



**School of
Engineering**

InIT Institute of Applied
Information Technology

Project work

Reviewing state of the art in voice generation

Authors

Bastian Aebischer
Ardi Jasari

Main supervisor

Martin Ochoa

Date

22.12.2023


Declaration of Originality

Project Work at the School of Engineering

By submitting this project work, the undersigned student confirm that this work is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the project work have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

Place	Date	 Name
Winterthur	22.12.2023	_____
Winterthur	22.12.2023	_____

Abstract

Voice generation through deep neural networks (DNN) has experienced the same advancements with the rise of public attention similar to other breakthroughs achieved by DNNs in fields like computer vision. Through voice conversion it is possible to synthesize a human-sounding voice based on short inputs from a targeted speaker. Does the increased quality of the end product make the difference between real speech and synthesized speech evanescent? We study multiple open-source implementations of DNN and compare them according to their features, ease of use, subjective quality, visual spectrograms and generation time. The generated products show a definitive improvements with the latest tools and their success in producing human-sounding voices. However our results also show that the speaker similarity to the target voice aspect is not improving in parallel. Generating a plausible imitation is possible, but not in the time frame that it can be used in an ongoing discussion. This paper concludes that the potential for voice spoofing attacks is higher with the use of DNNs and also presents possible defense options against such spoofing attacks.

Keywords: speech synthesis, text-to-speech, voice conversion, voice cloning, IT security

Contents

1	Introduction	1
1.1	Initial Situation	1
1.2	Objective of this Work	1
2	Theoretical Base	3
2.1	Text Analysis Module	3
2.2	Acoustic Model	3
2.3	Vocoder	4
2.3.1	UnivNet	5
2.3.2	WaveNet	5
2.4	Fully End-to-End TTS	5
3	Methods	7
3.1	Academic Research	7
3.2	Github	7
3.3	Output Evaluation	8
4	Results	10
4.1	TorToise	10
4.1.1	Ease of use	11
4.1.2	Generated speech	12
4.2	VALL-E (X)	13
4.2.1	Ease of use	14
4.2.2	Generated speech	14
4.3	Real-Time Voice Cloning	15
4.3.1	Ease of use:	16
4.3.2	Generated speech	16
5	Discussion	18
6	References	20
7	List of Figures	22
8	List of Tables	22
9	Glossary	23
10	Appendix	24
10.1	Official Assignment	24

1 Introduction

1.1 Initial Situation

Modern speech synthesis started in the 1960s, it began with large databases of recorded speech that were cut into words and afterwards stitched back together during synthesis. String of words were then reduced down to individual diphones, which minimized the size of the databases significantly. In later years there was a statistical approach using a hidden Markov model (HMM), where waveforms are formed based on probability. With the rapid expansion in the field of DNN in the past years, voice generation based on artificial intelligence (AI) improved immensely as well [1].

More and more successful AI voice scams have been reported undeniably because of the increasingly realistic spoofed voice attacks [2]. Impersonation attacks over phone calls can become even easier and identity verification protocols based on voice id will soon be unfeasible. This puts great importance on raising awareness of the capabilities of such attacks and technologies to further improve possible detection or defense strategies and if these render futile, the deprecation of such security systems [3].

1.2 Objective of this Work

The research question and objectives of this work are based on the official assignment, which is attached in the appendix in 10.1.

The primary research question guiding this thesis is: What is the current state of the art in voice generation, with a particular focus on the quality, ease of use, the potential limitations and risks of open source software implementations?

To address the research question effectively, the following objectives have been defined:

- Review state of the art for voice generation techniques:
 - Conduct an extensive search of literature on voice generation, focusing on recent developments in the field.
 - Systematically classify identified voice generation techniques based on their underlying technologies.
- Select and evaluate most recent open source implementations:
 - Compare the different open source implementations subjectively through qualitative tests.
 - Compare the most recent software against an outdated solution highlighting improvements in recent implementations.
- Analyze the potential limitations and risks:
 - Subjectively assess usability and ease of use, considering installation, intuitiveness of the

user interface and accessibility of documentation.

- Examine robustness and stability of the implementations, by testing it under potential stress conditions, such as low-quality audio of the voice to be cloned.

2 Theoretical Base

This section is intended to describe the key components that usually compose a text-to-speech (TTS) system. This section does not go into the details of the individual components, but rather provides the basics for understanding the later sections. For more in-depth details, the references cited in this section can be consulted.

A TTS system normally consists of three components: a text analysis module, an acoustic model and a vocoder.

2.1 Text Analysis Module

The text analysis module processes the input text to extract linguistic features. Traditionally in statistical parametric speech synthesis (SPSS) this process involves text processing, phonetic analysis and prosodic analysis [4].

Text processing: In this phase the document structure (e.g. headings, paragraphs) is analysed to apply appropriate prosody, rhythm, and intonation to the synthesized speech [4]. Subsequently, the non-standard words in the text (e.g. dates) are converted from written form into speakable form with text normalization [4], [5]. Linguistic analysis then extracts semantic information from the text [4].

Phonetic analysis: In phonetic analysis polyphone disambiguation is applied to distinguish the different pronunciations of the same word in different contexts. Furthermore, graphemes (sequence of characters) are converted into phonemes (sequence of pronunciation symbols), which facilitates speech synthesis [4], [6].

Prosodic analysis: Prosodic analysis analyses the rhythm, stress, and intonation of speech. These features are important for achieving naturalness in synthesized speech [4].

In neural TTS the text analysis module is often simplified, retaining only essential steps such as text normalization and grapheme-to-phoneme conversion. This approach enables the models to take phoneme sequences directly as input for the synthesis. This is driven by the capacity of neural networks to capture internal patterns and relationships directly from raw input data [4].

2.2 Acoustic Model

Traditionally, in SPSS the acoustic model takes the linguistic features as input to generate acoustic features, such as mel cepstral coefficients (MCC), band-aperiodicity (BAP), and fundamental frequency (F0), that represent the frequency components of the speech signal [7], [8]. Techniques such as HMM have been employed to generate these acoustic features [9]. Even though HMM-based SPSS are more flexible in changing the voice characteristics compared to concatenative speech synthesis, the naturalness of synthesized speech is compromised due to the challenge of accurately modeling the complex acoustic features [10].

