

TESI DI LAUREA MAGISTRALE

**Voice Cloning: Increasing Expressivity of Italian Text-to-Speech
with Phonemization**

Relatore:

Prof. Paolo Torroni

Correlatore:

Dott. Silverio Di Candilo

Candidato:

Martino M. L. Pulici

Matricola 985218

Sommario

La sintesi vocale partendo da un testo scritto è un concetto affascinante nell'ambito dell'elaborazione del linguaggio naturale. Per quanto sia simile alla sua controparte (riconoscimento vocale automatico), il passaggio da testo a voce risulta essere un processo più delicato. Rispetto al testo scritto, il linguaggio parlato trasmette uno spettro molto più ampio di informazioni, quali frequenza, volume, velocità, e pause. Nonostante a oggi la sintesi vocale abbia raggiunto un livello di intellegibilità molto alto, continua a presentare dei limiti, principalmente per quanto riguarda la mancanza di credibilità ed espressività. Ciò vale in particolar modo per le lingue diverse dall'inglese, in cui la raccolta di dati è già di per sé una sfida. Il presente lavoro mira ad affrontare il passaggio da testo a voce in italiano, sfruttando fonemizzazione e modelli neurali così da ottenere una migliore espressività nella clonazione vocale. Dopo aver addestrato i modelli, gli audio ottenuti vengono confrontati, attraverso la somministrazione di un questionario, con quelli forniti da altri servizi di sintesi vocale. La voce da noi sintetizzata si posiziona ai livelli più alti in tutte le caratteristiche testate: l'espressività e il punteggio globale medio presentano gli scarti maggiori rispetto alle altre voci. Sviluppi futuri potrebbero prevedere il miglioramento di alcuni degli aspetti meno premiati dai questionari, in particolare la naturalezza delle pause, così come la possibilità di clonare nuove voci utilizzando una quantità ridotta di dati.

Abstract

The synthesis of spoken words from a written text is a fascinating concept in natural language processing. Despite being similar to its counterpart (automatic speech recognition), text-to-speech tends to be a more subtle task. Compared to written text, spoken language conveys a much broader spectrum of information, such as pitch, volume, speed, and pauses. Even though synthetic speech has by now reached a very high level of intelligibility, it has some shortcomings, mainly the lack of believability and expressivity. This holds especially in languages other than English, where data gathering is a hard challenge by itself. The present work aims at tackling the text-to-speech task in Italian, leveraging phonemization and neural models to achieve better expressivity in voice cloning. After training the models, the obtained audios are compared, through the administration of a questionnaire, with the ones provided by other available synthetic speech services. The voice we synthesized leads the rankings in all the tested characteristics: expressivity and global mean opinion score show the widest difference compared to the other voices. Future developments might include working on some of the less successful questionnaire scores, particularly pause naturalness, as well as investigating the possibility of cloning new voices using a smaller amount of data.

Contents

List of figures	ix
List of tables	xi
Foreword	xiii
I BACKGROUND	1
1 Text-to-speech	3
1.1 The text-to-speech problem	3
1.1.1 Speech and writing	3
1.1.2 Reading aloud	5
1.1.3 Key problems in text-to-speech	6
1.2 Classical models	7
1.3 Neural models	8
1.3.1 Text analysis	9
1.3.2 Acoustic models	11
1.3.3 Vocoder	12
2 Phonemes	15
2.1 Phonemes and the IPA Alphabet	15
2.1.1 Consonants	15
2.1.2 Vowels	17
2.1.3 Other symbols	17
2.2 Italian phonemes	17

2.2.1	Consonants	17
2.2.2	Vowels	18
2.2.3	Other symbols	19
II	EXPERIMENTAL SETUP	21
3	Datasets	23
3.1	Phonemization	23
3.1.1	Words gathering	23
3.1.2	Phonemization	24
3.1.3	Dataset analysis	25
3.2	Text-to-speech	28
3.2.1	Preprocessing	28
3.2.2	Dataset analysis	29
3.3	Human evaluation	32
4	Architectures	33
4.1	Phonemization	33
4.1.1	Transformer-based networks	33
4.1.2	Attention	34
4.1.3	DeepPhonemizer	34
4.2	Text-to-speech	35
4.2.1	FastPitch	35
4.2.2	HiFi-GAN	37
III	RESULTS	41
5	Training	43
5.1	Phonemization	43
5.2	Text-to-speech	43
5.2.1	FastPitch main training	45
5.2.2	FastPitch fine-tuning	45
5.2.3	HiFi-GAN	48

6 Evaluation	49
6.1 Phonemization	49
6.2 Human evaluation	52
7 Conclusion	55
7.1 Phonemization	55
7.2 Human evaluation	56
7.3 Future work	57
A Interface	59
B Pie charts	63
C Items distributions	67
Bibliography	75

List of figures

1.1	Basic model of reading aloud	5
2.1	The International Phonetic Alphabet	16
3.1	Graphemic dataset length distributions	26
3.2	Phonemic dataset length distributions	27
3.3	Main dataset duration distributions	30
3.4	Fine-tuning dataset duration distributions	31
4.1	The FastPitch architecture	36
4.2	The HiFi-GAN generator	38
4.3	The HiFi-GAN discriminator	39
5.1	DeepPhonemizer loss	44
5.2	FastPitch main training loss	46
5.3	FastPitch fine-tuning loss	47
5.4	HiFi-GAN loss	48
6.1	CER validation values	50
6.2	Test set CER distribution	51
6.3	Questionnaire results	53
6.4	Questionnaire MOS distributions	54
A.1	The questionnaire introduction	60
A.2	The questionnaire interface	61
B.1	Partecipants age	64

B.2	Partecipants biological sex	64
B.3	Partecipants educational level	65
B.4	Partecipants native language	65
C.1	Expressivity distributions	68
C.2	Facilitation to focus on the content distributions	69
C.3	Facilitation to understand the text distributions	70
C.4	Reading naturalness distributions	71
C.5	Pause naturalness distributions	72
C.6	Presence of personality and/or emotion distributions	73

List of tables

2.1	Italian consonant phonemes	18
2.2	Italian vowel phonemes	19
3.1	Graphemic dataset length statistics	26
3.2	Phonemic dataset length statistics	27
3.3	Main dataset duration statistics	30
3.4	Fine-tuning dataset duration statistics	31
7.1	CER values	56
7.2	Questionnaire results	56

Foreword

The ability to synthesize spoken words from a text is a fascinating topic. Despite being seemingly similar to its reverse task (automatic speech recognition), text-to-speech is much more nuanced. Speaking is the way by which language first developed, with writing coming much later as an imperfect method to record what has been uttered. While speaking, a person conveys a huge amount of information that can not be easily embedded in our writing systems. Pitch, volume, speed, pauses: all these factors contribute to the meaning conveyance and they are almost completely lost when an oral text is transcribed. Especially for narration and poetry, the spoken word has always held an appeal unrivalled by the printed page: books can surely have their charm, but a good actor reciting a Shakespeare sonnet is a wholly different experience.

As of today, especially for the Italian language, text-to-speech models are still not as believable as a human reader. Sure, they can now produce all spoken sounds correctly, but they lack personality and expressivity. While this might not be a problem for some very cold and technical contexts, when reading a narrative passage current solutions (especially for Italian) show all their shortcomings. This is a pity, as a natural-sounding synthetic voice could be an excellent resource, in particular for the visually impaired, for whom reading a book now requires the time and patience of a close person or resorting to an audiobook. The ability to clone the voice of a given person might also have other applications, such as aiding journalists to provide an audio version of an article using their own voice or helping mute people to communicate more satisfactorily.

The current thesis aims to employ deep neural networks to convert Italian

written texts to audio, with special attention to voice expressivity. A single speaker is chosen for the task to exemplify the voice cloning capabilities of the pipeline. To achieve better results, a phonemizer will be created as well, focusing especially on the positioning of stresses, the hardest part of the otherwise phonetic Italian reading conventions.

Part I

BACKGROUND

Chapter 1

Text-to-speech

1.1 The text-to-speech problem

Text-to-speech (TTS) [1] is a process that creates audio representing speech, starting from written text input. If both written and spoken languages are considered signals, then TTS is the process of taking one type of signal (written) and converting it into another type (spoken). There are various issues to be discussed regarding speech synthesis, such as “does the written signal contain enough information to generate the spoken one?”, “what are the relationships between author, reader, and listener?”, and most importantly “what do listener *expect* from a synthetic voice?”.

1.1.1 Speech and writing

The first thing to be noted about the relationship between speech and writing is that human language first developed in its spoken form. This is important since written texts are (in most languages¹) a representation of spoken sounds: in other words, written text is a form of encoding of the auditory signal. In turn, spoken communication is a form of encoding meanings.

¹Even though some writing systems (e.g., Chinese characters) do not have a direct connection to spoken texts, the focus of this work is on Italian TTS, therefore this simplification holds. Anyway, it should be noted that despite lacking a *direct* phonetical encoding even logograms are readable aloud: they are of course less direct phonetical encodings, but phonetical encodings nonetheless.

Therefore, when writing something, meaning is first encoded as spoken words (which are not necessarily pronounced aloud), which are then encoded as a written text. This written text is then decoded to an auditive signal by the reader, and this signal is finally decoded to its meaning by the listener.

Keeping track of this double encoding and decoding process is crucial since audio produced by a TTS system must not only convey a sufficiently good representation of the written text but also of its meaning. This means that the single sounds themselves, although necessary, are not sufficient to create a robust synthetic speech: convincing prosody is just as important as correct graphemes-to-sounds decoding.

Here arises one of the main difficulties of TTS: the encoding of spoken to written language is somewhat impoverished, as prosodic aspects are almost completely ignored. One of the hypotheses on why prosody is not encoded in usual written transcriptions is that being it a continuous aspect it would be hard to express it with discrete symbols (a “happy” intonation is not an on/off switch, but rather a spectrum of intensities). Another explanation could be that, while important for emotive (and therefore literary) texts, prosody is not relevant for legal and commercial documents, which are the uses for which written language has been initially developed [2].

There are however some ways to convey limited prosodic information. The first one is of course punctuation, used to give structure to what would otherwise be a list of juxtaposed words. It is also possible to emphasize some words or phrases by using *italicized*, **bold**, or underlined text. Changing *formatting*, *font faces*, and *sizes* can have similar usages as well. What is completely lacking from basically all conventional written texts is affective prosody².

²This may be changing in the recent years with the widespread usage of electronic means of communication: in instant messaging, emojis, stickers, and GIFs have become common and are expressing a broader spectrum of emotive nuances than plain text. These solutions, in addition to being specific to their media, are almost impossible to express orally: albeit very interesting on the linguistic level, it will thus be necessary to ignore them in the current work.

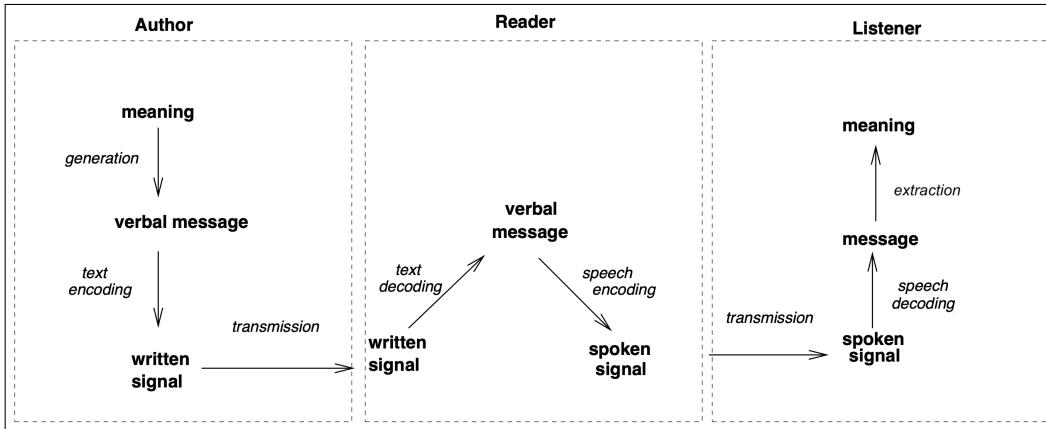


Figure 1.1: Basic model of reading aloud, which shows in the information flow between the author, the reader, and the listener.

Source: adapted from [1].

1.1.2 Reading aloud

When reading aloud, as opposed to silently, the process of encoding and decoding becomes more complex, especially because of the prosodic impoverishment just discussed. In this situation, there are three actors: the author, the reader, and the listener. The author and the listeners behave intuitively: the author generates the message and (double) encodes it to a written signal, while the listener decodes the spoken signals and extracts the meaning from the message.

One could then expect that the reader merely decodes the written signal to a spoken one, but this is not all that happens: actually, the reader has to both decode the written signal *and* encode the spoken one. This extra step is necessary precisely because of the prosodic impoverishment: when the written text is first encoded some information is lost and it needs to be reconstructed by the reader to generate a naturally sounding spoken signal. It then becomes clear why convincing TTS is so challenging: the synthetic speaker does not only have to provide accurate phoneme pronunciation, but it also needs to infer and add prosodic details to restore (as much as possible) the originally encoded information. A schematization of the described process can be found in fig. 1.1.

1.1.3 Key problems in text-to-speech

When building a TTS system, four key problems have been identified [1].

The first issue regards the use of encoding systems other than natural language: classic examples are numbers, dates and times, units of measurement, and currencies. The most common method to deal with these instances consists in applying text normalization: simply put, a normalizer tries to rewrite any non-standard text as proper linguistic text. For example, in Italian “28” will be rewritten as “ventotto” and “km” will be transformed to “chilometri”.

The second problem is ambiguity. One form of ambiguity is represented by homographs, i.e. words that are written in the same way but pronounced differently according to the context: in Italian examples of homographs are “leggere” and “leggére”, “prèsidi” and “presidi”, or “principi” and “princìpi”³. Another form of ambiguity is syntactic ambiguity: an Italian example is “ci scusiamo dei possibili fastidi causati porgendo cordiali saluti”, where it may be interpreted as both “ci scusiamo dei possibili fastidi causati [porgendo cordiali saluti]” or as “ci scusiamo dei possibili fastidi [causati porgendo cordiali saluti]”. Both homographs and syntactic ambiguity influence pronunciation and prosody and often create problems when dealing with TTS.

The third issue is naturalness. As already discussed, a lack of proper prosody might induce in the listener a sense of artificiality, which may be distracting and annoying. Fixing this problem is very hard since a complete semantic understanding of both the content and the context would be needed. Some systems try to infer prosodic information but the issue is still one of the main limitations of believable TTS.

The last problem is intelligibility. There is not as much to talk about here: this aspect is simpler to measure and modern TTS models are almost perfect in terms of intelligibility, so this is at this point more of a prerequisite than an actual issue.

³In Italian stresses are omitted, except when they fall on the last vowel of the word: here they are reported just to illustrate the aforementioned differences.

1.2 Classical models

When creating a computer model that deals with TTS tasks, multiple approaches are feasible [3]. They can be roughly divided into classical models and modern neural-network-based solutions.

Articulatory synthesis

Articulatory synthesis [4], [5] is ideally the most effective method of speech synthesis, as it simulates the behavior of human articulators (lips, tongue, glottis, and moving vocal tract). It is however hard to put these models into practice, especially because of the difficulty to collect the required data. Thus, systems using this method tend to perform poorly compared to more abstract models.

Formant synthesis

Formant synthesis [6]–[8] is based on a set of rules to control a source-filter model, usually following the formant structure and other spectral properties of speech. The speech is created by using an additive synthesis module and an acoustic model, with the possibility to tune some parameters such as frequency, voicing, or noise. The speech created is usually highly intelligible and requires moderate computational resources: therefore, these methods are well-suited for embedded systems. The main drawback of formant synthesis is that the output is unnatural and contains artifacts.

Concatenative synthesis

Concatenative synthesis [9]–[13] works by concatenating pre-stored pieces of speech. The database usually consists of voice actors' recordings, ranging from whole sentences to single syllables. While this method usually produces very convincing results, it requires a huge database and the voice often lacks emotions, as the recordings need to be adaptable to any context.

Statistical parametric synthesis

Statistical parametric synthesis [14]–[18] is based on the idea of first generating the acoustic parameters that are needed to produce speech and then recovering speech from these parameters by using some algorithms. There are usually three main components in a statistical parametric synthesis model: a text analysis module, a parameter prediction module, and a vocoder. The text analysis module performs some pre-processing, such as text normalization, grapheme-to-phoneme conversion, and word segmentation, before extracting some linguistic features. The acoustic module extracts the acoustic features from the linguistic ones, while the vocoder synthesizes speech from the acoustic features. This method gives more a natural speech, it is flexible and it does not require a large amount of data. On the other hand, the generated speech can be less intelligible due to artifacts and the generated voice can be easily detected as synthetic.

1.3 Neural models

In the 2010s, some works started to introduce neural networks into statistical parametric synthesis models, using both deep neural networks [19], [20] and recurrent neural networks [21], [22]. However, these approaches still use linguistic features: a more modern approach [23] consists in using graphemes or phonemes as inputs and performing an end-to-end generation of acoustic features.

Main taxonomy

There are four main categories into which neural network models for TTS can fall:

1. Text analysis models, which converts characters into phonemes or linguistic features;
2. Acoustic models, which generate acoustic features starting from linguistic features or characters/phonemes;

3. Vocoder, which generate waveforms from linguistic or acoustic features;
4. End-to-end models, which convert characters/phonemes into waveforms.

Following this classification, there are different paths data can follow to go from text to speech:

- Characters → linguistic features → acoustic features → waveform;
- Characters → phonemes → acoustic features → waveform;
- Characters → linguistic features → waveform;
- Characters → phonemes → waveform;
- Characters → acoustic features → waveform;
- Characters → waveform.

