REAL TIME VOICE CLONING

[1]Kaushik Daspute, [2]Hemang Pandit, [3]Shweta Shinde
[1]Student, [2]Student, [3]Student
[1] Department of Electronics and Telecommunication,
[1] Pune Institute of Computer Technology, Pune, India.

Abstract : Recent progress in deep learning has shown impressive results in the area of speech-to-text. For this reason, a deep neural network is usually trained from a single speaker using a corpus of several hours of voice recorded professionally. Giving such a model a new voice is highly expensive, as it needs a new dataset to be collected and the model retrained. A recent research has developed a three-stage pipeline that allows you to clone an unseen voice from just a few seconds of reference speech during practice and without retraining the template. The researchers share strikingly natural-sounding findings. The plan is to replicate this model and open source it to the public. With a new vocoder model, the aim is to adapt the framework to make it run in real time. The aim is to develop a three-stage deep learning system that will perform real-time voice cloning. This framework is the result of Google's 2018 paper, for which only one public implementation exists before ours. The system could capture a realistic representation of the voice spoken in a digital format from a speech utterance of only 5 seconds. Because of a text prompt, it can use any voice extracted from this process to perform text-to-speech. With our own implementations or open-source ones then plan is to replicate each of the three stages of the model. The plan is to implement successful models of deep learning and appropriate pipelines for pre-processing information. The next step is training these models from several thousand speakers for weeks or months on large datasets of tens of thousands of hours of speech. Instead examine their strengths and their drawbacks. The main focus on making this system function in real time, that is, to allow a voice to be captured and speech to be generated in less time than the duration of the speech produced. The framework will be able to clone voices it has never heard during training, and to generate speech from text it has never seen.

I. INTRODUCTION

In many fields of computational machine learning, deep learning models have become prevalent. The method of synthesizing artificial speech from a text prompt, Text-to-speech (TTS), is no exception. Deep models that would create more natural-sounding speech than the conventional concatenative methods begun emerging in 2016. Much of the research focus has been since gathered around making these deep models more effective, sound more natural, or training them in an end-to-end fashion. Inference on GPU (van den Oord et al., 2016)[2] has come from being hundreds of times slower than real-time on a mobile CPU (Kalchbrenner et al., 2018)[3] .Shen et al. (2017)[4] shows near-human naturalness as to the quality of the speech produced. Ironically, speech naturalness is better measured by subjective metrics; and correlation with actual human speech leads to the assumption that "speech more natural than human speech" can exist. Nonetheless, some argue that the boundary of human nature has already been reached (Shirali-Shahreza and Penn, 2018)[5] While a single-speaker TTS model's complete learning is theoretically a form of voice cloning, the purpose is rather to create a fixed model that can integrate new voices with little information. The common approach is to condition a TTS template that has been trained to generalize voice to clone to new speakers (Arik et al., 2017, 2018; Jia et al., 2018)[7]. Low-dimensional embedding is derived from a speaker encoder model which takes reference speech as input. Typically, this approach is more data-efficient than training a separate TTS model for each speaker, as well as faster and less computationally expensive orders of magnitude. Interestingly, there is a broad disparity between the length of reference speech necessary to clone a voice among the various methods, varying from half an hour per speaker to just a few seconds. Generally, this aspect determines the similarity of the voice produced with respect to the speaker's true voice.

II. LITERATURE SURVEY

To develop the idea, different methods were researched to find a suitable implementation algorithm, including a few research papers and a master's thesis from May 2019 [1]. Deep models that would make speech more natural-sounding than the conventional concatenative solutions that began to appear in 2016. Since that time, much of the research focus has been on making these deep models more effective, sounding more realistic, or training them in an end-to-end fashion. Inference on GPU (van den Oord et al., 2016)[2] has come from being hundreds of times slower than real-time on a mobile CPU (Kalchbrenner et al., 2018)[3].Shen et al. (2017)[4] shows near-human naturalness as to the quality of the speech produced. Ironically, speech naturalness is better measured by subjective metrics; and correlation with actual human speech leads to the assumption that "speech more natural than human speech" can occur. Nonetheless, some argue that the boundary of human nature has already been reached (Shirali-Shahreza and Penn, 2018)[5].