In neural TTS systems acoustic models are enhanced with DNNs. This eliminates the need for ex-

PLICIT alignments between linguistic and acoustic features, reducing pre-processing requirements. Furthermore, the representation of linguistic features simplifies to character or phoneme sequences, and acoustic features transition from low-dimensional representations to high-dimensional ones, such as mel-spectrograms or linear spectrograms [7].

Various model structures have been adopted in neural TTS acoustic models:

RNN-based models (e.g., Tacotron series): The Tacotron series leverages recurrent neural network (RNN), employing an encoder-attention-decoder framework, to generate linear-scale spectrograms from character sequences [7], [11]. Tacotron 2 uses the same acoustic model as in the original Tacotron to generate the mel-spectrograms. The main difference is in the vocoder utilized: Tacotron 2 utilizes a modified WaveNet vocoder, improving audio quality in comparison to the Griffin-Lim algorithm used in Tacotron [12].

CNN-based models (e.g., DeepVoice): DeepVoice integrates convolutional neural network (CNN) into the SPSS system, emphasizing the local dependencies of speech signals. Similarly to SPSS, DeepVoice predicts the F0 for the entire duration of phonemes, with the major difference that it employs CNNs to do so [13].

Transformer-based models (e.g., FastSpeech series): The previous models, which all use autoregressive generation, have limitations such as slow inference and word skipping [7]. The FastSpeech series introduces a feed-forward Transformer network that generates mel-spectrograms in parallel, speeding up training and inference, and improving robustness [14], [15].

Advanced generative models (GAN/Flow/VAE/Diffusion): To handle the distributional nature of mel-spectrogram data conditioned on phoneme sequences, advanced generative models, such as generative adversarial network (GAN), normalizing flow, variational auto-encoder (VAE), and diffusion model, are introduced. These models contribute to generating high-quality mel-spectrograms with fine-grained details. Diffusion models in TTS operate on the principle of enhancing the quality of the mel-spectrograms by iteratively removing noise. This denoising process necessitates a large number of iteration steps considerably slowing down inference speed [7].

2.3 Vocoder

The vocoder is the component of voice generation that takes the context from the outputs of text analysis modules and or from acoustic models, and synthesizes the actual audio, meaning waveforms that can be listened to on their preferred medium. In the focus of digital voice generation one does generally differentiate between vocoders from a signal-processing-based approach and vocoders from a neural-network-based approach [16].

The DNNs examined in this paper are using the following vocoders.

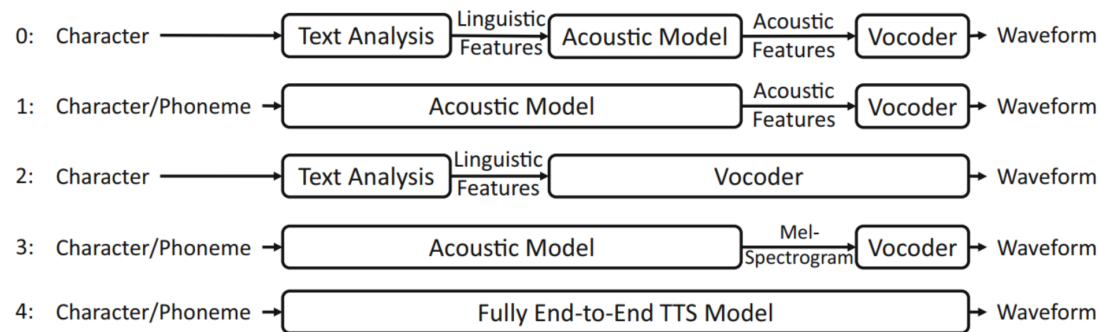
2.3.1 UnivNet

UnivNet is a neural vocoder developed in South Korea that synthesizes high-fidelity waveforms in real time. Instead of only using band-limited mel-spectrograms, which can produce over-smoothing problems; UnivNet accepts multiple spectrograms generated or real then applies a multi-resolution spectrogram discriminator. The main difference lies in the spectrograms used in UnivNet are chosen based on different spectral and temporal resolutions. Finally there is also a multi-period waveform discriminator [17].

2.3.2 WaveNet

WaveNet is a deep neural network developed by Google. It is an audio generative model based on the PixelCNN architecture that is fully probabilistic and auto-regressive. WaveNet combines causal filters with dilated convolutions to allow their receptive fields to grow exponentially with depth, which is important to model the long-range temporal dependencies in audio signals [18].

2.4 Fully End-to-End TTS



Stage	Models
0	SPSS [1, 2, 3, 4, 5]
1	ARST [6]
2	WaveNet [7], DeepVoice 1/2 [8, 9], Par. WaveNet [10], WaveRNN [11], HiFi-GAN [12]
3	DeepVoice 3 [13], Tacotron 2 [14], FastSpeech 1/2 [15, 16], WaveGlow [17], FloWaveNet [18]
4	Char2Wav [19], ClariNet [20], FastSpeech 2s [16], EATS [21], VITS [22], NaturalSpeech [23]

Figure 1: The progressively end-to-end process for TTS models[19]

Figure 1 provides an overview of the different modules that compose voice generation systems, and illustrates the components that perform similar functions. Even though the inputs are not the same for each process the overarching theme is still generally the same. Namely it starts from thought concepts that are written down and try to communicate a meaning within their context of application and it ends with the synthesizing of binary data to the finalized waveforms. Now comes the last process which incorporates all different modules into one model : a fully end-to-end TTS model.

An end-to-end system tries to optimize the different shortcomings of marrying multiple single components into one system. These are:

- Reducing human supervision of the feature alignments,
- Optimising of error propagation in cascaded models,
- Saving time and resources during training and deployment.

[19]

3 Methods

In this section, we describe the methods employed for the preliminary selection of different voice generation solutions and the refinement of this selection. The process started by identifying the key words used in the official assignment, see 10.1. In the description the key words are "generative AI", "voice generation / voice synthesis", "open source implementations" and "machine learning / deep neural networks". Additionally there is also a news report linked from CBS News [2], where CBS reports an uptick in elevated scams using cloned speech and also describes how with the new tools like VALL-E (X) a scammer only needs 3 seconds of the targets voice to create a copy that can then be used against potential victims.

3.1 Academic Research

With the guidance of our supervisor we started with multiple surveys in scientific papers and followed all the trails that promised us new knowledge. The most helpful were "Spoofing and countermeasures for speaker verification: A survey" by Zhizheng Wu et al. [20] and "A survey on voice assistant security: Attacks and countermeasures" by Chen Yan et al. [21].