1.3.1 Text analysis

Text analysis, also known in the TTS field as frontend, transforms the input characters into linguistic features or phonemes. Compared to parametric synthesis, where linguistic features are much more important for the final result, text analysis in neural models is greatly simplified. Many tasks can be carried out as part of text analysis, as discussed in the following sections.

Text normalization

As already mentioned in section 1.1.3, not all text is made up of characters corresponding directly to spoken words. Earlier text normalizers solved this issue with a rule-based pipeline, with later ones leveraging neural networks to work with a sequence-to-sequence approach. There are also some more recent works [24] proposing to combine both methods to improve performances.

Word segmentation

This step is essential when dealing with character-based languages, such as Chinese or Japanese: in these writing systems, word boundaries are not marked by spaces and thus need to be detected to facilitate later steps.

Part of speech tagging

Part of speech (POS) tagging consists in the identification of the POS corresponding to each word, such as noun, verb, adjective, and other classes common to most languages, as well as more specific ones, such as nominal classifiers for many Asian idioms.

Prosody prediction

As already discussed, prosody plays a big role in oral communication: thus, some approaches have been devised to address the task of predicting prosodic information. These methods vary from language to language, but in general, attention is put on tagging intonation and pauses, such as with the Tobi (tones and break indices) system for English [25].

Grapheme-to-phoneme conversion

Grapheme-to-phoneme conversion, or phonemization, is a very effective way to facilitate speech synthesis and consists in turning graphemes into phonemes. The approach is very different for alphabetic and character-based writing systems: for alphabetic languages, the main challenge is that a lexicon can not cover all possible word inflections, therefore a method to predict the phonemization of new word forms must be used; on the other hand, for character-based idioms, even though the lexicon is limited there are many polyphones⁴, therefore a way to infer the proper meaning depending on the context is necessary.

⁴Words with the same writing but different pronunciation.

1.3.2 Acoustic models

While in classical TTS the most common model for this step consists in a hidden Markov model [14], [15], in neural implementations different approaches are used, such as recurrent neural networks (RNN), convolutional neural networks (CNN), and transformer-based models. Compared to their classical counterparts, neural-based models present several advantages: linguistic-acoustic alignment must not be provided as the models implicitly learn it; linguistic features are simplified into only characters of phoneme sequences; acoustic features change from low-dimensional cepstrums⁵ to high-dimensional mel (or even linear) spectrograms.

RNN-based models

The most popular RNN-based models are the ones from the Tacotron series. The original architecture [26] utilizes an encoder-attention-decoder framework, outputting linear spectrograms starting from an input of characters. The Griffin-Lim algorithm [27] is also leveraged to generate waveforms. Subsequent variations such as Tacotron 2 [28] produce mel spectrograms instead and use the WaveNet model [29] to convert them into waveforms.

CNN-based models

Examples of CNN-based models are the architectures from the Deep Voice family. Deep Voice [30] first obtains linguistic features through the convolutional neural network and then converts them to waveforms using WaveNet [29]. Deep Voice 2 [31] improves the network structures and multi-speaker modeling. Deep Voice 3 [32] uses a fully convolutional network for speech synthesis, with a compact sequence-to-sequence model, and predicts mel spectrograms directly.

⁵A cepstrum is a result of computing the inverse Fourier transform of the logarithm of an estimated signal spectrum.

Transformer-based models

The first transformer-based TTS model (TransformerTTS [33]) was developed while trying to solve two issues of the RNN architectures: first, because of the nature of recurrency itself, the encoder and the decoder can not be trained in parallel; second, the text and speech sequences tend to be quite long, which is not ideal for RNNs. The main drawback in using transformers is that, despite achieving a similar quality as Tacotron 2 with less training time, parallel training makes the encoder-decoder attentions not robust. To solve this problem, alternative architectures have been proposed [34], [35].

A further evolution is represented by FastSpeech [36]: this model uses a feed-forward transformer network to generate mel spectrograms in parallel, removing the attention mechanism between text and speech to improve robustness. FastPitch [37] improves FastSpeech by using pitch information as additional decoder inputs.

1.3.3 Vocoder

Early neural vocoders [38]–[40] directly took linguistic features as inputs, skipping the explicit acoustic features extraction step. However, later models [41], [42] started to work with mel-spectrograms. Four main categories of vocoder models can be discerned: autoregressive, flow-based, GAN-based, and diffusion-based.

Autoregressive vocoders

WaveNet [38] was the first neural vocoder. It uses dilated convolutions to generate waveforms autoregressively. Contrary to previous statistical methods, almost no prior knowledge of audio signals is needed, resulting in an end-to-end learning approach. While the original model uses linguistic features as inputs, it can be adapted to process linear and mel spectrograms as well. The main drawback of WaveNet is the slow inference speed: for this reason, some improvements in terms of weight and speed have been proposed, such as SampleRNN [43], Char2Wav [44], and WaveRNN [45].

Flow-based vocoders

Flow-based vocoders work on the idea that normalizing flow is a kind of generative model, as it transforms a probability density with a sequence of invertible mappings [46]. Data are generated from a standard probability distribution during sampling, using the inverse of the transforms. Flow-based vocoders can be further divided into two subcategories: autoregressive [39] and bipartite transforms [41], [42]. Autoregressive methods are usually more expressive, but require teacher distillation during training; bipartite approaches have a simpler training pipeline but tend to require a larger amount of parameters to achieve comparable performances.

GAN-based vocoders

Generative adversarial networks (GANs) are one of the most utilized methods for generative tasks. They consist of a generator, which generates data, and a discriminator, which judges the authenticity of data from the generator. For the generator, most vocoders leverage dilated convolutions to increase the receptive field and transposed convolutions for upsampling. Regarding the discriminator, more varied approaches are used: random window [47], multi-scale [48], multi-period [49], and hierarchical [50] are all common methods.

Diffusion-based vocoders

The basic idea of diffusion-based vocoders [51] is to generate mappings between data and latent distributions with a diffusion process and an inverse process. During diffusion, the waveform data sample is gradually added with some random noises until it becomes Gaussian noise. In the reverse process, denoising is applied to the Gaussian noise to obtain waveform data samples. Despite generating very high voice quality, these approaches suffer from slow inference speed.

Chapter 2

Phonemes

2.1 Phonemes and the IPA Alphabet

In most languages, graphemes (the glyphs of the written language) do not directly traduce in phones (i.e. the sounds produced by the speakers). To transcribe phones more precisely, phonemes are used instead of graphemes.

The most widespread way to transcribe sounds is the International Phonetic Alphabet [52], maintained by the International Phonetic Association (IPA) [53]. The IPA, established in 1886 in Paris, is the oldest organization for phoneticians. The first version of the IPA Alphabet was first published in 1888 and underwent many changes, with the most recent revision dated 2020 (fig. 2.1). It consists of two families of symbols: letters and other symbols.

2.1.1 Consonants

In general, consonants are sounds articulated with a complete or partial closure of the vocal tract. They can in turn be divided into pulmonic and non-pulmonic consonants.

Pulmonic consonants are the most common ones, especially in European languages, and are articulated by using the glottis or the oral cavity together with the lungs. They can also be further categorized according to two additional criteria: manner and place of articulation.

THE INTERNATIONAL PHONETIC ALPHABET (revised to 2020)											
CONSONANTS (PULMONIC)											© 2020 IPA
	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		t̪ d̪	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	m̪		n		n̪	ɲ	ŋ	N		
Trill		B		r					R		
Tap or Flap		v		r̪		t̪					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ɟ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɭ							
Approximant		v		ɹ		ɻ	j	ɻ̪			
Lateral approximant				ɬ		ɬ̪	ɻ̪	ɻ̪	L		

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

CONSONANTS (NON-PULMONIC)

Clicks	Voiced implosives	Ejectives
ʘ Bilabial	ɓ Bilabial	,
ǀ Dental	ɗ Dental/alveolar	Examples: p' Bilabial
ǃ (Post)alveolar	ʄ Palatal	t' Dental/alveolar
ǂ Palatoalveolar	ɠ Velar	k' Velar
ǁ Alveolar lateral	ʄ ɠ Uvular	s' Alveolar fricative

OTHER SYMBOLS

ʍ Voiceless labial-velar fricative	ç ʐ Alveolo-palatal fricatives
w Voiced labial-velar approximant	ɿ Voiced alveolar lateral flap
ɥ Voiced labial-palatal approximant	ʃ ɷ Simultaneous ʃ and ɷ
h Voiceless epiglottal fricative	Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary.
ɸ Voiced epiglottal fricative	
ʗ Epiglottal plosive	

DIACRITICS

o Voiceless	ɳ ɖ	.. Breathy voiced	ɳ̪ ɖ̪	o Dental	ʈ ɖ
~ Voiced	ʂ ʈ	~ Creaky voiced	ʂ̪ ʈ̪	~ Apical	ʈ̪ ɖ̪
h Aspirated	tʰ dʰ	~ Lingualobial	t̪ ɖ̪	~ Laminar	ʈ̪ ɖ̪
› More rounded	ɔ	w Labialized	tʷ ɖʷ	~ Nasalized	ɛ̪
⟨ Less rounded	ɔ̪	j Palatalized	t̪j ɖ̪j	n Nasal release	dn̪
+ Advanced	u̪	Y Velarized	t̪y ɖ̪y	↑ Lateral release	dl̪
– Retracted	ɛ̪	ɸ Pharyngealized	t̪f ɖ̪f	~ No audible release	d̪
.. Centralized	œ̪	~ Velarized or pharyngealized	t̪ ɖ̪		
×					
Mid-centralized	ɛ̪	Raised	ɛ̪ (ɻ̪ = voiced alveolar fricative)		
Syllabic	ɳ̪	Lowered	ɛ̪ (ɸ̪ = voiced bilabial approximant)		
Non-syllabic	ɛ̪	Advanced Tongue Root	ɛ̪		
~ Rhoticity	œ̪	Retracted Tongue Root	ɛ̪		

Some diacritics may be placed above a symbol with a descender, e.g. ɳ̪

VOWELS

Where symbols appear in pairs, the one to the right represents a rounded vowel.

SUPRASEGMENTALS

↑ Primary stress	founə'tuʃən
↓ Secondary stress	eɪ
— Long	eː
‐ Half-long	e'
‐ Extra-short	ɛ̪
Minor (foot) group	
Major (intonation) group	
• Syllable break	ni.ækt
‐ Linking (absence of a break)	

TONES AND WORD ACCENTS

LEVEL	CONTOUR		
ጀ or ገ	Extra high	ጀ or ገ	Rising
ጀ	High	ጀ	Falling
ጀ	Mid	ጀ	High rising
ጀ	Low	ጀ	Low rising
ጀ	Extra low	ጀ	Rising-falling
↓	Downstep	↗	Global rise
↑	Upstep	↘	Global fall

Typefaces: Doulos SIL (metatext), Doulos SIL, IPA Kiel, IPA LS Uni (symbols)

Figure 2.1: The International Phonetic Alphabet.

Source: [54].

Non-pulmonic consonants, on the other hand, do not depend on the air-flow coming from the lungs. They include ejectives, clicks, and implosives.

2.1.2 Vowels

Vowels are sounds occurring in the center of a syllable and are classified according to the position of the tongue. In particular, tongue vertical (height) and horizontal (backness) positions are taken into consideration.

2.1.3 Other symbols

The IPA Alphabet also makes use of other symbols to convey phonetic details. These characters include diacritics (which specify articulatory details of given letters), pitch and tone symbols, and suprasegmentals. Suprasegmentals are particularly useful for Italian phonemization, as they provide information about the stress position as well as the duration of phonemes.

2.2 Italian phonemes

Of course, in Italian not all of the IPA symbols find usage. The used phonemes are shown in tables 2.1 and 2.2.

2.2.1 Consonants

Regarding the consonants, the phones associated with most of them are quite intuitive, as they correspond to how the letters are pronounced in standard written Italian. In particular, /m/, /n/, /p/, /b/, /t/, /d/, /f/, /v/, /l/, and /r/ do not need any clarification. Some others are less clear:

- /k/ and /g/ are always pronounced *dure*, such as in “cane” and “gatto”;
- /tʃ/ and /dʒ/ are the *dolce* versions of “c” and “g”, such as in “cena” and “gelato”;

	Labial	Dental alveolar	Post-alveolar palatal	Velar
Nasal	m	n	j̊	
Stop	p b	t d		k g
Affricative	ts dž	tʃ dʒ		
Fricative	f v	s z	ʃ	
Approximant			j	w
Lateral		l	ʎ	
Trill		r		

Table 2.1: Italian consonant phonemes.

- /s/ and /z/ both correspond to the “s” grapheme, but /s/ is voiceless (*sorda*, such as in “**sano**”) while /z/ is voiced (*sonora*, such as in “*causa*”);
- /ts/ and /dž/ both correspond to the “z” grapheme, but /ts/ is voiceless (*sorda*, such as in “*azione*”) while /dž/ is voiced (*sonora*, such as in “*zaino*”);
- /j̊/ represents the “gn” sound such as in “**gnomo**”;
- /ʃ/ represents the “sc” sound such as in “**sciare**”;
- /ʎ/ represents the “gl” sound such as in “**aglio**”;
- /j/ and /w/ represent semivowels or semiconsonants, such as in “**ieri**” and “**uovo**”.

2.2.2 Vowels

Vowels are even simpler than consonants, with just a caveat: while /i/, /u/, and /a/ are the same as their written counterparts, “e” and “o” present two versions. /e/ and /o/ correspond to the closed versions of the relative letters, such as in “*vero*” and “*ombra*”, while /ɛ/ and /ɔ/ correspond to the open versions, such as in “*etto*” and “*notte*”.

	Front	Central	Back
Close	i		u
Close-mid	e		o
Open-mid	ɛ		ɔ
Open		a	

Table 2.2: Italian vowel phonemes.

2.2.3 Other symbols

In addition to consonants and vowels, two other symbols are commonly used. The first one is the primary stress (//), written before the stressed syllable. The other one is the length mark (:/), written after a consonant whose length must be doubled (*doppia*).

Part II

EXPERIMENTAL SETUP

Chapter 3

Datasets

The experimental work can be roughly divided into three parts: the first one is the creation of a phonemizer for the Italian language, the second one deals with the training of the text-to-speech model, and the third one consists of data gathering regarding the human evaluation of the model.

3.1 Phonemization

To properly train a phonemizer, a dataset containing Italian words and their phonemic transcription must be exploited. Since this kind of dataset is not present online, it needs to be created manually.

3.1.1 Words gathering

The first step consists in gathering the Italian words. One approach would be, for example, to get the words from a dictionary or some lists containing the most frequently used terms. This would, however, create an issue: in Italian most parts of speech present inflections, and it is important to train the phonemizer on the inflected words as well. The conjugation of verbs is especially complex in Italian, with some lemmas completely changing their form and stress position for different tenses.

A more realistic sample of inflected words needs to be chosen: the whole body of Wikipedia's Italian featured articles [55] is used, which as of today

consists of a total of 539 voices. Starting from this list, all articles are parsed and the words they contain are stored.

3.1.2 Phonemization

When trying to phonemically transcribe the gathered words, two main issues arise. The first one is that not all phonemic transcription sources include stresses, which are crucial for training the phonemizer, while the second one is that virtually no phonemization of inflected terms is available.

Both these problems are solved by using the WordReference.com Italian to English dictionary [56]: in addition to the inclusion of stresses, when an inflected term is looked for the dictionary returns the main lemma, which in turn contains the phonemic transcription. Unfortunately, phonemic transcription of inflected words is rarely provided, so some processing is needed.

The processing pipeline is quite complex, but its main idea is to obtain the phonemized lemma corresponding to each word and then to adapt the phonemic transcription to the inflected term. This is achieved by exploiting the fact that in Italian most verbs, nouns, and adjectives have almost standard desinences, so the final parts of words can usually be deterministically phonemized.

The last thing that must be done is to deal with the stress position since in some verbal tenses it tends to move from its original location: this is obtained by using a dedicated function. The algorithm first checks for inflected verbs ending in “-iamo”, “-iate”, “-iono”, or “-iano” and moves the stress right before the desinence. For example “cuocere” is phonemized as /'kwɔtʃere/, but its inflected form “cuociamo” is pronounced with the stress moved to the next syllable: /kwo'tʃamo/. The same is done with other desinences, such as “-ssimo”, “-ate”, “-ete”, “-ite”. On the other hand, some forms make the stress move backward, such as “-ano”, “-ono”, “-ino”, or “-ero”: “vedere” is phonemized as /ve'dere/, but “vedono” becomes /'vedono/. The same is done with forms ending in “-a”, “-e”, “-i”, or “-o”. Of course, these deterministic rules do not cover all exceptions of the Italian language, but when dealing with a large amount of data a margin of error is always unavoidable.

3.1.3 Dataset analysis

The final dataset is then split into training, validation, and test sets. Firstly, the dataset is split in 25 % for test and 75 % for training and validation. Then, the latter is split again in 75 % for training and 25 % for validation. The final proportions are 56.25 % for training, 18.75 % for validation and 25 % for test.

The training set consists of 18 173 pairs, each containing a word and its phonemized pronunciation. For the graphemic transcription, the mean length is 8.57 with a standard deviation of 2.46, the minimum length is 1, while the maximum is 20. For the phonemic transcription, the mean length is 9.75 with a standard deviation of 2.64, the minimum length is 2, while the maximum is 23.

The validation set consists of 6058 pairs, each containing a word and its phonemized pronunciation. For the graphemic transcription, the mean length is 8.59 with a standard deviation of 2.46, the minimum length is 2, while the maximum is 19. For the phonemic transcription, the mean length is 9.79 with a standard deviation of 2.65, the minimum length is 2, while the maximum is 21.

The test set consists of 8078 pairs, each containing a word and its phonemized pronunciation. For the graphemic transcription, the mean length is 8.58 with a standard deviation of 2.44, the minimum length is 1, while the maximum is 20. For the phonemic transcription, the mean length is 9.77 with a standard deviation of 2.61, the minimum length is 2, while the maximum is 23.

