hours of recorded speech data, annotations of the audio data in text, careful lexicon and phoneme set development, or complex rules for text normalization, among others (Gutkin et al., 2016). Thus, we find that most TTS resources are exclusively available in English. It is difficult for a developer who wants to create a speech synthesizer in a new language to find a model, a phoneme set in the target language, or even data.

The goal of this thesis is to train a TTS model in Spanish and, while doing so, provide a clean dataset and tools for the cleaning, processing and adaptation of future datasets in Spanish for the TTS domain. With this project, I contribute to the field of Spanish text-to-speech with following resources and experiments:

1. Resources for TTS in Spanish: methods, means, databases.
2. Tools for the dictionary creation, phonetic transcription and linguistic module implementation in Spanish.
3. Tools for audio and text data curation.
4. A cleaned Spanish dataset that is ready to be used by any other TTS model.

5. End-to-end TTS model in Spanish.
6. Quality evaluation of FastPitch with two vocoders in Spanish: WaveGlow and HiFi-Gan.
7. Quality evaluation of three TTS acoustic models in Spanish: FastPitch, FastSpeech2, Tacotron2.
8. Prosodic transformations in Spanish at inference.

This work is accomplished by re-purposing and adapting the following main resources:

1. **TC-STAR dataset**, a Spanish TTS database consisting of the recordings and annotation of approximately 10 hours of read speech in Spanish.
2. **Aholab** `Modulo1y2`, a text processing module that transcribes text into phonemes, which we use for the dictionary creation as well as transcription of text at inference.
3. **FastPitch text-to-speech**, a text-to-speech acoustic model that learns and models the duration and pitch features of input text and audio (from our corpus), to subsequently predict mel-spectrogram values from a given text at inference.
4. **WaveGlow/HiFiGan**, vocoders that create the final speech waveform from the predicted mel-spectrograms.

1.3 Research questions

1. With the available dataset and resources in Spanish, what are the procedures needed to train and develop a TTS model in Spanish based on FastPitch?
2. How does quality of three state-of-art TTS acoustic models (FastPitch, Tacotron2, FastSpeech2) compare when trained using the same Spanish dataset and conditions?

1.4 Master's thesis outline

This document is organised as follows: in section 2, we introduce the state-of-the-art techniques in text-to-speech. In section 3, we explain the methodology and processes followed for the training and assessment of our TTS system in Spanish, including the corpus and data processing methods, the acoustic model training procedure, as well as the evaluation of results. Section 4 contains the results of the different experiments and the

analysis and evaluations performed. To conclude, section 5 provides an overview of this project and suggests further steps for continuing the study.

2 Literature review

This cutting-edge piece of technology that transforms text into speech is the result of centuries of interdisciplinary application and advances in linguistic theory, acoustic–phonetic characterization of target language sound patterns, perceptual psychology, mathematical modeling of speech production, structured programming, and computer hardware design (Klatt, 1998). In the following subsections, we briefly introduce the history of the Text-to-Speech technology, including the different trends and efforts that have contributed to the advances in speech synthesis. Then, we explain the models and techniques of current state-of-the-art TTS. Finally, we address Spanish TTS by reviewing the available work and resources in this regard.

2.1 History of TTS systems

Human curiosity over generating synthetic speech began centuries ago, dating back to the 12th century (Tan et al., 2021). During the 18th century, mechanical talking machines started to be built (Mache et al., 2015). In 1791, Wolfgang von Kempelen created one of the first speaking machines that produced "speech" by building mechanical equivalents of the parts of the human vocal system (Tarnóczy, 2005). Subsequent early attempts of speech synthesis technology used parametric synthesis methods that construct a model of the acoustic properties of the human vocal tract, and then analyze speech by determining the values of the parameters of the model (Berkley). In 1939, Homer Dudley introduced parametric techniques by creating the first device to synthesize speech sounds via electrical means, inspired by VOCODER (Voice Coder), which represent the stepping stone of the vocoders used nowadays for speech synthesis (Hoffmann and Mehnert, 2010). It wasn't until late 1960's, though, that first full computer-based TTS systems started to be developed (Mache et al., 2015).

Until the arrival of the DNN techniques, computer-based speech synthesis methods can be divided into two main techniques: parametric and concatenative. Concatenative systems, as the name suggests, aim to concatenate speech units selected from a pre-recorded speech database together for synthesis (Ning et al., 2019). Parametric synthesis uses digital signal processing to synthesize speech from text and describes the speech using parameters, rather than stored exemplars (King, 2011). Early computer-based speech synthesis methods include articulatory synthesis, formant synthesis and concatenative synthesis (e.g. diphone concatenation or unit selection) (Tan et al., 2021). With the increase in power

and resources of computer technology, more sophisticated engineered, robust and fast TTS systems started to emerge, such as Statistical Parametric Synthesis (Hidden-Markov Model or neural network-based) and Neural Speech Synthesis, which are the most popular TTS techniques nowadays (Zen et al., 2009). The timeline of all the different TTS systems and their types can be observed in Figure 6 below[1]. Following the technological progress, creating natural sounding synthetic voices has shifted from a knowledge-based focus to a data-based one (Zen et al., 2009).

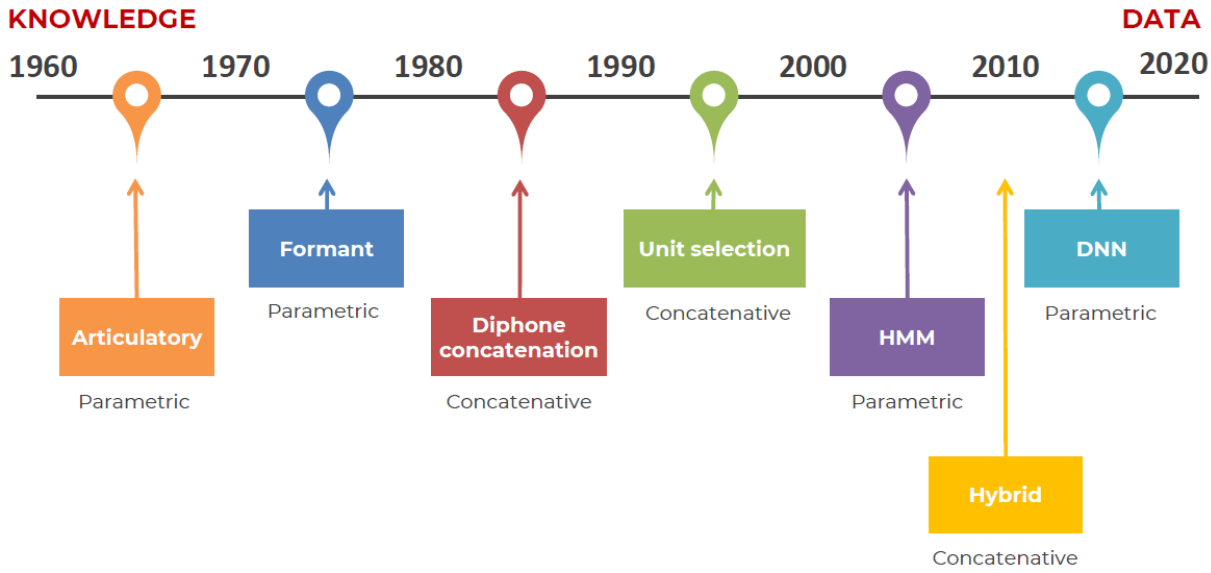


Figure 6: Waveform generation techniques over history

Articulatory TTS synthesis is a parametric and rule based technique that aims to model the human vocal organs as accurately as possible, including the areas of the glottis, mouth, and vocal chords. Simulating computationally the neurophysiology and biometrics of speech production should obtain quality speech results. However, there are inconsistencies between the format of the collected and the synthesized data. In addition, the modeling of the tongue movements is largely complex, which causes an output still far from natural sound. Hence, articulatory synthesis methods did not achieve a high level in success (Karjalainen, 1999; Tan et al., 2021).

Formant synthesis refers to a set of rules that control a simplified source filter model to produce speech, a method for speech synthesis that became popular in the last decades of the 20th century (Karjalainen, 1999). Formant TTS synthesis measures formant trajec-

[1]Figure extracted from Speech Technologies course in Language Analysis and Processing master's degree (2020) at EHU/UPV

tories for different sounds and transitions, designs rules to describe them mathematically, and, during synthesis, simulates formants through digital filters. This parametric method uses an additive synthesis module, i.e. a module for adding waves together, and an acoustic model with different parameters such as fundamental frequency, voicing or noise level to synthesize speech. The output quality, however, is not ideal, and it has a long development time and difficulties to accurately estimate the correct value for all the parameters (Tan et al., 2021).

Concatenative synthesis, as mentioned, consists on selecting the most appropriate speech unit from a pre-recorded and labeled speech corpus, and concatenating each unit into the final synthesized speech (Ning et al., 2019). This selection depends on a unit selection algorithm that finds the characteristics extracted from the source sounds or the high level descriptions attributed to them and, at inference, searches speech units to match the given input text, to then produce speech waveform by concatenating these units together. Diphone synthesis refers to the presence of a single segment per unit (e.g. diphone); whereas unit selection synthesis is the presence of multiple segments per unit (Zen, 2015). The concatenative synthesis method faces several drawbacks including segmentation (phoneme boundaries not well detected), descriptors (accurate labels), efficiency (selection is computationally expensive when the number of units in the inventory is very large), data mining (requires huge recording database in order to cover all possible combinations of speech units for spoken words), and lack of flexibility (only recorded data is available for synthesis) (Schwarz, 2006; Tan et al., 2021).

Statistical Parametric Speech Synthesis (SPSS) is proposed to address the weaknesses of concatenative synthesis. The model is parametric and statistical because it uses parameters to describe speech, and these parameters are described using statistics (e.g., means and variances of probability density functions) for calculating the distribution of the parameter values in the training data (King, 2011). This way, instead of concatenating units to generate the waveform, SPSS generates the acoustic parameters necessary to produce speech, then recovers the speech waveform from these parameters using a vocoder. Early Statistical Parametric Speech Synthesis models are formed by three main parts: a standard text analysis module for linguistic feature extraction, an HMM-based acoustic model, and the vocoder for analysis and speech waveform synthesis.

The overall architecture of a SPSS system is shown in Figure 7. As we can see in this figure, during training, the Hidden Markov Model (HMM)-based acoustic module performs a mapping from l linguistic contexts extracted from text, i.e. text analysis module, to

probability densities of speech parameters o such as the frequency spectrum (vocal tract), fundamental frequency (voice source), and duration (prosody) of speech, i.e. vocoder analysis module (Zen et al., 2013; Karagiannakos, 2021) During synthesis, HMMs generate a set of l linguistic parameters from the input text sequence, extracted from the text analysis, and vocoders synthesize speech from the predicted acoustic features $\hat{o}$ (Karagiannakos, 2021; Tan et al., 2021).

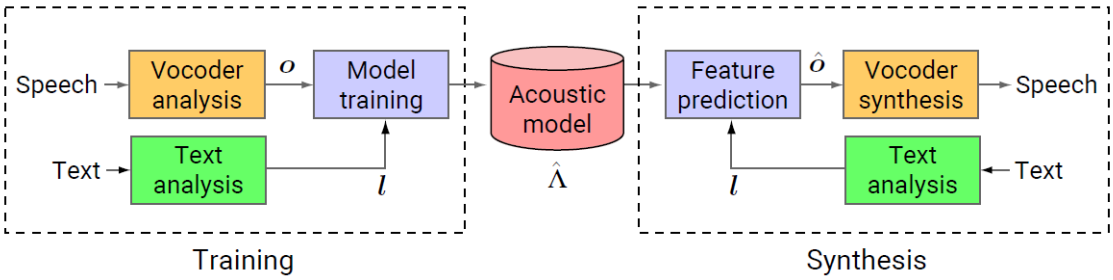


Figure 7: Statistical Parametric Speech Synthesis Zen (2015)

In near 2010s, with the rapid development of neural networks and deep learning, the HMM-based acoustic model of SPSS starts to be replaced by early Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs) or Generative Adversarial Networks (GANs) that predict the acoustic features from linguistic features (Saito et al., 2017; Zen et al., 2013; Tan et al., 2021). As the deep learning field continues to develop, the following logical step for TTS systems is to adopt (deep) neural networks as the model backbone for speech synthesis, giving birth to state-of-the-art Neural Speech Synthesis (Tan et al., 2021). In the following subsection, we further deepen into neural TTS by reviewing the different types of models and frameworks of the last decade.

2.2 State-of-the-art TTS systems

State-of-the-art text-to-speech models are most commonly formed by three main components divided into two modules: the text analysis and processing part as the TTS frontend module, and the acoustic model and the vocoder as the TTS backend module. We can observe these two modules in Figure 8 below. As it is shown in the figure, the text processing module takes a discrete symbol sequence (raw text) as input and transforms it into a discrete linguistic feature sequence that is easier to interpret by the acoustic model, such as phonetic, prosodic or linguistic labels. The acoustic model processes the linguistic and acoustic features together, learns the relationship between them, and uses predictions

to map linguistic features to acoustic features at inference (synthesis) time. The vocoder takes this discrete symbol sequence containing the predicted acoustic features and turns it into a waveform (Zen, 2015; Tan et al., 2021).