There are three main approaches for AI voice imitation: Voice conversion models that need an audio sample from the target voice (the voice to be imitated) and an audio recording of the target speech (the sentences to be imitated); singing voice synthesis that work similar to a voice conversion model but are more fine-tuned for intonation used when singing; and lastly there is the TTS synthesis that needs a target voice sample and the target speech in text format [20].

3.2 Github

As one focus of this paper is to find out how easy it would be for an attacker to fake the voice of a potential victim, we concentrate on open source implementations. These are commonly found on <https://www.github.com>. First method was searching for keywords via a search engine, some of the keywords were: "AI voice generation", "voice synthesis", "text-to-speech", "voice conversion", etc.. Finding interesting repositories it is then to evaluate them based on objective criteria. For us these criteria were:

- **Actively maintained:** When was the latest feature merge? Is the project still in development or is it finished? How is the activity on open issues?
- **Amount of stars:** Staring is the github equivalent to liking or giving a thumbs up. It is a way of adding a repository (repo) to the users list or for showing their appreciation. Generally the more popular a repo is the more stars it has.
- **Amount of forks:** This metric can be similar to the amount of stars, but it also shows how a repos is established as a solution that it is trying to solve. The more forks a project has, the more it can be looked at as a standard for this specific solution.
- **Available training data:** Is the training data integrated in the repo? Are there links to tried and

tested training data?

- **Available pre-trained models:** Are there pre-trained models available for download? Is one expected to train the model by themselves?
- **Available languages:** Does it only support English? Does it only support Chinese? Does a repo support a wide range of languages?
- **Implementations of papers:** Are there any papers referenced in the repo?
- **Possible affiliation with companies/governments:** Is a repo linked to a paper that was published with the special funding of companies or governments? Is it part of a business solution that a company is offering in exchange for money?

After finding good candidates most often the repo are well maintained and with that, they usually are correctly categorized into the appropriate topics. The connected topics are thereupon a great follow-up, if the repo was not good enough although the topic shows promise. We identified the following topics: "speech", "text-to-speech", "deep-learning", "speech-synthesis", "voice-synthesis", "melgan", "multi-speaker-tts", "voice-cloning", "tacotron" and "vocoder".

3.3 Output Evaluation

After our selection of DNN to test, we defined a system to compare the results of each model. For the target voice input and for the target speech input pangrams and phoneme pangrams were used. Pangrams are short sentences that include all the letters from the alphabet. Since our tests were concluded in English the language for the target voice was also English. The models were evaluated using voices of unseen speakers, i.e. voices that were not used during the training of the models.

These were :

1. **Pangram1:** The quick brown fox jumps over the lazy dog.
2. **Pangram2:** Sphinx of black quartz, judge my vow.
3. **Pangram3:** Are those shy Eurasian footwear, cowboy chaps, or jolly earth-moving headgear.
4. **Pangram4:** With tenure, Suzie'd have all the more leisure for yachting, but her publications are no good.

Pangram 3 and 4 are phoneme pangrams meaning that instead of every letter, every phoneme that is used in the language, is in the sentence. The phoneme pangrams especially are a very good indicator of a DNN that can generate speech from the ground up and is not over-trained on the most common words and phrases.

To properly compare the different chosen models a system of six different criteria were defined:

- I Subjective description of the audio outputs: For the reason that sound is complex to be perceived by humans, a subjective description has a higher impact than an objective calculation on the output audio.
- II Any negative trends: Does the output audio have obvious patterns that sound robot-y, echo-y or choppy? Does one hear a difference between a recorded voice and a generated voice?
- III Outcomes with emotional inputs: How does the model process audio inputs that are spoken with emotions? Does it have special settings?
- IV Visual comparison of mel-spectrogram: A comparison between the target speech recorded by the actual person and the generated voice.
- V Used settings of the model: Are there any settings within the model? Which settings produce better outcomes? Which settings produce indistinguishable outputs?
- VI Generation time: What is the generation time of the output audio after the finished training of the model?

4 Results

After applying the methods described in the previous section 3.2, three open source implementations were selected for evaluation: TorToise [22], VALL-E (X) [23] and Real-Time Voice Cloning [24]. All three offer pretrained models and support English synthesis. While TorToise and VALL-E (X) are actively maintained, Real-Time Voice Cloning, an outdated tool, was chosen for comparative analysis with the more recent implementations. In this section, we delve into each open-source implementation, starting with a comprehensive description of their respective model architectures. Following this, a thorough assessment of usability and ease of use is conducted. Conclusively, the models are evaluated as described in 3.3. All synthesis tasks were performed on an NVIDIA GeForce MX350 with 2GB VRAM.

In table 1 the results and characteristics of the three open-source implementations are summarized.

	TorToise	VALL-E (X)	Real-Time Voice Cloning
Intermediate representation	mel-spectrograms	acoustic tokens	mel-spectrograms
Training data (only English)	49'896 h	704 h	2960 h
Long text synthesis	✓	✓	✓
Quality selection	✓	✗	✗
Prompt engineering for emotions	✓	✗	✗
Cross-lingual synthesis	✗	✓	✗
Synthesis time (average)	110 minutes	170 seconds	10 seconds
Naturalness of output	high	medium	low
Similarity to target voice	medium	medium	low
Expressiveness of output	low-medium	medium	low

Table 1: Comparison of the three open-source implementations: TorToise, VALL-E (X) and Real-Time Voice Cloning

4.1 TorToise

The author of TorToise [25] states that TorToise is a TTS System that combines an auto-regressive decoder with a denoising diffusion probabilistic model (DDPM), both often used in image generation models. The auto-regressive decoder utilizes a GPT-2 architecture for generating speech tokens based on the target text and on audio clips of the target speaker. The DDPM is used as the acoustic model to generate mel-spectrograms and Univnet as the vocoder.

During inference the auto-regressive decoder generates a large number of speech tokens. The highest quality speech token is then selected using contrastive language-voice pretrained transformer (CLVP), a model similar to CLIP from DALL-E. The selected speech tokens are then fed into the DDPM and finally Univnet generates the audio waveforms.

TorToise is trained on a dataset comprising LibriTTS, HiFiTTS, which combined give 896 hours of transcribed speech, and an extended dataset of 49'000 hours of cleaned audio from audiobooks and podcasts, transcribed with a wav2vec2-large model.