The higher length encountered in the phonemized data is to be expected, since stresses are added and some graphemes are rendered with multiple phonemes. The length statistics and distributions can be found in tables 3.1 and 3.2 and figs. 3.1 and 3.2.

	Training	Validation	Test
Mean	8.57	8.59	8.58
Standard deviation	2.46	2.46	2.44
Minimum	1	2	1
25 %	7	7	7
50 %	8	9	8
75 %	10	10	10
Maximum	20	19	20

Table 3.1: Graphemic dataset length statistics.

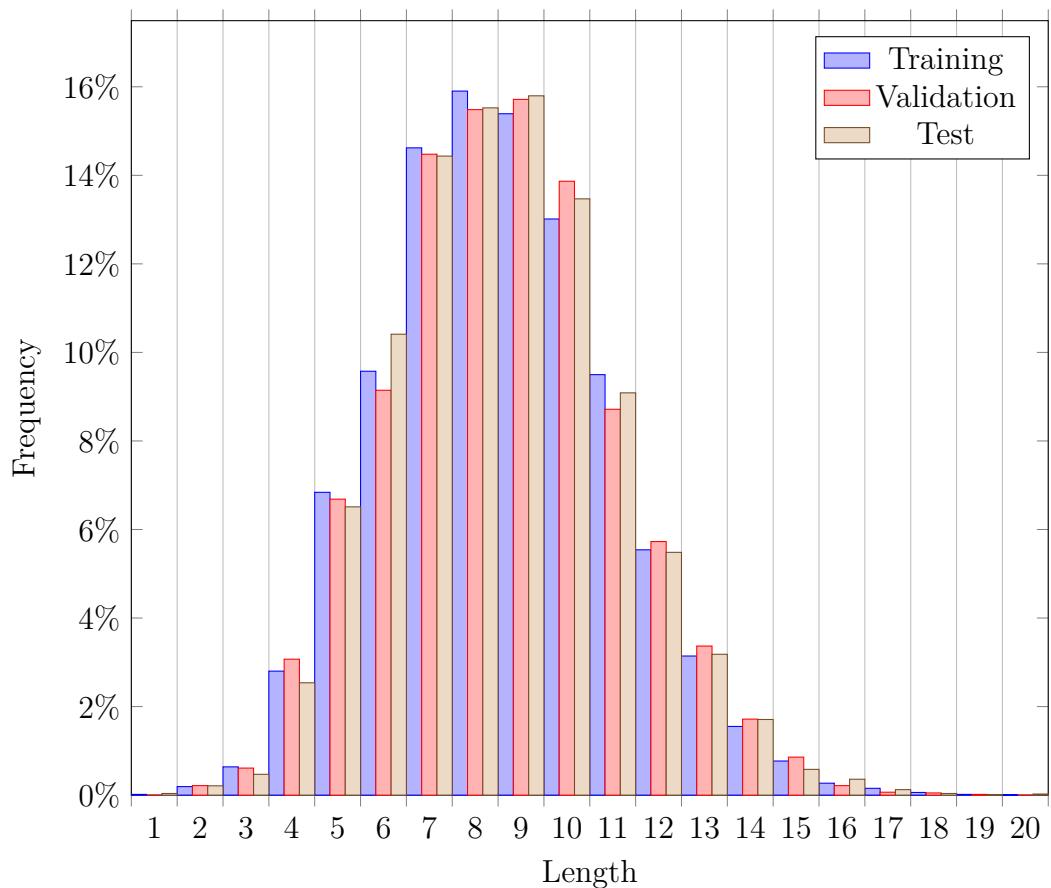


Figure 3.1: Graphemic dataset length distributions.

	Training	Validation	Test
Mean	9.75	9.79	9.77
Standard deviation	2.64	2.65	2.61
Minimum	2	2	2
25 %	8	8	8
50 %	10	10	10
75 %	11	11	11
Maximum	23	21	23

Table 3.2: Phonemic dataset length statistics.

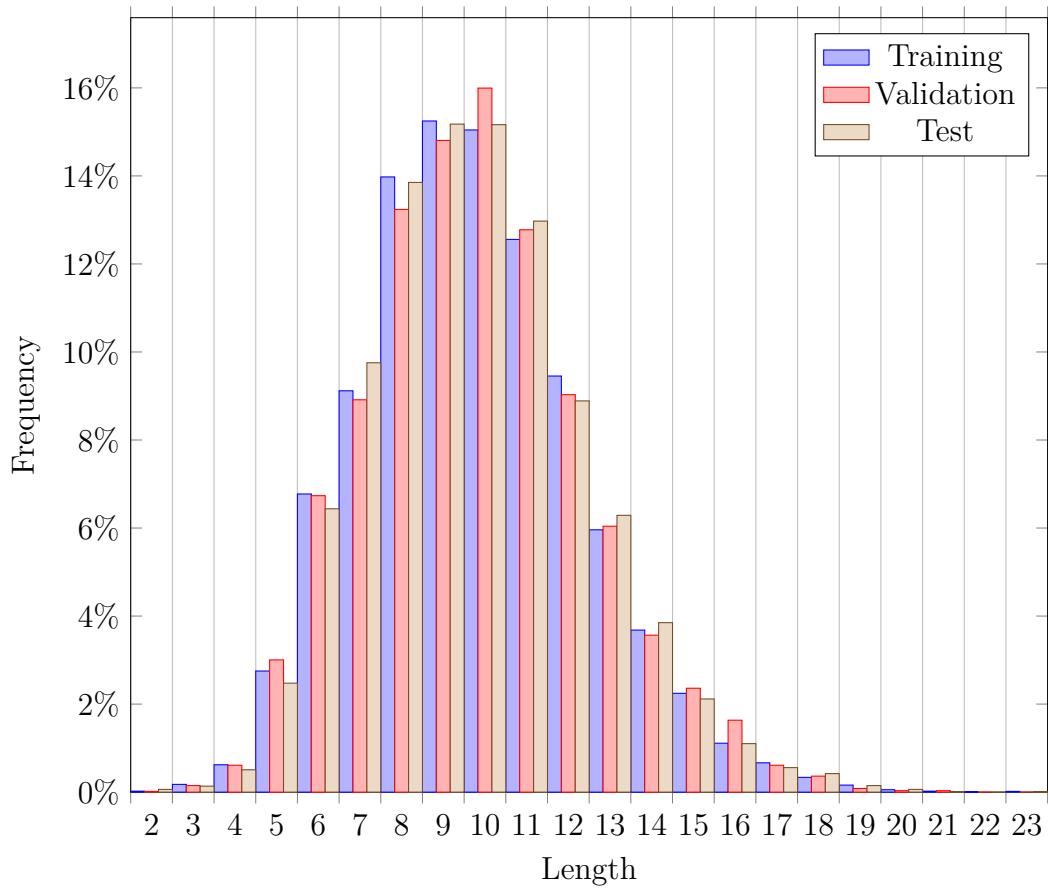


Figure 3.2: Phonemic dataset length distributions.

3.2 Text-to-speech

For the text-to-speech dataset, some requirements need to be fulfilled. First, all data have to be audio clips of the same speaker whose voice is to be cloned. Second, the files must be reasonably short, not exceeding 20s in duration. Moreover, the audios need to come with a corresponding graphemic transcription, which is then converted to a phonemic one.

For this purpose, several audio clips of a single speaker reading a narrative text are chosen. The original audios are `.wav` files, with a sample rate of 22 050 Hz, 16 bit of depth, and 2 channels.

3.2.1 Preprocessing

The text-to-speech pipeline works with relatively short audio clips (under 20s). Since the original files are much longer (tens of minutes), they need to be split and cleaned.

To obtain the synchronization map between the text strings and the audio, the Aeneas tool is used [57]. After the maps are created, the audio files are split using the Pydub Python package [58]. The Pydub package is also exploited to convert the audio from stereo to mono. To trim silence at the beginning and the end of clips, the `silenceremove` feature of FFmpeg is employed [59].

To achieve better training, clips that are too long or too short are removed from the datasets. In particular, a minimum length of 1.00s and a maximum length of 14.00s are chosen. Since the synchronized text is not always optimally parsed, the text content is inspected as well, removing all strings which do not contain readable characters.

Lastly, `.json` files are produced, with objects containing the following keys: `audio_filepath`, `duration`, and `text`. The `audio_filepath` value corresponds to the path of the audio file described by the object, `duration` expresses the length of the clip in seconds (computed using Pydub), while `text` contains the phonemized text corresponding to the audio. Before being phonemized, all texts are properly normalized (section 1.1.3) with a custom normalizing function. This normalization pipeline is adapted to Italian

from the English normalizer provided by the NVIDIA NeMo framework [60]. Because of the narrative nature of the texts used for training, only basic normalization is used: if the TTS model is applied in more technical areas, more complex normalization will be needed.

3.2.2 Dataset analysis

A part of the dataset is used for the main training, while a smaller part is reserved for fine-tuning. The main dataset consists of 41 750 clips for a total duration of 60.31 h, while the fine-tuning one has 10 217 audios for a total duration of 14.84 h. Both subsets are then split into training, validation, and test. The percentages are 70 % for training, 10 % for validation, and 20 % for test.

The main training set consists of 29 237 clips for a total duration of 42.38 h, while the fine-tuning one has 7154 audios for a total duration of 10.45 h. For the main set, the mean duration is 5.22 s with a standard deviation of 3.01 , the minimum duration is 1.00 s, while the maximum is 13.96 s. For the fine-tuning set, the mean duration is 5.26 s with a standard deviation of 3.08 , the minimum duration is 1.00 s, while the maximum is 13.96 s.

The main validation set consists of 4173 clips for a total duration of 5.96 h, while the fine-tuning one has 1023 audios for a total duration of 1.46 h. For the main set, the mean duration is 5.15 s with a standard deviation of 2.98 , the minimum duration is 1.00 s, while the maximum is 13.96 s. For the fine-tuning set, the mean duration is 5.14 s with a standard deviation of 3.00 , the minimum duration is 1.00 s, while the maximum is 13.96 s.

The main test set consists of 8340 clips for a total duration of 11.97 h, while the fine-tuning one has 2040 audios for a total duration of 2.92 h. For the main set, the mean duration is 5.17 s with a standard deviation of 3.01 , the minimum duration is 1.00 s, while the maximum is 13.96 s. For the fine-tuning set, the mean duration is 5.15 s with a standard deviation of 3.00 , the minimum duration is 1.00 s, while the maximum is 13.96 s.

The duration statistics and distributions can be found in tables 3.3 and 3.4 and figs. 3.3 and 3.4.

	Training (s)	Validation (s)	Test (s)
Mean	5.22	5.15	5.17
Standard deviation	3.01	2.98	3.01
Minimum	1.00	1.00	1.00
25 %	2.76	2.76	2.80
50 %	4.56	4.52	4.48
75 %	7.04	7.00	6.92
Maximum	13.96	13.96	13.96

Table 3.3: Main dataset duration statistics.

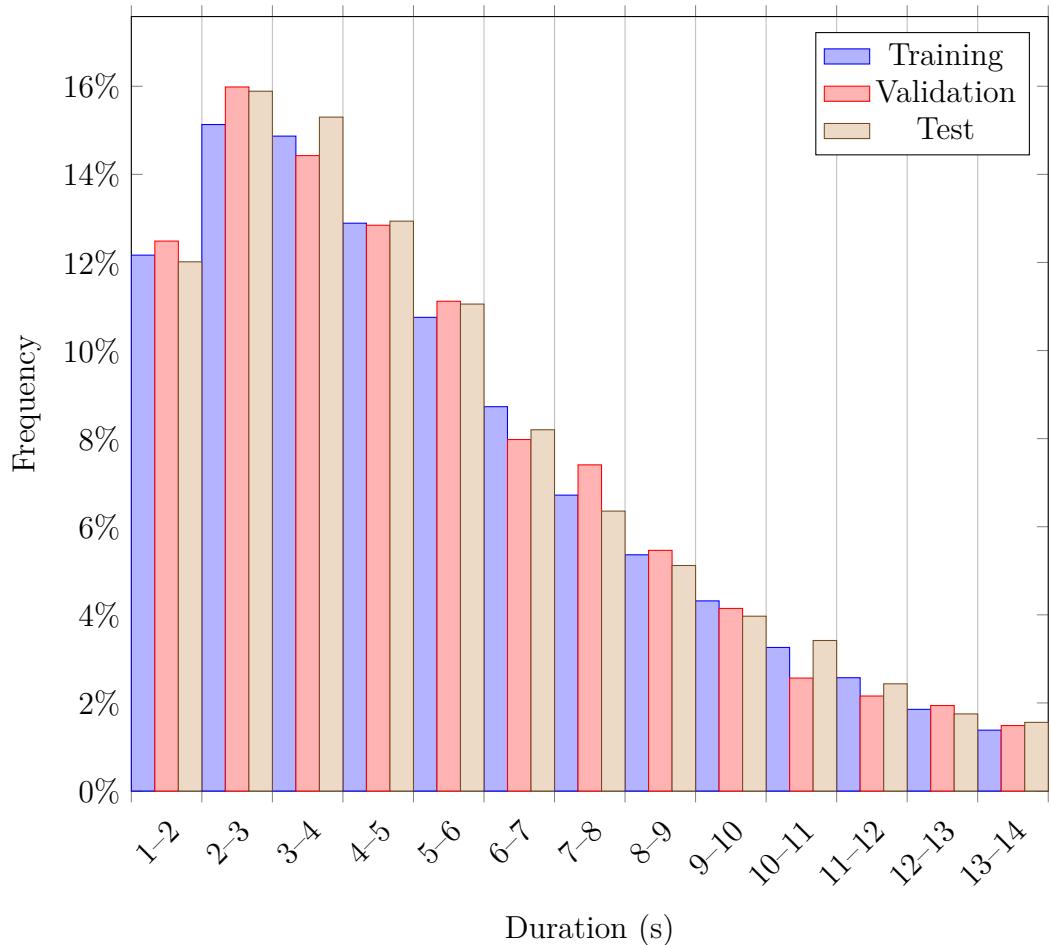


Figure 3.3: Main dataset duration distributions.

	Training (s)	Validation (s)	Test (s)
Mean	5.26	5.14	5.15
Standard deviation	3.08	3.00	3.00
Minimum	1.00	1.00	1.00
25 %	2.76	2.84	2.76
50 %	4.56	4.44	4.48
75 %	7.20	7.00	7.04
Maximum	13.96	13.96	13.96

Table 3.4: Fine-tuning dataset duration statistics.

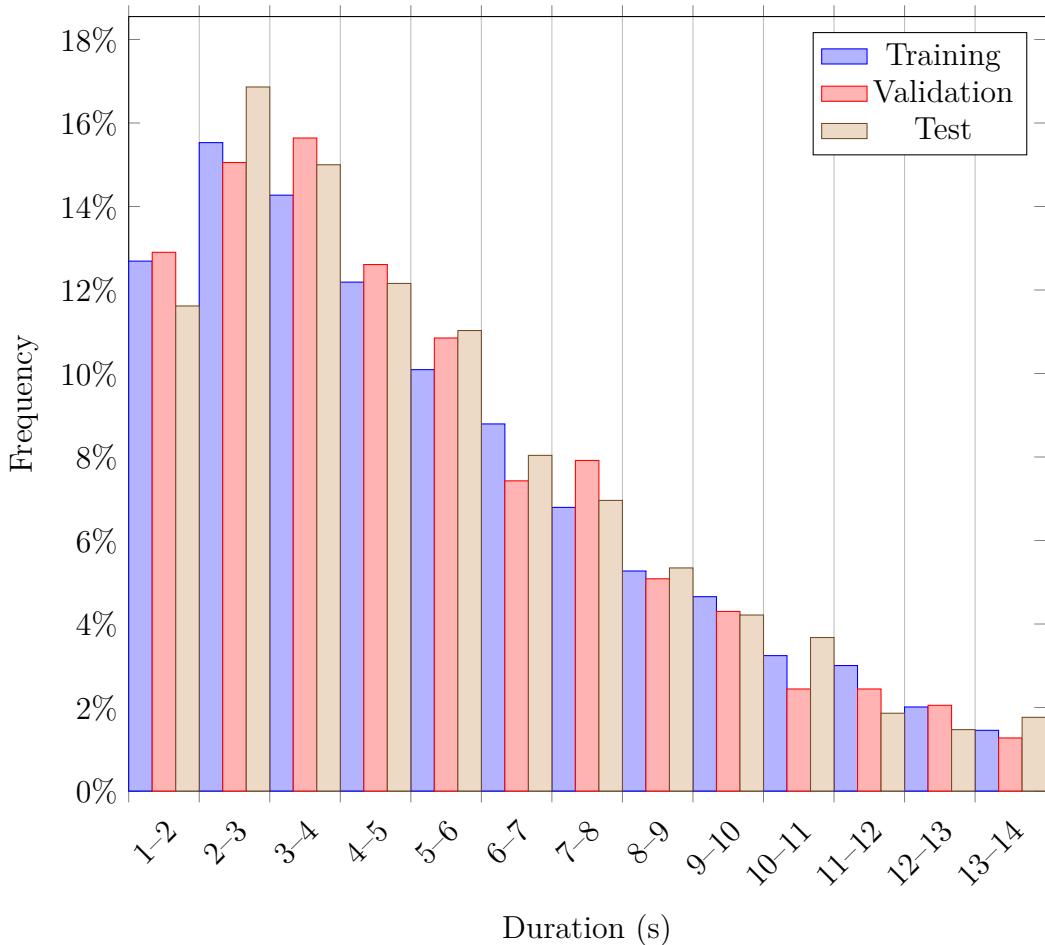


Figure 3.4: Fine-tuning dataset duration distributions.

