iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VIII **Month of publication:** August 2022

DOI: https://doi.org/10.22214/ijraset.2022.44524

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VIII Aug 2022- Available at www.ijraset.com

Voice Cloning in Real Time

Varad Naik[1], Aaron Mendes[2], Saili Kulkarni[3], Saiesh Naik[4], Saiesh Prabhu Verlekar[5]
[1, 2, 3, 4]*Students,* [5]*Project Guide, Department of Information Technology, Shree Rayeshwar Institute of Engineering and Information Technology Goa, India*

Abstract: ***Deep learning models for natural voice cloning methods were first developed in 2016, and since then, the researchers' main attention has been on making the voice more realistic and obtaining the output voice in real time. Previously it used to take many hours of voice samples to clone a few seconds. It was decreased to a few seconds after utilizing deep learning models. We shall look at various voice cloning techniques in this paper. Multi-speaker generative models, speaker adoption, speaker encoding, vector quantization, and other techniques are among them.***
Index Terms: ***Voice cloning, text-to-speech, voice conversation, speaker adaption, speaker encoding, vector quantization, neural network.***

I. INTRODUCTION

Voice conversion is defined as changing the voice of the source speaker to the voice of the target speaker without any change in language speaker. The voice conversion system should modify the tone, accent and pronunciation of the source speaker's voice in order to imitate the target speaker. Even after decades of research the process of producing a natural speech from a written input remains a challenging task[2]. There are many test-to-speech (TTS) methods which can give great results in terms of synthesis of natural voices similar to human ones. Many of these systems only manage to learn to synthesis text with a single voice. The project goal is creating a system which effectively generates a natural and data efficient speech for different speakers. This technology can be applied in various fields like entertainment and creativity. With this technology restoring of voice or customizing digital assistance such as Siri is also possible[5]. A voice conversion consists of supervised and unsupervised scenario's. one to one or many to one conversion systems such as GMM based and regression based models which converts one or many speakers voice into specific targets speakers is supervised learning. Higher voice similarity and speech quality is achieved by learning to map the data into specific target distribution, but frame level alignment on training data is needed which may lack flexibility[10]. The many to many voice conversion problem was successfully tackled by unsupervised vc which does not need parallel data and most of the work achieved good performance on seen speakers. Generative adversarial network (GAN) is one of the leading voice conversion techniques moving to its distribution guarantee amongst the produced data and true data and the difference between the speaker is most obvious. Some works use vector quantization technique to extract content information and feed into WaveNet by combining with one-hot speaker embedding to synthesise the voice. This output produced is remarkably closer to natural human voice. However, they still face one problem that is they cannot synthesise the voice that does not exist in the training data[2]. One-shot technique can resolve the unseen speaker problem. One-shot model only requires one utterance from the source speaker and the target speaker. Proposed IN-based oneshot VC, which is a technique widely used in the computer vision, and it shows that this approach can learn the meaningful speaker embedding. The important elements are: Multi speaker generative model- In speech synthesis, generative models (speaker generative model) can be conditioned on text and speaker identity. Although the text contains language proficiency and controls the content of the produced speech, the speaker identity captures features such as tone, speech rate, and pronunciation[1]. Speaker adaption - Speaker adaptation is based on fine-tuning a multi-speaker generative model with a few cloning samples [1]. Speaker encoding - 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 [1]. Vector quantization - Vector quantization is a method of constraining an input from a continuous or otherwise large set of values by using a shared codebook [2].

II. LITERATURE SURVEY

To develop the idea, many different methods were researched and tried to find a suitable implementation, an attempt was made to perform voice conversion that could transform voice from audio to source without losing the essential contents in a language. They used different methods such as voice conversion, vector quantization, contrastive predictive coding, augmentation. The content encoder achieves very good results in capturing the content information.[7] speaker encoding gives fast output but the voice feels little synthetic as compared to speaker adoption [1].

The research also showed that VQVC learns a meaningful embedding space without any supervision. However, some researcher's say that the naturalness of the human voice has already been reached[5]

III. IMPLEMENTATION

The implementation of the entire system involves processes such as installing required programs, Creating an environment to run the project properly, integrating the datasets, coding, and implementing of Encoder, Synthesizer, and Vocoder. All the coding and required implementations have been done using Python language, as mentioned Python language is used for coding and implementation. Open-source technologies such as Google Tacotron 2 and wavenet modules also have been used.