III. BLOCK DIAGRAM

The basic building blocks of the whole program can be shown as follows.

The important elements are

- Feature Extractor: -The role of the feature extractor is to provide data that is more indicative of what the speech produced by the model is expected to sound like.
- Acoustic Model:- The relationship between the features computed on the input text and the output acoustic features is learned by a statistical generative model called the Acoustic Model.
- Vocoder: A system able to reconstruct an audio waveform from the acoustic features produced by the acoustic model. It infers an audio waveform from the spectrograms generated by the synthesizer.

- Speaker Encoder: Derives an embedding from the short utterance of a single speaker. The embedding is a meaningful representation of the voice of the speaker, such that similar voices are close in latent space.
- Synthesizer: Conditioned on the embedding of a speaker, generates a spectrogram from text. This model is the popular Tacotron 2 (Shen et al., 2017) without WaveNet (which is often referred to as just Tacotron due to its similarity to the first iteration).

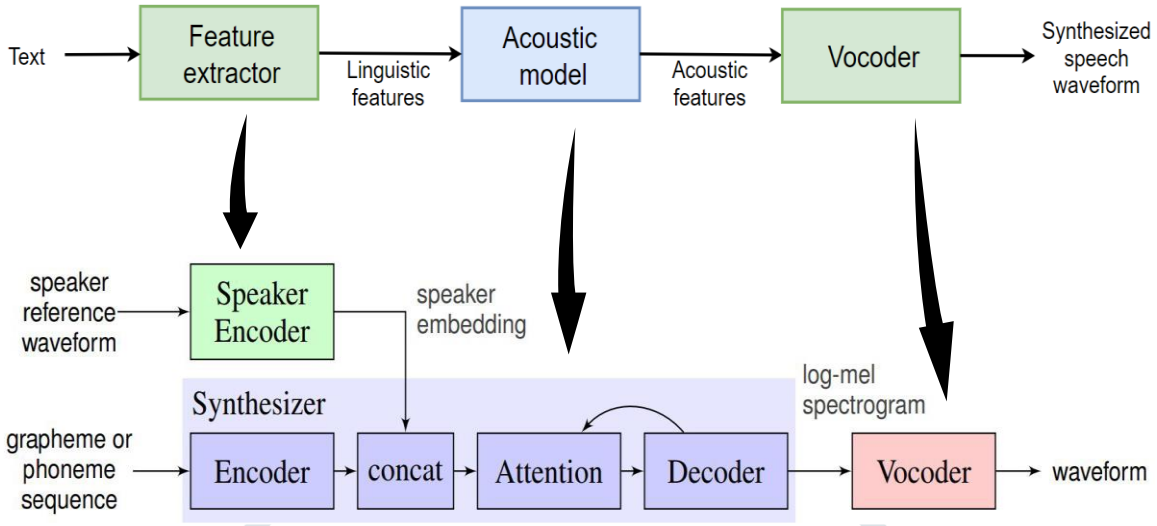

Figure 3.1: System Block Diagram

IV. FLOW CHART

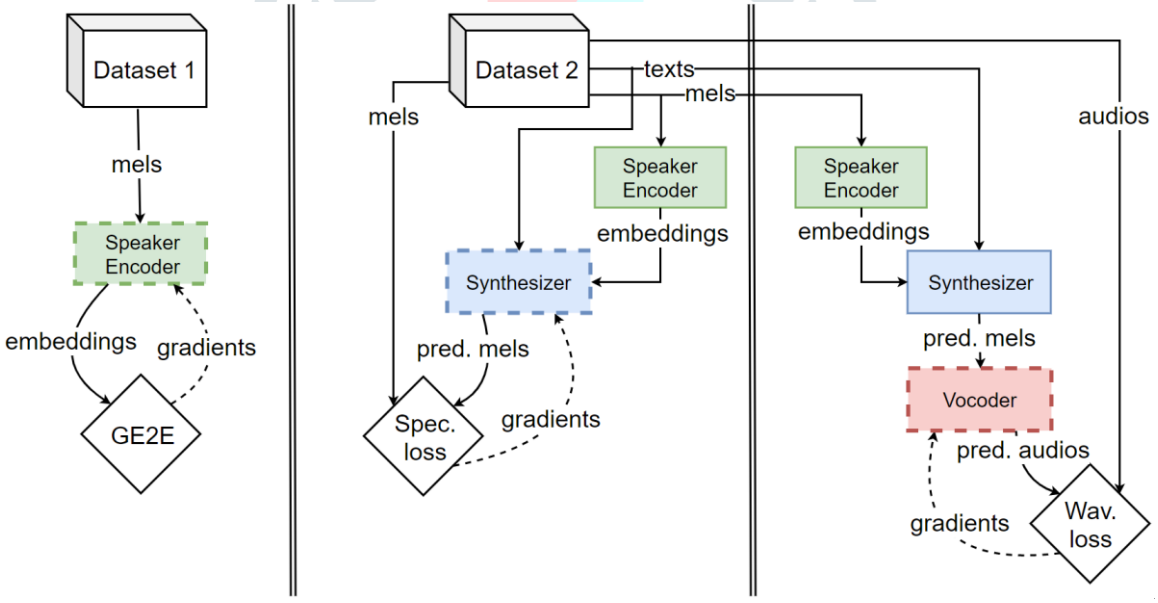

Figure 4.1: The System Flowchart

The above diagram explains the flow of the processes we plan to use to execute our project. The Encoder takes in the input audio and creates voice embeddings which have the characteristics of the unique speaker voice. The Synthesizer creates a grapheme or phoneme sequence from the input text using machine learning algorithms. The outputs of the Encoder and Synthesizer form a Mel-Spectrogram that's utilized by the Vocoder to give the final cloned voice output in the desired voice without training the system again.

V. IMPLEMENTATION

V.I Implementation

The implementation of the whole system involves processes like pre-requisite installations, configuring the environment for proper working of the project, acquiring of datasets, coding and implementation of Encoder module, Synthesizer module and the Vocoder module. All the coding and required implementations have been done using Python 3 language, as mentioned

earlier, and involves implementation of open-source technologies like Google Tacotron 2 and Wavenet modules which have openly available pre-coded packages that have been used.

V.II Installations