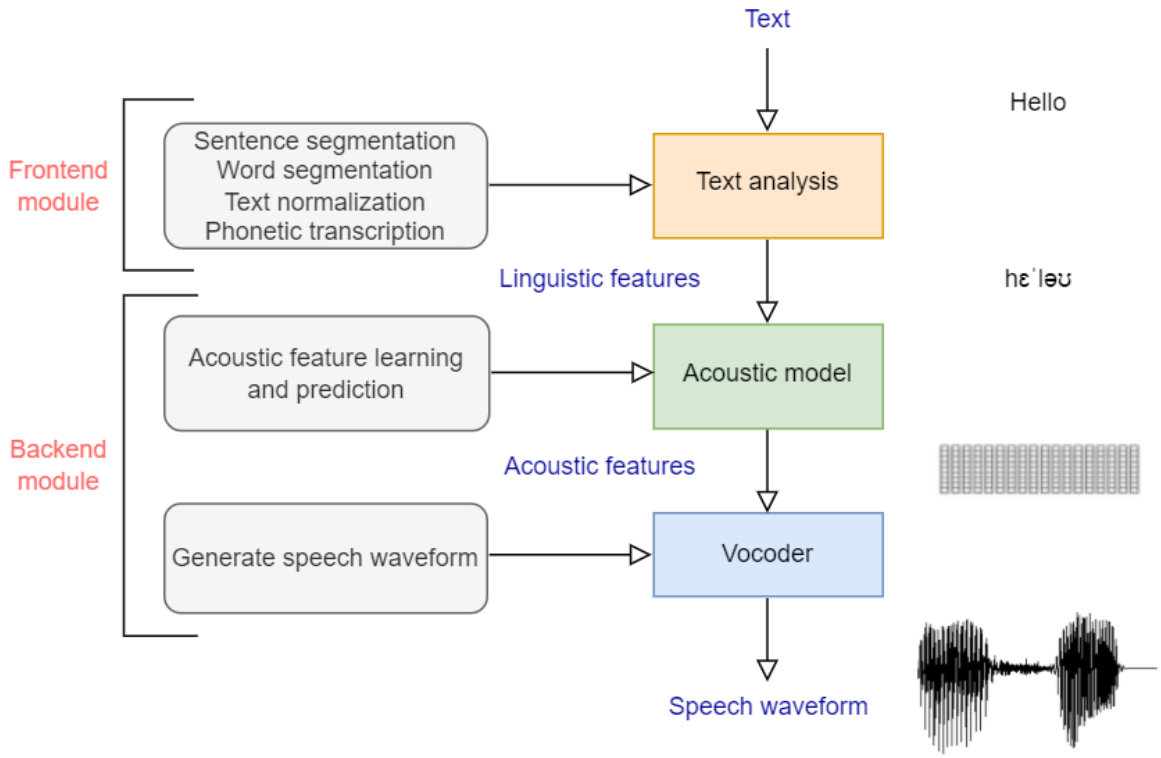


Figure 8: General scheme of a TTS system

In this section, we explore each part and the most relevant models and processes in the pipeline of state-of-the-art neural-based TTS synthesis, following a brief overview of the diachrony of these systems. By taking this information into consideration, we can understand the characteristics of our system, as well as the motivations behind the selection of our text processing method, acoustic model and vocoder.

2.2.1 Text analysis

The main objective of the text analysis part, also called frontend in the TTS domain, is to turn text into linguistic features, including pronunciation and prosody features to be used at inference for synthesis. Statistic parametric models extract several features from the frontend step through different processes such as text normalization, word segmentation, part-of-speech (POS) tagging, prosody prediction and grapheme-to-phoneme (G2P) con-

version. The large modeling capacity of end-to-end neural models allows for these to take phoneme sequences directly as input, therefore the preprocessing step can be simplified (Tan et al., 2021). The two main processes required for neural models are reduced to text normalization and clean up and G2P conversion.

Text normalization is the process of transforming raw text from character format into spoken word format. There are different categories in text normalization. For instance, numbers need to be converted from digit to word, e.g. the year "1970" is translated as "nineteen seventy" and "5/16" as "May sixteenth". Acronyms and abbreviations such as "Dept." and "UK" are normalized to "Department" and "United Kingdom". To conclude, text may contain punctuation and special characters such as: hyphen "-", plus "+", apostrophe " ' ", slash "/", counterslash "\", parentheses "(" and ")", ampersand "&" and at "@ ". These symbols contain no specific phonemic sound, therefore the acoustic model will not be able to draft and learn a correlation between these graphemes and a sound feature, and require normalizing or simply getting removed.

Grapheme-to-phoneme conversion refers to the conversion of characters (graphemes) into pronunciation tokens (phonemes) with the aim of facilitating the speech synthesis. Common examples of the phoneme tokens to which graphemes are converted include the International Phonetic Alphabet (IPA), Speech Assessment Methods Phonetic Alphabet (SAMPA) or ARPAbet from the Advanced Research Projects Agency (ARPA) . Thus, for instance, the word "Thursday" will take the following phonetic transcription according to each alphabet format: "/ˈθɜːzdeɪ/" in IPA, "T3:zdeI" in SAMPA and "TH ER1 Z D EY2" in ARPAbet. Each of the tokens has a correlation with specific sound values, which the model will learn during training. The model can leverage a manually collected G2P lexicon for the transcriptions, but this cannot cover the pronunciation of all the words in the language (Tan et al., 2021). Hence, it is common for this to include a pre-trained G2P backend that will generate the pronunciations of out-of-vocabulary (OOV) words, such as Phonemizer and G2P-seq2seq (cf. Section 2.3.2 for more information on these G2P softwares).

2.2.2 Acoustic models

Acoustic models generate acoustic features from linguistic features, which will then be further converted into speech waveform by the vocoder. Different acoustic model types have been developed over the history of TTS, from early HMM and DNN based models in statistical parametric speech synthesis (SPSS), to sequence to sequence models based on

encoder-attention-decoder framework (including LSTM, CNN and self-attention), and the latest feed-forward networks (CNN or self-attention) for parallel generation (Tan et al., 2021). In the following subsections we will further review the latest state-of-art acoustic models, amongst which we can find well-known models in the field like Tacotron and Tacotron2 (RNN-based), DeepVoice (CNN-based) or FastSpeech, FastSpeech2 and the present study's model, Fastpitch (transformer based).

Recurrent Neural Network-based models The main idea behind recurrent neural networks is to use the output of previous states in the network as inputs of the following ones, while having hidden states (Amidi and Amidi). At the same time, there are two types of models for sequence generation; autoregressive and non-autoregressive. Autoregressive models forecast a variable of interest depending on the past values of it, whereas non-autoregresive ones can generate a sequence in parallel, without explicitly depending on the previous element, which allows for creating faster inferences. The sequence to sequence learning of these models is based on an encoder-decoder framework: The encoder takes the source sequence as input and generates a set of representations, and the decoder takes these representations and its preceding elements to estimate the conditional probability of each target element (Ren et al., 2019).

Tacotron is one of the first end-to-end acoustic models based on a seq2seq model with attention. Instead of generating samples in an autoregressive way, Tacotron generates vocoder parameters. Tacotron's building block consists of a bank of 1-D convolutional filters, followed by highway networks and a bidirectional gated recurrent unit (GRU) recurrent neural net (RNN). It takes characters as input of the encoder and the attention-based encoder produces spectrogram frames that later on are converted into waveforms (Wang et al., 2017).

Both Tacotron and Tacotron2 map character embeddings to mel-scale spectrograms as the conditioning input to the vocoder instead of linguistic, duration, and F0 features. Tacotron2 improves the previous model architecture by simplifying building blocks (Paperswithcode, a). As we can see in Figure 9, Tacotron2 gets character embeddings as input and uses bidirectional vanilla LSTM and several convolutional layers in the encoder and decoder. Each decoder step corresponds to a single spectrogram frame (Shen et al., 2018). These structure and components of Tacotron2 can be observed in Figure 9.

Convolutional Neural Network-based models Convolutional Neural Networks (CNNs) refer to a type of algorithm popular to the field of computer vision that learns to assign a relevance to different inputs and differentiate from one another. By perform-

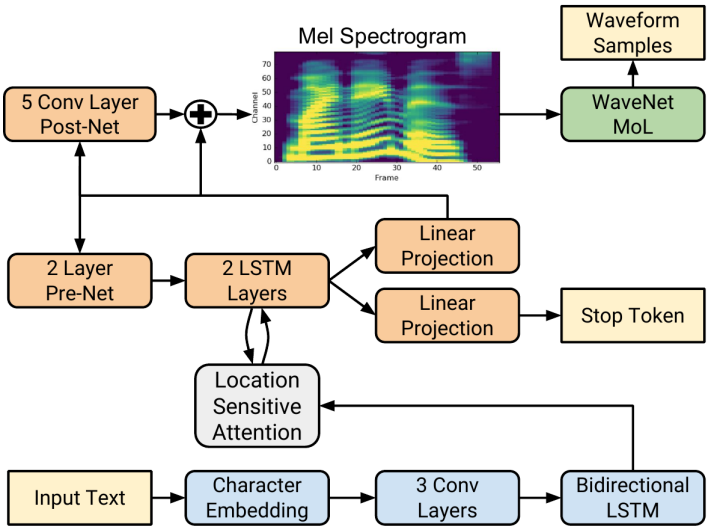


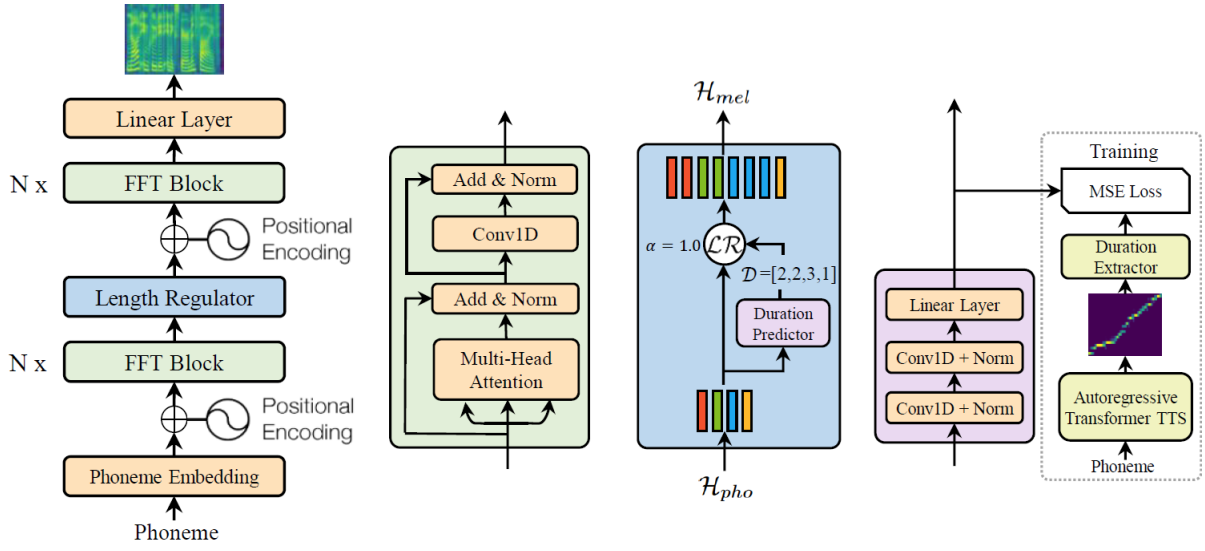
Figure 9: Tacotron2 system architecture (Shen et al., 2018)

ing this simple classification process, CNNs can learn filters and characteristics without requiring extensive hand-engineering (Saha, 2018). In the TTS domain, the use of CNNs has been proposed to solve the excessive training time of RNN models, which are less suited for parallel computation using GPUs (Tachibana et al., 2017). For instance, DeepVoice implements a CNN to enhance a statistical parametric synthesis model, which, after obtaining linguistic features, uses a WaveNet based vocoder to generate waveforms. Subsequent CNN-based type of models improve DeepVoice by integrating multi-speaker modeling, Tacotron + WaveNet model pipelines, or more compact seq2seq models for direct mel-spectrogram predictions (Tan et al., 2021; Arik et al., 2017).

Transformer-Based models Transformer-based models aim to replace RNN structures due to their low efficiency at training and inference, as well as their difficulties in modeling long dependencies (Li et al., 2018). These systems are formed by the same two main components as RNNs, an encoder and a decoder, but both are now built by stacks of several identity blocks. They do not use conventional encoder-attention-decoder framework for sequence to sequence learning but a feed-forward network to generate a sequence in parallel (Ren et al., 2019).

FastSpeech is a feed-forward network based on Transformer to generate mel-spectrograms non-autoregressively, that is, in parallel, and adopts a 1D convolution. Its entire architecture is shown in Figure 10. In this Figure, we can observe that the model consists of multiple feed-forward Transformer (FFTr) blocks on the phoneme and mel-spectrogram sides for the transformation of the former to the latter, and has a length regulation in

between the two to bridge the length gap between the phoneme and mel-spectrogram sequence. This length regulator is built on a phone duration predictor and up-samples the phoneme sequence according to the phoneme duration (i.e., the number of mel-spectrogram frames that each phoneme corresponds to) to match the length of the mel-spectrogram sequence. These duration predictions are obtained by extracting attention alignments from an encoder-decoder based teacher model (Ren et al., 2019).



(a) Feed-Forward Transformer (b) FFT Block (c) Length Regulator (d) Duration Predictor

Figure 10: FastSpeech system architecture (Wang et al., 2017)

However, there are a few shortcomings in the way FastSpeech processes information. FastSpeech has a knowledge distillation system, which aims to transfer knowledge from a large model to a smaller one in a more computationally efficient way in terms of knowledge capacity (Paperswithcode, b). This system is complicated and time consuming. In addition, it has a teacher model for duration prediction. The teacher model is expected to learn the true probability distribution from natural speech data, which is sent to the student model for more informative predictions. Nevertheless, the duration extracted from the teacher model is not accurate enough, and the target mel-spectrograms distilled from teacher model suffer from information loss due to data simplification, resulting in a limited voice quality (Ren et al., 2020).

FastSpeech2 aims to tackle the constraints present in FastSpeech, although it maintains the core feed-forward Transformer block with a stack of self-attention layer and 1D-convolution. In order to improve the aforementioned knowledge processing steps, Fast-

Speech2 trains directly using ground-truth as training targets instead of the simplified output from the teacher. As Figure 11 demonstrates, FastSpeech2 introduces features such as pitch, energy, or more accurate duration, as conditional inputs through a variance adaptor. These features are extracted from the waveform and the predicted values are used at inference (Ren et al., 2020).