3.3 Human evaluation

For the human evaluation part, an online questionnaire is used. The sample consists of 321 participants, 1 is younger than 18, 86 are between 18 and 24, 66 between 26 and 39, 139 between 40 and 59, with 29 being 60 or older. 199 identify themselves as biologically female, 121 as biologically male, while 1 person prefers not to specify their biological sex. Regarding the educational level, 227 possess at least a university degree, 90 a high school diploma, with 4 participants having lower certificates. Out of the 321 participants, only 2 are not native Italian speakers. The pie charts for these statistics are shown in appendix B. The technical questions asked are presented in detail in section 6.2.

Chapter 4

Architectures

4.1 Phonemization

In typical grapheme-to-phoneme approaches, the conversion is subdivided into three parts: aligning, training, and decoding. In particular, alignment is not always straightforward, therefore an empty symbol is often used in addition to phonemes by phonemizer models to facilitate the process.

Long short-term memory

One of the most promising approaches when dealing with this problem consists in using long short-term memory (LSTM) [61], a class of recurrent neural networks especially suited for sequence modeling. This family of networks avoids the need for explicit alignment before training since its dynamic contextual window makes it possible to see several graphemes before outputting a phoneme.

4.1.1 Transformer-based networks

An alternative method to achieve grapheme-to-phoneme conversion is the use of transformers [62]. The difference between a transformer-based solution and prior encoder-decoder architectures consists in stacking self-attention and fully connected layers for both the encoder and the decoder. Since no

recurrent layers are employed, positional encoding is added to the input and output embeddings.

4.1.2 Attention

The attention function is described as mapping a query Q and a set of key-value pairs (K, V) to an output:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (4.1)$$

where $\sqrt{d_k}$ is added to prevent softmax into regions with very small gradients.

Multi-head attention

Multi-head attention obtains parallel attention layers for learning different representations, computes scaled dot-product attention for each representation, concatenates the results, and projects the concatenation with a feedforward layer:

$$h_i = \text{Attention} \left(QW_i^Q, KW_i^K, VW_i^V \right) \quad (4.2)$$

$$\text{MultiHead}(Q, K, V) = \text{concat}(h_1, h_2, \dots, h_h) W^O \quad (4.3)$$

increasing the power of attention without computational overheads.

4.1.3 DeepPhonemizer

The model used to obtain grapheme-to-phoneme conversion is DeepPhonemizer [63], based on transformer models. The architecture consists of an embedding module followed by a positional encoder and several self-attention layers, each followed by a ReLU. In the end, a fully connected layer is used to predict the phonemes. The loss function used to train the model is the connectionist temporal classification (CTC) [64].

4.2 Text-to-speech

For the text-to-speech task, a two-stage approach is chosen. The first step consists in generating the mel spectrogram starting from the phonemized text, while the second employs the vocoder.

4.2.1 FastPitch

The phoneme-to-spectrogram step is achieved using the FastPitch model [37], a feed-forward model based on the previous FastSpeech architecture. The main idea of the model is to apply conditioning on a fundamental frequency estimated for every input symbol (referred to as pitch contour), addressing the quality shortcomings of the plain feed-forward transformer architectures. In addition, convergence is improved, and the need for knowledge distillation is eliminated. This architecture enables spectrogram synthesis at a pace 60 times faster than real-time. In addition, offsetting the reference frequency enables the production of naturally sounding low- and high-pitched variations of voices, preserving the perceived speaker identity.

The FastPitch architecture (shown in fig. 4.1) is composed of two feed-forward transformer (FFTr) stacks. One stack operates in the resolution of the input tokens, while the second one in the resolution of the output frames. The first FFTr block produces a hidden representation \mathbf{h} which is used to make predictions about both average pitch and duration for every input character using 1-dimensional CNNs (composed of 1-D convolutions and a fully connected layer):

$$\hat{\mathbf{d}} = \text{DurationPredictor}(\mathbf{h}), \quad \hat{\mathbf{p}} = \text{PitchPredictor}(\mathbf{h}) \quad (4.4)$$

where $\hat{\mathbf{d}}$ is a vector of natural numbers and $\hat{\mathbf{p}}$ is a vector of real numbers. The pitch is then projected to match the dimensionality of the hidden representation ($n \times d$) and it is added to it. The result is upsampled and converted in a mel-spectrogram $\hat{\mathbf{y}}$ by the output FFTr.

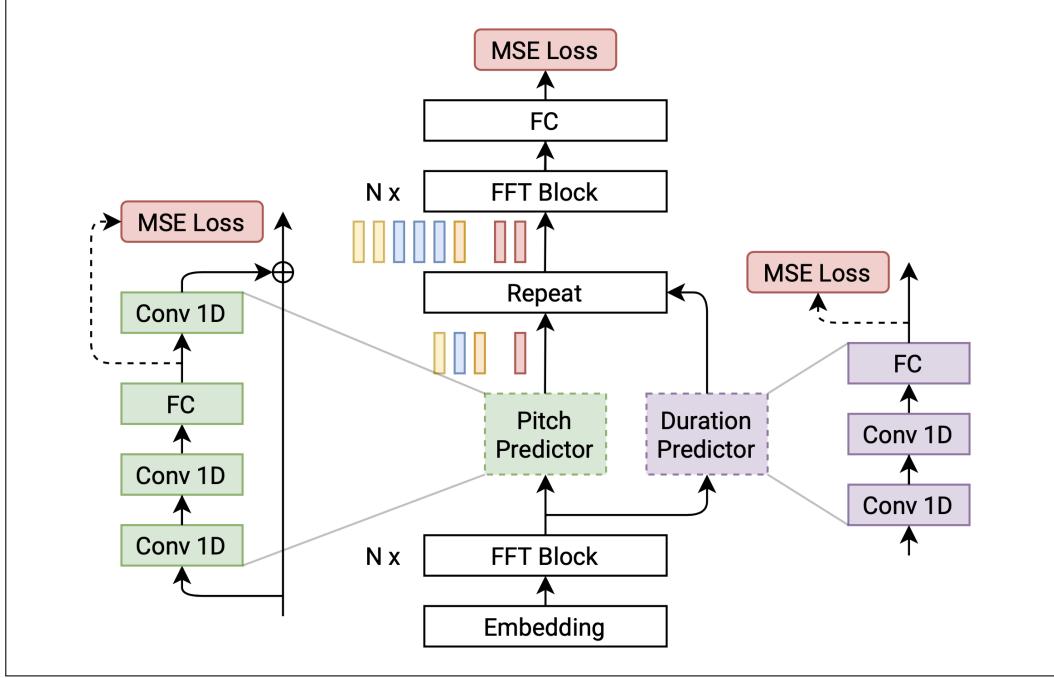


Figure 4.1: The FastPitch architecture.

Source: [37].

Duration predictor

The durations are inferred using a Tacotron model [26], trained on LJSpeech-1.1 [65]. The duration of the input symbol i is given by:

$$d_i = \sum_{c=1}^t \left[\arg \max_r \mathbf{A}_{r,c} = i \right] \quad (4.5)$$

where \mathbf{A} is the attention matrix.

Pitch predictor

Pitch ground truth is obtained using the accurate autocorrelation method [66], with Hann windows used to calculate the windowed signal. The maxima of the normalized autocorrelation function then become the candidate frequencies, with the lowest-cost path through the array of candidates calculated with the Viterbi algorithm.

Loss function

The loss function chosen to optimize the model is the summation of mean-squared errors between mel-spectrograms, pitches, and durations:

$$\mathcal{L} = \left\| \frac{\hat{\mathbf{y}} - \mathbf{y}}{5} \right\|_2^2 + \alpha \|\hat{\mathbf{p}} - \mathbf{p}\|_2^2 + \gamma \|\hat{\mathbf{d}} - \mathbf{d}\|_2^2 \quad (4.6)$$

where \mathbf{y} , \mathbf{p} , and \mathbf{d} represent the ground truth vectors.

4.2.2 HiFi-GAN

The spectrogram-to-audio step employs the High Fidelity Generative Adversarial Network (HiFi-GAN) model [49], which achieves higher sample quality than autoregressive or flow-based model while also increasing the computational efficiency. The architecture proposes two discriminators consisting of small sub-discriminators, each obtaining only a specific periodic part of the raw waveform. A generator trained adversarially is also used.

Generator

The generator (fig. 4.2) consists of a fully convolutional neural network, using a mel spectrogram as input and upsampling it with transposed convolutions to match the temporal resolution of raw waveforms. A multi-receptive field fusion (MRF) module follows each transposed convolution, observing patterns of various lengths in parallel: it returns the sum of outputs from multiple residual blocks.

Discriminator

Since audio consists of sinusoidal signals with various periods, the need to identify the diverse periodic patterns underlying the data arises. The multi-period discriminator (MPD) relies on sub-discriminators, each dealing with a portion of periodic signals of the input. An additional multi-scale discriminator (MSD) is used to capture consecutive patterns and long-term dependencies.

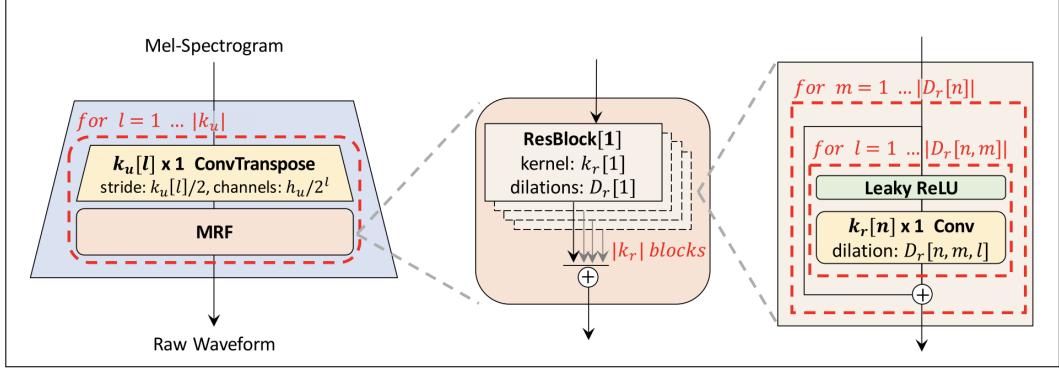


Figure 4.2: The HiFi-GAN generator.

Source: [49].

Each sub-discriminator of the MPD only accepts equally spaced samples as input, where the space corresponds to the period of the sub-discriminator. As shown in fig. 4.3a, the raw audio of length T is reshaped into 2-dimensional data of shape $\frac{T}{p} \times p$; then, a 2-dimensional convolution is applied. The width of the kernels in every layer is set to 1 to process the periodic samples independently. Strided convolutional layers with leaky ReLU are used in the sub-discriminators, before applying weight normalization.

The MSD is added to consecutively evaluate the audio sequence, operating on different input scales: raw, $\times 2$ average-pooled, and $\times 4$ average-pooled audios are used (fig. 4.3b). Each sub-discriminator is a stack of strided and grouped convolutions with leaky ReLUs. Weight normalization is applied to the second and third sub-discriminator.

Loss function

Three different sub-losses are used to compute the final loss function.

Least squares loss functions for non-vanishing gradient flows are used as GAN losses. The losses for the generator and the discriminator are:

$$\mathcal{L}_{\text{Adv}}(D; G) = \mathbb{E}_{(x,s)} [(D(x) - 1)^2 + (D(G(s)))^2] \quad (4.7)$$

$$\mathcal{L}_{\text{Adv}}(G; D) = \mathbb{E}_s [(D(G(s)))^2 - 1] \quad (4.8)$$

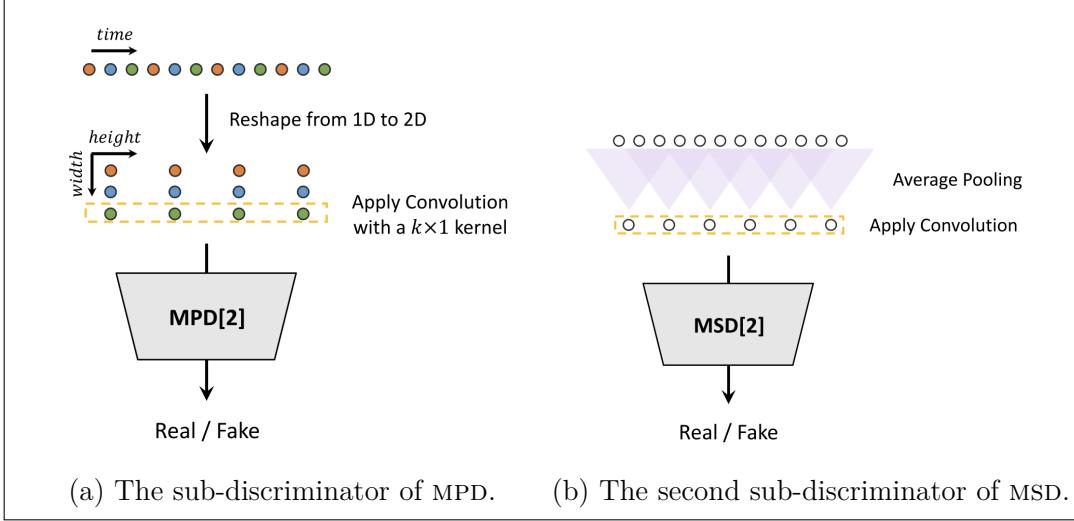


Figure 4.3: The HiFi-GAN discriminator.

Source: [49].

where x is the ground truth audio and s is the mel-spectrogram of the ground truth audio.

A mel spectrogram loss is also used to improve the training efficiency of the generator and the generation fidelity. It consists of an L_1 distance between the generated waveform mel spectrogram and that of the ground truth audio:

$$\mathcal{L}_{\text{Mel}}(G) = \mathbb{E}_{(x,s)} [\|\phi(x) - \phi(G(s))\|_1] \quad (4.9)$$

where ϕ is the waveform-to-spectrogram function.

The third loss function is the feature matching loss, a learned similarity metric measured by the difference in features of the discriminator between a ground truth sample and a generated one, defined as:

$$\mathcal{L}_{\text{FM}}(G; D) = \mathbb{E}_{(x,s)} \left[\sum_{i=1}^T \frac{1}{N_i} \|D^i(x) - D^i(G(s))\|_1 \right] \quad (4.10)$$

where T is the number of layers, D^i and N_i are the features, and the number of features in the layer i respectively.

The final losses for the generator and the discriminator are:

$$\mathcal{L}_G = \mathcal{L}_{\text{Adv}}(G; D) + \lambda_{\text{FM}} \mathcal{L}_{\text{FM}}(G; D) + \lambda_{\text{MEL}} \mathcal{L}_{\text{MEL}}(G) \quad (4.11)$$

$$\mathcal{L}_D = \mathcal{L}_{\text{Adv}}(D; G) \quad (4.12)$$

where $\lambda_{\text{FM}} = 2$ and $\lambda_{\text{MEL}} = 45$.

Part III

RESULTS

Chapter 5

Training

5.1 Phonemization

To create a proper Italian phonemizer, DeepPhonemizer is trained for 500 epochs. Each epoch consists of 567 steps, as the 18 173 grapheme-phoneme pairs are divided into batches of size 32. The chosen optimizer is Adam [67], with the default β_1 , β_2 , and ϵ values of 0.9, 0.999, and 1×10^{-8} respectively. The learning rate is set to 1×10^{-4} , with 1000 warm-up steps. The embedding dimension for the encoding is 512, while the dimension of the feedforward network is 1024. The number of encoder layers is 6, with 4 attention heads. A dropout probability of 0.1 is also used in the transformer. The training loss is shown in fig. 5.1. The evolution of the loss value in function of the number of steps highlights that probably a shorter training might be sufficient. However, training the model is pretty fast (with the setup used, an epoch takes less than 20 s) and a longer training might be beneficial, as shown in section 6.1.

5.2 Text-to-speech

For the text-to-speech task, three trainings are run. The first two regard FastPitch, while the last one is for HiFi-GAN. FastPitch is trained twice, as suggested by the documentation since this empirically gives better results.

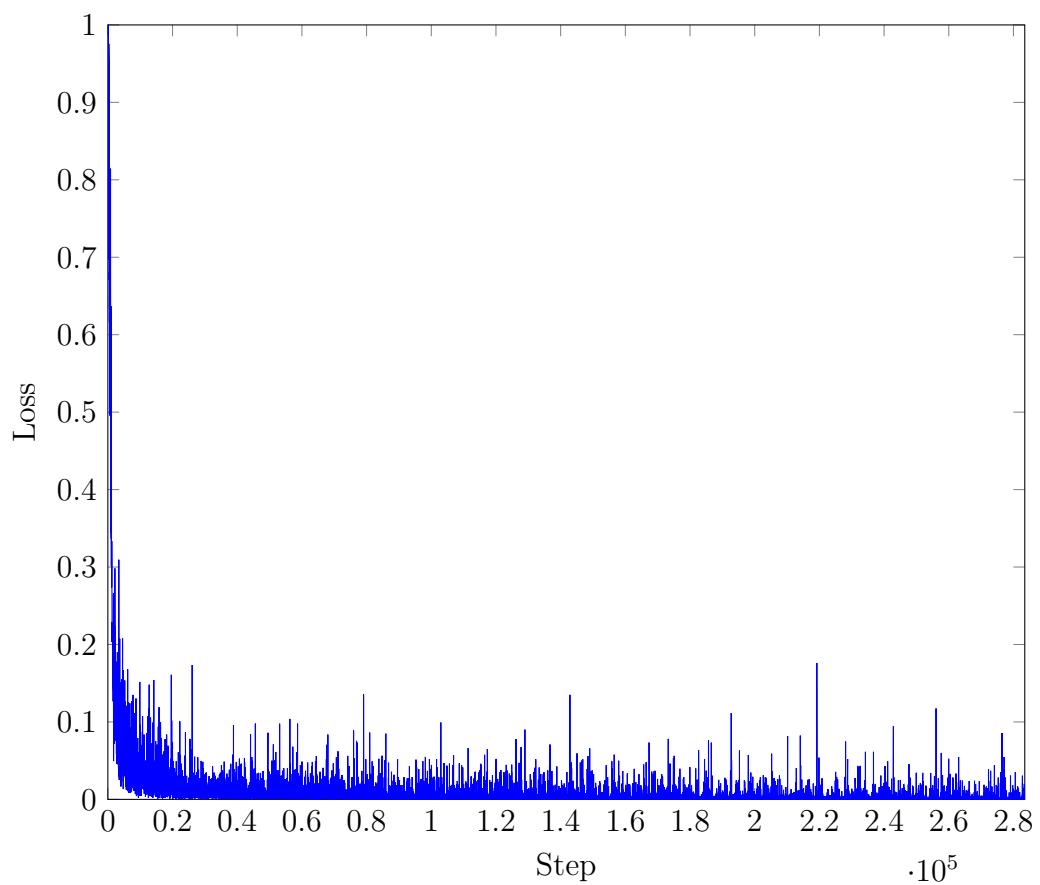


Figure 5.1: DeepPhonemizer training loss.

5.2.1 FastPitch main training

To start, FastPitch is trained from scratch for 1000 epochs. The batch size is 32. The optimizer is AdamW [68], with the default β_1 and β_2 values of 0.9 and 0.999 respectively and a weight decay of 1×10^{-6} . The initial learning rate is 0.001, with 1000 warm-up steps. The number of mel channels is set to 80, with a window size of 1024 and a window stride of 256. The embedding dimension is 384, with an embedding kernel size of 3. The FFTrs are formed of 6 layers each, with a head dimension of 64 and an inner dimension of 1536. The duration and pitch predictors both have a kernel size of 3, a filter size of 256, and 2 layers. A dropout of 0.1 is also used in all layers.

Regarding the loss, fig. 5.2 shows that the validation one is always smaller than the training one. This is peculiar since usually training loss tends to be smaller, but in this case, the training value takes into consideration additional parameters (such as CTC loss) which are not computed for validation. Other than this peculiarity, both losses seem to behave in a quite normal way, imparting confidence about correct training behavior.

5.2.2 FastPitch fine-tuning

After having trained the model from scratch, the weights are fine-tuned on the second dataset, as shown in section 3.2.2. For this task, the hyperparameters are kept the same, except for the number of epochs (decreased to 100) and the initial learning rate (which is lowered to 2×10^{-4}).

The losses shown in fig. 5.3 are even stranger than the ones in fig. 5.2: in this case, a proper explanation is hard to formulate, but looking at the documentation this is apparently an expected behavior. In fact, it is suggested to fine-tune the weights checking the outputs by ear until a satisfactory result is achieved, without relying on the loss values. An investigation of this behavior is out of the scope of this thesis, but empirical evaluations confirm the claims of the documentation.

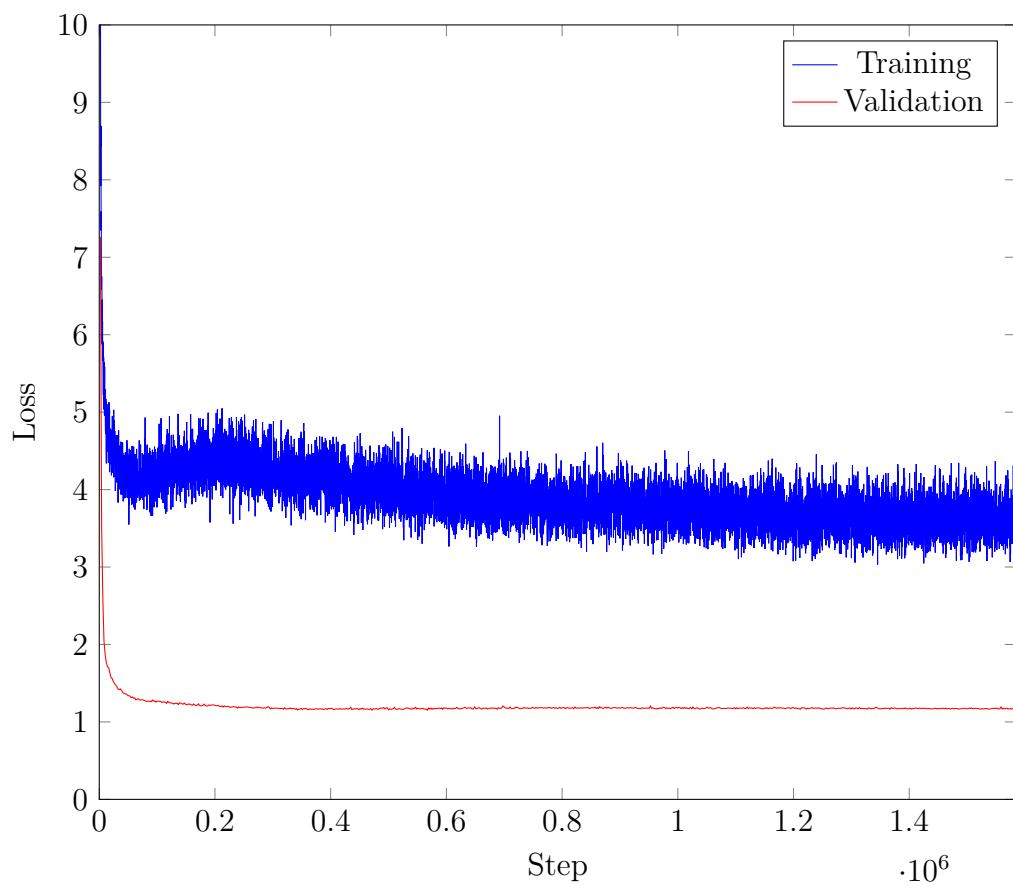


Figure 5.2: FastPitch main training loss.

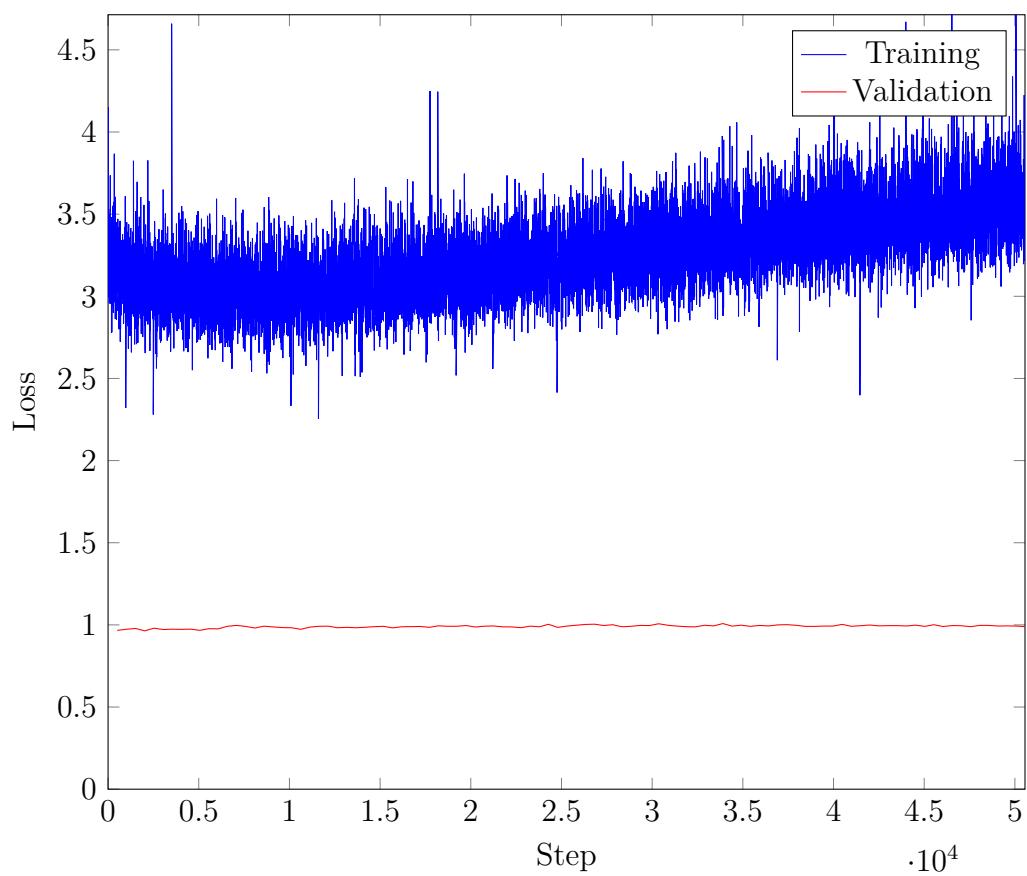


Figure 5.3: FastPitch fine-tuning loss.

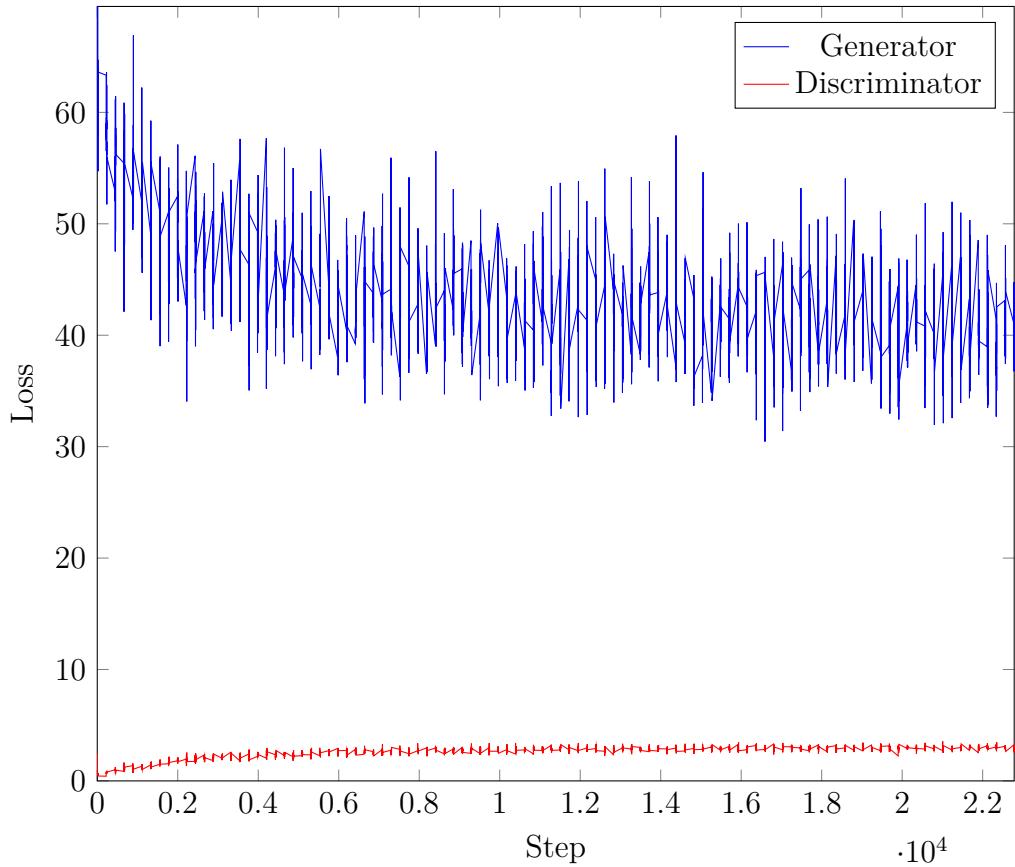


Figure 5.4: HiFi-GAN loss.

5.2.3 HiFi-GAN

The HiFi-GAN model is trained for a total of 2 500 000 steps. The optimizer is AdamW, with $\beta_1 = 0.8$ and $\beta_2 = 0.99$. The initial learning rate is 1×10^{-5} and the warm-up lasts for 50 000 steps. The number of mel channels is 80, with a window size of 1024 and a window stride of 256.

In fig. 5.4 the losses are shown only for the training set. The reason for this is that the loss on the validation set is only computed as the L_1 loss between the two mel spectrograms, while the training \mathcal{L}_G and \mathcal{L}_D losses are the ones shown in section 4.2.2. Because of the completely different nature of the two values, reporting the validation loss would not be useful.

Chapter 6

Evaluation

In this chapter, results are shown only for the phonemizer and the human evaluation steps, since the text-to-speech does not have powerful metrics to be analyzed by itself, which is the reason why human evaluation is needed in the first place.

6.1 Phonemization

To evaluate the phonemizer, the character error rate (CER) metric is chosen. The CER is based on the Levenshtein distance and is defined as:

$$\text{CER} = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \quad (6.1)$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and C is the number of correctly predicted characters. If the lengths of the two strings match, the CER can only take values between 0 and 1, while it can grow bigger when the lengths are different. A value of 0 corresponds to a perfect score.

Mean CER is evaluated regularly on both training and validation sets, and its evolution is shown in fig. 6.1. As already anticipated in section 5.1, the longer training brought some advantages. The epoch with the best CER is found to be located towards the end of the training, at a moment when the training loss is not improving anymore: the choice of a longer training,

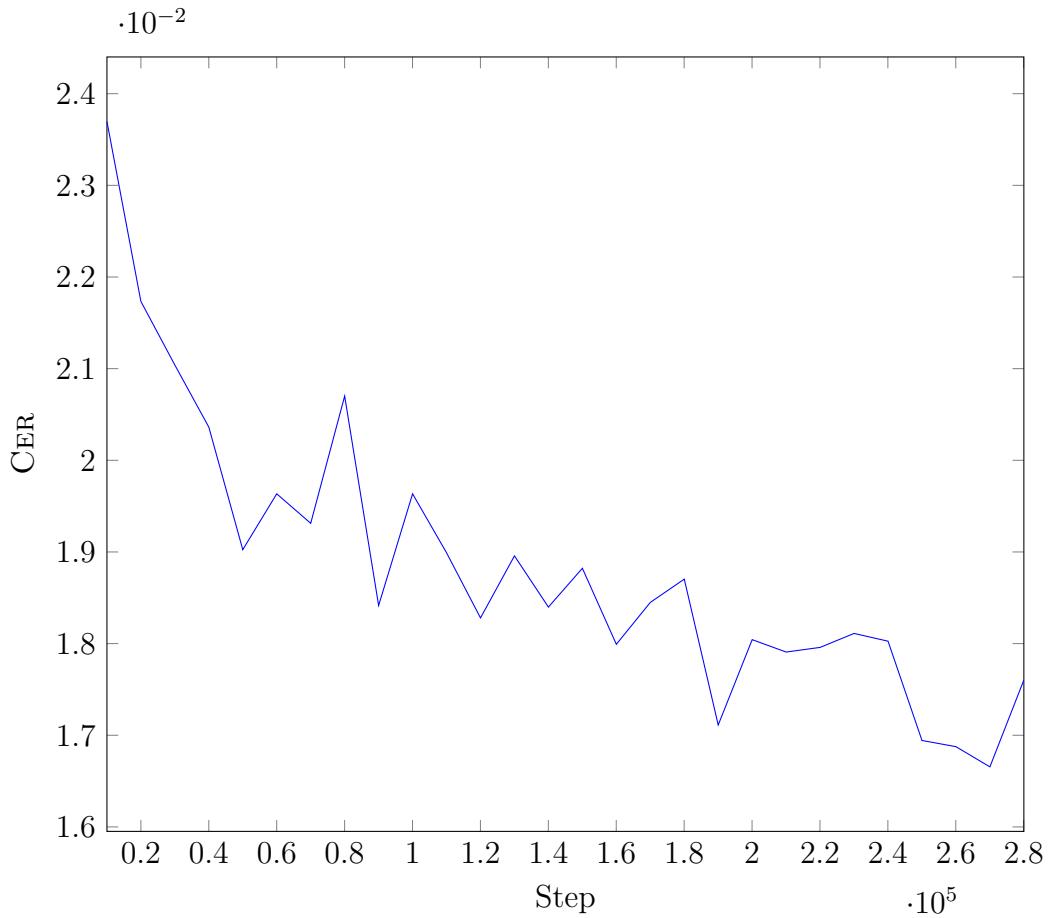


Figure 6.1: CER validation values.

therefore, pays back, resulting in a more accurate model. After the training is finished, and the checkpoint with the best mean CER is chosen, the metric is computed on all the words of the test set. The distributions are shown in fig. 6.2.

The results are very satisfactory: it is particularly positive that the distribution of fig. 6.2 shows a very evident peak at 0, representing a large amount of completely correctly phonemized words. The reason why the graph itself is shown, despite being almost completely blank, is to highlight this astonishing result: even when rendered at a high scale, the size of other bins is almost invisible, providing a visual representation of the impressive accuracy of the model.