The mandatory installations that are required for the working of the project included

- TensorFlow GPU(1.10.0=<Version<=1.14.0)
- Umap-learn
- Visdom
- Webrtcvad
- Librosa (0.5.1=<Version)
- Sounddevice
- Unidecode
- PyTorch
- Inflect

The above-mentioned installations were done using Python pip in the python shell. After the installations are done it is necessary to configure the new installations so that they form a proper environment that can be used to work and access the project workings. This is done with the help of the demo_cli.py file. If the demo_cli.py file runs successfully then we have successfully installed and configured the requirements.

V.III Dataset used

The researchers find two datasets in SV2TTS to train the synthesizer as well as the vocoder. These are LibriSpeech-Clean, which has been mentioned earlier, and VCTK10, a body of only 109 native English speakers recorded with professional equipment. Throughout their tests, the VCTK speech is sampled at 48kHz and downsampled to 24kHz, which is still higher than LibriSpeech's 16kHz sampling. It is decided to find that when it comes to similarity, a synthesizer trained on LibriSpeech generalizes better than on VCTK, but at the cost of natural speech. The datasets are public and a simple request email to University of Edinburg(the original owner of the datasets) was enough to get a download link for the database.

V.IV Implementation of Speaker Encoder

The Speaker Encoder is the first module to be trained. It deals with the audio input provided to the system and hence contains the preprocessing, audio training and visualization models. The Speaker Encoder is a LSTM3-layer with 768 hidden nodes followed by a 256-unit projection layer. While there is no reference to what a projection layer is in any of the articles, our understanding is that it is simply a full-connected layer of 256 outputs per LSTM that is repeatedly applied to each LSTM output. Instead of implementing the speaker encoder for the first time, it can be directly used 256 LSTM layers for quick prototyping, simplicity and a lighter training load. The outputs are 40-channel log-mel spectrograms with a window width of 25ms and a stage of 10ms. The output is the last layer's L2-normalized hidden state, which is a 256-element vector. Our implementation also features a pre-standardization ReLU layer with the goal of making embedding sparse and thus easier to interpret.