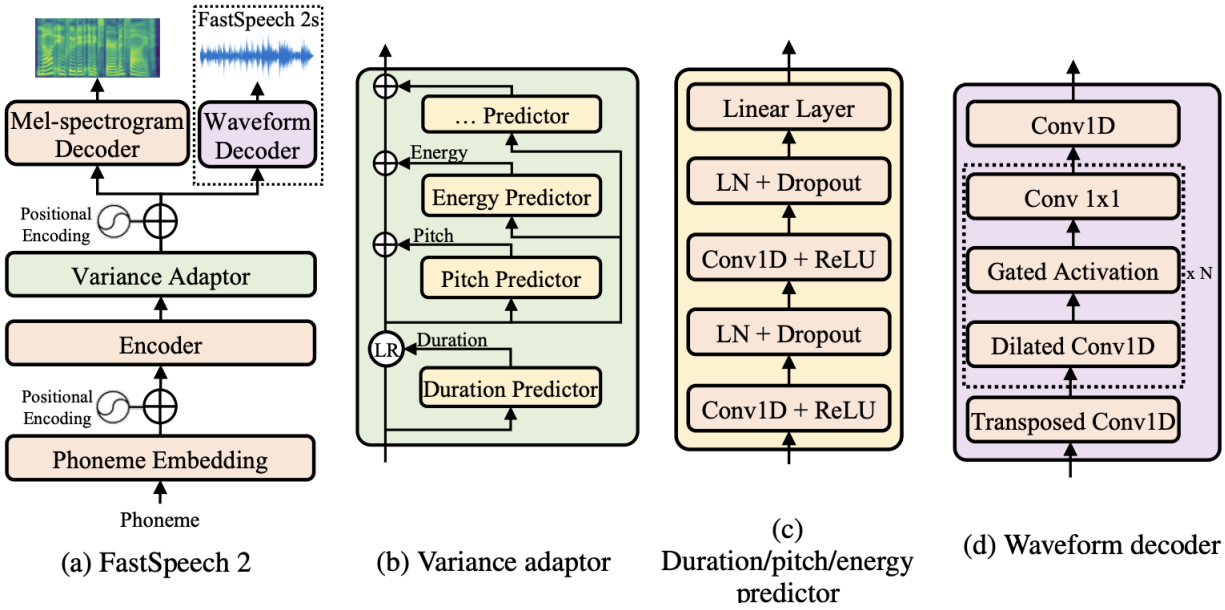


Figure 11: FastSpeech2 system architecture (Ren et al., 2020)

2.2.3 FastPitch model

FastPitch, the acoustic model selected for the training of the Spanish TTS system in the present study, is a feed-forward model with a fully-parallel Transformer architecture also based on FastSpeech. Equally to FastSpeech2, it eliminates the knowledge distillation system of mel-spectrogram targets present in FastSpeech. FastSpeech only predicted duration as input, which derives in pitch contours that collapse different pronunciations of the same unit. FastPitch prevents the absence of linguistic information by performing an explicit modeling of pitch contours. Besides, the approach to the conditioning on F0 is differentto FastSpeech2. The latter predicts contour with a resolution of one value for every mel-spectrogram frame, whereas FastPitch has one value for every input symbol. FastPitch is not conditioned on energy like FastSpeech2 (Łańcucki, 2021).

The architecture of is formed of two main FFTr stacks, one at the encoding level and another one at the decoding level, as we can observe in its architecture in Figure 12.

The first FFTr stack is operating at the input token resolution. It produces a hidden representation $h = \text{FFTr}(x)$ between x, the input lexical unit sequence, and y, the target mel-scale spectrogram frame sequence. The hidden representation h predicts the duration and average pitch of characters in Equation 1

$$\hat{d} = DurationPredictor(h), \quad \hat{p} = PitchPredictor(h) \tag{1}$$

where $\hat{d} \in \mathbb{N}^n$ and $\hat{p} \in \mathbb{N}^n$, with an integrated 1D CNN. The duration of the input tokens is predicted through the integration of Tacotron2. Using the extracted durations d, the model averages F0 values over every input symbol for pitch prediction. Ground truth p and d are used during training, and predicted $\hat{p}$ and $\hat{d}$ are used during inference. The result of the predictions is passed to the second stack of FFTr, which operates at the output frame resolution, and produces the output mel-spectrogram sequence (Łańcucki, 2021).

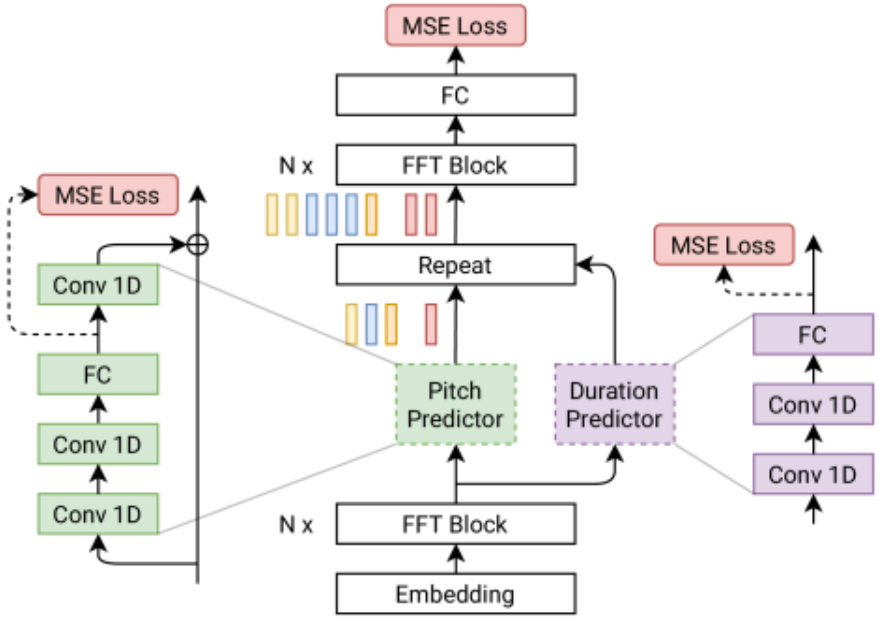


Figure 12: FastPitch system architecture (Łańcucki, 2021)

An distinctive feature of Fastpitch with respect to the other acoustic models is that this model generates mel-spectrograms and predicts a pitch contour from raw input text. During training, for every input character, the model predicts a pitch cue, i.e. an average pitch over a phoneme in Hz. In Figure 13, we can observe an example on how FastPitch is estimating the pitch for every mel-spectrogram frame in the phrase "in being comparatively" (in blue) averaged over characters (in green). Silent letters are assigned a duration

0 and omitted (Łańcucki, 2021). Subsequently, these pitch cues then can be transformed at inference to adjust prosody.

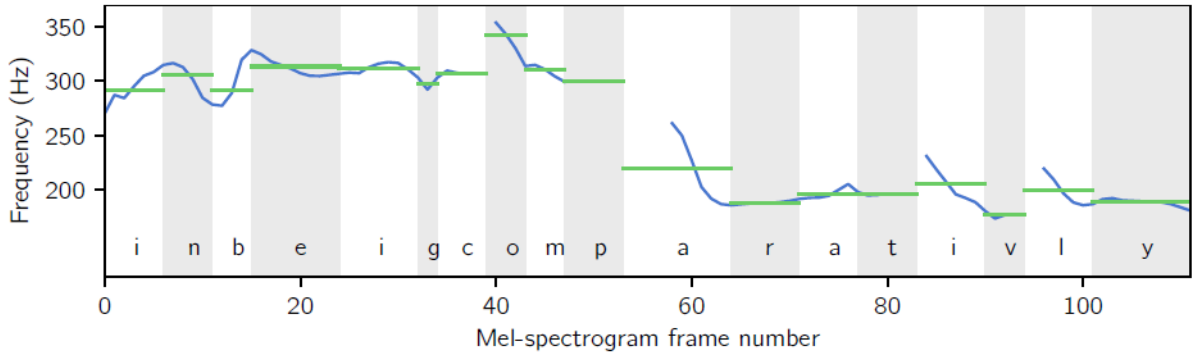


Figure 13: FastPitch pitch predictions (Łańcucki, 2021)

The prosodic cues that can be adjusted include[2]:

- Amplify pitch with respect to the mean pitch
- Invert pitch with respect to the mean pitch
- Raise/lower pitch
- Flatten the pitch to a constant value
- Change the rate of speech

2.2.4 Vocoders

Vocoders are models that transform speech features into raw waveform (You et al., 2021). There are two types of vocoders: the ones used in statistical parametric speech synthesis (SPSS) and the neural network-based vocoders. Introduced in Tacotron2, WaveNet is one of the earliest neural-based vocoders. WaveNet is probabilistic and autoregressive and purely relies on end-to-end learning with almost no prior knowledge on audio signals. It predicts each audio sample conditioning on the previous ones, and the waveform samples generated are conditioned on linguistic features using dilated convolution (van den Oord et al., 2016). Despite WaveNet's good output quality, its inference speed is too slow.

[2]Please refer to the original English version of the pitch transformation examples by Nvidia: https://fastpitch.github.io/

Subsequent efforts of improving WaveNet integrate, for instance, RNNs for unconditional waveform generation, conditioning on acoustic features, linear prediction coefficients to calculate the next waveform samples, or conditioning on BFCC (bark-frequency cepstral coefficients) to ease conditioning on mel-spectrograms . Other type of systems leverage neural-based source-filter model for waveform generations. More recently diffusion-based models have been introduced but they need long inference times (Tan et al., 2021). However, most current state-of-the-art vocoders are flow-based or GAN-based.

Flow-based vocoders Flow-based vocoders are a type of parallelizable statistical generative models that normalize flow and transform a probability density with a sequence of invertible operations Luong and Tran (2021); You et al. (2021). There are autoregressive flow-based vocoders, such as Parallel WaveNet, which model dependency between data distribution and standard probability distribution with teacher distillation; as well as bipartite flow-based vocoders, such as WaveGlow, which leverage the affine coupling layers that ensure the output can be computed from the input and vice versa.

WaveGlow is a flow-based generative model that generates audio from mel-spectrograms by sampling from a distribution. Samples are obtained from a zero mean spherical Gaussian, matching the same number of dimensions as the target output. These samples are then sent through n layers to transform the original sample distribution to the desired one. The modelling of the distribution of the audio samples is conditioned on mel-spectrograms(Prenger et al., 2018).

GAN-based vocoders Generative Adversarial Network-based vocoders aim to outperform autoregressive and flow-based vocoders in both qualitative and quantitative measures, including speed of synthesis (You et al., 2021). GANs are formed by a generator for data generation and a discriminator for the evaluation of the authenticity of the generated data. From the generator's side, the receptive field for modeling long-dependency is typically increased by dilated convolution. The upsampling of the condition information (e.g., linguistic features or mel-spectrograms), is performed by transposed convolution. The discriminator aims to capture the characteristics of the waveform, to then provide a better guiding signal for the generators. GAN losses are also leveraged to improve the stability and efficiency of the adversarial training as well as the perceptual audio quality (Tan et al., 2021). Amongst the vocoders that use GANs to ensure quality audio generation, we can find WaveGAN (Donahue et al., 2018), GAN-TTS (Li et al., 2018), MelGAN (Kumar et al., 2019), Parallel WaveGAN (Yamamoto et al., 2019) or HiFi-GAN (Kong et al., 2020).

HiFi-GAN, one of the vocoders used in the present study, aims to generate high fidelity

speech efficiently (Kong et al., 2020). Supposing that a speech audio consists of sinusoidal signals with various periods, HiFi-Gan models periodic patterns of an audio for enhancing sample quality. It's composed by one generator and two discriminators: multi-scale and multi-period discriminators. Each of the latter obtains only a specific periodic part of raw waveforms in paralell. Discriminators accept equally spaced samples of an input audio with a period, and then they are applied to the generator (Kong et al., 2020; Tan et al., 2021). The generator of this model processes different patterns of various lengths in parallel through a multi-receptive field fusion module, and also has the flexibility to trade off between synthesis efficiency and sample quality (Tan et al., 2021).

2.3 TTS in Spanish

Contemporary end-to-end systems have achieved remarkable quality standards and can produce almost human-like natural sounding speech in real time. Thanks to the aforementioned research and advancements in the field, these systems nowadays do not require excessively high demands on the quality, amount, and preprocessing of training data. This allows researchers to investigate other techniques such as expressiveness, controllability, voice conversion, or few-shot voice cloning. When extending these models to support multiple languages or a language other than English, we may find several obstacles, including mismatch in the input representations or pronunciations, and imbalanced amounts of training data per language (Nekvinda and Dušek, 2020). In this section, I will provide an overview of the available resources in Castilian Spanish in the field of text-to-speech.

2.3.1 Databases

In this section I present a list of available TTS datasets in Spanish. These are usually formed by a number of minutes/hours of speech recordings in Spanish and their corresponding orthographic transcription in text files. In some occasions, there are additional annotations of the data, such as phonetic transcriptions, lexicons or part-of-speech tags. The number of the available databases in Spanish is limited, especially open source free-to-use ones, as we can see in Table 2.

Open source work is Creative Commons Zero (CC0) license, a public domain license that is not adapted to the laws of any particular legal jurisdiction, therefore allowing the free distribution of an otherwise copyrighted "work" (CreativeCommons). The rest of the licenses are more restricted, from permitting only noncommercial uses of the work,

to requiring credit attribution to authors, sharing adaptations under the same terms, or no derivatives or adaptations of the work at all. Each database name has its source link embedded in it for further information on their respective license permissions, including the details for obtaining and using them.

Database	**Source**	**License**	**Duration (h)**	**Speakers**
CommonVoice ES	Mozilla	CC0	3,050	83,790 mixed
CSS10	LibriVox	CC0	23.82	1 male
MAILABS (ES)	LibriVox & Project Gutenberg	CC0	108.5	2 male 1 female
MLS (ES)	LibriVox	CC BY 4.0	1,438.41	120 mixed
Spanish TTS Speech Corpus	Appen	CC	1.45	1 male
TC-STAR	UPC	CC	20	2 male 1 female

Table 2: List of available Spanish databases

2.3.2 Tools

There are several tools that can be used for TTS training and quality improvement purposes that are language dependent, particularly in the text processing TTS frontend module. The objective of these TTS tools is to prepare the data in a way that the Spanish linguistic feature extraction from the audio and text data is performed correctly. Among these tools, we can find text cleaning tools for the normalization of numbers and character clean up, grapheme to phoneme conversion tools for the phonetic transcription of orthographic text, and other NLP tools for text parsing and linguistic feature extraction such as tokenization, lemmatization or part-of-speech tagging.

Text preprocessing Preprocessing the text involves several steps, including transcribing numbers or abbreviations/acronyms into verbalized orthographic form, and removing certain characters or symbols. This might be a challenging task, as it involves replacing the raw target part of speech with a new text, while maintaining its original coherence and adequate syntactical or grammatical forms. To perform this task, automatic text normalization models are usually rule or neural based.

In addition, punctuation plays a key role in the way texts are interpreted by the TTS system in Spanish (LumenVox). As it is shown in Table 3, intonation depends heavily on punctuation. Depending on each type, we might have longer or shorter pauses between

utterances, or rising or falling intonation at the end of a sentence. For instance, the spoken difference between “Quieres venir” versus “¿Quieres venir?” is marked with intonation. If we input this information for the model to learn it as well, we can have a more natural intonation at inference. It is therefore desirable to maintain original punctuation in our post-processed text. Another option to input these prosodic features to the model would be to annotate intonation patterns and features manually together with input text for its explicit learning, for example, using Tone and Break Indices (ToBI) tags (cf. (Hwang and Yu, 2020; Zou et al., 2021) for information on prosodic modeling with ToBI and (Beckman et al., 2002) for additional information on Spanish ToBI). This requires long and detailed annotation work, however.

Punctuation marks	**Pause**	**Intonation**
,	small	slightly rising
; :	medium	falling
. !	long	falling
¿ ?	long	rising or falling

Table 3: Intonational patterns in Spanish depending on punctuation (LumenVox)

While it is common in English, we can find very few tools that will perform these two tasks (numeric/abbreviation/acronym translations and text cleanup) simultaneously in Spanish. One of the libraries that performs a complete normalization task is the Nemo text processing toolkit from Nvidia. This toolkit has recently added Spanish to their supported languages, English being the most complete one with formatting and normalization of a larger number of cases. The NeMo text normalizer in Spanish covers the verbalizing of numeric cases such as regular or roman numbers, dates, time, and ordinals. It also takes care of other characters and symbols including money, email domains, and abbreviations. NeMo is based on Word Finite State Transducer (WFST) grammars. This allows to easily modify and customize the underlaying grammars and normalization cases covered, as well as to manually add new cases (versus having to retrain a new model as in the case of neural normalizers). The diagram in Figure 14 shows an example on how the NeMo performs the text normalization task in Spanish sentences with different cases of text normalization.

There are also a few options for treating each text processing steps separately. For writing numbers into orthographic form, a widely used tool is num2words. This package deals with numbers exclusively, simply converting different types (cardinal, ordinal, fractions, etc.) into verbalized form. For text cleanup, we can find other options such as writing a replacing script for unwanted symbols or characters, or tokenizing and lemma-

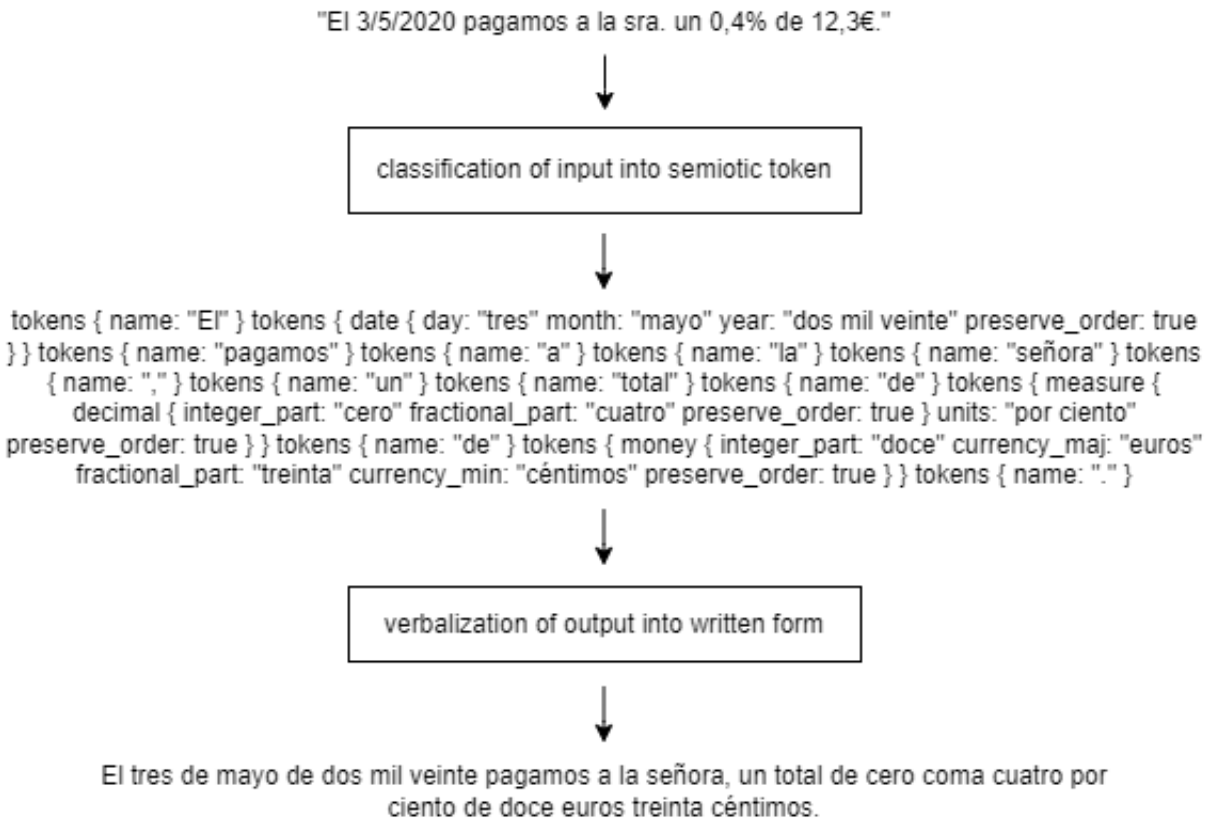


Figure 14: Example of text normalization in Spanish with NeMo

tizing our text and using other libraries such as Cucco. Cucco removes stop words, extra white spaces or undesired punctuation. For tokenizing and lemmatizing text in Spanish, we might use NLTK, Spacy, or Flair. However, this second option involves more manual steps and packages.

Grapheme-to-phoneme conversion Phonetic transcriptions convert graphemes into phonemes in the target language. These are needed for latest neural TTS models, since the acoustic model gets phoneme representations as input. Obtaining the phoneme representations are obtained either from a grapheme-to-phoneme software or from a lexicon that can be created manually or with the help of a G2P software. Phonetic transcriptions are created with spelling-to-phoneme translation rules. It is possible to create one's own phonetic rule set or dictionary in Spanish to be used by a TTS model by writing a script that maps a list of words in Spanish to phonemes. In the case of Spanish this task is easier than in other languages, as each sound correlates with the same sound representation most times. However, there are always exceptions that need rules (e.g. the pronunciation of

/r/ is different in the words “rojo” or “aroma”). As we can infer, creating these rules is a laborious work, as it involves understanding the main phonetic and phonological aspects of the language, being aware of its features, and having patience and attention to detail to carefully craft them. The most efficient option is to use open source phonetic transcriptors that have been already well developed and tested by experts in the field.

In the case of Spanish, the tools selected for transforming input text graphemes into phonemes are the following: for transcriptions into IPA, we can find eSpeak-ng, a text-to-speech software supporting a lot of languages and IPA phoneme output with information such as pitch and length. Transcriptions into SAMPA can be performed by AhoLab’s Libbertso `Modulo1y2`. This module performs a transcription that maintains the accent marking, as well as taking into account word and phoneme boundaries. The MBROLA backend of eSpeak-ng (eSpeak-ng-MBROLA) is another useful resource that provides SAMPA spelling-to-phoneme translation and intonation, although it does not preserve word boundaries. These two backends are gathered into the Phonemizer library. Although phonemization might be slow for phonemizing a large corpus in real time, Phonemizer allows a simple phonemization of words and texts in Spanish with separators for phones and words (latter not supported by the espeak-mbrola backend), as well as preserving punctuation (excepted espeak-mbrola).

2.3.3 Acoustic models and frameworks

To conclude, in this subsection, I have included some suggestions and guidance to take into consideration in order to train a model in Spanish or any other language. The most important part to take into consideration are the previous two aforementioned in the linguistic module, that is, input data processing and phonetic transcription. Because the input data is language specific, the data processing step and the linguistic module need to be carefully adapted before training. Once we have our text preprocessed, we can select the desired acoustic model and adapt its linguistic module to integrate our G2P method. Once the acoustic model learns the correlation between the phonemes of the target language and sound, the next steps of the training are no longer language dependent. The available acoustic models that can be fairly easily adapted for training end-to-end TTS in different languages are listed in Table 4.

Model Name	Source	Framework
ESPnet	(Watanabe et al., 2018) (Hayashi et al., 2019)	FastSpeech2, Tacotron2
Coqui	Coqui	FastSpeech2, Tacotron2
FastPitch	Nvidia	FastPitch
IMS-Toucan	University of Stuttgart	FastSpeech2

Table 4: Available end-to-end TTS models

2.4 Chapter summary

In this chapter, I introduced key studies about text-to-speech. I provide a historical context, starting from the early experiments and models. Thereafter, I review current state of art TTS architectures, models, and vocoders. Finally, I provide an overview of the state of the art in Spanish TTS, including available resources in this field. In the following section, I will explain the methodology and procedures followed for the completion of this study.

3 Methodology

In the previous section, I have reviewed and summarized the evolution of the development of text-to-speech technology, as well as current state-of-the-art TTS systems and their features. In addition, I provided an overview of the resources available for speech synthesis in Spanish, including databases and tools. In this section, I explain the setup for this project and introduce the data and tools used for its successful completion. This includes methods used for the cleanup of the data, the setup and training of the acoustic model, and the evaluation of the different systems.

3.1 Project setup

The setup of this project begins with the preparation, study and literature review of the topic. The study, work and progress carried out has been documented weekly with detailed examples and explanations on the progress for tutor-ships and follow-ups. I did extensive reading about state-of-the-art acoustic models, as explained in the previous section, and documented myself to select an acoustic model for this work. As a result, I concluded that most competitive TTS acoustic models include FastPich and FastSpeech2.

Fastitch and FastSpeech2 models have been developed almost parallel in time and share good quality results in the synthesized speech. They both take phoneme as inputs. FastSpeech2's predicted F0 contour has a resolution of one value for every mel-spectrogram frame, discretized to 256 frequency values; FastPitch's predicted F0 contour has one value for every symbol (Łańcucki, 2021; Ren et al., 2020). However, while both models have different approaches to conditioning on F0 with good results, FastPitch has a unique feature, which is that it explicitly models pitch contours. This feature allows for post-training pitch modification control at inference. I consider this an innovative technique since, in most cases, the most common way of manipulating or improving prosody is by vocoder training and adjustments, and is the main reason why I decided to move forward using FastPitch as acoustic model.

The handling of the data, the training of the acoustic model and the creation of the evaluation system for the three acoustic model was entirely carried out in the AhoLab server. In this server, I first create symlinks to the raw data and perform an initial testing of FastPitch in English using the default setup provided by Nvidia, to make sure that both model and server configuration work well. In order to set up FastPitch, I cloned the source FastPitch repository, and built and ran the FastPitch PyTorch NGC docker

container (which contains all the requirements and dependencies of the model). The Spanish FastPtich acoustic model was trained using one GPU provided by AhoLab, GeForce GTX TITAN X. Specifics of the configuration for each experiment carried out are further explained in Section 4.

3.2 Data

The database for the training of this acoustic model in Castillian Spanish consists of the audio recordings and annotation text files of approximately 10h of speech from a female baseline speaker. The source and information of this TC-STAR original database can be found in Table 2. This makes a total of 3,675 initial audio and text files. The database is fully documented and validated, the annotation text files of the database include a manual orthographic transcription of what was really pronounced by the speakers. The speech text files are aligned with the audio files: both file types have the same filename, and each text file has the aligned audio ID and transcription of the speech data, as shown in Example 3.1. There is one label file named SAM assigned to each text file that includes relevant information to our project such as the transcription and the time stamps in milliseconds of the start and end of each transcribed utterance, in the format shown in Example 3.2.

Example 3.1
T6B72110035—El presidente Santer inspira confianza. Se trata de una cualidad muy importante. Pero es también una cualidad que, hasta el momento, no ha utilizado suficientemente.

Example 3.2
LBO: 0.826000,2.034,3.242000,el presidente Santer inspira confianza .
LBO: 3.794000,4.906,6.018000,se trata de una cualidad muy importante .
LBO: 6.350000,8.602,10.854000,pero es también una cualidad que , hasta el momento , no ha utilizado suficientemente .

The initial audio format of the original dataset has the specifics show in Example 3.3. As shown, all signals have a sampling rate of 96 kHz, 24 bits precision with the least significant byte first ("lohi" or Intel format) as (signed) integers. The recording tool used is NannyRecords. Recordings were done in a silence room and the system records simultaneously a close talk microphone, a mid distance microphone and a laryngograph signal. Channel 1 contains the recordings from the Far microphone, Channel 2 contains

the recordings from the Laryngograph, and Channel 3 contains the recordings from the Close talk microphone.

Example 3.3
Input File : 'T6B72110035.wav'
Channels : 1
Sample Rate : 96000
Precision : 24-bit
Duration : 00:00:04.99 = 479232 samples 374.4 CDDA sectors
File Size : 1.44M
Bit Rate : 2.30M
Sample Encoding: 24-bit Signed Integer PCM

3.2.1 Data pre-processing

Data preparation procedures are of great relevance for the output quality of a TTS system. In the case of the present study, the original corpus needs several modifications in order to be used as input to our acoustic model. For instance, the text files contain punctuation marks and special characters that provide no specific phonemic sound (unless normalized to the orthographic pronunciation of the symbol), and hence, the acoustic model will not be able to draft and learn a correlation between this grapheme and a mel-spectrograms sound value. I therefore remove these during this data processing step.

In addition, there are multiple utterances separated with full stops inside the same audio and transcription files, as it can be observed in Example 3.1. These need to be separated into separate single audio and text files. I use the SAM documentation files for this task, since they contain the time information on the beginning and end of the utterances in seconds as seen in Example 3.2. In the first of the three utterances of this example, we can observe three comma separated numeric values. The first and third value indicate the beginning and the end of the utterance expressed in seconds, so we can know that "el presidente Santer inspira confianza" is spoken within the first 0.826 and 3.242 seconds. This was confirmed by listening and testing a few audio files. Therefore, using these time stamps, we can split the audios at the beginning and end of each utterance and save them as new individual audios.

The Wave library allows for audio splitting in seconds with decimals (Wave). I adapted the code from the source Wave library documentation and provided these duration stamps to divide the utterances. Then, I follow the same procedure with the text files, dividing

them following the SAM annotations and saving the file with the same ID as the audio file to maintain a correct audio/text alignment.

During this process, I performed several modifications to the audio file format. For example, I converted the audio data to a 16-bit pulse code modulation (PCM) sample encoding format, because the 24-bit Floating Point PCM sample encoding is not accepted by the Wave library. The term pulse code modulation (PCM) refers to the use of a specific set of rules to digitally represent sampled analog signals, transforming a waveform into a stream of digits and vice versa (Eugene and sneps sneppe, 2019). In addition, because the input audio format of the FastPitch acoustic model is 22050, I resampled the audios as well. I used SoX for the bit-depth and sample rate format modification task of all the audio files. SoX is a cross-platform command line utility that, among other signal handling and processing tasks, can convert various formats of computer audio files into other formats (Sox). The end result of the audio pre-processing is shown in Example 3.4

Example 3.4
Input File : 'T6B72110035.wav'
Channels : 1
Sample Rate : 22050
Precision : 16-bit
Duration : 00:00:02.42 = 53273 samples 181.201 CDDA sectors
File Size : 107k
Bit Rate : 353k
Sample Encoding: 16-bit Signed Integer PCM

Subsequent to the audio format changes, the sentences are split. The resulting number of files was 5,334. Some of the audios had to be removed because they contained several long utterances separated by commas and semicolons, which caused the files to be too long (time wise). This means that they are not only difficult for the model to process in terms of Random Access Memory (RAM) space, requiring a smaller batch size, but also contain speech of sentences with unnatural intonation. After discarding those files, the final corpus contains a total number of 4,268 audio files and same number of text files (one sentence per audio file). The original corpus does not include validation and test sets, but it was divided for this experiment into train, test and validation subsets. The final amount of audio/text files of each subset after dataset division is:

- Train set: 4,141

- Validation set: 127
- Test set: 40

3.3 End-to-end text-to-speech

It has been established in Section 2 that there are three main components in a TTS system: the text processing part, where I obtain the linguistic representation of the input data, the acoustic model, which learns the relationship between the linguistic representation and the acoustic data, and the vocoder, which generates the synthetic speech from the linguistic representation. As such, FastPitch, the acoustic model selected for this work, follows the same flow. However, this model is originally designed to train in English data, and its linguistic module needs some adaptation in order for it to learn Spanish linguistic features as well. In addition, FastPitch needs an implementation with our dataset for TTS, as well as a tailored evaluation system from which we obtain results. In this subsection, I explain the methodology and the tools used for the end-to-end implementation and evaluation of our Spanish TTS based on FastPitch.

3.3.1 Linguistic feature configuration

FastPitch is designed to predict mel-spectrograms from input symbols. The input symbols for this model can be selected as graphemes or phonemes. In the case of graphemes, FastPitch takes the raw text and turns it into pure graphemes, that is, written symbols that represent a sound. If the selected input is phoneme, the format by default in FastPitch is ARPAbet (Advanced Research Projects Agency phonetic transcription). The model has a local dictionary file: the Carnegie Mellon University (CMU) Pronouncing Dictionary (Carnegie-Mellon-University). The CMU is a dictionary that contains a large set of words and their corresponding phonetic transcription based in ARPAbet symbols. The text processing module of FastPitch processes the input raw text word by word, looks up each word in the CMU dictionary, and returns its ARPAbet representation. Then, this output gets encoded, and the model will align the encodings to output mel-spectrogram frames automatically.

As it has been established in the review of literature, phonemes result easier and more accurate for the model to learn a relationship with acoustic features. This is because there is a clearer mapping from spelling to phonemes than to pure orthographic characters or even graphemes. For instance, the phoneme `/k/` in English can be represented by the

orthograpic/grapheme forms `k` (kite), `c` (case), `ck` (lock) or `ch` (choir). Thus, I selected the method of using phonemes as input.

However, instead of using ARPAbet as the phonetic annotation format, the text files of the original corpus are phonetically transcribed to Speech Assessment Methods Phonetic Alphabet (SAMPA) format. I chose to transcribe the data into SAMPA format and replace ARPAbet with it for two main reasons: first, because this ARPAbet backend is only supporting North American English, and our Spanish corpus needs Spanish phoneme representation for words; second, because I was provided a linguistic module developed by the AhoLab center for this task, which transcribes directly and efficiently orthographic text into Spanish SAMPA.

AhoLab's SAMPA phonetic transcription module[3], `Modulo1y2` within the `Speech` functionality, is part of AhoLab's in-house text processing fronted package named Libbertso, and transcribes text to Spanish and Basque SAMPA using a personalized Spanish dictionary as backend. The tool allows for choosing these two fundamental parameters as arguments, `Lang` for the language, which can be set as `es` for Spanish or `eu` for Basque, and `HDicDB` for selecting the desired dictionary file. In addition, this module cleans and normalizes the text, i.e. expands numerals from digit to orthographic form (e.g. `8` as `eight`) and removes special characters, and allows for phonemizing sentence-by-sentence or word-by-word, with or without stress markers. Therefore, prior to the training of the model, I used the `Modulo1y2` transcriber to create my own pronunciation dictionary in Spanish SAMPA offline, which contained each unique word in the corpus, alphabetically sorted and followed by its phonetic notation.

3.3.2 Model implementation

After the original corpus has been pre-processed and the dictionary with the words in Spanish SAMPA has been created, the data is ready for modeling. In this subsection, I explain the process of implementing our data and modules. Next, I overview the steps for the training of the FastPitch acoustic model, as well as for the inference process. All three experiments conducted throughout this work, which are further developed in Section 4, follow the same implementation methodology.

The first step is to organize and place the training data in the right place within the model. With the datasets in place, I then proceed to the implementation of the custom

[3]The AhoLab linguistic processing module can be found in this source: https://sourceforge.net/projects/ahotts/

dictionary in SAMPA extracted from the database. Within the FastPitch model, there are three main modules that interact with the linguistic features: the `symbol` module, which defines the set of symbols used in text input to the model; the `CMU dict` module, which looks up for the target word in the dictionary file and returns its phonetic transcription; and the `text processing` module, which handles the input words, calls the previous two modules for their processing, and encodes the output to return it to the next learning steps. To conclude, I set up the model parameters for training of the model, as well as for the creation of the synthesized inferences. These steps are further developed in the following subsections.

Training and validation sets I hereby divide the audio and text data into train and validation sets, and place the two files in the corresponding folder of the FastPitch model named `filelists`. The filelists contain transcripts and paths to `.wav` files in the format `‘<audio file path>|<transcript>‘` and define training/validation split of the data. Furthermore, because this model uses ground truth pitch values and performs explicit modeling of such pitch contours, the folder also needs to have file lists with paths to pre-calculated pitch files. The pitch gets calculated by simply running a pre-processing script provided in the FastPitch model.

Symbols module The symbol module of FastPitch is tailored to English set of symbols only. The default is a set of ASCII characters that works well for English or text that has been run through Unidecode. This includes punctuation marks, math and special characters, accented letters and common letters. I therefore replicate the same symbol set but replacing and adding the default characters by Spanish ones, such as inverted question `¿` and exclamation `¡` marks, accented vowels or the letter `ñ`.

Dictionary module I first add the Spanish dictionary file to the same directory where the CMU dictionary file is. As mentioned, the default dictionary module will parse the input words, access that file, and look for the word to return its phonetic transcription. If there are heteronym words (i.e. words with the same spelling but different pronunciations and meanings), it looks for the correct pronunciation for the target heteronym. I deactivated this latter function because we did not add information on heteronyms in my dictionary file due to time constraints. I modified the set of valid symbols and, instead of having the ARPAbet symbol set, I added the SAMPA set by obtaining the all unique characters in my phonetically transcribed data.

Text processing module The text processing module handles the symbols and dictionary modules. It first parses and cleans input text with the symbols and cleaner modules.

Before the word gets transcribed into phonemes, the text processing module also calls other modules that further normalize the text by expanding abbreviations, acronyms, numbers and letters such as measurements, magnitudes or other currency symbols, into written forms. However, these have been all designed for English and creating a Spanish version of it would need a extensive detailed engineering of each form and context.

In addition, the text has been previously cleaned and normalized in our data pre-processing step, so I eliminated the cleaning modules from my processing. For the same reason, I also deactivated the cleaner module by creating a None type of cleaner that applies no post-processing to our text. Next, the text processing module calls the dictionary module and obtains the returned trascribed ARPAbet form of the word. Finally, it encodes the phonemized word output by mapping it into a sequence of IDs.

Model training The parameter setup of FastPitch used in the following experiments does not differ much with the default configuration by Nvidia. Please refer to FastPitch model's source code for detail specifications on the model configuration. However, due to our GPU and setup conditions, as specified in 3.1, as well as our previous changes and implementations in the linguistic module, a few parameters needed to be adjusted. I modified some arguments that the training script takes, including the paths for the dataset and training files (adjusted to our audio path and training and validation filenames), the number of speakers (set to one), and the symbol set and dictionary paths (adjusted to my new Spanish symbol class and dictionary file). The training file configuration also needed adjustments for input audio data. Additionally, I changed the values for the batch size for a single forward-backward step and the gradient accumulation (i.e. the number of steps over which gradients are accumulated), taking into account the computational power of the current setup. Details on parameters are further described in 3.3.3.

Model selection The model was trained up to 1,000 epochs and the latest checkpoint of 1,000 epochs was selected.

Inference For the inference process, I adapted an inference example file provided by Nvidia, and adjusted the paths to those of the trained acoustic model, vocoder and files with the phrases to synthesize. The model is trained using a dictionary with the phonetic transcription of all the unique words contained in the input dataset. This means that new out-of-vocabulary (OOV) words will not have a phonetic transcription. I tested how these words were synthesized, and as expected the output of the words resulted unintelligible.

I then implemented a real time phonetic transcription solution to this problem by adding AhoLab's `Modulo1y2` backend as support for OOV words in the linguistic module

of FastPitch. This way, when the model looks up for a word in the vocabulary and does not find any transcription, AhoLab's G2P `Modulo1y2` will be called and return a phonetic transcription for any word. This is also useful for retraining the model directly with a different dataset without having to create a custom dictionary for it.

3.3.3 FastPitch configuration

Audio configuration depends on the input dataset and on the input configuration of the vocoder. FastPitch has two pre-set configurations for 44100 and 22050 Hz sample rate input audio. These two configurations include adjusted filter length, hop length, window length or maximum and minimum mel frequency values. These values are carefully selected for audio data at those two specifics sample rates. I used the 22050 preset configuration for our data, taking into account that the vocoders to use for synthesis have been trained at 22050 Hz as well.

Audio files are transformed into mel-spectrogram frame feature vectors. These audio sequences are encoded following the configuration of the audio, i.e. sample rate, filter length, hop length, window length, and mel frequency minimun and maximum values. The encoded vectors are smaller, which make them more manageable to be process by the model, but contain the important information on the audio signals. Text files get processed, each word is looked up in the dictionary, and encoded into feature vectors as well.

I then selected the environment variable configuration for the batch size. Batch size refers to the amount of data sent at each time to the model for learning. A training step is one gradient update, and in one step batch size examples are processed. Gradient accumulation refers to the training steps to accumulate gradients for. An epoch consists of one full cycle through the training data. This is usually many steps. As an example, if you have 2,000 audios and use a batch size of 10 an epoch consists of 200 steps:

$$\frac{2,000 \text{ (audios)}}{10 \text{ audios / step}} = 200 \text{ steps} \tag{2}$$

In our case, the GPU available was one, so I set the batch size to 8, and the gradient accumulation to 32. This decision was based on two factors. First, FastPitch is required to maintain a global batch size, which means: `NUM_GPUS x BATCH_SIZE x GRAD_ACCUMULATION = 256`. Second, I tried to optimize the RAM memory and train at lesser time by sending the largest possible batch size as input. By following the first re-

quirement, I adjusted the batch size to the bigger amount allowed by our RAM memory, which is 8, resulting in a gradient accumulation value of 32, therefore meeting required global batch size (1 x 8 x 32 = 256).

In this case, the total amount of training data is 4,141. Since the gradient accumulation is set to 32, per each GPU (in our case just one), I am accumulating gradients 32 times in one training step. In each step batch size examples get processed, and I set batch size to 8 as per the maximum processing data allowed at once. This means that at at each gradient update, as shown in Example 3, I am accumulating 32 steps of 8 batches of input data, resulting in a total batch size of 256. For our input data of 4,141, one iteration of the entire data is completed in 16 steps (MarvMind).

$$32 \text{ gradients accumulated } \times 1 \text{ step (8 batch-size)} = 256 \tag{3}$$

$$\frac{4,141 \text{ (input vectors)}}{256 \text{ (batch size)}} = 16 \text{ steps} \tag{4}$$

The model learns the relationship between the audio and text feature encodings once after it gets as input 16 iterations through the entire data. As mentioned, this is one epoch. I trained the model up to 1,000 epochs, which results in 16,000 iterations of the data. In the model parameters, I set saving the checkpoint every 20 epochs so I can then select the best one for evaluation.

No further network configuration was adjusted. A summary of the the selected environment variables and the parameters of the network selected (besides the default configuration of FastPitch) are shown in Table 5.

In the training process, the mel-spectrogram generation neural network is fed with lexical unit label vectors as input, $x = (x1, \ldots, xn)$, and their corresponding sequence of target mel-scale spectrogram frames as outputs, $y = (y1, \ldots, yn)$. Data is first fed to the first *N* feed-forward Transformer (FFTr) stack which operates in the resolution of input tokens. These FFTr blocks will encode input and produce the hidden representation $\boldsymbol{h = FFTr(x)}$. This hidden representation $\boldsymbol{h}$ is used to predict duration and average pitch of every phoneme with a 1-D CNN, having a predictor for each feature type as shown in Example 5, where $\hat{d} \in \mathbb{N}^n$ and $\hat{p} \in \mathbb{R}^n$.

$$\hat{d} = DurationPredictor(h), \quad \hat{p} = PitchPredictor(h) \tag{5}$$

Pitch is then projected to match the dimensionality of the hidden representation and

Parameter type	Parameter configuration
Text	- Phone = true - Text cleaners = none
Audio	- Sampling rate = 22050 - Filter length = 1024 - Hop-length = 256 - Win-length = 1024 - Minimum mel-frequency = 0.0 - Maximum mel-frequency = 8000.0
Data processing	- Number of GPUs = 1 - Batch size = 8 - Gradient accumulation = 32 - Epochs = 1000 - Epochs per checkpoint = 20

Table 5: Parameter configuration of FastPitch model network

added to h, resulting in g. Subsequently, g is discretely upsampled and goes through another set of output *N* FFTr blocks, with the goal of smoothing out the upsampled signal, and constructing a mel-spectrogram sequence as output (Paperswithcode, c; Łańcucki, 2021). For training, only ground truth **p** and **d** are used.

$$\mathbf{g} = \mathbf{h} + PitchEmbedding(\mathbf{p}) \tag{6}$$

$$\hat{y} = FFTr\left(\underbrace{[g_1, ..., g_1}_{d_1}, ..., \underbrace{g_n, ..., g_n]}_{d_n}\right) \tag{7}$$

During inference, FastPitch uses the predictions on pitch ($\hat{\mathbf{p}}$) and duration ($\hat{\mathbf{d}}$) calculated in the model training step. The mean-squared error (MSE) between the predicted and ground-truth modalities will then be optimized by the model. Predicted values are then provided to a waveform synthesizer such as WaveGlow or HiFiGan at inference, which will construct the output synthesized speech waveform.

3.3.4 Evaluation tool

Non-Intrusive Speech Quality Assessment (NISQA) is used for the evaluation of the results (Mittag et al., 2021). NISQA is a deep learning model/framework for speech quality prediction trained on a large dataset containing mean opinion scores (MOS) and the audios

for which they provide the evaluation. MOS is a numerical measurement of a human evaluation widely used to rank voice and video quality. The scale used for the quality evaluation typically ranks from, where 0 means poor and 5 means excellent; then the average number is calculated (Karim and Saleh, 2022). Trained on this type of MOS evaluations, the NISQA-TTS model weights can be used to estimate the Naturalness of synthetic speech generated by a text-to-speech system like the one presented in this work.