A. Installations

The mandatory installations which are required to implement the project are:

1) TensorFlow GPU(1.10.0=<Version<=1.14.0).
2) Umap-learn: provides the UMAP manifold based dimension reduction algorithms.
3) Visdom: is a visualization tool that generates rich visualization of live data.
4) Webrtcvad: is a python interface for WebRTC VAD (Voice Activity Detector).
5) Librosa (0.5.1=<version): is a python package for music and audio analysis.
6) Sounddevice: python module to play and record numpy arrays containing audio signals.
7) Unidecode: Library that decodes Unicode.
8) PyTorch: machine learning library.
9) Inflect: it is a module that correctly generates plurals, ordinals, singular nouns, etc.

The above installations were done using the Python pip command in the python shell. After the installation has been completed it is necessary to configure the new installations to build the right environment. This is done with the help of a demo_cli.py file. If the demo_cli.py file is working properly then we have successfully installed and configured the requirements.

B. Dataset Used

The researchers found two different datasets in SV2TTS to train the synthesizer and vocoder. These are LibriSpeech-Clean, and VCTK10, a group of only 109 native English speakers recorded their voices using professional equipment. In all their tests, the VCTK expression was sampled at 48kHz and reduced to 24kHz, which is much higher than LibriSpeech's 16kHz sampling. When it comes to similarity, the synthesizer which is trained on LibriSpeech generalizes is better than on VCTK, however, this comes at the expense of natural speech.

C. Implementation Of Speaker Encoder

The first module to be trained is the Speaker Encoder. It deals with the system's audio input and so includes preprocessing audio training, and visualization models. The Speaker Encoder consists of a 768-hidden-node LSTM3 layer followed by a 256-unit projection layer. While no mention of a projection layer is made in any of the publications, we believe it is just a full-connected layer with 256 outputs per LSTM that is applied to each LSTM output repeatedly. Instead of developing the speaker encoder from scratch, 256 LSTM layers can be used directly for quick prototyping, simplicity, and a lower learning curve. The results are 40-channel log-Mel spectrograms with a 25-ms window width and a 10-ms stage. The L2-normalized hidden state of the last layer, which is a 256-element vector, is the output. A pre-standardization ReLU layer is also included in our solution, with the purpose of making embedding sparse and hence easier to read.

D. Implementation Of Synthesizer

The Synthesizer is Google Tacotron 2 model and Wavenet is not used. Tacotron is a system that predicts a text-based Mel spectrogram using a repeating sequence-to-sequence method. From the text string, specific characters are initially added as vectors. Standard layers are used to extend the length of a single encoder block. These frames are run via a bidirectional LSTM to create the encoder output frames. The embedding of a speaker is concatenated with each frame that the Tacotron encoder creates, which is where SV2TTS makes a change to the architecture. The attentiveness function pays to the encoder output frames in order to generate the decoder input frames. The input texts are not tested for pronunciation in our implementation, and the characters are fed as is. However, cleaning methods include replacing acronyms and numerals with their full-text form, converting all characters to ASCII, normalizing white spaces, and reducing all characters. Punctuation may be utilized, however, it is not included in our datasets.

E. Implementation Of Vocoder

Because the modules are supposed to be taught in the order Encoder Synthesizer-Vocoder, the Vocoder module is the final to be trained.. In both SV2TTS and Tacotron2, the vocoder is WaveNet. The vocoder model utilized is an open-source PyTorch implementation15 that is based on WaveRNN but has a few different user fat chord design options. "The alternate WaveRNN" is the name given to this architecture. Each training step divides a Mel spectrogram and its related waveform into the same number of segments. The design inputs are segment t of the spectrogram to imitate and segment t-1. The waveform segment t must be the same length throughout the design. The Mel spectrogram goes through an upsampling network to adjust the length of the target waveform (the number of Mel channels remains equal). The spectrogram is also used as an input by a resnet-like model to create features that will condition the layers during the Mel spectrogram's transformation into a waveform. The resulting vector is repeated to fit the length of the waveform segment. This conditioning vector is then divided into four parts, each of which is concatenated with the upsampled spectrogram and the waveform segment from the preceding time step. The generated vector undergoes many modifications as a result of skip connections: first two GRU layers, then a dense layer.

