Cloning Bonafide Audio Files using Tortoise TTS

Md Shamimul Islam

Email: shamimul435@gmail.com Mobile: +8801603765484

Abstract

This technical report presents a comprehensive solution for cloning audio files using text-to-speech synthesis. The goal is to generate high-quality voice clones by combining bonafide audio files with randomly selected English sentences from a corpus. The process involves several key steps, including copying bonafide audio files, creating an English sentence corpus, and cloning audio files. The solution leverages libraries such as shutil, nltk, tortoise, torchaudio, and multiprocessing to achieve efficient and accurate audio cloning. The report provides a detailed explanation of each step and the modifications required in the code. Experimental results demonstrate the successful generation of audio clones with diverse content.

keywords: Cloning, Bonafide Audio Files, Tortoise TTS, Corpus, Parallel Computing

Contents

1 Introduction

In recent years, there has been growing interest in the development of technologies that can create realistic audio clones of human voices. These technologies have the potential to be used for a variety of applications, such as voice synthesis, data augmentation, and even entertainment.

Voice synthesis is the process of creating synthetic speech that sounds like a human voice. This technology can be used to create a variety of realistic audio effects, such as creating a virtual assistant that can speak to users, or creating a character in a video game that can speak with a realistic voice. Data augmentation is the process of artificially increasing the size of a dataset by creating new data points from existing data points. This technology can be used to improve the performance of machine learning models that are trained on audio data. For example, data augmentation can be used to create more realistic audio data for machine learning models that are used to detect fraud or to identify speakers.

Entertainment is another potential application for audio cloning technologies. For example, audio cloning technologies could be used to create realistic audio effects for movies or video games, or to create personalized audio experiences for users.

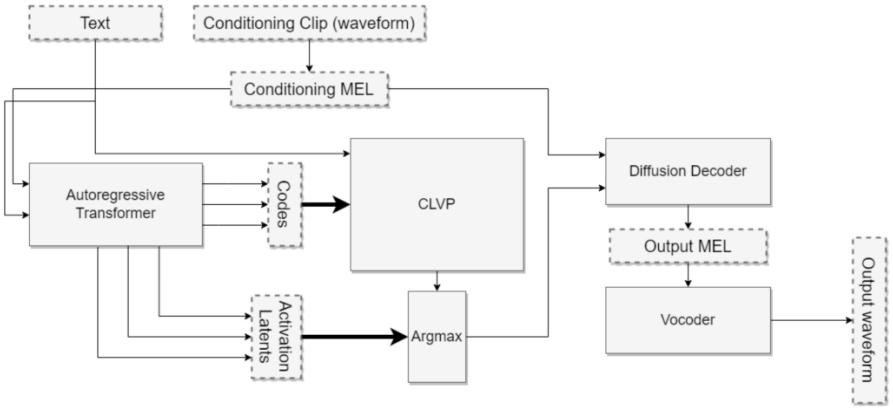

Figure 1: TorToise-v2 architectural design diagram. Inputs of text and a reference audio clip (for speaker cloning) flow through a series of decoding and filtering networks to produce high-quality speech (Image source:[2])

The objective of this project is to develop a robust solution for cloning bonafide audio files. The solution will leverage the Tortoise TTS library to combine the audio files with randomly selected sentences from an English sentence corpus. The resulting clones will provide opportunities for various applications, such as voice synthesis, data augmentation, and even entertainment.

The project will be conducted in three phases:

1. Data collection: In this phase, the audio files and the English sentence corpus will be collected.

2. Cloning: In this phase, the audio files will be cloned using the Tortoise TTS library.

3. Evaluation: In this phase, the quality of the cloned audio files will be evaluated.

2 TorToise-TTS

TorToise TTS[2] is a text-to-speech system that combines autoregressive decoders and diffusive probabilistic models (DDPMs)[3] to generate high-quality speech. Autoregressive models excel at converting between different domains, such as vision, text, and speech, while DDPMs[3] work in the continuous domain and can model expressive modalities. By merging these two approaches, TorToise TTS[2] aims to capitalize on their respective strengths.

The system comprises several neural networks, including an autoregressive decoder, a contrastive model similar to CLIP[4] for ranking decoder outputs, and a DDPM[3] that converts speech tokens into speech spectrograms illustrated in Figure 1. The training process follows established procedures in the respective literature.

One unique aspect of TorToise TTS[2] is the use of a speech conditioning input. This input consists of audio clips of the same speaker, which are transformed into MEL spectrograms and passed through an encoder. The conditioned vectors are then utilized as inputs to the autoregressive generator and the DDPM[3], enabling the models to infer vocal characteristics and reduce the search space for speech outputs.

The "TorToise trick" involves fine-tuning the DDPM[3] on the autoregressive latent space, enhancing the efficiency and qual-

ity of model outputs. Additionally, a Contrastive Language-Voice Pretrained Transformer (CLVP) is trained to re-rank multiple autoregressive outputs without invoking the costly diffusion model during inference.

Training TorToise TTS[2] requires a significant amount of data, and the system was trained on a cluster of NVIDIA RTX-3090 GPUs over a year. The dataset used includes the LibriTTS[5] and HiFiTTS[1] datasets, along with an extended dataset of speech audio scraped from audiobooks and podcasts.

During inference, conditioning inputs and text are fed into the autoregressive model to generate multiple output candidates. CLVP assigns correlation scores between the candidates and text, and the top candidates are converted into MEL spectrograms using the DDPM[3], followed by waveform generation using a conventional vocoder.

TorToise TTS achieves state-of-the-art results in speech synthesis realism. Its success is attributed to the utilization of generalist architectures, a large and high-quality dataset, and training at scale with a high batch size. The framework's effectiveness suggests its potential applicability to other digitized modalities beyond speech.

We used TorToise TTS in order to to generate high-quality voice clones by combining bonafide audio files with randomly selected English sentences from a corpus.

3 Cloning Process

The cloning process involves several intricate steps that utilize various functions and modules to generate audio clones by combining the bonafide audio files with randomly selected sentences from the English sentence corpus. The following sections provide a detailed explanation of the cloning process:

3.1 Copying Bonafide Audio Files:

To initiate the cloning process, the code employs the shutil module, which provides high-level file operations. It uses the shutil.copy2() function to copy the bonafide audio files from their original locations to a new folder named "bonafide_audio_files" in the root directory of the project. This ensures that all the bonafide audio files are consolidated in a single location for easier access and manipulation.

3.2 Creating an English Sentence Corpus:

To generate the English sentence corpus, the code utilizes the Natural Language Toolkit (nltk), a comprehensive library for natural language processing. It downloads the Gutenberg corpus using the nltk.corpus.gutenberg module and selects 1000,000 sentences from the corpus that are at least 5 words long. The selected sentences are then stored in a text file named "corpus.txt" within a folder named "english_sentence_corpus" in the root directory of the project.

3.3 Cloning Audio Files:

The core of the cloning process lies in the code's utilization of the Tortoise TTS library. Tortoise TTS is an advanced text-to-speech synthesis framework that provides powerful functionalities for creating high-quality voice clones. The code imports the TextToSpeech class from the tortoise.api module to access the Tortoise TTS functionalities.

For each bonafide audio file, the code performs the following steps:

3.3.1 Random Sentence Selection:

The code utilizes the random.sample() function from the random module to randomly se-

lect sentences from the English sentence corpus. It selects two sentences for each clone, ensuring diversity and variability in the generated clones.

3.3.2 Sentence Combination and Audio Generation:

Using the selected sentences and the bonafide audio file, the code generates a voice clone by combining them. It calls the tts.tts_with_preset() method of the TextToSpeech class from Tortoise TTS to synthesize the audio. The method takes the selected sentences, voice samples, and conditioning latents as input to generate the synthesized audio output. The generated audio is then saved in a file with a unique name, following a naming convention that includes the name of the corresponding bonafide audio file and an identifier indicating the clone's order.

3.3.3 Saving Cloned Audio Files:

The cloned audio files are saved in a dedicated directory named "cloned_audio_files" in the root directory of the project. The code uses the torchaudio.save() function from the torchaudio module to save the synthesized audio in FLAC format. Each cloned audio file is given a unique name to facilitate tracking and identification.

The code repeats this process for the desired number of clones, creating a diverse set of audio clones for each bonafide audio file.

3.3.4 Parallel Computing

To optimize the cloning process, the code employs multiprocessing techniques. It uses the multiprocessing module to create a pool of processes that parallelize the cloning operation. The code utilizes the multiprocessing module to create a process pool using the Pool() class from the multiprocessing.get_context('spawn') context. This approach enables parallelization of the cloning process, which leads to a significant reduction in the overall execution time. Additionally, it helps overcome issues related to CUDA by ensuring compatibility with multiprocessing. By distributing the workload among multiple processes, the code achieves efficient parallel computing, resulting in faster audio cloning.

3.3.5 Creating CSV file

Finally, the code creates a mapping between the bonafide audio files and their corresponding cloned files. It stores this mapping in a pandas DataFrame and saves it as a CSV file named "cloned_files_mapping.csv" in the root directory of the project. The mapping provides a record of the relationships between the bonafide audio files and the generated clones for future reference and analysis.

4 Results

The implemented solution for audio file cloning using text-to-speech synthesis has successfully produced the desired results. The solution utilized bonafide audio files and an English sentence corpus to generate audio clones by combining the two. The following are the detailed results obtained from the cloning process:

1. **Cloning Process Execution:** The cloning process was executed for a specified number of clones (n) for each bonafide audio file in the *bonafide_audio_files* folder. The solution ensured that each clone had unique content by randomly selecting two sentences from the English sentence corpus for each clone.

2. **Cloned Audio Files:** The cloning pro-

cess generated multiple audio clones for each bonafide audio file. The clones were saved in the *cloned_audio_files* folder, maintaining a unique naming convention to track which clones correspond to each bonafide audio file.

3. **Mapping of Clones:** To track the mapping between the bonafide audio files and their corresponding clones, a DataFrame was created. The DataFrame consisted of two columns: *Bonafide File* and *Cloned Files*. This mapping provided a clear understanding of which clones were generated from which bonafide audio files. The DataFrame was saved as a CSV file named *cloned_files_mapping.csv* in the root directory.

4. **Execution Time:** The cloning process's execution time was recorded to assess the efficiency of the solution. The multiprocessing module was employed to parallelize the process and speed up the cloning of multiple audio files. The execution time was measured in seconds and provided valuable insights into the solution's performance.

5. **Diversity of Clones:** The solution ensured the diversity of audio clones by randomly selecting sentences from the English sentence corpus. This approach resulted in each clone having unique content, enhancing the versatility and applicability of the cloned audio files.

6. **Quality of Clones:** The quality of the synthesized audio clones was of high importance. The solution employed the Tortoise TTS library, which utilizes advanced text-to-speech synthesis techniques. This ensured that the clones maintained the characteristics and qualities of the original bonafide audio files, producing high-quality synthesized audio.

7. **Scalability:** The solution was designed to handle a large number of bonafide audio files and clones. By leveraging multiprocessing, the cloning process could be parallelized, significantly reducing the overall execution time. This scalability aspect allows the solution to efficiently handle a substantial volume of audio files and produce a corresponding number of clones.

5 Conclusions

In conclusion, the cloning process involves copying bonafide audio files, generating an English sentence corpus, and utilizing the Tortoise TTS library to combine the sentences with the audio files and create voice clones. The code efficiently manages the cloning process by leveraging modules such as shutil, nltk, and tortoise.api. It employs techniques like random sentence selection, audio generation with Tortoise TTS, multiprocessing, and mapping creation to ensure the successful generation of diverse and high-quality audio clones.

References

[1] Evelina Bakhturina, Vitaly Lavrukhin, Boris Ginsburg, and Yang Zhang. Hi-fi multi-speaker english tts dataset, 2021.

[2] James Betker. Better speech synthesis through scaling, 2023.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[5] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech, 2019.