3.4 Chapter summary

In this chapter, I described the methodology followed for the setup of this project, as well as the processing of the data, the configuration of the linguistic modules and their implementation in the model. In addition, I explained the broad configuration of the training and inference scripts of FastPitch. In the following section, I further develop on the experiments and the specifics of their configurations and the evaluation and results of the experiments.

4 Findings

This chapter details the findings of this project. There are several findings which correspond to three distinct experiments detailed in the following subsections. Each of these experiments implements different strategies for neural-based text-to-speech synthesis in Spanish, following the process described in Section 3. The first experiment, described in Section 4.1, outlines the initial experiments where I implement the Spanish dictionary and train FastPitch for the first time with the preprocessed dataset. The output of this training is then synthesized using WaveGlow and HiFiGan vocoders, and resulting audios are compared. The second experiment, described in Section 4.2, aims to compare the performance of three different acoustic models (FastPitch, Tacotron2 and FastSpeech2). The three are trained under the same conditions for quality comparison of the different frameworks using the same Spanish dataset. In addition, this experiment implements the same architecture but uses a different vocoder, HiFiGan, for the inference step. The third experiment, described in Section 4.3, explores the possibilities and flexibility of the pitch alteration application offered by the FastPitch model at inference in our trained Spanish model. Each of these sections is followed by an analysis of the evaluation and results obtained from each experiment, as well as information on sources where sample audios can be listened to.

4.1 Experiment 1: WaveGlow and HiFiGan

The first experiment aims to first test the performance of FastPitch on our Spanish dataset. Besides the initial testing of FastPitch, the objective of this experiment is to compare its performance between the default WaveGlow vocoder provided by the model and a HiFiGan vocoder fine tuned with the same dataset. For this task, the model is trained for the first time and output speech waveform is synthesized with WaveGlow. Later, the mel-spectrograms are synthesized using HifiGan.

4.1.1 Experimental setup

After the initial processing of the data and implementing the custom dictionary and linguistic module, I prepared the test and validation sets. During the training process, I debugged a few last issues in the dataset, and checked the logs to make sure the learning is progressing as expected and iterations were successfully completed. I then trained FastPitch, selected the latest checkpoint, and synthesized the speech waveform of the 40 test

sentences, first with WaveGlow, second with HiFiGan. This vocoder has been fine tuned using the same dataset with which the FastPitch acoustic model was trained [4].

4.1.2 Evaluation and results

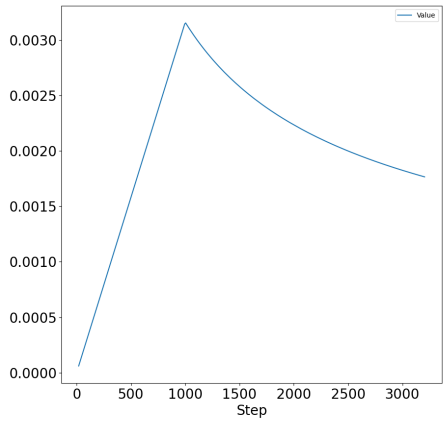
In the initial steps of the training, we can observe the learning rate, mean gradient, loss and mel losses to assess the evolution of the learning during the first epochs. The learning rate controls how much to change the model in response to the estimated error each time the model weights are updated (Brownlee, 2019). The mean gradient represents the changes in all weights with regard to the change in error (Donges, 2021). Loss is the penalty for a bad prediction. In other words, it represents a value that indicates how bad the model's prediction was on a single example (Google). Therefore, we are looking for our model to minimize loss during training. The overall loss is computed on training and validation by summing the errors made for each example of the two sets. The mel loss refers to the mel-spectrogram reconstruction loss between the predicted and the ground truth mel-spectrogram.

The initial 200 epochs are first analyzed. As mentioned in Section 3.3.3, the number of steps per epoch of our model is 16. At 200 epochs, the total of training steps is 3,200 (i.e. epochs times the batch step of 16). In the following figures we can observe the progress on the gradient mean, the loss and the learning rate of the training. The 3,200 steps represent X axis of the figures, projected in wall time (Y axis).

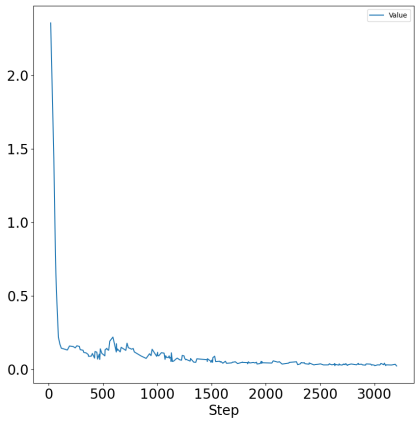
In Figure 15a, we observe that the learning rate increases linearly over the first 1,000 epochs. This means that, at every weight update, the acoustic model is moving faster towards the optimal weights with respect to the loss gradient descent. After 1,000 epochs, the learning is not so fast and starts slowing down progressively. The mean gradient increases and decreases weights depending in error, as we can see in Figure 15b. Because of the gradual adjustment, the changes in weights gets smaller at each epoch, especially after 1,500, showing stability in the learning. To conclude, loss and mel-loss follow a similar progress. as shown in Figures 15c and 15d. The losses decrease dramatically during the first 500 epochs. Subsequently, they continue decreasing slower and with a lower improvement, especially in the case of mel-spectrogram reconstruction losses. This means that penalty for the predictions made by the model is decreasing, showing a positive progress during its training.

Training is done until epoch 1,000. The progress on the training of this model is

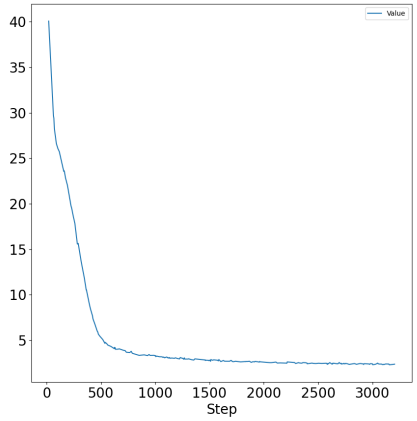
[4] HiFiGan's fine tuning was performed by Victor García, a fellow student in AhoLab.



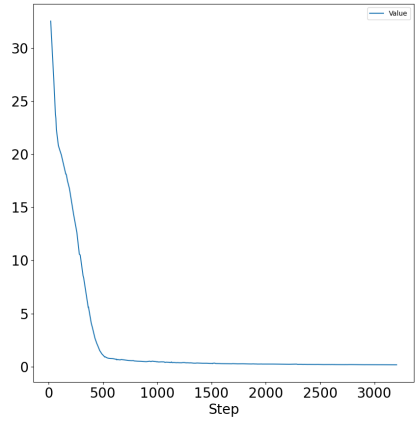
(a) Learning rate



(b) Mean gradient



(c) Loss



(d) Mel-loss

Figure 15: Training progress at 200 epochs

equally monitored by looking at the learning rate, mean gradient, loss and mel-loss, which continue to follow similar trends after the 3,200 steps. The learning rate continues to decrease progressively in a nonlinear scale. The mean gradient becomes more stable and epoch 2,000 onward the weight adjustments are minor. Similarly, the overall and mel-spectrogram reconstruction losses continue to decrease but in a minimal way.

After selecting the latest checkpoint, I synthesized the 40 test audios with WaveGlow and with HiFiGan. I ran the NISQA TTS evaluation on these two set of audios and collected the mean and standard deviation values of each. The results are shown in Table 6. The evaluation scale, as indicated in 3.3.4, ranges from 0 to 5, where 0 stands for the

lowest quality and 5 for the highest quality. As we can see, HiFiGan outperforms WaveGlow by only 0.01. The mean result for FastPitch with both vocoders is rather similar.

Model	**Mean**	**Standard deviation**
WaveGlow	4.32	0.34
HiFiGan	4.33	0.33

Table 6: Statistical results of NISQA evaluation of WaveGlow and HiFiGan

4.2 Experiment 2: FastPitch, Tacotron2 and FastSpeech2

The objective of the second experiment is to compare and evaluate the quality of three different Spanish TTS systems: FastPitch, FastSpeech2 and Tacotron2. These models are trained under the same conditions (i.e. audio features, linguistic features, data training and validation sets, vocoder) and evaluated using the same system (NISQA). The results of this evaluations are statistically analyzed as well. Additionally, this experiment tests the quality of synthesizing the same model with HiFiGan as a vocoder.

4.2.1 Experimental setup

This experiment required coordination with other members of AhoLab. First, we ensured that we use the same utterances for the train, validation, and test sets. In order to do this with FastPitch, I was provided with the list of utterances used by the other models, and I wrote a script to create the same sets with the selected sentences. I followed the same procedure to create the sets with the pitch audio files. All the models were then trained using the sample sample frequency of 22050 Hz and SAMPA phonemes as linguistic features. To conclude, the same 40 test audios were synthesized with the same HiFiGan vocoder.

4.2.2 Evaluation and results

The evaluation of these three models was once again performed using NISQA. The 40 test audios were synthesized with the three models and the NISQA TTS evaluation was ran on each. In Table 7 we can observe the mean and standard deviation results. For the 40 audio tested, the mean opinion scores of FastPitch are 4.33 out of a maximum of 5 score points, which is the highest evaluation among all the acoustic models. Tacotron2 follows closely

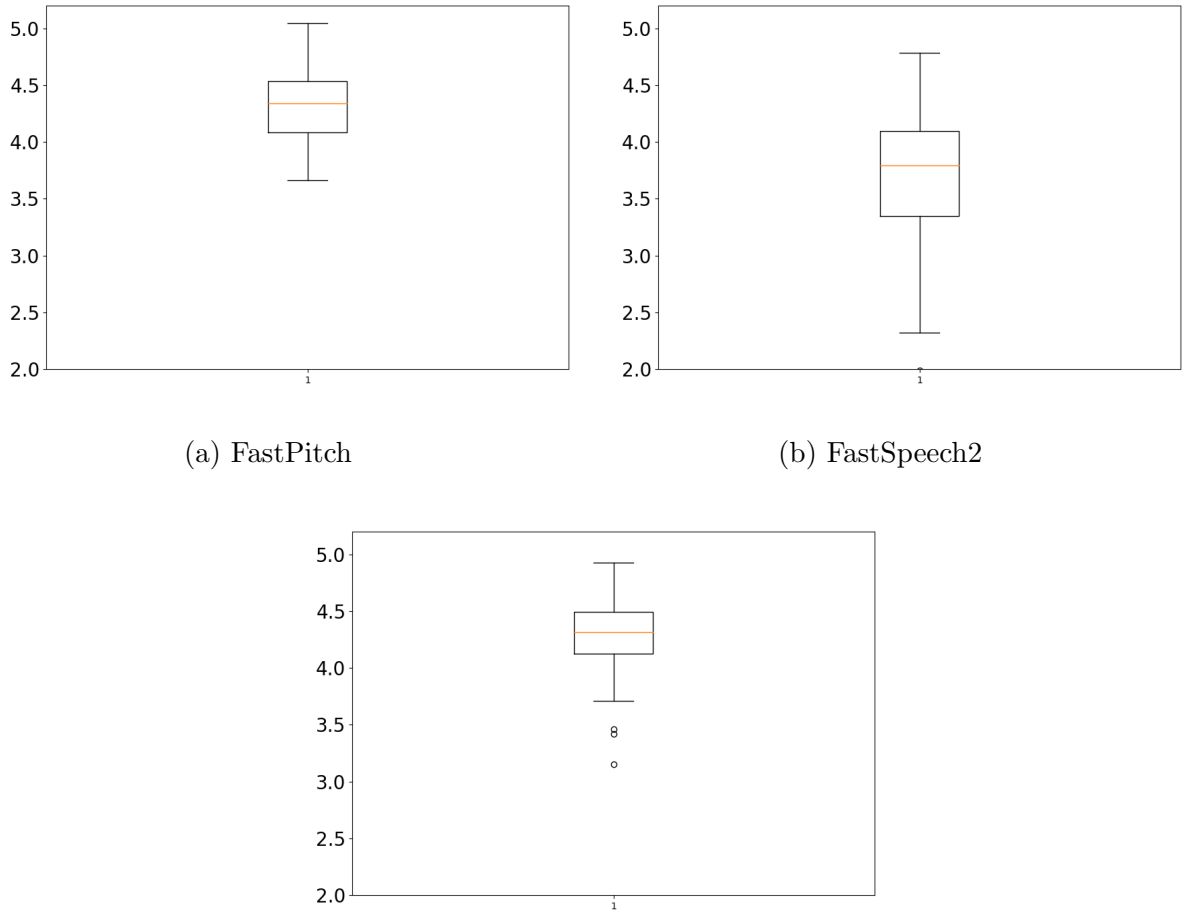
with a slightly lower evaluation of 4.28. FastSpeech2 is evaluated with the lowest quality score, 3.67, which results in a 13.2% lower scored evaluation compared to FastPitch.

Model	**Mean**	**Standard deviation**
FastPitch	4.33	0.33
FastSpeech2	3.67	0.63
Tacotron2	4.28	0.38

Table 7: Statistical results of NISQA evaluation of the three acoustic models

The standard deviation scores reflect the distribution of the data, that is, how dispersed is the data (evaluation scores, in this case), in relation to the mean. If the standard deviation is low, it means that the evaluation scores of a model are clustered around the mean score, which makes them to be more likely closer to the mean. If standard deviation is high, it indicates that the evaluation scores are more spread out. In this case, a lower standard deviation may indicate that the model is more stable, as evaluation scores are more consistent. For instance, in the case of FastPitch, the data is likely to be 0.33 evaluation scores more or less away from the mean, 4.33. It means the MOS of the evaluations are likely to be between 4 and 4.66. This is similar for Tacotron2, where the data will deviate from the mean in 0.38, i.e. it is more likely that the scores will be found between 3.9 and 4.66. FastSpeech2 shows a higher dispersion of the MOS evaluations with a standard deviation of 0.63, meaning that the scores could range from 3.04 and 4.3.

The results of the NISQA evaluation of FastPitch, FastSpeech2 and Tacotron2 are plotted in Figure 16. In these plots, we can analyze the probability distributions of the model evaluations aforementioned, that is, the probabilities of a model being evaluated with higher or lower scores. In the case of FastPitch (see Figure 16a, we can observe that the mean quality MOS resides at 4.33. In addition, the probability of the evaluations being higher than the mean is lower than the probability of lower evaluation scores. In the case of this model, most of the data is fairly distributed around a 3.6 and 5.0 evaluation, where 25% of evaluations are more likely to be between 4.33 and almost 4.6, and the other 25% between 4.33 and 4.1 approximately. With regard to FastSpeech2, as seen in Figure 16b, the evaluation results are more likely to be found between the scores of 2.3 and 4.8. The highest part of the probability distribution of the evaluations relies under the value of 4. The highest interval shows a 25% of likelihood of the model being scored between the mean 3.67 and 4.2, and the lowest 25% of the evaluation predictions are more likely to be between 3.3 and 3.67. To conclude, Tacotron2 scores are likely to be found between 3.7



(a) FastPitch

(b) FastSpeech2

(c) Tacotron2

Figure 16: Results of NISQA MOS evaluation

and 4.9, despite a few outliers in the lower scoring side between 3 and 3.5. For this model, 50% of the evaluations are likely to be scored between 4.1 and 4.5, approximately.

4.3 Experiment 3: Pitch modifications

The last experiments aim to test the pitch transformation application of FastPitch. As mentioned in Section 2.2, this model allows to exert additional control over the synthesized utterances at inference by modifying the pitch contour, which enables us to control the output prosody. For instance, we can increase or decrease the fundamental frequency in a naturally sounding way, that preserves the perceived identity of the speaker, adjust the

energy or alter the rate of speech (i.e slower or faster pace). In the following subsection, the evaluations and results of the different alterations of our FastPitch model in Spanish will be shown.

4.3.1 Experimental setup

It has been mentioned in Section 2.2.3 that, thanks to FastPitch's explicit modeling of pitch from input data, the model allows for prosody transformations applied at inference. In this experimental section, I have synthesized three audios extracted from the test set using each of these prosody transformations. All the speech files have been synthesized with the FastPitch trained model's latest checkpoint and WaveGlow as vocoder, changing the transformation parameters exclusively. FastPitch allows the application of more than one transformation at the same time, but for the purpose of this study, I have selected only one transformation at each inference time. The three sentences analyzed are short, medium and long, and have been carefully selected in order to test the efficiency of this features in declarative sentences with different lengths in Spanish. The sentences and the applied prosody transformations with their values are as follows:

Test sentences:

- No lo entiendo.
- Me imagino que usted habrá explicado esto.
- También es cierto que durante mucho tiempo nuestro grupo ha sido de los primeros.

Transformations:

- pitch-transform-amplify: 2
- pitch-transform-invert
- pitch-transform-shift: +50Hz, -50Hz
- pitch-transform-flatten
- pace: 1.5 (times faster), 0.5 (times slower)

4.3.2 Evaluation and results

The prosody alterations performed are most likely not suitable for NISQA MOS evaluation. This is because NISQA has not been trained in this type of data with different prosodic features and transformations like the ones allowed by FastPitch. Hence, the evaluation of these audios has been performed as a self perceptual listening activity. This means the evaluation described in this section is subjective and qualitative. Nonetheless, all samples

have been made available for the public in a website for anyone to listen and evaluate [5]. After listening to the samples, the self qualitative evaluation of each transformation is the following:

Pitch amplification The pitch amplification parameter raises expressiveness of pitch by increasing the range of fundamental frequency (F0) of the speech signal at inference. This means that the speech signal has a higher and lower range of Hz. The transformation applied to the model trained in Spanish is intelligible and the synthesized examples sound more expressive, but in a lower note voice. The expressiveness seems to increase the longer the input utterance is. However, I believe the quality of the speech signal is affected negatively. The output presents some artifacts and the speech appears to have a background noise, making it sound less clear and natural.

Pitch inversion Pitch inversion consists in inverting the frequency around the mean value for a single utterance (Łańcucki, 2021). This means that if the sound of the target token has only mid to high frequencies, its inversion will only have mid to low frequencies after the transformation, and vice versa. The intonation of the synthesized speech is not adequate for a neutral declarative sentence in Spanish, as the downstepping of the pitch at the end of the utterance becomes a high pitch, making it sound as a question. Nevertheless, the results of applying this transformation itself are positive. The intonation is altered while maintaining the original quality.

Pitch shifting in Hz By shifting pitch in Hz, we increase or lower the pitch, resulting in deeper or high-pitched voices. To shift the pitch up or down by some Hz, we can just add or subtract as needed. The result of increasing the pitch in 50 Hz in our trained TTS model is positive and we obtain a good quality higher pitched voice. However, when lowering the pitch by 50 Hz, the output of the synthesized voice is not as good, particularly in the short and long sentences. The output speech appears to be echoed and tinny.

Pitch flattening Pitch flattening removes the sentence intonation of the speech, with no dynamic pitch variations. The output is a monotone voice. To flatten the pitch at inference, therefore, the parameter needs to be set at 0. The synthesized voice has fair quality, but does not sound natural because of the monotonous perceptual effect of flattening pitch, resulting in a robot-like unnatural voice. The transformation applied works as expected, and performs the intonation removal as other audio editing software such as Audacity, without altering original standards or adding any sound artifacts.

[5] https://annemnvz-blog.vercel.app/blog/fastpitch-samples
Note: it is recommend to open this url in Google Chrome as other browsers might not be supported.

Pace changes The change in pace results in a faster or slower speed of the speech. The pace at normal speed is 1, and this can be adjusted from 0.1 to 2. The values are multiplied by 1 resulting in a slow down or speed up speech. For instance, setting the pace at 1.5 increases the pace and make each token duration shorter, therefore the output speech will be two times faster. If the pace is set at x0.5, the output is 0.5 times slower, taking double time of the normal speed. The quality of the synthesized audios at these two paces is as expected. The audios are slower and faster accordingly without any additional noisiness, glitches or changes in frequency that might affect their quality.

4.4 Chapter summary

In this chapter, I have detailed three experiments, each targeting different areas of text-to-speech in Spanish. The first experiment introduces the initial training of the FastPitch acoustic model, and includes a comparison of the performance of the model with two different vocoders: WaveGlow and HiFiGan. The second experiment evaluates and compares the results of three acoustic models: FastPitch, FastSpeech2 and Tacotron2. Lastly, the third experiments of section explores prosody transformations of the trained model at inference, which are self-evaluated and made publicly available for evaluation. The next chapter combines a more detailed discussion of these results with critical reflection on shortcomings of the experiments, suggestions for further research, and the implications of the thesis for TTS researchers, especially in the context of Spanish.

5 Discussion, future work, and conclusion

In the previous chapter, I presented the results of the work accomplished for this master's thesis. In this section, we revise our work and how this could be continued in the future. In order to do so, I first discuss the findings of these three experiments described in Section 4, identifying the strengths and shortcomings of the approach, as well as the implications and contributions of this work in the field of text-to-speech in Spanish. In addition, a set of suggestions and potential areas for further research in this topic is included. Lastly, a brief summary and conclusion of this work follows.

5.1 Discussion

The results of the previous section have been divided per experiment for discussion in the current section, following the same subsection structure. Therefore, in following subsections, I further discuss the results of the first experiment, analyzing the training of the model and synthesis of test audios with WaveGlow and HiFiGan vocoders; as well as the second experiment, explaining in detail the main differences among the FastPitch, FastSpeech2 and Tacotron2 acoustic models. To conclude, I briefly discuss the results obtained from the third experiment, the prosody transformations at inference.

5.1.1 Experiment 1: WaveGlow and HifiGan

This first experiment represents the first steps to train FastPitch in Spanish. Once the aforementioned problems during the training were resolved, the training of the model was successful and the quality of the synthesized speech was adequate. As shown in Figure 15, the initial 200 epochs progress as expected: the learning rate increases during the first 1,000 steps, the mean gradient changes up to the point where changes start to be minimal, and the losses decrease dramatically during the first 500 steps. After the initial 500-1,000 steps, all these measurements stabilize and the changes are steady, less drastic. This is positive as it indicates that the model is settled and therefore more robust.

After the completion of the training of FastPitch in Spanish, the model was further used to synthesize the test sentences with two different vocoders: WaveGlow and HiFiGan. The evaluation of the output performed by NISQA has been laid out in Table 6. There we can observe that the mean MOS evaluation of the synthesized speech is very similar: WaveGlow's mean evaluation is 4.32, with a probability of distribution of evaluations between 3.98

and 4.66; HifiGan's mean is 4.33, with a probability distribution of evaluations between 4 and 4.66. This means both have a good output quality.

5.1.2 Experiment 2: FastPitch, Tacotron2 and FastSpeech2

The comparison of the three acoustic models can be analyzed in detail in Table 16. This table shows the mean and standard deviation. The mean of the evaluations of FastPitch is 4.33, slightly better than Tacotron2's evaluations, 4.28. FastSpeech2 is at the bottom of the rank with 3.67. Besides, the standard deviation of FastPitch relies between 4 and 4.66, the one of FastSpeech2 is between 3.04 and 4.3, and Tacotron2's is between 3.9 and 4.66. These result mean that the preliminary automatic prediction of NISQA for the best evaluation MOS scores is FastPitch, while the lowest scored model would be FastSpeech2.

I analyzed the results of the NISQA evaluation for the 40 test audios, and detected a few results with a evaluation of 1 and 2. After listening to these, I found that the audios have important glitches and unnatural robotic alike sounds. Nonetheless, some other audios have scores above 4 points and their quality is adequate. This randomized quality performance issue in FastSpeech2 would require further investigation.

5.1.3 Experiment 3: Pitch modifications

In section 4.3.2, it has been explained in detail the results of each prosody transformation applied at inference with FastPitch. A close perceptual study of the output suggests that the transformations effectively transform pitch as intended. The synthesized speech after applying pitch inversion, pitch flattening, pitch shifting in +50Hz and pace changes maintains the good quality of the model trained in Spanish. The results of pitch amplification and shifting in -50 Hz, however, alter the original quality, as the speech is synthesized in a less clear and natural manner.

5.2 Limitations of the work

After the implementation and evaluation of a Spanish TTS system based on FastPitch, followed by three different experiments, we have identified three main limitations to note after the conducted work. These include the training and checkpoint selection of the models, the evaluation system, and time constraints. Regarding the first limitation, the training of the acoustic models (FastPitch, FastSpeech2 and Tacotron2) was performed independently for each system. This is principally related to time constraints for the

coordination of the training of these models. As a result, the checkpoint selected among the three trained output models was not the same. In other words, the acoustic models used during inference are inconsistent in their trained time, epochs and losses. Besides, training parameters and other thresholds such as steps, batch size, learning rate, loss scales, or energy conditioning, were not compared and aligned among the models.

Regarding the second limitation, human subjective assessment is the most adequate option for the evaluation of synthesized speech. The original evaluation for this project was designed as a paired comparison MOS task. The aim was to compare the synthesized speech of the three models and the original speech audios. These would have been randomized and shown to evaluators in sets of two pairs, and the task of these evaluators would be to choose the audio with the most natural speech. Ultimately, finding an open source evaluation system that could present the pair of audios in a controlled randomized way was finally a challenging task. Due to time limitations, we decided to conduct the evaluation using NISQA.

The reliability of the NISQA model's MOS evaluation model is not guaranteed, however. NISQA is widely used for this type of evaluations in the speech community and, in the case of this project, it has proven its efficiency in detecting problematic sentences. As previously mentioned, the low rating of the speech synthesized with FastSpeech2 corresponded to faulty and unclear audios. Even so, NISQA has been trained in English and German audio data, and our data is in Spanish. This means that the model would not be able to tell if a word or a part of the speech has not been pronounced correctly, or if the prosody of the sentence is incorrect for a given context.

5.3 Future work

Based on the previous analysis and results obtained for the different experiments, there are two primary types of steps that can be remarked in order to continue with this work. First, it is recommended that, in future experiments that compare two or more acoustic models, the training parameters are adjusted as close to each other as possible: e.g. steps, batch size, learning rate, loss scales. Second, the selected checkpoint of the three acoustic models for synthesis needs to be carefully chosen in order to ensure the fairness of their evaluation. In order to do this, it is recommended to make sure the number of epochs and the progress of each model is similar, including time, losses or other statistical measurements.

There are more options to select a good model. For instance, one possibility is to consider testing time distortion measures such as the losses of the training, which helps

estimate how well the model's predictions are. In addition, we could also perform a word error rate (WER) or character/phoneme error rate (CER/PER) test of the synthesized output. This test is good to monitor intelligibility of the synthesized speech at the word level. However, the output of this test could be good even with poor quality or robot-like voice, so it does not completely correspond to naturalness. This makes choosing the best model based on the validation values a difficult task, as it requires manual opinion based evaluation. A last option would be to combine the WER or CER/PER tests with NISQA.

The second recommendation for completing this project is to evaluate audios with a paired audio assessment test. The aim would be for this test to be performed as a subjective human evaluation. Ideally, in this test, we would present two audios at each time in a random way for their evaluation, but every audio from each model and the original speech are presented the same amount of times. The minimum number of evaluations can vary, but it is suggested that more than 10 native Spanish speakers perform the evaluation.

To conclude, the research and the experiments has been limited to a set of phonemes, acoustic settings, training parameters and network configurations. The training of FastPitch with the same data, however, allows for a wide range of configurations. Future research would also benefit from modifying hyper-parameters such as phoneme type, text cleaners especialized in Spanish, energy conditioning, number of layers, or loss and learning rate, to determine which model configurations result in a better overall performance.

5.4 Contributions of this thesis

This thesis aims to train and evaluate a TTS model in Spanish based on FastPitch. In order to do this, we have provided a set of tools for the cleaning and processing of a raw dataset in Spanish, which can be used in future raw dataset cleanup in Spanish. These tools have been used to provide a preprocessed Spanish dataset that is now available for use with any other TTS framework for speech synthesis in Spanish. As one of the main tasks, this project also provides a well trained model in Spanish based on FastPitch, a framework that allows for post-training adjustments of prosody such as pitch flattening, pitch inversion, pitch amplification, pitch shift and pace changing. FastPitch does not include Spanish as part their recipes or pre-trained models, nor there is feedback on this topic online, which makes this an innovative research. Alongside, the comparison on the evaluation among FastPitch, FastSpeech2 and Tacotron2 presented in this project helps to shed light on the performance of latest state-of-art models in the field of Spanish TTS.

To conclude, this project has provided synthesized research on state-of-art models in

this field. Particularly in the research of TTS in Spanish, as it is not ad abundant as in English, and open source resources are scarce. In this work, we have included a set of resources in Spanish, which include: Spanish transcribed audio databases, tools and methods for data processing, phonetic transcriptions and dictionary creations, and models or frameworks that can be used for the training of a TTS system in Spanish.

5.5 Summary and conclusion

In this thesis, I presented a text-to-speech model in Spanish trained in FastPitch. For the completion of this work, the dataset used for the training, composed by audio and text files in Spanish, has been first preprocessed and prepared. The linguistic module of FastPitch required to be adjusted to the Spanish language and, in order to do so, a dictionary and an online transcription module in Spanish were implemented. The training was successfully conducted and this enabled three different experiments: a comparison of the performance of creating inferences with the vocoders WaveGlow and HiFiGan; a comparison of the performance of FastPitch, FastSpeech2 and Tacotron2 acoustic models; a test of the prosody transformations the FastPitch framework allows, including pitch flattening, pitch inversion, pitch amplification, pitch shift and pace changing.

The results of the experiments suggest that FastPitch performs best as acoustic model, particularly when combined with the HiFiGan vocoder. The performance and quality of this Spanish TTS system is high in general. This allows for a satisfactory for immediate adoption in a downstream application in the future. Furthermore, this work has provided a list of learnings, resources, methods and tools, carefully designed for the future development of a TTS in spanish and for the replication of the current study.

References

Afshine Amidi and Shervine Amidi. Cs 230 - recurrent neural networks cheatsheet. URL https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#overview.

Sercan Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, Shubho Sengupta, and Mohammad Shoeybi. Deep voice: Real-time neural text-to-speech. *34th International Conference on Machine Learning, ICML 2017*, 1:264–273, 2 2017. doi: 10.48550/arxiv.1702.07825. URL https://arxiv.org/abs/1702.07825v2.

Kurniawati Azizah, Mirna Adriani, and Wisnu Jatmiko. Hierarchical transfer learning for multilingual, multi-speaker, and style transfer dnn-based tts on low-resource languages. *IEEE Access*, 8:179798–179812, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3027619.

Mary E. Beckman, Manuel DÍAz-Campos, Julia Tevis, and Mcgory Terrell A. Morgan. Intonation across spanish, in the tones and break indices framework. *Probus*, 14:9–36, 6 2002. ISSN 16134079. doi: 10.1515/PRBS.2002.008. URL https://www.researchgate.net/publication/254325237_Intonation_Across_Spanish_in_the_Tones_and_Break_Indices_Framework.

UC Berkley. Parametric speech synthesis. URL https://ptolemy.berkeley.edu/eecs20/speech/voder.html.

Paola Bonifacci, Elisa Colombini, Michele Marzocchi, Valentina Tobia, and Lorenzo Desideri. Text-to-speech applications to reduce mind wandering in students with dyslexia. *Journal of Computer Assisted Learning*, 38:440–454, 4 2022. ISSN 1365-2729. doi: 10.1111/JCAL.12624. URL https://onlinelibrary.wiley.com/doi/full/10.1111/jcal.12624https://onlinelibrary.wiley.com/doi/abs/10.1111/jcal.12624https://onlinelibrary.wiley.com/doi/10.1111/jcal.12624.

Britannica. phoneme, linguistics. URL https://www.britannica.com/topic/phoneme.

Jason Brownlee. Understand the impact of learning rate on neural network performance, 1 2019. URL https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

Carnegie-Mellon-University. cmusphinx/g2p-seq2seq: G2p with tensorflow. URL https://github.com/cmusphinx/g2p-seq2seq.

Yuan Jui Chen, Tao Tu, Cheng Chieh Yeh, and Hung Yi Lee. End-to-end text-to-speech for low-resource languages by cross-lingual transfer learning. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2019-September:2075–2079, 4 2019. ISSN 19909772. doi: 10.48550/arxiv.1904.06508. URL https://arxiv.org/abs/1904.06508v2.

CreativeCommons. Cc0 - creative commons. URL https://creativecommons.org/share-your-work/public-domain/cc0/.

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *7th International Conference on Learning Representations, ICLR 2019*, 2 2018. doi: 10.48550/arxiv.1802.04208. URL https://arxiv.org/abs/1802.04208v3.

Niklas Donges. Gradient descent: A quick, simple introduction — built in, 7 2021. URL https://builtin.com/data-science/gradient-descent.

Thierry Dutoit and Thierry Dutoit. High-quality text-to-speech synthesis : an overview. *JOURNAL OF ELECTRICAL ELECTRONICS ENGINEERING*, 1997. URL http://130.203.136.95/viewdoc/summary?doi=10.1.1.495.9289.

Shirly Edward. Text-to-speech device for visually impaired people. *International Journal of Pure and Applied Mathematics*, 119, 7 2018.

Tikhonov Eugene and Manfred sneps sneppe. Introduction to signal processing: sampled signals. 05 2019.

Google. Descending into ml: Training and loss — machine learning — google developers. URL https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss.

Alexander Gutkin, Linne Ha, Martin Jansche, Knot Pipatsrisawat, and Richard Sproat. Tts for low resource languages: A bangla synthesizer – google research, 5 2016. URL https://research.google/pubs/pub45300/.

Tomoki Hayashi, Ryuichi Yamamoto, Katsuki Inoue, Takenori Yoshimura, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Yu Zhang, and Xu Tan. Espnet-tts: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2020-May:7654–7658, 10 2019. ISSN 15206149. doi: 10.48550/arxiv.1910.10909. URL https://arxiv.org/abs/1910.10909v2.

Rüdiger Hoffmann and Dieter Mehnert. Early experiments on prosody in synthetic speech. 2010.

Hee Hwang and Kristine M. Yu. Word-based neural prosody modeling with tobi. *Proceedings of the International Conference on Speech Prosody*, 2020-May:1019–1023, 2020. ISSN 23332042. doi: 10.21437/SPEECHPROSODY.2020-208.

Sergios Karagiannakos. Speech synthesis: A review of the best text to speech architectures with deep learning — ai summer, 5 2021. URL https://theaisummer.com/text-to-speech/.

Abdulamir A. Karim and Suha Mohammed Saleh. Text to speech using mel-spectrogram with deep learning algorithms. *Periodicals of Engineering and Natural Sciences (PEN)*, 10:380, 6 2022. ISSN 23034521. doi: 10.21533/PEN.V10I3.3113. URL http://pen.ius.edu.ba/index.php/pen/article/view/3113.

Matti Karjalainen. Review of speech synthesis technology. *Helsinki University of Technology, Department of Electrical and Communications Engineering*, 1999.

Simon King. An introduction to statistical parametric speech synthesis. *Sadhana*, 36:837–852, 10 2011. ISSN 0256-2499. doi: 10.1007/S12046-011-0048-Y. URL https://www.research.ed.ac.uk/en/publications/an-introduction-to-statistical-parametric-speech-synthesis.

Dennis H. Klatt. Review of text-to-speech conversion for english. *The Journal of the Acoustical Society of America*, 82:737, 6 1998. ISSN 0001-4966. doi: 10.1121/1.395275. URL https://asa.scitation.org/doi/abs/10.1121/1.395275.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 2020-December, 10 2020. ISSN 10495258. doi: 10.48550/arxiv.2010.05646. URL https://arxiv.org/abs/2010.05646v2.

Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in Neural Information Processing Systems*, 32, 10 2019. ISSN 10495258. doi: 10.48550/arxiv.1910.06711. URL https://arxiv.org/abs/1910.06711v3.

Adrian Łańcucki. Fastpitch: Parallel text-to-speech with pitch prediction. volume 2021-June, 2021. doi: 10.1109/ICASSP39728.2021.9413889.

Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 6706–6713, 9 2018. ISSN 2159-5399. doi: 10.48550/arxiv.1809.08895. URL https://arxiv.org/abs/1809.08895v3.

LumenVox. Tts1 spanish text normalization — lumenvox knowledgebase. URL https://www.lumenvox.com/knowledgebase/index.php?/article/AA-01896/0/TTS1-Spanish-Text-Normalization.html.

Manh Luong and Viet Anh Tran. Flowvocoder: A small footprint neural vocoder based normalizing flow for speech synthesis. 9 2021. doi: 10.48550/arxiv.2109.13675. URL https://arxiv.org/abs/2109.13675v2.

Suhas R Mache, Manasi R Baheti, and C Namrata Mahender. Review on text-to-speech synthesizer. 4:54–59, 2015. doi: 10.17148/IJARCCE.2015.4812.

Harsh Maheshwari. Basic text to speech, explained. learn the cool technology behind alexa... — by harsh maheshwari — towards data science, 5 2021. URL https://towardsdatascience.com/text-to-speech-explained-from-basic-498119aa38b5.

MarvMind. Machine learning - what is the difference between steps and epochs in tensorflow? - stack overflow. URL https://stackoverflow.com/questions/38340311/what-is-the-difference-between-steps-and-epochs-in-tensorflow.

Gabriel Mittag, Babak Naderi, Assmaa Chehadi, and Sebastian Möller. Nisqa: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced

datasets. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 4:2818–2822, 4 2021. doi: 10.21437/Interspeech.2021-299. URL http://arxiv.org/abs/2104.09494http://dx.doi.org/10.21437/Interspeech.2021-299.

Tomáš Nekvinda and Ondrej Dušek. One model, many languages: Meta-learning for multilingual text-to-speech. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2020-October:2972–2976, 8 2020. ISSN 19909772. doi: 10.48550/arxiv.2008.00768. URL https://arxiv.org/abs/2008.00768v1.

Yishuang Ning, Sheng He, Zhiyong Wu, Chunxiao Xing, and Liang Jie Zhang. Review of deep learning based speech synthesis. *Applied Sciences (Switzerland)*, 9, 10 2019. ISSN 20763417. doi: 10.3390/APP9194050.

Michael H. O'Malley. Text-to-speech conversion technology. *Readings in Human–Computer Interaction*, pages 539–545, 1 1995. doi: 10.1016/B978-0-08-051574-8.50056-X.

Paperswithcode. Tacotron2 explained — papers with code, a. URL https://paperswithcode.com/method/tacotron-2.

Paperswithcode. Knowledge distillation — papers with code, b. URL https://paperswithcode.com/task/knowledge-distillation.

Paperswithcode. Fastpitch explained — papers with code, c. URL https://paperswithcode.com/method/fastpitch.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2019-May:3617–3621, 10 2018. ISSN 15206149. doi: 10.48550/arxiv.1811.00002. URL https://arxiv.org/abs/1811.00002v1.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie Yan Liu. Fastspeech: Fast, robust and controllable text to speech. volume 32, 2019.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. 6 2020. doi: 10.48550/arxiv.2006.04558. URL https://arxiv.org/abs/2006.04558v6.

Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 12 2018. URL https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari. Statistical parametric speech synthesis incorporating generative adversarial networks. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26:84–96, 9 2017. ISSN 23299290. doi: 10.48550/arxiv.1709.08041. URL https://arxiv.org/abs/1709.08041v1.

Diemo Schwarz. Concatenative sound synthesis: The early years. *Journal of New Music Research*, 35:3–22, 3 2006. ISSN 09298215. doi: 10.1080/09298210600696857.

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. volume 2018-April, 2018. doi: 10.1109/ICASSP.2018.8461368.

Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018-April:4784–4788, 10 2017. doi: 10.1109/ICASSP.2018.8461829. URL http://arxiv.org/abs/1710.08969http://dx.doi.org/10.1109/ICASSP.2018.8461829.

Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. A survey on neural speech synthesis. 6 2021. doi: 10.48550/arxiv.2106.15561. URL https://arxiv.org/abs/2106.15561v3.

T. H. Tarnóczy. The speaking machine of wolfgang von kempelen. *The Journal of the Acoustical Society of America*, 21:461, 6 2005. ISSN 0001-4966. doi: 10.1121/1.1917078. URL https://asa.scitation.org/doi/abs/10.1121/1.1917078.

TechTarget. What is vocoder? - definition from whatis.com, 2005. URL https://www.techtarget.com/whatis/definition/vocoder.

Kohsheen Tiku, Jayshree Maloo, Aishwarya Ramesh, and R. Indra. Real-time conversion of sign language to text and speech. *Proceedings of the 2nd International Conference on Inventive Research in Computing Applications, ICIRCA 2020*, pages 346–351, 7 2020. doi: 10.1109/ICIRCA48905.2020.9182877.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 9 2016. doi: 10.48550/arxiv.1609.03499. URL https://arxiv.org/abs/1609.03499v2.

Sarah E. Wallace, Karen Hux, Kelly Knollman-Porter, Jessica A. Brown, Elizabeth Parisi, and Rebecca Cain. Reading behaviors and text-to-speech technology perceptions of people with aphasia. *Assistive technology : the official journal of RESNA*, 2021. ISSN 1949-3614. doi: 10.1080/10400435.2021.1904306. URL https://pubmed.ncbi.nlm.nih.gov/33724912/.

Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, and Quoc Le. Tacotron: Towards end-to-end speech synthesis. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2017-August:4006–4010, 3 2017. ISSN 19909772. doi: 10.48550/arxiv.1703.10135. URL https://arxiv.org/abs/1703.10135v2.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. Espnet: End-to-end speech processing toolkit. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018-September:2207–2211, 3 2018. ISSN 19909772. doi: 10.48550/arxiv.1804.00015. URL https://arxiv.org/abs/1804.00015v1.

Wave. wave — read and write wav files — python 3.10.5 documentation. URL https://docs.python.org/3/library/wave.html.

Ryuichi Yamamoto, Eunwoo Song, and Jae Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2020-May:6199–6203, 10 2019. ISSN 15206149. doi: 10.48550/arxiv.1910.11480. URL https://arxiv.org/abs/1910.11480v2.

Jaeseong You, Dalhyun Kim, Gyuhyeon Nam, Geumbyeol Hwang, and Gyeongsu Chae. Gan vocoder: Multi-resolution discriminator is all you need. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*,

5:3236–3240, 3 2021. ISSN 19909772. doi: 10.48550/arxiv.2103.05236. URL https://arxiv.org/abs/2103.05236v2.

Heiga Zen. (pdf) acoustic modeling for speech synthesis: from hmm to rnn, 12 2015. URL https://www.researchgate.net/publication/290441279_Acoustic_Modeling_for_Speech_Synthesis_from_HMM_to_RNN.

Heiga Zen, Keiichi Tokuda, and Alan W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51:1039–1064, 11 2009. ISSN 0167-6393. doi: 10.1016/J.SPECOM.2009.04.004.

Heiga Zen, Andrew Senior, and Mike Schuster. Statistical parametric speech synthesis using deep neural networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 7962–7966, 10 2013. ISSN 15206149. doi: 10.1109/ICASSP.2013.6639215.

Yuxiang Zou, Shichao Liu, Xiang Yin, Haopeng Lin, Chunfeng Wang, Haoyu Zhang, and Zejun Ma. Fine-grained prosody modeling in neural speech synthesis using tobi representation. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 5:3346–3350, 2021. ISSN 19909772. doi: 10.21437/INTERSPEECH.2021-883.