V.V Implementation of Synthesizer

The Synthesizer used is the Google Tacotron 2 model which is used without Wavenet. Tacotron is a repeated sequence-to-sequence system predicting a text-based mel spectrogram. Specific characters are first inserted as vectors from the text string. In order to increase the length of a single encoder block, standard layers follow. To build the encoder output frames, these frames are passed through a bidirectional LSTM. This is where SV2TTS adds a change to the architecture: the embedding of a speaker is concatenated with each frame that the Tacotron encoder creates. In order to generate the decoder input frames, the attention function attends to the encoder output frames. In our implementation, the input texts are not checked for pronunciation and the characters are fed as they are. Nonetheless, there are a few cleaning procedures: replacing abbreviations and numbers with their full text form, moving all characters into ASCII, normalizing white spaces, and lowering all characters. Punctuation may be used, but in our datasets, it is not present.

V.VI Implementation of Vocoder

The Vocoder module is the last to be trained as the modules are expected to be trained in the sequence of Encoder-Synthesizer-Vocoder. WaveNet is the vocoder in both SV2TTS and Tacotron2. The vocoder model used is an open source PyTorch implementation15, based on WaveRNN, but with a few different user fatchord design choices. This architecture will be referred to as "the alternative WaveRNN." A mel spectrogram and its corresponding waveform are split into the same number of segments at each training phase. The design inputs are the segment t of the spectrogram to simulate and segment t-1. The design is required to produce the same length of the waveform segment t. To suit the length of the target waveform, the mel spectrogram passes through an upsampling network (the number of mel channels remains equal). A resnet-like model also uses the spectrogram as an input to generate features that will condition the layers during the mel spectrogram's transformation into a waveform. To match the length of the waveform segment, the resulting vector is repeated. This conditioning vector is then split into the channel dimension equally by four ways, and the first part is concatenated with the upsampled spectrogram and the previous time step's waveform segment. With skip connections, the resulting vector undergoes several transformations: first two layers of GRU, then a dense layer.

VI. TESTING AND DEBUGGING

To be sure that the initial model was giving appropriate results before committing to the larger process, the model was tested on an online machine learning environment provided by Google, known as Google Collab. The initial issues in testing were related to audio input errors and environment configurations. Those were ironed out after relevant problematic sections of code. The screenshots of the testing process are attached below:

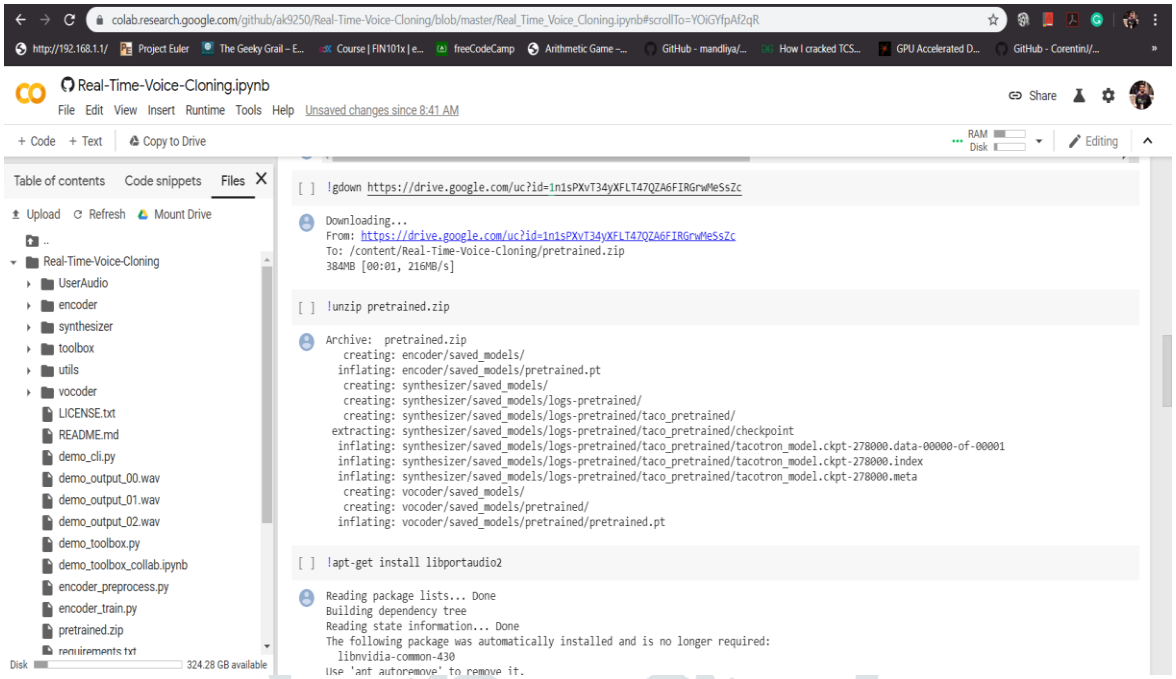

Figure 6.1: Uploading the project for testing on Google Collab

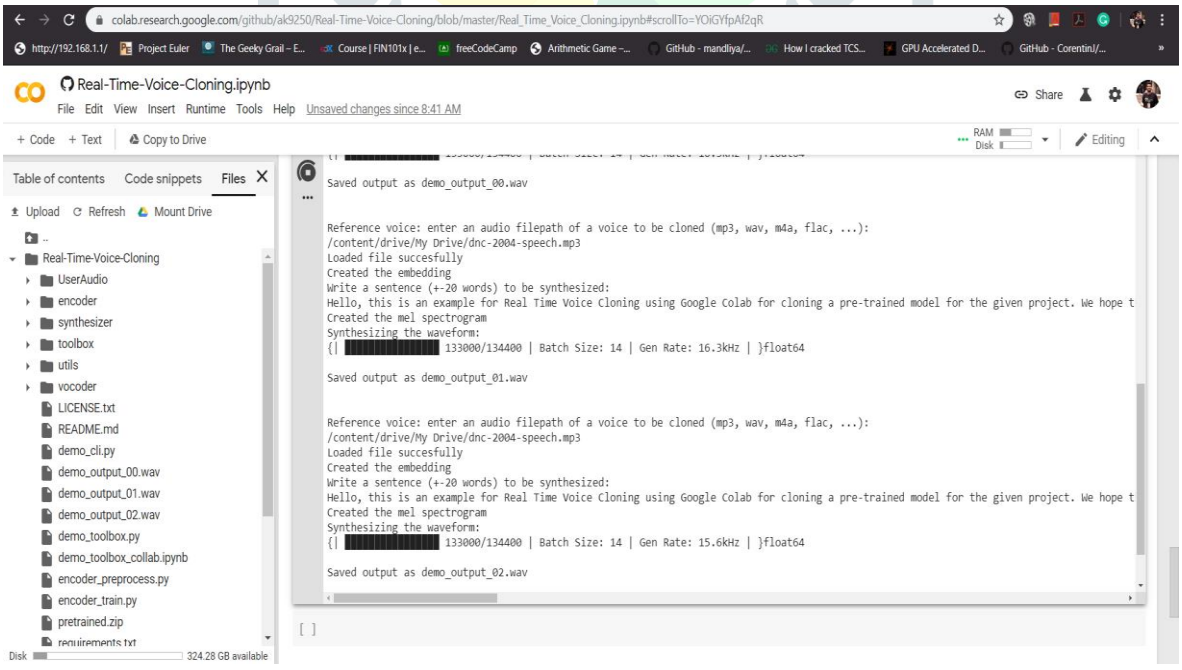

Figure 6.2: The output obtained on Google Collab while testing. This signified successful testing of the initial project.

VII. DEBUGGING

The major problems that occurred during the implementation and testing were environment and installation related issue. Hence reinstalling modules like PyTorch and Webrtcvad with lower version spec and customization helped. Another error that we encountered was the toolbox used to open for an instance before closing again immediately. After adequate research it was found that this happened because no root directory had been passed to the dataset path. After passing the required root directory command

we found the system to be working properly. Other errors that were encountered were trivial and were solved with just a review of the code.

VIII. RESULTS

With the successful working and execution of the project, the system was able to successfully clone a given voice audio through text-to-speech synthesis. As quality of voice is a subjective property, it is not possible to assign a quantitative measure for the accuracy reached by the results. The best approach would be to calculate the Mean Opinion Score (MOS) where a survey can be held to get an opinion whether the obtained cloned audio output compares with the natural human voice. After an extensive survey of MOS it can be said that the voice cloned through the given system is closely comparable to original human voice but lacks in naturalness and accent, parameters which can be worked upon.

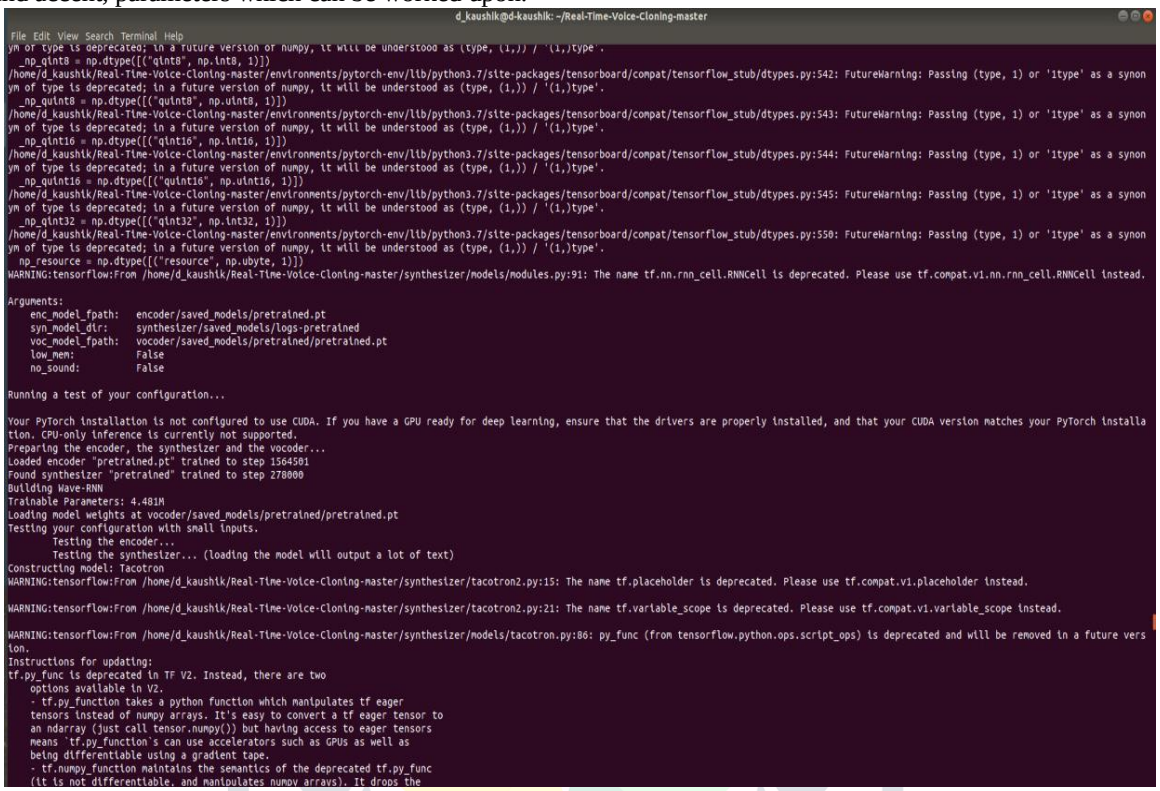

Figure 8.1: The working of the project in shell environment using CUDA.

The shell working of the project affirms that the processes are working in desired manner and further gives the chance of development of the project if necessary. A graphical interface was created allowing users to quickly, and without first having to study it to, get their hands on the framework. It is named the "SV2TTS toolbox." The toolbox interface can be seen in Figure 22. It is written with the graphical interface Qt4 in Python and is thus cross-platform.

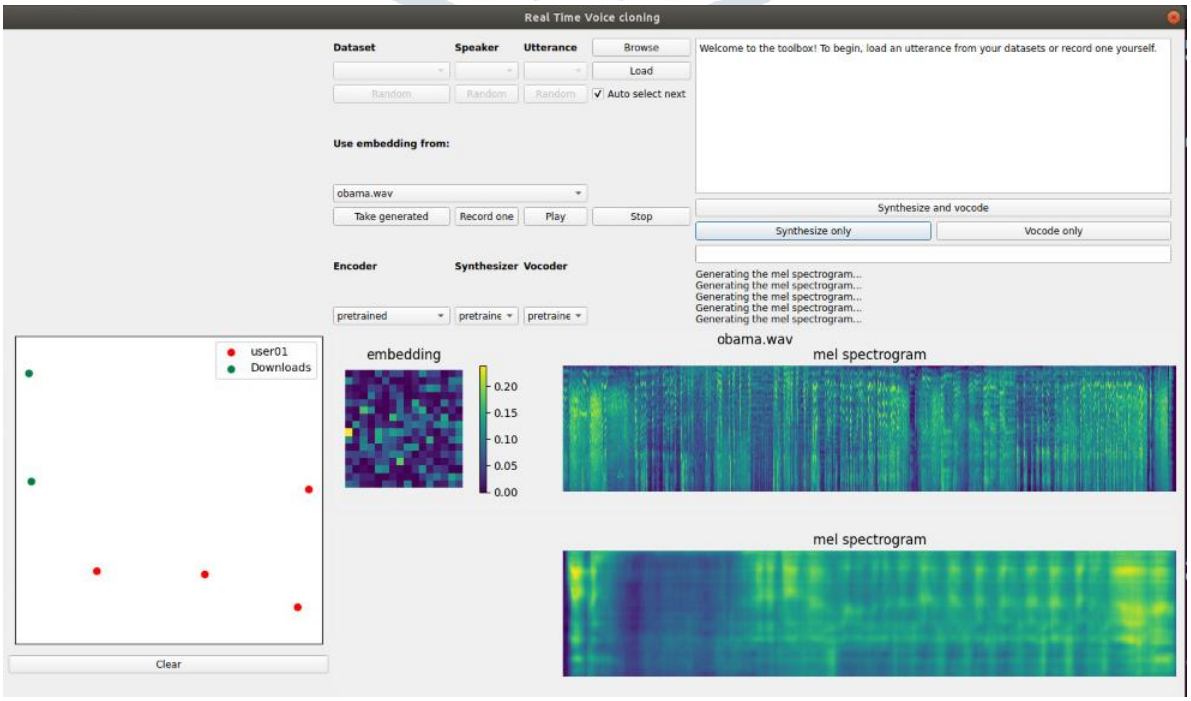

Figure 8.2: The SV2TTS Toolbox used for graphical interfacing of the project. It is a cross platform

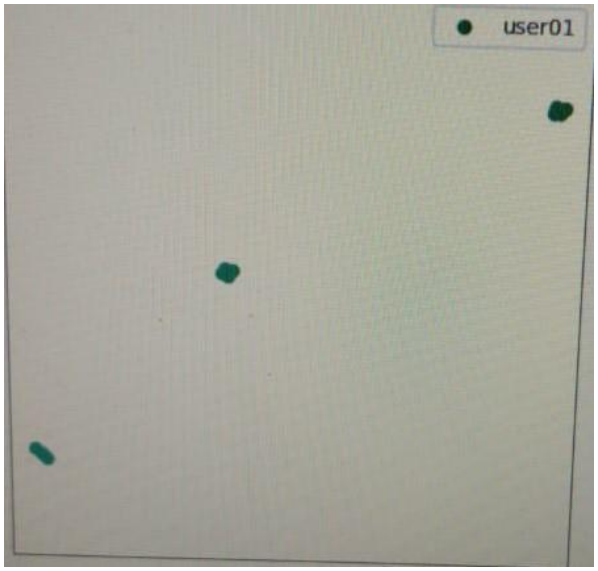

Figure 8.3: User audio clustering based on similarity of embeddings

A client starts by choosing an audio utterance from any of the data sets stored on their disk. The toolbox manages many common voice datasets, and new ones can be modified to include. In addition, the client can record utterances to replicate their own speech as well.