IV. TESTING

The model was evaluated on Google Collab, an online machine learning environment provided by Google, to ensure that the initial model was producing adequate results before moving forward with the bigger procedure. The first problems encountered during testing were audio input failures and environment setups. After relevant problematic code sections were smoothed out, those were ironed out.

V. DEBUGGING

The biggest issues that arose during implementation and testing were connected to the environment and installation. As a result, reinstalling PyTorch and Webrtcvad with a lower version spec and customizing them helped. Another issue we encountered was the toolbox, which would open for a brief period of time before closing again. After further investigation, it was discovered that this occurred because no root directory was supplied to the dataset path. We discovered that the system was operational after giving the required root directory command. Other issues were minor and could be resolved by simply going over the code again.

VI. RESULTS

After a thorough examination of MOS, it can be concluded that the voice cloned by the system is very similar to the original human voice, but it lacks naturalness and accent, both of which are variables that can be improved. The project's shell working confirms that the procedures are operating as expected, as well as providing the opportunity to expand the project if necessary. A graphical interface was built to allow people to get their hands on the framework fast and without having to read it first. The "SV2TTS toolbox" is its name. A client begins by selecting an audio utterance from any of their disk's data sets. Many common voice datasets are managed by the toolbox, and new ones can be added. Additionally, the customer can record utterances in order to mimic their own speech. Once an utterance has been loaded, it will calculate its embedding and update the UMAP projections automatically.

The Mel spectrogram for the speech is presented (middle row on the right), however, it is simply for comparison purposes because nothing is measured. Because embedding is a one-dimensional vector, the square shape has no structural importance when it comes to embedding values. The difference between two embeddings can be visually shown by drawing embedding.

Any arbitrary text could be written by the client to be synthesized (top right of the interface). Our template, as a reminder, does not support punctuation and will be discarded. To change the prosody of the generated speech, the user must enter line breaks between elements that should be synthesized independently. The full spectrogram is then the concatenation of those segments. It will be displayed in the bottom right corner of the interface as a spectrogram. Finally, the client will generate the vocoder segment that matches the synthesized spectrogram.

The progress of the generating is indicated by a loading bar. The synthesized utterance embedding is created (on the left of the synth esized spectrogram) and projected with UMAP once it's finished. The client can use the embedding as a template for future generati ons.