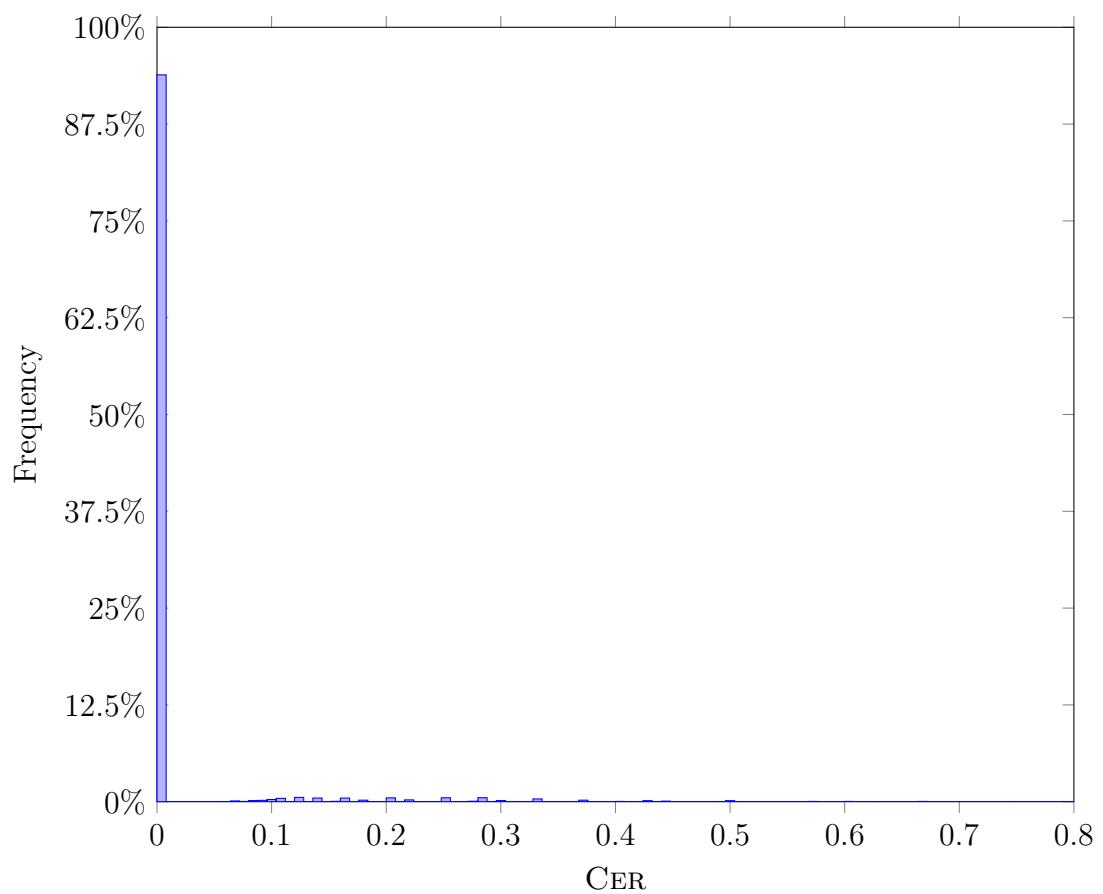


Figure 6.2: Test set CER distribution.

6.2 Human evaluation

Regarding human evaluation, the questionnaire is structured as follows. The voice produced by the training pipeline exposed in section 5.2 together with the ones generated by three other high-quality Italian TTS systems are used to record the same article by Gianfranco Moroldo:

Sono sempre stato un istintivo. Ho imparato che conviene affidarsi a un segno, a un odore, a un sapore. Piccoli campanelli d’alarme che ti dicono “vai”, oppure “non andare”. Bene, quell’elicottero che doveva portarci sulla collina 1383 aveva davvero un brutto rumore. «Allora, si va?», ci urla Pip, l’accompagnatore. «Andiamo», risponde Oriana Fallaci. Io sto seduto su un mucchio di casse e mangio la mia razione C, pollo e dolce alla crema. E non mi muovo. «Si può prendere il prossimo?», chiedo con aria distratta. «Forse sì. Ma perché?». Mentre contrattiamo, l’elicottero parte, e io ho risolto il mio problema. «Ma ti sembra il momento di mangiare...», sbraità l’Oriana. In quella, sulla pista di Dak To traforata dai mortai, atterra un secondo elicottero. Scarica i morti impacchettati come liquirizie in sacchi di plastica nera. Il pilota apre lo sportello. Ha un sorriso nervoso, quasi un ghigno: «Chi mena buono di voi? L’altro elicottero è andato in pezzi. Una mitragliata alle pale...». [69]

Then, the four audio clips are randomly ordered and the participants are asked to give a value on a 1–5 Likert scale to the following attributes of each voice:

1. Expressivity;
2. Facilitation to focus on the content;
3. Facilitation to understand the text;
4. Reading naturalness;

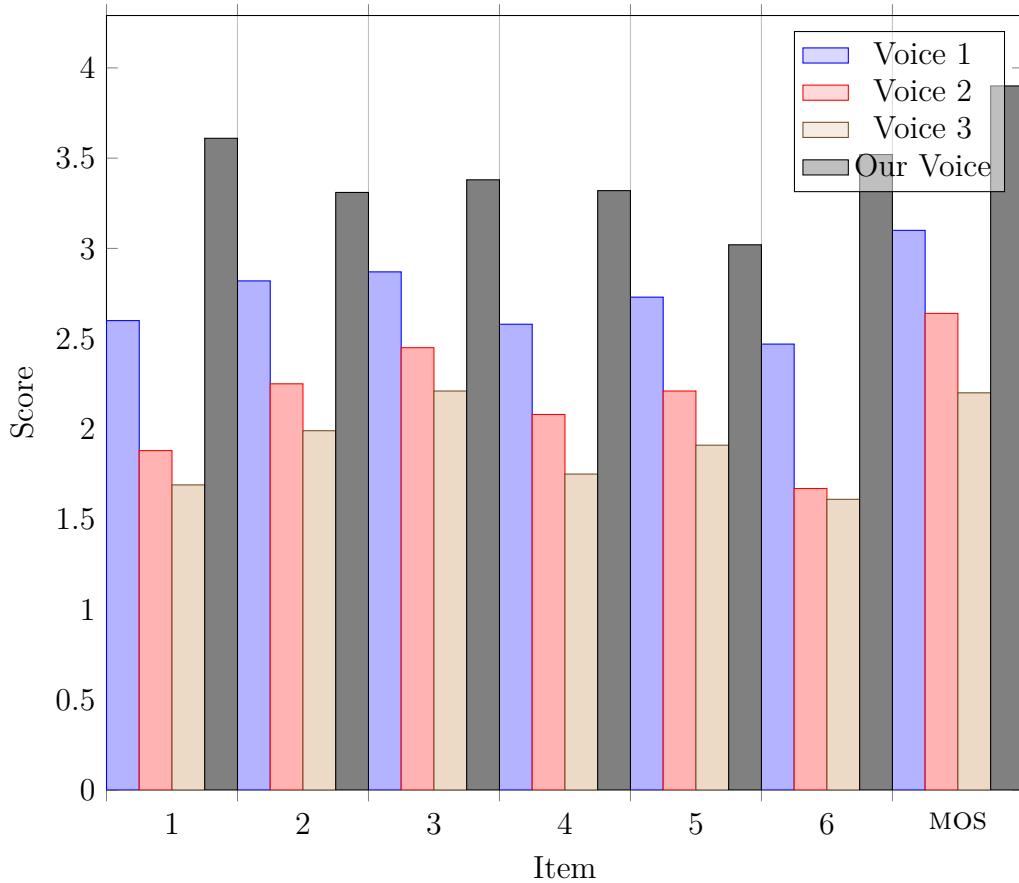


Figure 6.3: Questionnaire results. The column number corresponds to the questionnaire item.

5. Pause naturalness;
6. Presence of personality and/or emotion.

A final global score from 1 to 5, used to compute the mean opinion score (MOS), is requested as well. The aforementioned questions are chosen taking inspiration from previous TTS quality studies [70], [71], adapting them to the peculiarities of the present work. The results are reported in fig. 6.3.

The data show that our model decisively outperforms the other existing voices. This is particularly evident in items 1 and 6, corresponding to “expressivity” and “presence of personality and/or emotion”, but even for the other items, the better performance is clear. Even for the MOS, arguably

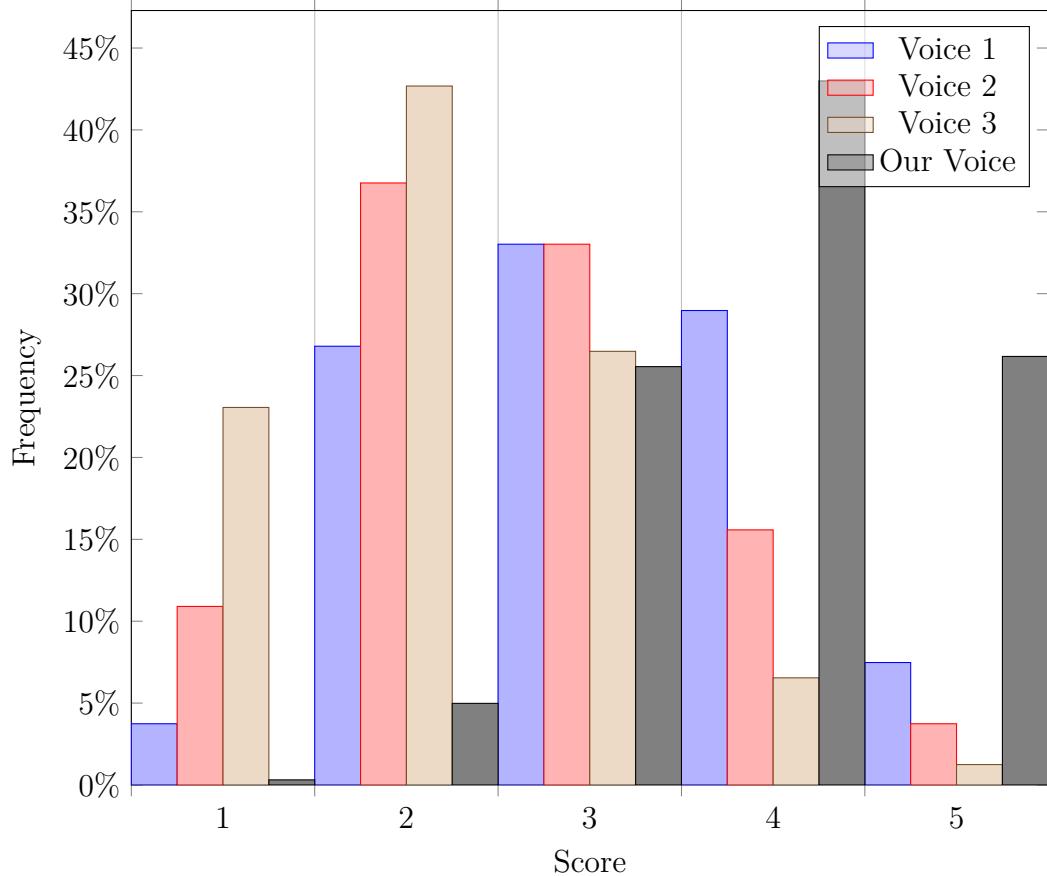


Figure 6.4: Questionnaire MOS distributions.

the most important metric, the advantage of our voice is glaring. This extreme difference in the MOS values is particularly blatant when looking at fig. 6.4, where the peak of the distribution of our voice is completely shifted to the right compared to the other three candidates. The questionnaire interface, together with the distributions for all the other items, can be found in appendices A and C.

Chapter 7

Conclusion

The aims of this thesis were mainly two. The first one was to create a robust Italian grapheme-to-phoneme converter, with special attention reserved for stress positioning. The second one was to exploit the phonemizer to increase expressivity in text-to-speech and apply it to voice cloning.

7.1 Phonemization

Regarding the phonemization step, the results can be considered absolutely satisfactory (table 7.1). The test mean CER of 0.0138 is very good even compared to state-of-the-art values [62], though it must be noted that the existing baselines (which are on average around 0.05) refer to English phonemization, which is by nature more difficult than the Italian one, being English a less phonetic language. An additional positive note can be derived by looking at the wrongly predicted words: most of the mistakes regard phonemes which even native speakers often get wrong, such as /'mets:e/ instead of /'medz:e/ or /'lenta/ instead of /'lenta/. When these mistakes are not considered, therefore treating as the same phoneme (as most Italian speakers would) open and closed vowels, /s/ and /z/, and /ts/ and /dʒ/, the test mean CER goes as low as 0.0080. This means that only 0.80 % of all the phonemes are wrongly predicted, a very impressive result.

Dataset	Mean CER	Standard deviation
Training	0.0025	0.0280
Validation	0.0021	0.0244
Test	0.0138	0.0629

Table 7.1: CER values. The discrepancy between the validation CER of the table and that of fig. 6.1 arises from the different computation method: during training, CER is computed globally on all phonemes (not grouped in words), while the values in this table are a mean of the values for each word.

Voice	1	2	3	4	5	6	MOS
Voice 1	2.60	2.82	2.87	2.58	2.73	2.47	3.10 ± 0.06
Voice 2	1.88	2.25	2.45	2.08	2.21	1.67	2.64 ± 0.06
Voice 3	1.69	1.99	2.21	1.75	1.91	1.61	2.20 ± 0.05
Our voice	3.61	3.31	3.38	3.32	3.02	3.52	3.90 ± 0.05

Table 7.2: Questionnaire results.

7.2 Human evaluation

The results of the human evaluation step are shown in table 7.2. Looking at the MOS values, our voice surpasses the second best by 0.80, quite a wide margin in a 1–5 Likert scale, especially when considering the standard errors of the means, which are an order of magnitude smaller than the difference between the two values. The results met and even exceeded our expectations: the most positive one is probably the “expressivity” score (item 1), which is not only the highest one across the various voices but is also the highest scoring item for our voice, a *unicum* compared to the other candidates. Therefore, the main aim of the work, which was to increase expressivity, can be considered fully achieved. The only relatively negative result regards item 5, “pause naturalness”: even though our model surpassed all the other voices, this value is quite low compared to the other items.

7.3 Future work

Regarding the phonemization step, there is probably not much room for improvement, since most of the wrongly predicted phonemes do not follow any sort of rule in Italian. Some fine-tuning on a dataset containing many words with the aforementioned phonemes might be considered.

For text-to-speech, there are two main ways future work might explore. The first one consists in improving pause naturalness, the item with the lowest score in the questionnaire: some specific training regarding the treatment of punctuation could be a solution to this, maybe paired with some rule-based processing of silences. The other area to investigate is the possibility to make voice cloning for new speakers easier: in this thesis, a huge dataset with paired audio and text was used, but this is not a viable solution for most speakers. A first step might be trying to re-use the weights of our main FastPitch training as a backbone and to fine-tune the model on other speakers, which requires much fewer data. Another approach could consist in exploiting an automatic speech recognition architecture to produce the written text for stand-alone audio clips, which are much easier to obtain for most speakers, especially for journalists. The usage of a custom end-to-end model, using raw audio of new speakers as the sole input to fine-tune FastPitch and HiFi-GAN and achieve voice-cloning, would also be interesting.

Appendix A

Interface

Questionario tesi Text-to-Speech

Il presente questionario è parte di una tesi sul Text-to-Speech in italiano (sintesi vocale artificiale partendo da un testo scritto). Ai partecipanti sarà chiesto di valutare la qualità di 4 voci differenti.

Saranno presentati 4 audio contenenti lo stesso articolo letto dalle diverse voci. Al termine di ogni audio, che durerà circa 1 minuto, verrà chiesto di valutare 6 affermazioni e di giudicare la qualità complessiva della voce.

Le voci potrebbero non essere in ordine (ad es. la voce 3 potrebbe essere presentata prima della voce 2).

In caso di compilazione da smartphone potrebbe essere necessario scorrere orizzontalmente per visualizzare tutti i valori delle scale.

Si raccomanda l'utilizzo di un paio di cuffie.

Grazie per la collaborazione!

Per info: martinomare.pulici@studenti.unibo.it

[Accedi a Google per salvare i risultati raggiunti. Scopri di più](#)

***Campo obbligatorio**

Figure A.1: The questionnaire introduction.

Attribuire un punteggio da 1 (pessimo) a 5 (ottimo) alle seguenti caratteristiche della voce *

	1	2	3	4	5
Naturalezza delle pause	<input type="radio"/>				
Espressività	<input type="radio"/>				
Aiuta a comprendere il testo	<input type="radio"/>				
Aiuta a concentrarsi sul contenuto del testo	<input type="radio"/>				
Presenza di personalità e/o emozione	<input type="radio"/>				
Naturalezza della lettura	<input type="radio"/>				

Che punteggio assegneresti alla qualità complessiva della voce (tenendo conto della sua natura sintetica)? *

	1	2	3	4	5
Pessima	<input type="radio"/>				
Ottima					

Figure A.2: The questionnaire interface.

Appendix B

Pie charts

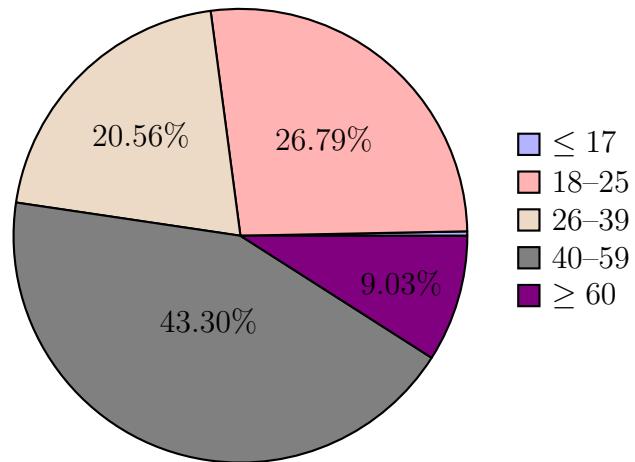


Figure B.1: Partecipants age.

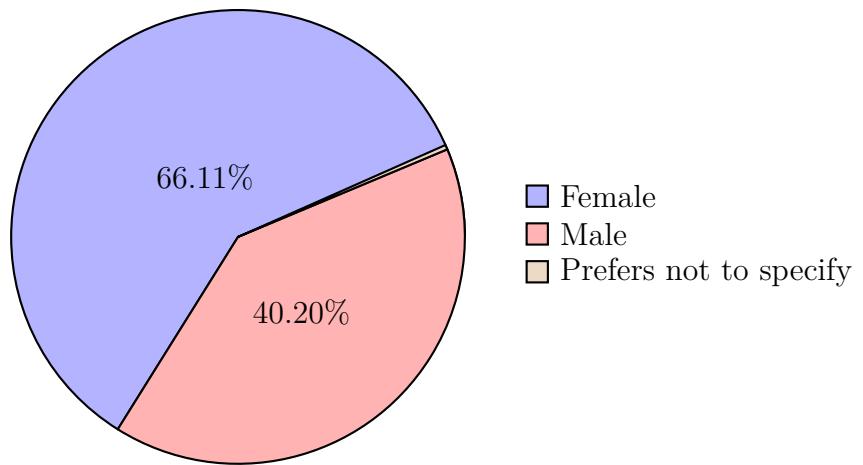


Figure B.2: Partecipants biological sex.

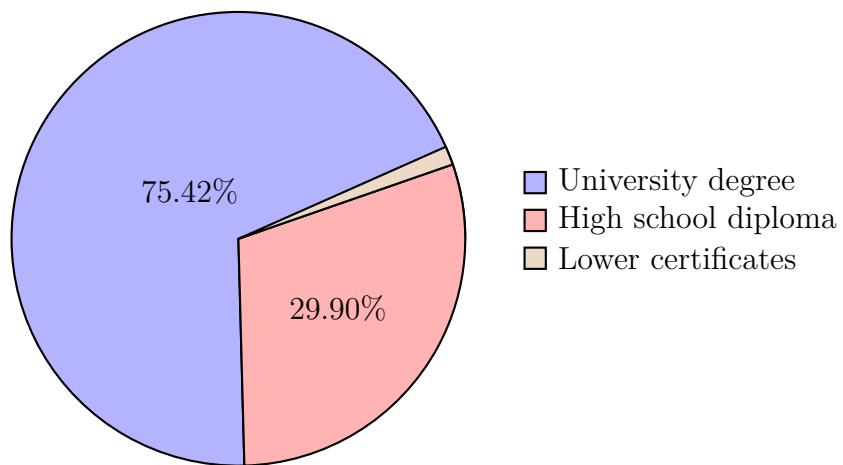


Figure B.3: Partecipants educational level.

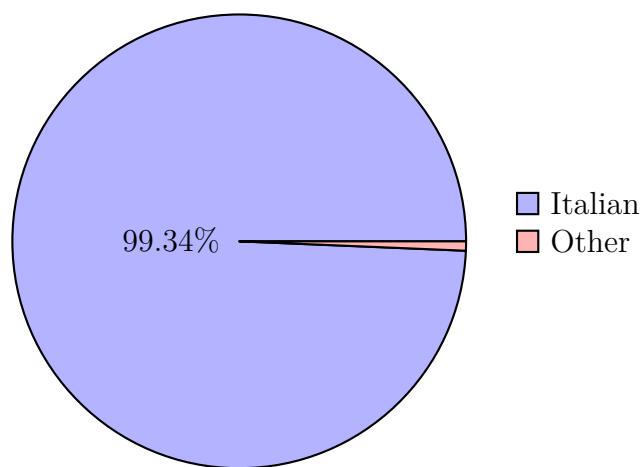


Figure B.4: Partecipants native language.

Appendix C

Items distributions

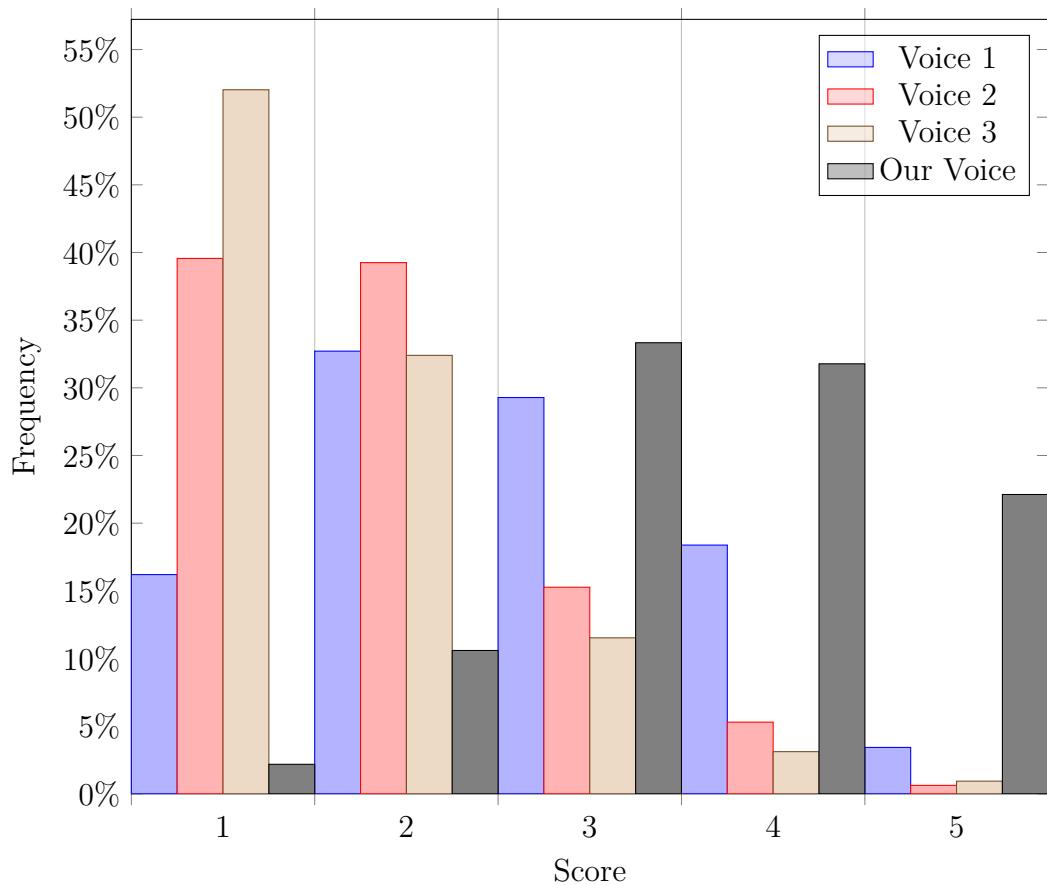


Figure C.1: Expressivity distributions.

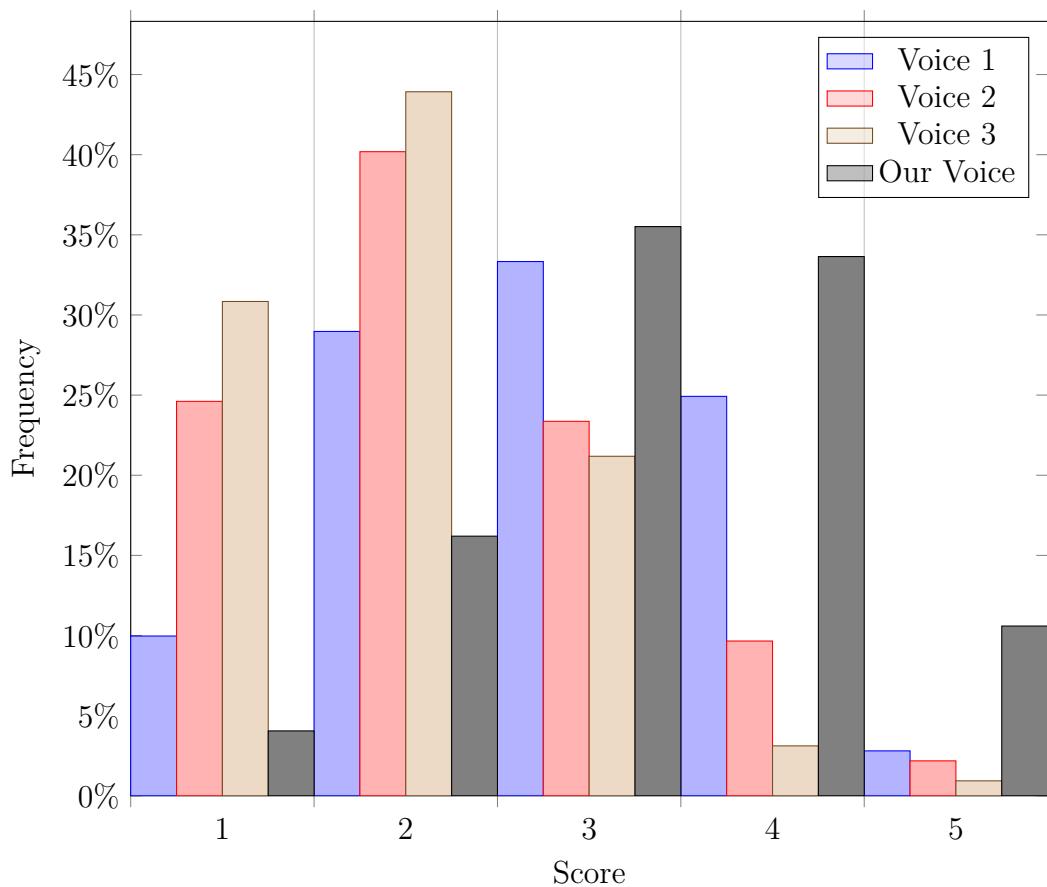


Figure C.2: Facilitation to focus on the content distributions.

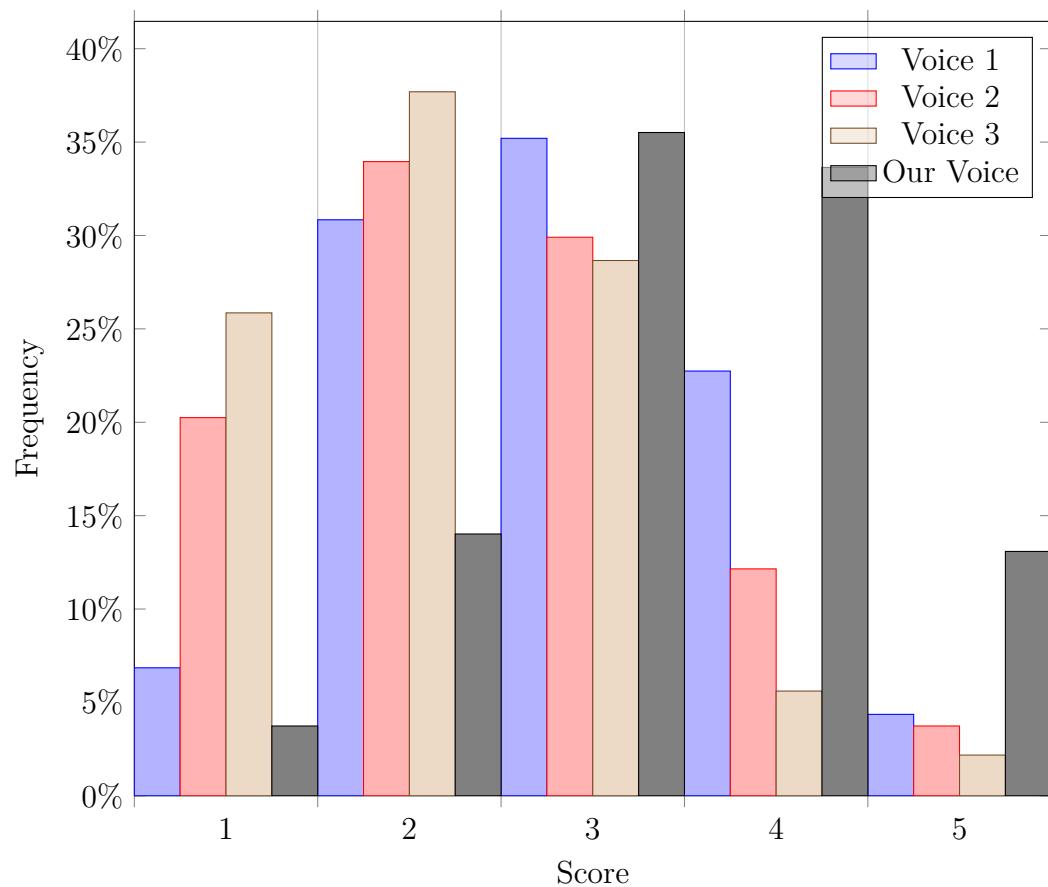


Figure C.3: Facilitation to understand the text distributions.

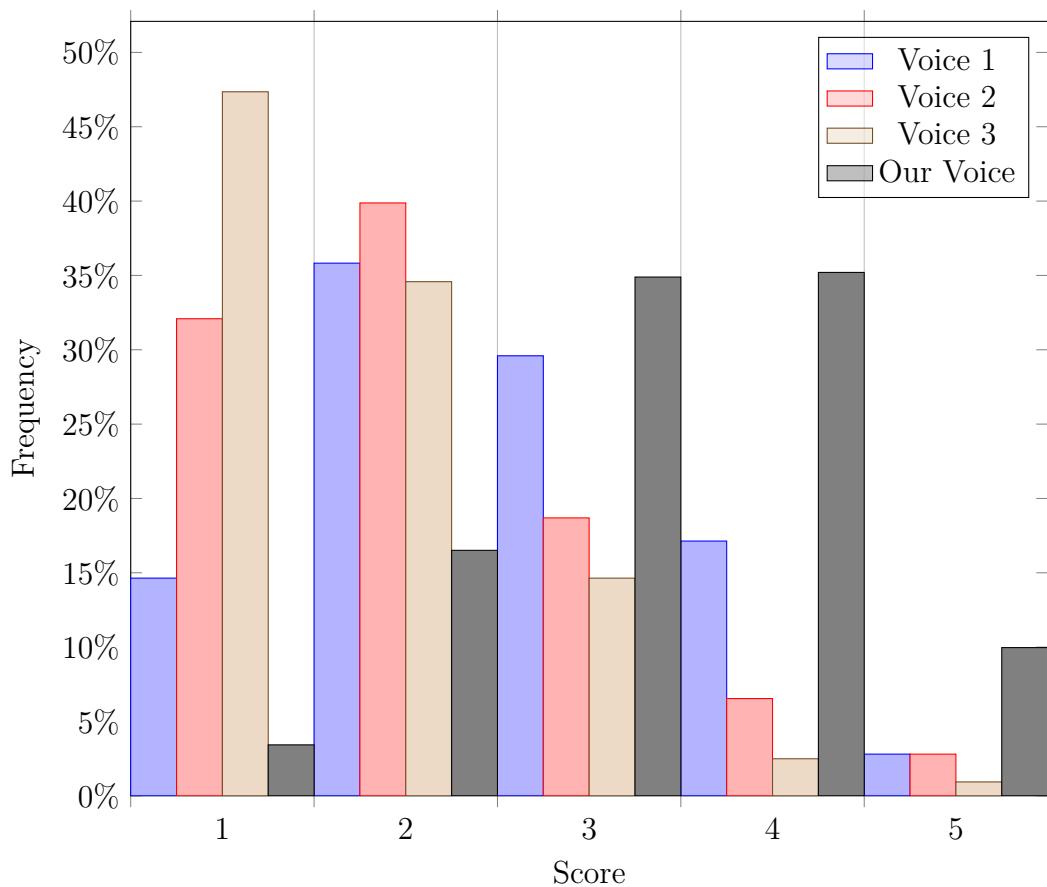


Figure C.4: Reading naturalness distributions.

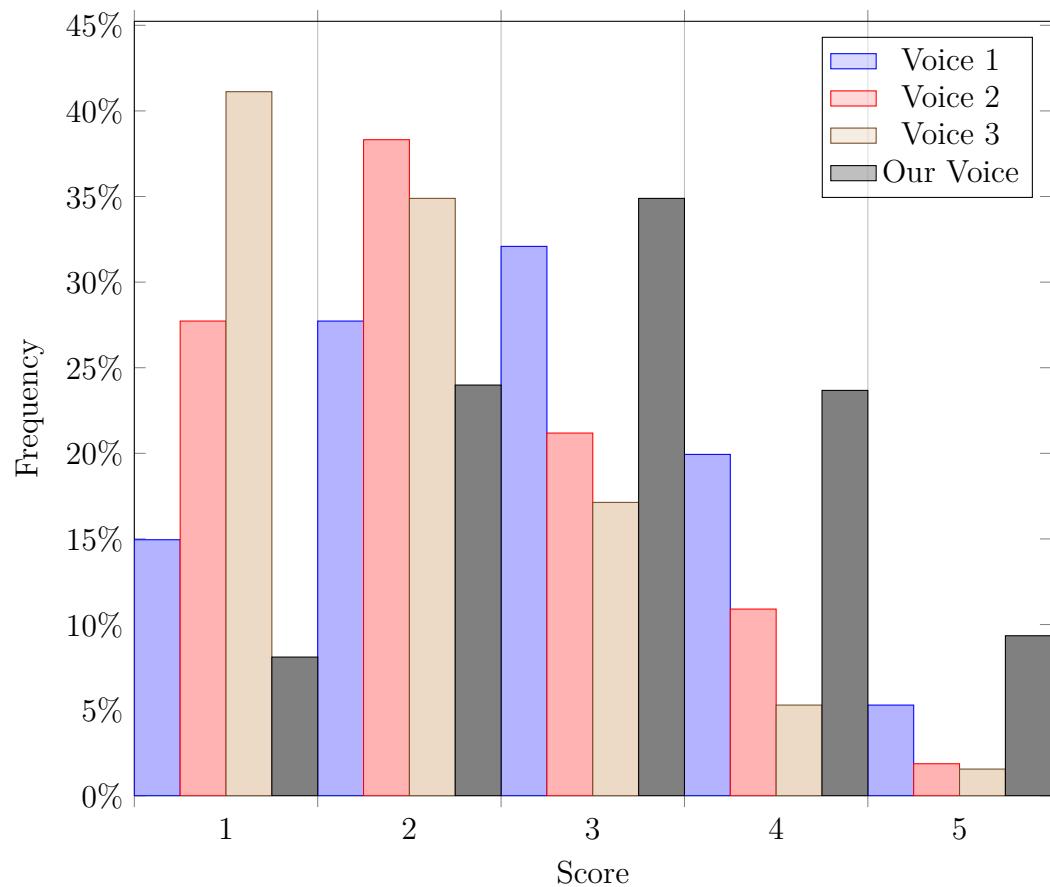


Figure C.5: Pause naturalness distributions.

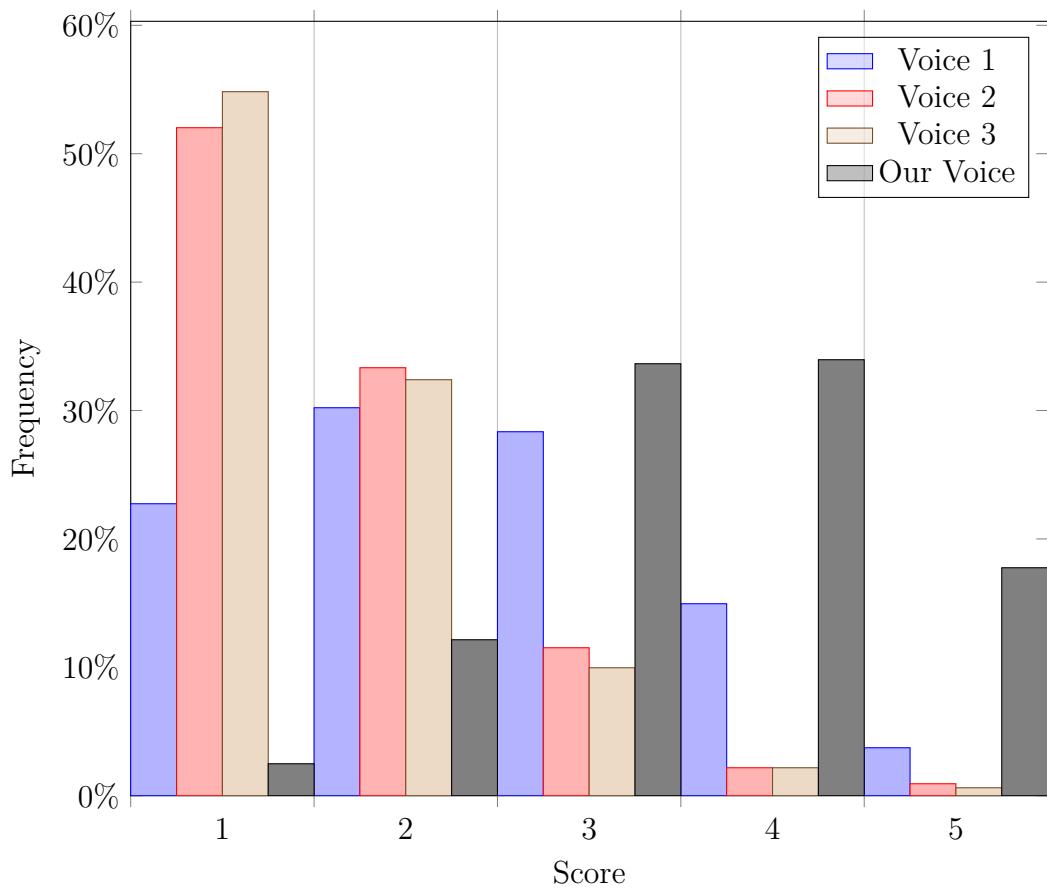


Figure C.6: Presence of personality and/or emotion distributions.

Bibliography

- [1] P. Taylor, “The text-to-speech problem”, in *Text-to-Speech Synthesis*. Cambridge, United Kingdom: Cambridge University Press, 2009, ch. 3, pp. 26–51.
- [2] D. Schmandt-Besserat, “An archaic recording system and the origin of writing”, *Syro-Mesopotamian Stud.*, vol. 1, no. 1, pp. 1–32, May 1977.
- [3] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, “A survey on neural speech synthesis”, Jun. 2021. arXiv: 2106.15561 [eess.AS].
- [4] C. H. Coker, “A model of articulatory dynamics and control”, *Proc. IEEE*, vol. 64, no. 4, pp. 452–460, Apr. 1976.
- [5] C. H. Shadle and R. I. Damper, “Prospects for articulatory synthesis: A position paper”, presented at the 4th ISCA Tut. and Res. Workshop Speech Synthesis, Perthshire, United Kingdom, Aug. 29–Sep. 1, 2001.
- [6] J. Allen, S. Hunnicutt, R. Carlson, and B. Granstrom, “MITalk-79: The 1979 MIT text-to-speech system”, *J. Acoust. Soc. Amer.*, vol. 65, no. S1, p. 130, Jun. 1979.
- [7] E. Battenberg, R. Skerry-Ryan, S. Mariooryad, *et al.*, “Location-relative attention mechanisms for robust long-form speech synthesis”, in *Proc. 45th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Barcelona, Spain, May 4–8, 2020, pp. 6194–6198.
- [8] P. Seeviour, J. Holmes, and M. Judd, “Automatic generation of control signals for a parallel formant speech synthesizer”, in *Proc. 1st IEEE Int. Conf. Acoust., Speech and Signal Process.*, Philadelphia, PA, USA, Apr. 12–14, 1976, pp. 690–693.

- [9] J. Olive, “Rule synthesis of speech from dyadic units”, in *Proc. 2nd IEEE Int. Conf. Acoust., Speech and Signal Process.*, vol. 2, Hartford, CT, USA, May 9–11, 1977, pp. 568–570.
- [10] E. Moulines and F. Charpentier, “Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones”, *Speech Commun.*, vol. 5, no. 5, pp. 453–467, Dec. 1990.
- [11] Y. Sagisaka, N. Kaiki, N. Iwahashi, and K. Mimura, “ATR μ -talk speech synthesis system”, in *Proc. 2nd Int. Conf. Spoken Lang. Process.*, Banf, Canada, Oct. 13–16, 1992, pp. 483–486.
- [12] A. J. Hunt and A. W. Black, “Unit selection in a concatenative speech synthesis system using a large speech database”, in *Proc. 21st IEEE Int. Conf. Acoust., Speech and Signal Process.*, vol. 1, Atlanta, GA, USA, May 7–10, 1996, pp. 373–376.
- [13] P. Taylor, A. W. Black, and R. Caley, “The architecture of the festival speech synthesis system”, in *Proc. 3rd ESCA/COCOSDA Workshop Speech Synthesis*, Canterbury, Australia, Nov. 26–29, 1998, pp. 147–152.
- [14] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, “Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis”, presented at the 6th European Conf. Speech Communication Technology, Budapest, Hungary, Sep. 5–9, 1999.
- [15] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, “Speech parameter generation algorithms for HMM-based speech synthesis”, in *Proc. 25th IEEE Int. Conf. Acoust., Speech and Signal Process.*, vol. 3, Istanbul, Turkey, Jun. 5–9, 2000, pp. 1315–1318.
- [16] T. Yoshimura, “Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for HMM-based text-to-speech systems”, Ph.D. dissertation, Dept. Elect. Comp. Eng., Nagoya Institute of Technology, Nagoya, Japan, 2002.

- [17] A. W. Black, H. Zen, and K. Tokuda, “Statistical parametric speech synthesis”, in *Proc. 32nd IEEE Int. Conf. Acoust., Speech and Signal Process.*, vol. 4, Honolulu, HI, USA, Apr. 15–20, 2007, pp. 1229–1232.
- [18] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, “Speech synthesis based on hidden markov models”, *Proc. IEEE*, vol. 101, no. 5, pp. 1234–1252, Jan. 2013.
- [19] H. Zen, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks”, in *Proc. 38th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Taipei, Taiwan, Jan. 15–16, 2013, pp. 7962–7966.
- [20] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, “On the training aspects of deep neural network (DNN) for parametric TTS synthesis”, in *Proc. 39th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Singapore, May 20–21, 2014, pp. 3829–3833.
- [21] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, “TTS synthesis with bidirectional LSTM based recurrent neural networks”, in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, Singapore, Sep. 14–18, 2014, pp. 1964–1968.
- [22] H. Zen and H. Sak, “Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis”, in *Proc. 40th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Brisbane, Australia, Apr. 19–24, 2015, pp. 4470–4474.
- [23] W. Wang, S. Xu, and B. Xu, “First step towards end-to-end parametric TTS synthesis: Generating spectral parameters with neural attention”, in *Proc. 19th ISCA Tut. and Res. Workshop Speech Synthesis*, San Francisco, CA, USA, Sep. 8–12, 2016, pp. 2243–2247.
- [24] J. Zhang, J. Pan, X. Yin, *et al.*, “A hybrid text normalization system using multi-head self-attention for Mandarin”, in *Proc. 45th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Barcelona, Spain, May 4–8, 2020, pp. 6694–6698.

- [25] K. Silverman, M. Beckman, J. Pitrelli, *et al.*, “Tobi: A standard for labeling English prosody”, in *Proc. 2nd Int. Conf. Spoken Lang. Process.*, Banf, Canada, Oct. 13–16, 1992, pp. 867–870.
- [26] Y. Wang, R. Skerry-Ryan, D. Stanto, *et al.*, “Tacotron: Towards end-to-end speech synthesis”, in *Proc. 20th ISCA Tut. and Res. Workshop Speech Synthesis*, Stockholm, Sweden, Aug. 20–24, 2017, pp. 4006–4010.
- [27] D. Griffin and J. Lim, “Signal estimation from modified short-time Fourier transform”, *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 32, no. 2, pp. 263–243, Apr. 1984.
- [28] P. Liu, X. Wu, S. Kang, G. Li, D. Su, and D. Yu, “Maximizing mutual information for Tacotron”, Aug. 2019. arXiv: 1909.01145 [eess.AS].
- [29] J. Shen, R. Pang, R. J. Weiss, *et al.*, “Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions”, in *Proc. 43rd IEEE Int. Conf. Acoust., Speech and Signal Process.*, Calgary, Canada, Apr. 15–20, 2018, pp. 4779–4783.
- [30] S. Ö. Arik, M. Chrzanowski, A. Coates, *et al.*, “Deep Voice: Real-time neural text-to-speech”, in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, Australia, Aug. 6–11, 2017, pp. 195–204.
- [31] S. Ö. Arik, G. Diamos, A. Gibiansky, *et al.*, “Deep Voice 2: Multi-speaker neural text-to-speech”, presented at the 31st Conf. Neural Inf. Process. Syst., Long Beach, CA, USA, Dec. 4–9, 2017.
- [32] W. Ping, K. Peng, A. Gibiansky, *et al.*, “Deep Voice 3: Scaling text-to-speech with convolutional sequence learning”, presented at the 6th Int. Conf. Learn. Representations, Vancouver, Canada, Apr. 30–May 3, 2018.
- [33] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, “Neural speech synthesis with transformer network”, in *Proc. 33rd AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, Jan. 27–Feb. 1, 2019, pp. 6706–6713.
- [34] M. Chen, X. Tan, Y. Ren, *et al.*, “MultiSpeech: Multi-speaker text to speech with transformer”, in *Proc. 23rd ISCA Tut. and Res. Workshop Speech Synthesis*, Shanghai, China, Oct. 25–29, 2020, pp. 4024–4028.

- [35] N. Li, Y. Liu, Y. Wu, S. Liu, S. Zhao, and M. Liu, “RobuTrans: A robust transformer-based text-to-speech model”, in *Proc. 34th AAAI Conf. Artif. Intell.*, New York, NY, USA, Feb. 7–12, 2020, pp. 8228–8235.
- [36] Y. Ren, Y. Ruan, X. Tan, *et al.*, “FastSpeech: Fast, robust and controllable text-to-speech”, presented at the 33rd Conf. Neural Inf. Process. Syst., Vancouver, Canada, Dec. 8–14, 2019.
- [37] A. Łańcucki, “FastPitch: Parallel text-to-speech with pitch prediction”, Jun. 2020. arXiv: 2006.06873 [eess.AS].
- [38] A. van den Oord, S. Dieleman, H. Zen, *et al.*, “WaveNet: A generative model for raw audio”, Sep. 2016. arXiv: 1609.03499 [cs.SD].
- [39] A. van den Oord, Y. Li, I. Babuschkin, *et al.*, “Parallel WaveNet: Fast high-fidelity speech synthesis”, in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 10–15, 2018, pp. 3918–3926.
- [40] N. Kalchbrenner, E. Elsen, K. Simonyan, *et al.*, “Efficient neural audio synthesis”, in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 10–15, 2018, pp. 2410–2419.
- [41] R. Prenger, R. Valle, and B. Catanzaro, “WaveGlow: A flow-based generative network for speech synthesis”, in *Proc. 44th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Brighton, United Kingdom, May 12–17, 2019, pp. 3617–3621.
- [42] S. Kim, S.-G. Lee, J. Song, J. Kim, and S. Yoon, “FloWaveNet: A generative flow for raw audio”, in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, Jun. 9–15, 2019, pp. 3370–3378.
- [43] S. Mehri, K. Kumar, I. Gulrajani, *et al.*, “SampleRNN: An unconditional end-to-end neural audio generation model”, presented at the 5th Int. Conf. Learn. Representations, Toulon, France, Apr. 24–26, 2017.
- [44] J. M. R. Sotelo, S. Mehri, K. Kumar, *et al.*, “Char2Wav: End-to-end speech synthesis”, presented at the 5th Int. Conf. Learn. Representations, Toulon, France, Apr. 24–26, 2017.

- [45] Z.-R. Zhang, M. Chu, and E. Chang, “An efficient way to learn rules for grapheme-to-phoneme conversion in Chinese”, presented at the 3rd Int. Symp. Chin. Spoken Lang. Process., Taipei, Taiwan, Aug. 23–24, 2002.
- [46] D. Rezende and S. Mohamed, “Variational inference with normalizing flows”, in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, Jul. 6–11, 2015, pp. 1530–1538.
- [47] M. Bińkowski, J. Donahue, S. Dieleman, *et al.*, “High fidelity speech synthesis with adversarial networks”, presented at the 7th Int. Conf. Learn. Representation, New Orleans, LA, USA, May 6–9, 2019.
- [48] K. Kumar, R. Kumar, T. de Boissiere, *et al.*, “MelGAN: Generative adversarial networks for conditional waveform synthesis”, presented at the 33rd Conf. Neural Inf. Process. Syst., Vancouver, Canada, Dec. 8–14, 2019.
- [49] J. Kong, J. Kim, and J. Bae, “TTS synthesis with bidirectional LSTM based recurrent neural networks”, in *Proc. 33rd Advances Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 6–12, 2020, pp. 17 022–17 033.
- [50] J. Yang, J. Lee, Y. Kim, H. Cho, and I. Kim, “VocGAN: A high-fidelity real-time vocoder with a hierarchically-nested adversarial network”, in *Proc. 23rd ISCA Tut. and Res. Workshop Speech Synthesis*, Shanghai, China, Oct. 25–29, 2020, pp. 200–204.
- [51] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models”, presented at the 34th Conf. Neural Inf. Process. Syst., Dec. 6–12, 2020.
- [52] *The International Phonetic Alphabet and the IPA chart*, International Phonetic Association. [Online]. Available: <https://www.internationalphoneticassociation.org/content/ipa-chart> (visited on 11/25/2022).
- [53] *IPA home*, International Phonetic Association. [Online]. Available: <https://www.internationalphoneticassociation.org> (visited on 11/25/2022).

- [54] *The International Phonetic Alphabet (revised to 2020)*, International Phonetic Association, 2020. [Online]. Available: https://www.internationalphoneticassociation.org/IPAcharts/IPA_chart_orig/pdfs/IPA_Kiel_2020_full.pdf.
- [55] *Wikipedia:Vetrina*, Wikipedia. [Online]. Available: <https://it.wikipedia.org/wiki/Wikipedia:Vetrina> (visited on 11/25/2022).
- [56] *Dizionario inglese-italiano WordReference*, WordReference.com. [Online]. Available: <https://www.wordreference.com/iten/> (visited on 11/25/2022).
- [57] *aeneas: automagically synchronize audio and text*, ReadBeyond. [Online]. Available: <https://readbeyond.it/aeneas> (visited on 11/25/2022).
- [58] *jiaaro/pydub @ GitHub*. [Online]. Available: <http://pydub.com> (visited on 11/25/2022).
- [59] *FFmpeg Filters Documentation*, FFmpeg. [Online]. Available: <https://ffmpeg.org/ffmpeg-filters.html> (visited on 11/25/2022).
- [60] *NVIDIA NeMo*, NVIDIA Corporation. [Online]. Available: <https://developer.nvidia.com/nvidia-nemo> (visited on 11/25/2022).
- [61] K. Rao, F. Peng, H. Sak, and F. Beaufays, “Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks”, in *Proc. 40th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Brisbane, Australia, Apr. 19–24, 2015, pp. 4225–4229.
- [62] S. Yolchuyeva, G. Németh, and B. Gyires-Tóth, “Transformer based grapheme-to-phoneme conversion”, in *Proc. 22nd ISCA Tut. and Res. Workshop Speech Synthesis*, Graz, Austria, Sep. 15–19, 2019, pp. 2095–2099.
- [63] *DeepPhonemizer*, GitHub. [Online]. Available: <https://as-ideas.github.io/DeepPhonemizer> (visited on 11/25/2022).
- [64] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks”, in *Proc. 23rd Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 25–29, 2006, pp. 369–376.

- [65] *The LJ Speech Dataset*, FFmpeg, 2017. [Online]. Available: <https://keithito.com/LJ-Speech-Dataset>.
- [66] P. Boersma, “Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound”, in *Proc. IFA Congr.*, Amsterdam, Netherlands, 1993, pp. 97–110.
- [67] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization”, presented at the 3rd Int. Conf. Learn. Representation, San Diego, CA, USA, May 7–9, 2015.
- [68] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization”, presented at the 7th Int. Conf. Learn. Representation, New Orleans, LA, USA, May 6–9, 2019.
- [69] G. Moroldo, “Il Vietnam sulla mia pelle”, *L'Europeo*, no. 16, Apr. 1987.
- [70] W. Cardoso, G. Smith, and C. Fuentes, “Evaluating text-to-speech synthesizers”, in *Proc. 23rd Eur. Assoc. Comput. Assisted Lang. Learn. Conf.*, Padova, Italy, Aug. 26–29, 2015, pp. 108–113.
- [71] J. Cambre, J. Colnago, J. Maddock, J. Tsai, and J. Kaye, “Choice of voices: A large-scale evaluation of text-to-speech voice quality for long-form content”, presented at the 39th Conf. Human Factors Comput. Syst., Honolulu, HI, USA, Apr. 25–30, 2020.