The author of TorToise further highlights the extreme slowness of the tool compared to other TTS systems, attributed to the use of both an auto-regressive decoder and a diffusion decoder. Furthermore, driven by ethical concerns about potential misuse of the voice-cloning text-to-speech system, he created Tortoise-detect, a classifier model to detect if a speech was generated by TorToise. For the same reason he refrains from releasing training configuration details of TorToise [22].

4.1.1 Ease of use

Installation: To utilize the latest version of TorToise, users can either install it locally or access an official live demo hosted on Hugging Face Spaces [26]. The demo hosted on Hugging Face Spaces does not offer certain features, such as cloning a custom voice and selecting presets, which affect the quality of the generated speech. An alternative demo of TorToise hosted on Replicate [27] include these features but does not have the latest TorToise version. Therefore the tool was installed locally in a conda environment following the repository README [22].

Main features: The key features of TorToise are:

- Generation of a sentence or a large text file with one or more voices separately.
- Combination of multiple voices when generating a synthesized voice.
- Use of four different presets (*ultra_fast*, *fast*, *standard*, *high_quality*). The presets differ in the number of speech tokens generated by the auto-regressive decoder, and in the number of iterations of the DDPM. These factors affect the quality of the generated speech and the synthesis time.
- Selection of the number of output candidates to generate.
- Prompt engineering to generate speech with emotions.

User interface: The locally installed tool does not offer a graphical user interface (GUI) and can only be used via terminal instructions.

Accessibility of documentation: Documentation on how to use the model is available on the repository [22]. The README provides well-documented installation instructions. Although the author of TorToise specifies the need for an NVIDIA GPU, our tests with CPUs produced comparable results, albeit with longer synthesis times. The documentation describes how to add custom voices. The terminal commands for the generation of speech are listed in the documentation. However, additional command arguments, such as presets selection and the number of output candidates, are not extensively detailed and require the consulting of the source code for more information. In the documentation, a brief section is dedicated to utilizing prompt engineering for emotions. However, a detailed list of all supported emotions is absent, apart from a few examples provided by the author.

4.1.2 Generated speech

All generated speeches were produced using the *high_quality* preset. In each synthesis three output candidates were generated and the optimal result was manually selected for evaluation.

When using high-quality monotone audio as target voice, the generated speeches exhibit a natural sounding voice with both male and female voices. Although the voices of the generated speeches bear a resemblance to the target voices, they are distinguishable upon attentive listening. Notably, the monotonous tone of the original voice is altered, adopting a book reading style with increased emphasis on certain words. This book reading style is less pronounced with the female voice. Some of the generated speeches manifest a British accent, a feature not present in the reference voice.

Speeches generated using telephone-quality audio as reference similarly retain the book reading style. Moreover, the audio sounds more distorted than the reference audio, significantly diminishing the voice similarity to the target voice.

Attempts to generate speech with an angry emotion yielded mixed results. Firstly, when using an angry-voiced audio as reference, the original voice was not preserved very well in the output. The voice did not sound similar to that of the target and the emotion of anger was not as evident as in the original audio. Alternatively, employing prompt engineering with a neutral-voiced reference, and appending "[I am so angry,]" to the text, produced a more faithful reproduction of the original voice. However, there was not the intended emotion of anger.

Using a longer audio as reference didn't affect the quality and naturalness of the generated speech.

The synthesis of one sentence with the preset *high_quality* takes on average about 110 minutes, going from 66 minutes for the shortest sentence (pangram2) to 160 minutes for the longest sentence (pangram4).

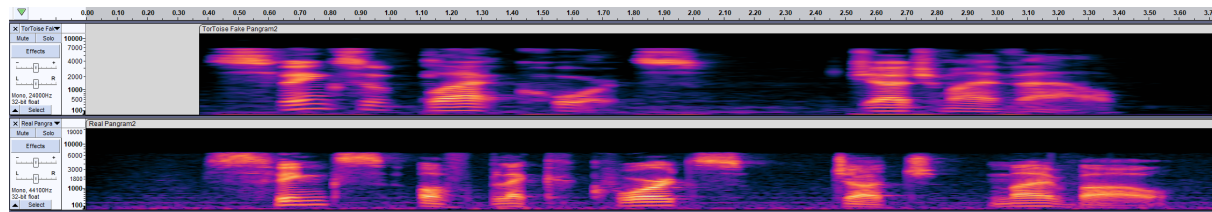


Figure 2: Mel-spectrogram of the real(bottom) and the synthesized voice(top) of pangram2

In Figure 2 the mel-spectrograms of the real voice and of the synthesized voice, both uttering pangram2, can be seen. The synthesized voice was generated with pangram1 as input. From the mel-spectrogram one can see that the synthesized voice begins the sentence without any initial silence. One can visualize the phonemes pronounced. One can visually map each phoneme from the top to the bottom, some pauses are different however over the grand scheme of things these two mel-spectrograms are related to each other. The main difference lies in the different scales of the y-axis, the synthesized voice features a higher ceiling and also is pitched higher than the original voice throughout. This is also validated while

listening to the samples.

4.2 VALL-E (X)

VALL-E [28] is a recent TTS framework introduced by Microsoft, that differs from previous TTS frameworks in its distinctive approach to synthesizing speech. Instead of treating TTS as a continuous signal regression, where mel-spectrograms are generated and used as an intermediate representation, VALL-E treats TTS as a conditional language modeling task, generating acoustic tokens.

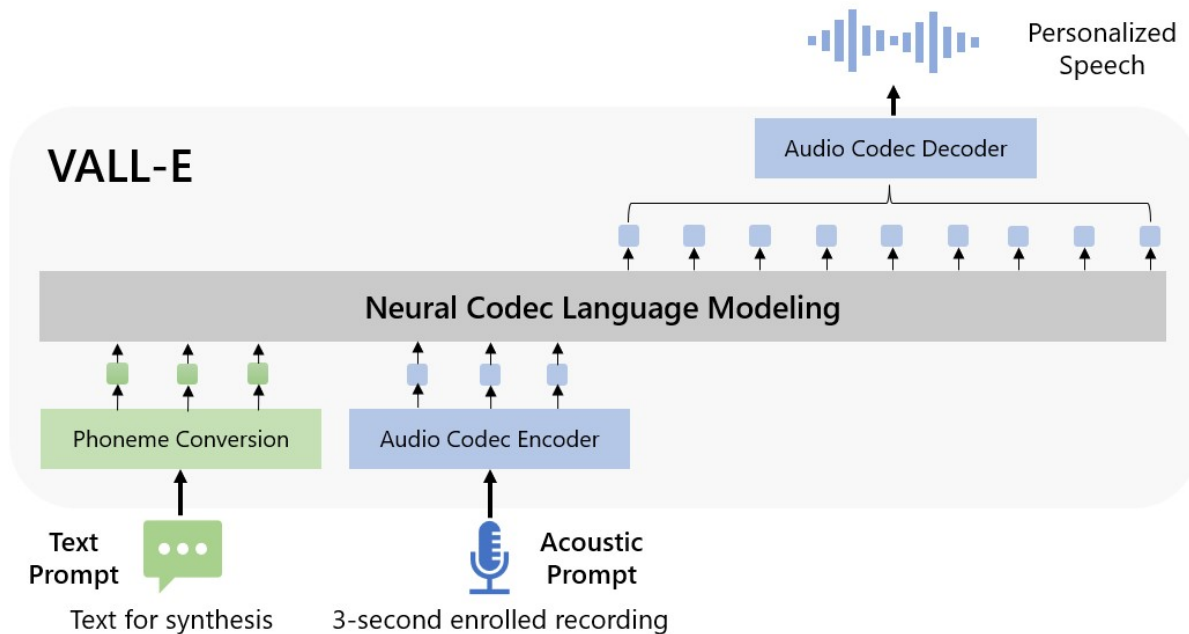


Figure 3: Inference process of VALL-E [28]

Figure 3 shows the inference process of VALL-E. The text to synthesize and a few seconds of speech of the target speaker are taken as input. The text input and the acoustic input are respectively converted into phonemes and acoustic tokens. Based on these phonemes and acoustic tokens, VALL-E generates additional acoustic tokens, which are fed into the audio codec decoder to synthesize personalized speech. The neural codec language model (the grey rectangle) comprises an auto-regressive Transformer and a non-auto-regressive Transformer. EnCodec [29] is used as the Audio Codec Encoder/Decoder (blue rectangles).

Although there is no official public implementation of VALL-E, several unofficial open-source implementations are available on GitHub [23], [30], [31]. To test the VALL-E framework this GitHub repository [23] was selected, primarily because it offers a pretrained model. It is essential to note that this particular repository implements VALL-E X [32], an extension of VALL-E that enables cross-lingual speech synthesis, i.e. speech synthesis in a language different from the source speaker's language. The inference process remains similar to VALL-E, the only difference being that the transcription of the input speech and the target language ID are additionally given as input. The author of the repository replaced the EnCodec decoder with a Vocos decoder [33], stating it improved the audio quality.

The pretrained model of the unofficial implementation was trained on 704 hours of English speech from LibriTTS and self-gathered audios, on 598 hours of Chinese speech from AISHELL-1, AISHELL3, Aidatatang and self-gathered audios, and on 437 hours of Japanese speech from Japanese Common Voice and self-gathered audios [34].

4.2.1 Ease of use

Installation: The author of the repository provide a demo hosted on Hugging Face Spaces [35]. Alternatively, users can follow the instructions in the README of the repository for local installation [23]. To benchmark synthesis time against the other open-source implementations, it was decided to install VALL-E X locally in a conda environment.

Main features: The key features of VALLE X are:

- Sentence or long text generation with a single voice.
- Cross-lingual speech synthesis.
- Accent selection of the generated speech.
- Downloading the encoded prompt, allowing the skipping of the encoding process in subsequent inferences with the same acoustic prompt.

User interface: This implementation can be used directly as code or through a GUI. The latter option streamlines the synthesis process for users unfamiliar with command-line terminals.

Accessibility of documentation: Comprehensive documentation for the tool is available in the repository [23]. It covers the use of the tool in Python code and details its features. In the documentation, the author refers to this repository [30] for the training code for users to train their own model.

4.2.2 Generated speech

All generated speeches were produced using the *no-accent* setting for the output.

Speeches generated with VALL-E X yielded mixed results. Notably, VALL-E X introduced a breathing sound at the start of some outputs, enhancing the realism of the speech, even if the reference audio lacked this characteristic. However, VALL-E X struggled in generating pangram3 and pangram4, exhibiting instances of slurred articulation and mispronouncing words, leading to an unnatural delivery. The audio quality, while generally good, did not match the crispness observed with TorToise, and some outputs exhibited a hoarse quality in the male voice. The female voice displayed a higher level of similarity to its reference voice than the male voice, but noticeable differences are apparent to someone familiar with the original speaker.

When using telephone-quality audio as a reference, the output speech took on a more robotic quality, but still performed better than TorToise.

Generating speech with an angry tone, using an angry-voiced reference audio resulted in a more dissimilar output voice compared to TorToise. However, VALL-E X performed better in reproducing the emotion of anger.

Using a longer audio as a reference, in some cases, negatively impacted the naturalness of the generated speech.

The synthesis time of one sentence averages about 170 seconds, ranging from 90 seconds for the shortest sentence (pangram2) to 220 seconds for the longest sentence (pangram4). In contrast to TorToise, in VALL-E X at times, the initial synthesis may fail to produce an optimal output, requiring multiple attempts (typically around three) to achieve a satisfactory result.

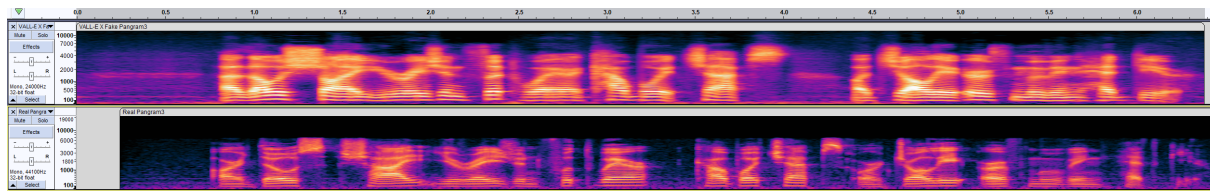


Figure 4: Mel-spectrogram of the real(bottom) and the synthesized voice(top) of pangram3

Figure 4 shows the comparison with pangram3, here the different auditory effects of each voice sample are visually apparent. The real voice has clear pronunciation, which can be seen by the absence of signals between each word and a clear line when syllables are linked together for more complicated words. Whereas in the upper mel-spectrogram the clear contours are missed, which can also be heard in the sample in a coarse and slurred nature.

4.3 Real-Time Voice Cloning

Real-Time Voice Cloning is the result of a Master's thesis [36] from 2019, where the author implements the framework described in this work [37] from 2018 by Google. As illustrated in figure 5 the framework consists of:

- A speaker encoder, based on generalized end-to-end loss(GE2E) [38], that takes as input a few seconds of speech of the target speaker and generates an embedding vector containing the speaker's characteristics.
- An acoustic model, in the work referred to as a synthesizer, based on a modified version of Tacotron 2, where the embedding vector is concatenated to the acoustic model in the attention layer.
- A vocoder based on WaveRNN, differing from the original work by Google [37] which used WaveNet.

The three components were trained independently. The speaker encoder was trained on the three datasets LibriSpeech-Other, VoxCeleb1 and VoxCeleb2, which combined have over 2'500 hours of audio. The acoustic model and the vocoder were trained on 460 hours of audio from the LibriSpeech-Clean

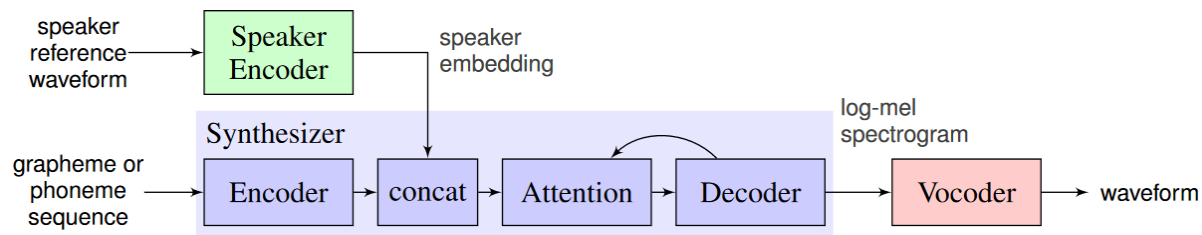


Figure 5: framework components [37]

Dataset.

4.3.1 Ease of use:

Installation: Setting up Real-Time Voice Cloning locally in a conda environment proved challenging due to missing dependencies and incompatible versions listed in the requirements.txt file. The installation process involved manual installation of missing modules and adjustment of library versions.

Main features: The key features of Real-Time Voice Cloning are:

- Sentence or long text generation with a single voice.
- Visualization of mel-spectrograms and embeddings for both the reference and generated audio.
- Visualization of embeddings in a 2D-space, where embeddings from the same speaker are grouped into clusters.

User interface: While the tool includes a GUI toolbox for generating personalized speech, compatibility issues prevent the toolbox from functioning on Windows 11 due to the repository's lack of maintenance. The tool can be used via terminal commands, but without access to the visualization features mentioned above.

Accessibility of documentation: The README of the repository [24] contains a concise video explaining the toolbox. For detailed information on training configurations and instructions to train a personalized model, users can refer to the repository's Wiki.

4.3.2 Generated speech

All the speeches generated with Real-Time Voice Cloning yielded poor results. All outputs present a robotic, hoarsed voice with a quick pace and slurred articulation. The original voice is very slightly resembled in the generated speech, but because of the robotic rhythm, the speeches don't show any naturalness. Using an angry-voiced reference does not affect the emotion of the generated speeches, which always have a monotone emotion.

To synthesise one sentence Real-Time Voice Cloning takes about 10 seconds.

The outputs of the speech synthesis process using Real-Time Voice Cloning consistently yielded unsatisfactory results. The generated speeches consistently display a robotic, hoarse voice characterized by a rapid pace and slurred articulation. While there exists a subtle resemblance to the original voice, the outputs lack the natural cadence inherent in human speech, primarily attributed to the persistent robotic rhythm.

When an audio with an angry voice was utilized as a reference, the emotion in the generated speeches remained consistently monotone, indicating a limitation in conveying emotions.

The synthesis of a single sentence through Real-Time Voice Cloning requires approximately 10 seconds. Generating multiple candidates for a sentence by manually running multiple inferences did not improve the result.

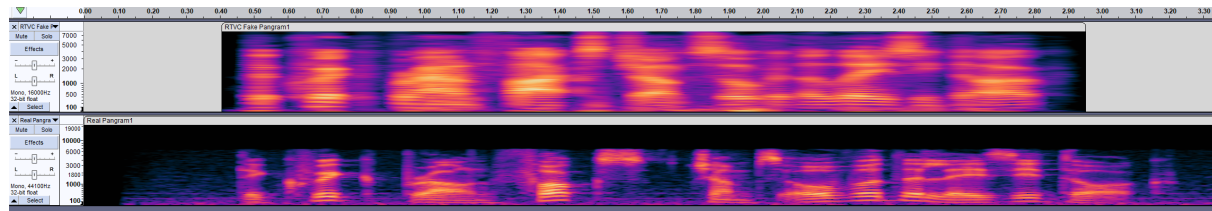


Figure 6: Mel-spectrogram of the real(bottom) and the synthesized voice(top) of pangram1

For the best sounding sample of Real-Time Voice Cloning (figure 6) the used input was pangram3 and the best output from this batch was pangram1. This is the most frequent case, as pangram3 has all the necessary phonemes included and pangram1 uses simple words that are often included in the training sets for DNNs. This specific sample sounds choppy like it was built frame by frame that is also present in the mel-spectrogram where one can see the individual jumps per frame instead of the more connected signal strengths in the original voice.

5 Discussion

The evaluation of speech is subjective due to the complexity of human perception. The influence of training data and inference iterations on the quality of speech synthesis is a crucial aspect to consider. This can be observed with TorToise, which demonstrated remarkable naturalness in its synthesized output, attributed to the extensive training data set and a high number of inference iterations. However, the speaker similarity aspect did not exhibit a proportional improvement, as evidenced by the comparable speaker similarity results observed in TorToise, trained on a significantly larger data set, and VALL-E X, trained on a comparatively smaller data set. The nature of the training data further influences the output characteristics, evident in TorToise's distinctive book reading style, a reflection of its primary training on audio books.

The candidate generation played a significant role in obtaining natural synthesized speech. TorToise seamlessly handled this step through the Contrastive Language-Voice Pretrained Transformer (CLVP), automating the generation and selection of optimal speech tokens. Additionally three output candidates were generated and the best one was manually selected. In contrast, VALL-E X required manual generation of multiple candidates by synthesizing the same sentence repeatedly. Multiple candidates were needed especially for sentences featuring complex phonemes and uncommon words in VALL-E X, highlighting potential limitations in the diversity of its training data concerning these linguistic elements. The quantity of training data alone does not guarantee success, as observed in Real-Time Voice Cloning. Despite being trained on a larger dataset compared to VALL-E X, Real-Time Voice Cloning, being an outdated tool, did not generate natural sounding speech. This underscores the importance of recent advancements in model architectures in DNN.

The results from this research show the current capabilities of DNNs synthesizing speech and how the quality can be, when generating speech for potential scams or spoofed voice attacks. Most of the generated audio is not usable in spoofed voice attacks due to clearly being recognized as generated speech with robotic-sounding artefacts, wrong intonations or bad-sounding prosody. These negative properties can be diminished when using them in situations, where worse audio quality is a plausibility. For example phone calls with bad audio quality either due to a bad microphone or bad reception while going through tunnels or calling from remote areas with less signal. With modern voice-over-internet-protocol calls the quality is usually good enough to notice synthesized speech. Considering the long generation times the need of a pre-generated response set is mandatory for a spoofing attack over a phone call. Additionally this means generating a lot of different responses to accommodate many situations in a phone call. A more ideal situation for an attacker is the use of voice messages, so sending recorded speech in replacement of a phone call. There, an attacker has much more time to generate the best possible response while also having the option to generate speech displaying emotions. Though with voice messages one does need to have access to a verified device from the targeted person to properly impersonate them. Sending voice messages with synthesized speech can be interpreted as more plausible as just plain text, when having access to a verified device through theft or spoofing. The best defense against these attacks is to ask the possible attacker on the phone a question from a shared experience or recently told stories between them. For example: "Which ski resort did you finally decide to go last weekend?".

All these reasons naturally change when one tries to imitate a well-known person with a lot of recorded material readily available, for example celebrities, politicians, CEOs or people in general working in front of a camera or microphone such as actors, news anchors, journalists, podcast hosts et cetera. With extensive audio material one can even train DNNs on these persons to achieve even more convincing synthesized speech from these models. If one abstracts this imitation problem to the extreme, it condenses into a probabilistic problem where each audio file is a sequence of binary data that translated into sound waves could be unrecognizable to the binary sequence of a recording of a real person. Obviously with so many variables it will take a lot of time to produce such results, but at a certain point the digital sequence of a fake voice will be equivalent to the digital sequence of the original voice. Such problems can most often only be solved by cryptography, meaning in the future where such perfect imitations are possible, the best way to verify the authenticity of any voice transmission will be a Diffie-Hellman key exchange with verified sources [39].

6 References

- [1] Wikipedia contributors, *Speech synthesis — Wikipedia, the free encyclopedia*, [Online; accessed 19-December-2023], 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Speech_synthesis&oldid=1189644011.
- [2] CSBS News. "Cybercriminals are using ai voice cloning tools to dupe victims." (2023), [Online]. Available: <https://www.cbsnews.com/news/ai-scam-voice-cloning-rising/> (visited on 10/08/2023).
- [3] Joseph Cox. "How i broke into a bank account with an ai-generated voice." (2023), [Online]. Available: <https://www.vice.com/en/article/dy7axa/how-i-broke-into-a-bank-account-with-an-ai-generated-voice> (visited on 12/19/2023).
- [4] X. Tan, "Text analyses," in *Neural Text-to-Speech Synthesis*. Singapore: Springer Nature Singapore, 2023, pp. 67–80, ISBN: 978-981-99-0827-1. DOI: 10.1007/978-981-99-0827-1_4. [Online]. Available: https://doi.org/10.1007/978-981-99-0827-1_4.
- [5] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. D. Richards, "Normalization of non-standard words," *Computer speech & language*, vol. 15, no. 3, pp. 287–333, 2001.
- [6] H. Sun, X. Tan, J.-W. Gan, *et al.*, "Token-level ensemble distillation for grapheme-to-phoneme conversion," *arXiv preprint arXiv:1904.03446*, 2019.
- [7] X. Tan, "Acoustic models," in *Neural Text-to-Speech Synthesis*. Singapore: Springer Nature Singapore, 2023, pp. 81–100, ISBN: 978-981-99-0827-1. DOI: 10.1007/978-981-99-0827-1_5. [Online]. Available: https://doi.org/10.1007/978-981-99-0827-1_5.
- [8] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis," in *Sixth European conference on speech communication and technology*, 1999.
- [9] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, "Speech synthesis based on hidden markov models," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1234–1252, 2013.
- [10] H. Zen, "Acoustic modeling in statistical parametric speech synthesis-from hmm to lstm-rnn," 2015.
- [11] Y. Wang, R. Skerry-Ryan, D. Stanton, *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.
- [12] J. Shen, R. Pang, R. J. Weiss, *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, pp. 4779–4783.
- [13] S. Ö. Arık, M. Chrzanowski, A. Coates, *et al.*, "Deep voice: Real-time neural text-to-speech," in *International conference on machine learning*, PMLR, 2017, pp. 195–204.
- [14] Y. Ren, Y. Ruan, X. Tan, *et al.*, "Fastspeech: Fast, robust and controllable text to speech," *Advances in neural information processing systems*, vol. 32, 2019.
- [15] Y. Ren, C. Hu, X. Tan, *et al.*, "Fastspeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=piLPYqxtWuA>.
- [16] X. Tan, "Vocoders," in *Neural Text-to-Speech Synthesis*. Singapore: Springer Nature Singapore, 2023, pp. 101–114, ISBN: 978-981-99-0827-1. DOI: 10.1007/978-981-99-0827-1_6. [Online]. Available: https://doi.org/10.1007/978-981-99-0827-1_6.

- [17] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, "Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation," *arXiv preprint arXiv:2106.07889*, 2021.
- [18] A. v. d. Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.
- [19] X. Tan, "Fully end-to-end tts," in *Neural Text-to-Speech Synthesis*. Singapore: Springer Nature Singapore, 2023, pp. 115–122, ISBN: 978-981-99-0827-1. DOI: 10.1007/978-981-99-0827-1_7. [Online]. Available: https://doi.org/10.1007/978-981-99-0827-1_7.
- [20] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *speech communication*, vol. 66, pp. 130–153, 2015.
- [21] C. Yan, X. Ji, K. Wang, Q. Jiang, Z. Jin, and W. Xu, "A survey on voice assistant security: Attacks and countermeasures," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–36, 2022.
- [22] J. Betker, *TorToiSe text-to-speech*, version 2.0, Apr. 2022. [Online]. Available: <https://github.com/neonbjb/tortoise-tts> (visited on 11/29/2023).
- [23] Songting, *VALL-e x: Multilingual text-to-speech synthesis and voice cloning*, Dec. 5, 2023. [Online]. Available: <https://github.com/Plachtaa/VALL-E-X> (visited on 12/05/2023).
- [24] C. Jemine, *CorentinJ/real-time-voice-cloning*, Dec. 14, 2023. [Online]. Available: <https://github.com/CorentinJ/Real-Time-Voice-Cloning> (visited on 12/14/2023).
- [25] J. Betker, "Better speech synthesis through scaling," *arXiv preprint arXiv:2305.07243*, 2023.
- [26] J. Betker. "Tortoise tts - a hugging face space by manmay." (), [Online]. Available: <https://huggingface.co/spaces/Manmay/tortoise-tts> (visited on 12/12/2023).
- [27] C. Mullis. "Afiaka87/tortoise-tts – run with an API on replicate." (), [Online]. Available: <https://replicate.com/afiaka87/tortoise-tts> (visited on 12/12/2023).
- [28] C. Wang, S. Chen, Y. Wu, *et al.*, "Neural codec language models are zero-shot text to speech synthesizers, 2023," URL: <https://arxiv.org/abs/2301.02111>. doi: doi, vol. 10,
- [29] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.
- [30] F. Li, *Vall-e: A neural codec language model*, 2023. [Online]. Available: <http://github.com/lifeiteng/vall-e>.
- [31] Z. Niu, *VALL-e*, Dec. 1, 2023. [Online]. Available: <https://github.com/enhuiz/vall-e> (visited on 12/05/2023).
- [32] Z. Zhang, L. Zhou, C. Wang, *et al.*, "Speak foreign languages with your own voice: Cross-lingual neural codec language modeling," *arXiv preprint arXiv:2303.03926*, 2023.
- [33] H. Siuzdak, "Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis," *arXiv preprint arXiv:2306.00814*, 2023.
- [34] "Demo of reproduced VALL-e x," VALL-E. (), [Online]. Available: <https://plachtaa.github.io/> (visited on 12/06/2023).
- [35] Songting. "Vall e x - a hugging face space by plachta." (), [Online]. Available: <https://huggingface.co/spaces/Plachta/VALL-E-X> (visited on 12/12/2023).
- [36] C. Jemine, "Master thesis: Real-time voice cloning," 2019.
- [37] Y. Jia, Y. Zhang, R. Weiss, *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," *Advances in neural information processing systems*, vol. 31, 2018.

- [38] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4879–4883.
- [39] U. M. Maurer and S. Wolf, "The diffie–hellman protocol," *Designs, Codes and Cryptography*, vol. 19, no. 2-3, pp. 147–171, 2000.
- [40] Ars Technica. "Microsoft's new ai can simulate anyone's voice with 3 seconds of audio." (2023), [Online]. Available: <https://arstechnica.com/information-technology/2023/01/microsofts-new-ai-can-simulate-anyones-voice-with-3-seconds-of-audio/> (visited on 11/07/2023).
- [41] J. Yi, C. Wang, J. Tao, X. Zhang, C. Y. Zhang, and Y. Zhao, "Audio deepfake detection: A survey," *arXiv preprint arXiv:2308.14970*, 2023.
- [42] A. Hannun, C. Case, J. Casper, *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.
- [43] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 6706–6713.
- [44] S. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, "Neural voice cloning with a few samples," *Advances in neural information processing systems*, vol. 31, 2018.
- [45] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, "Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone," in *International Conference on Machine Learning*, PMLR, 2022, pp. 2709–2720.

7 List of Figures

1	The progressively end-to-end process for TTS models[19]	5
2	Mel-spectrogram of the real(bottom) and the synthesized voice(top) of pangram2	12
3	Inference process of VALL-E [28]	13
4	Mel-spectrogram of the real(bottom) and the synthesized voice(top) of pangram3	15
5	framework components [37]	16
6	Mel-spectrogram of the real(bottom) and the synthesized voice(top) of pangram1	17

8 List of Tables

1	Comparison of the three open-source implementations: TorToise, VALL-E (X) and Real-Time Voice Cloning	10
---	---	----

9 Glossary

AI artificial intelligence 1

BAP band-aperiodicity 3

CNN convolutional neural network 4

DDPM denoising diffusion probabilistic model 10, 11

DNN deep neural networks , 1, 3, 4, 8, 17, 18, 19

F0 fundamental frequency 3, 4

GAN generative adversarial network 4

GUI graphical user interface 11, 14, 16

HMM hidden Markov model 1, 3

MCC mel cepstral coefficients 3

repo repository 7, 8

RNN recurrent neural network 4

SPSS statistical parametric speech synthesis 3, 4

TTS text-to-speech 3, 4, 5, 7, 10, 11

VAE variational auto-encoder 4

10 Appendix

10.1 Official Assignment

Titel

Reviewing state of the art in voice generation

Beschreibung

In recent months, the topic of generative AI has gained increased attention, given impressive advances in the field. In particular, voice generation has sparked press attention, for example due to impressive songs released where the voice of a famous artist is artificially generated. These advances have also sparked worries, for instance due to their potential for fraud. It has been recently reported that scammers are using generative AI for impersonation [1]. In this work you will review the state of the art in voice generation, with a particular focus on approaches that provide open source software implementation. The goal is to understand the ease of use and potential risks posed by this technology, and to understand potential limitations. This will lay the basis for future works that will attempt to implement countermeasures or detection techniques for artificially generated voices.

<https://www.cbsnews.com/news/ai-scam-voice-cloning-rising/>

Voraussetzungen

General knowledge of machine learning

Knowledge of generative machine learning is a plus

Vereinbarungen

- Review state of the art (literature) for voice generation techniques
- Select most recent open source implementations
- Compare main features between leading implementations
- Highlight pros and cons of various solutions
- Write final report
- Prepare final presentation