Once an utterance is loaded, it will calculate its embedding and automatically update the UMAP projections. The utterance's mel spectrogram is drawn (middle row on the right), but this is for comparison only, as nothing is measured. Remember that embedding is one-dimensional vector and therefore the square shape has no structural significance with respect to embedding values. Drawing embedding gives visual signs of the difference between two embeddings.

The client could write any arbitrary text to be synthesized (top right of the interface). As a reminder, our template does not support punctuation and it will be discarded. The user must insert line breaks between parts that should be synthesized individually to adjust the prosody of the generate utterance. Then the concatenation of those parts is the complete spectrogram. It will be shown at the bottom right of the interface by synthesizing a spectrogram.

The client will finally create the section with the vocoder that corresponds to the synthesized spectrogram. A loading bar shows the generation advancement. When completed, the synthesized utterance embedding is produced (on the left of the synthesized spectrogram) and is also projected with UMAP. The client can take the embedding as a guide for generation to come.

Opinion Set	Naturalness	Similarity	Source
VCTK	4.28 ± 0.05	1.82 ± 0.08	Jia et al 2018 paper
LibriSpeech	4.01 ± 0.06	2.77 ± 0.08	
This Project	4.10 ± 0.12	2.19 ± 0.20	Survey (25 people)

Table 8.1: Final MOS results obtained from a personal survey

The above table is the summary of the final result. It tries to give the best subjective measure that can be obtained to understand the quality of the audio generated through our project.

IX. CONCLUSION

The project successfully developed a framework for real-time voice cloning that had no public implementation. The results are found to be satisfying despite some unnatural prosody, and the voice cloning ability of the framework to be reasonably good but not on par with methods that make use of more reference speech time. There is still scoping to improve the given framework beyond the scope of this project, and possibly to implement some of the newer advances in the field that were made at the time of writing. It can be inferred that the aforementioned project and research is one of the latest approaches to audio processing (text-to-speech) and voice cloning using sophisticated deep learning networks and improving previously tried approaches that give us better similarity and naturalness of the generated speech. While it's agreed that our design and toolbox will be one of the improved TTS prototype versions, it can also be affirmed the hypothesis that better and more advanced models will be developed in the future for the same technology field. Therefore, it's found that the approach to be an attempt to understand, implement and innovate using the expertise that has been acquired in researching this venture. We believe that even more powerful forms of voice cloning will become available in a near future.

X. ACKNOWLEDGEMENT

We would express our gratitude towards our mentor, Mrs. A. M. Kulkarni for being of great support and guiding us through the research. She gave this paper the insight and the expertise it needed for making it a presentable one. Her advice, professional acumen, and encouragement proved to be a valuable guidance.

REFERENCES

[1] Corentin Jemine, Gilles Loupe. Master Thesis: Automatic Multispeaker Voice Cloning. Faculty of Science Applications, University of Liege. URL http://hdl.handle.net/2268.2/6801

[2] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. CoRR, abs/1609.03499, 2016. URL http://arxiv.org/abs/1609.03499.

[3] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis, 2018.

[4] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and YonghuiWu. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. CoRR, abs/1712.05884, 2017. URL http://arxiv.org/abs/1712.05884.

[5] S. Shirali-Shahreza and G. Penn. Mos naturalness and the quest for human-like speech. In 2018 IEEE Spoken Language Technology Workshop (SLT), pages 346 to 352, Dec 2018. doi: 10.1109/SLT.2018.8639599.

[6] Sercan Arik, Gregory Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech, 2017.