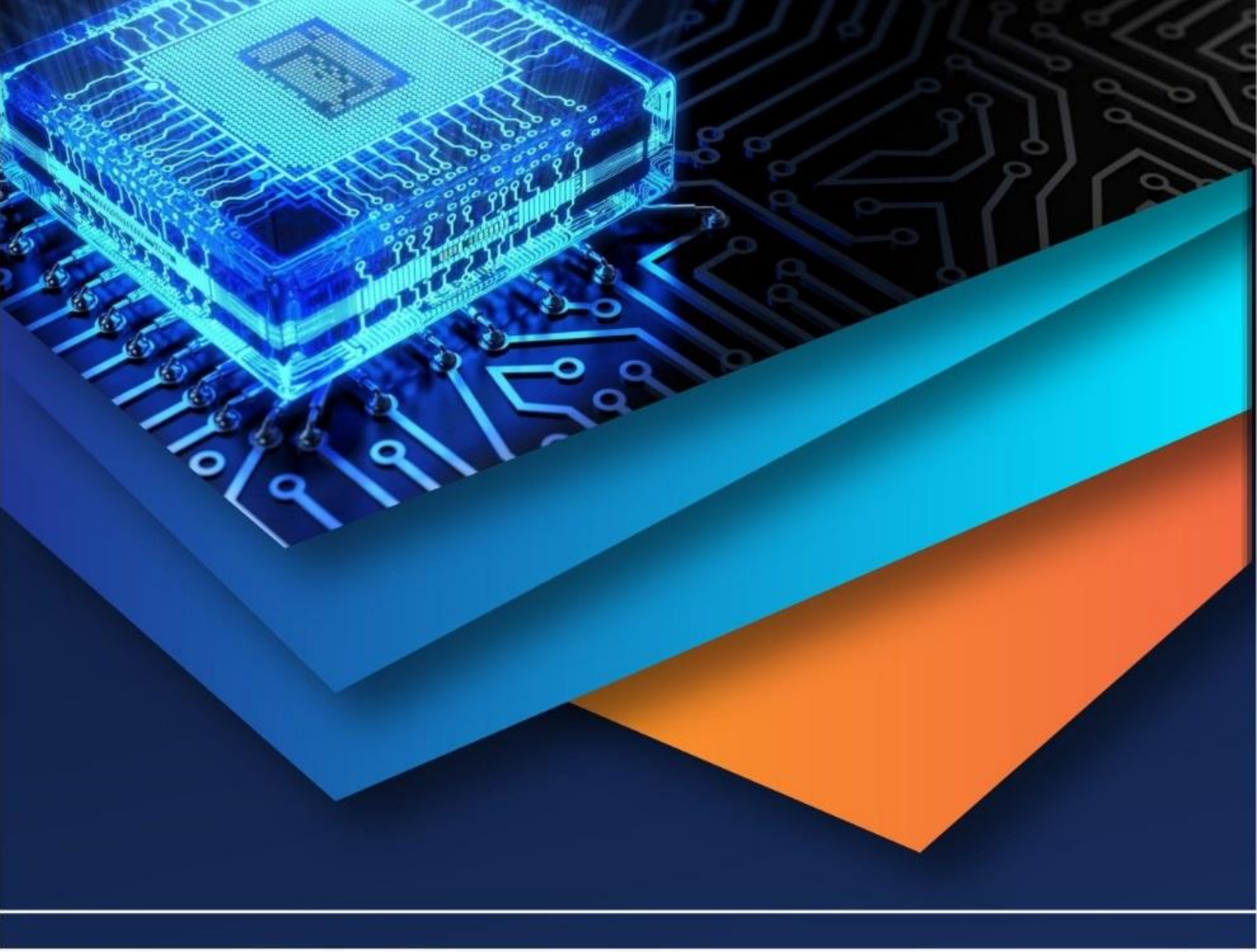
VII. CONCLUSION

This project has successfully developed a real-time voice integration framework that did not have a public implementation. Despite some unusual prosody, the outcomes are satisfactory, and the framework's voice cloning ability is reasonable but not on par with methods that require more reference speech time. Beyond the scope of this project, there is still space to improve the supplied framework and possibly apply some of the more recent improvements in the field that have been made since the writing of this document.

The aforementioned project and research can be inferred to be one of the most recent 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 is accepted that our design and toolset will be one of the enhanced TTS prototype versions, it can also be stated that better and more advanced models for the same technology sector will be produced in the future. As a result, the method has been determined to be an endeavor to comprehend, execute, and develop using the knowledge gained while exploring this enterprise. We believe that in the near future, even more powerful kinds of voice cloning will become available. As a result, the method has been determined to be an endeavor to comprehend, execute, and develop using the knowledge gained while exploring this enterprise. We believe that in the near future, even more powerful kinds of voice cloning will become available.

REFERENCES

[1] Sercan Ö. Arık, Jitong Chen, Kainan Peng, Wei Ping, Yanqi Zhou. Neural Voice Cloning with a Few Samples. Baidu Research 1195 Bordeaux Dr. Sunnyvale, CA 94089.
[2] Da-Yi Wu, Hung-yi Lee. One-shot Voice Conversion by Vector Quantization. College of Electrical Engineering and Computer Science, National Taiwan University (x07922119, hungyilee)@ntu.edu.tv.
[3] Ju-chieh Chou, Cheng-ChiehYeh. To implement one shot voice cloning by separating speaker from sourse and target speaker.
[4] Paarth neekhara, Shehzeen Hussain. Expressive neural voice cloning.
[5] Giuseppe Ruggiero, Enrico Zovato. Voice cloning: A multi speaker TTS synthesis approach based on transfer learning.
[6] Merlin Blaauw, Jordi Bonada, and Ryunosuke Daido. Data Efficient Voice Cloning For Neural Singing Synthesis.
[7] Shilun Wang, Damian Borth. Towards High Quality Zero-Shot Voice Conversion.
[8] Hieu-Thi Luong and Junichii Yamagishi. Nautilus: A Versatile Voice Cloning System.
[9] Kaushik Daspute, Hemang Pandit. Real Time Voice Cloning.
[10] Yen-Hao Chen, Da-vi wu. Again-VC A one shot voice conversion using activation guidance and AdalN.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY