REAL TIME VOICE CLONING

[1]Dr.T NAnitha, [2]Amilio Dsouza, [3]Ashutosh, [4]Akshay Gole

[1]Professor, [2,3,4]UG Students
Department of Information Science and Engineering
Atria Institute of Technology, Bangalore, Karnataka, India.

***Abstract*: A latest study has advanced a three-level pipeline that lets in you to clone an unseen voice from just a few seconds of reference speech all through exercise and without retraining the template. The researchers percentage strikingly natural sounding findings. The plan is to copy this model and open source it to the public. With a new vocoder version, the aim is to adapt the framework to make it run in real time. The purpose is to expand a three degree deep getting to know the gadget so one can perform real-time voice cloning.**

The recent development of the Deep Study has shown amazing results in the localization of Speech to Text. For this reason, the deep neural community usually consists of a single speaker using a professionally recorded hours of audio corpus. Revoking this type of model is very costly as it requires accumulating a whole new dataset and retraining the version. This framework is the final result of Google's 2018 paper, and just a single public implementation exists before us. The system may want to capture a practical representation of the voice spoken in a virtual layout from a speech of just 5 seconds. Now that the text content has been dropped, you can use any voice extracted from this process to perform a text-to-speech. Looking forward for our own implementation or the Open-source implementation plans to duplicate each of the three stages of release.

The plan is to implement a successful model of the corresponding pipeline with in-depth knowledge for the preprocessing of facts. The next step is to train these models of thousands of audio systems for weeks or months with large datasets of tens of hours of audio. Instead, observe their strengths and weaknesses. We are primarily aware that this system works in real time, that is, it allows us to capture and generate audio in a time that is much shorter than the length of the produced audio. The framework may be able to duplicate audio that you have never heard at any stage of training and generate audio from text content that you have never seen.

***Keywords*: Neural Networking, 3-stage pipeline, Text-to-Speech, Deep Learning.**

INTRODUCTION

Deep knowledge of models has been established in many areas of computational machine learning. The approach of synthesizing synthetic speech from text-to-speech (TTS), which is a spark of text content, is no exception. In 2016, there was a deep epidemic that could produce more natural speech than traditional consolidation strategies. Much research has focused on improving the performance of these low-end models, making the sound more natural, and training in a stop-to-give manner. The example on GPU is because many of the instances are slower than real time on mobile CPUs. Paradoxically, the naturalness of speech is better measured by subjective metrics. And the correlation with the actual human speech leads to the assumption that "a speech that is more natural than human speech" may exist. Some argue that the limits of humanity have already been reached, but getting a TTS version on a single speaker is all theoretically in the form of a voice clone, but the goal is to use a fixed version instead and develop something that allows you to integrate your new voice with fewer information. A known method is to tune a TTS template trained to generalize speech and replicate it to a new audio. The low-dimensional embedding is derived from the speaker-encoder version that receives the reference audio as input. In general, this technique is in fact more efficient, faster, and less computationally expensive than creating a separate TTS version for each speaker. Apparently, there can be a big difference between the length of the reference speech that is important for cloning the voice and the various strategies that vary from 30 minutes to just a few seconds depending on the speaker. This usually determines the similarity between the cognitively generated voice and the speaker's real voice 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.

LITERATURESURVEY

Paper I: Real Time Voice Cloning

A three-step pipeline that allows you to replicate invisible audio from reference audio in just a few seconds during practice without retraining the template. The researchers shared a very natural sounding audio output. The plan for this project is to clone this model and make it publicly available. The new vocoder model adjusts the framework to run in real time. The goal is to develop a three-stage deep learning system that performs real-time voice cloning.

Paper II: Neural Voice Cloning With a Few Samples

Voice Cloning is a preferred feature in personalized voice interfaces. Neural network-based speech synthesis has been shown to produce high quality speech for large numbers of speakers. This article introduces a neural voice cloning system that takes fewer audio samples as input. We consider and study two approaches: speaker adaptation and speaker encoding. Speaker adaptation is based on fine-tuning the generated multi-speaker model using several clone samples. The speaker encoding is based on training another model to infer a new speaker embedding directly from the audio duplication and use it in a multi-speaker generation model.

In terms of the genuineness of the speech and its likeness to the original speaker, both approaches turn out well even with very few cloned audio. Speaker adaptation can achieve better naturalness and similarity, but requires significantly less cloning time or memory for the speaker encoding approach, making it suitable for resource-poor deployments.

Paper III: WAVENET: A Generative Model For Raw Audio
This article presents WaveNet, a deep neural network for generating raw audio waveforms. The model is completely probabilistic and autoregressive, and the predicted distribution of each audio sample depends on all previous samples. Still, it shows that you can efficiently train your data with tens of thousands of samples per second of audio.When applied to text-to-speech, it provides state-of-the-art performance and is rated by human listeners as a much more natural sound than the best English and Mandarin parametric and concatenated systems. With one WaveNet, you can capture the characteristics of different speakers with equal accuracy and switch speakers by adjusting the speaker ID. When we are trained to model music, we find that it produces novel and often very realistic pieces of music. It can also be used as a discriminative model, showing promising results for phoneme recognition.

Paper IV: Efficient Neural Audio Synthesis:
The Sequential Model provides state-of-the-art results in the audio, image, and text domains, both in terms of data distribution estimation and high quality sample generation. However, efficient sampling of this class of models remains an elusive problem. Focusing on text-to-speech synthesis, we will discuss a set of common techniques for reducing sample time while maintaining high output quality. First, we will discuss WaveRNN, a single-layer recurrent neural network with a double softmax layer that matches the quality of the state-of-the-art WaveNet model. The compact format of the network allows GPUs to generate 24kHz 16-bit audio four times faster than real time. Then apply a weight pruning technique to reduce the number of weights in the WaveRNN. You can see that large sparse networks perform better than small dense networks for a certain number of parameters. This relationship also applies to sparsity levels above 96%. The low number of weights on the Sparse WaveRNN allows you to sample hi-fi audio in real time on your mobile CPU. Finally, we put forward a new generation scheme based on sub-scaling. This collapses a long sequence into a stack of short sequences, allowing multiple samples to be generated at the same time. Subscale WaveRNN produces 16 samples per step without compromising quality and provides an orthogonal method to increase sample efficiency.

Paper V: Natural TTS synthesis by conditioning Wavenet onMel spectrogram predictions:
This paper illustrates the Tacotron 2, a neural network architecture for speech synthesis directly from text. The system consists of an iterative inter-sequence functional prediction network that maps character embeddings to Melscale spectrograms and a modified WaveNet model that acts as a vocoder that synthesizes time domain waveforms from these spectrograms. Our model scores a Mean Opinion Score (MOS) of 4.53. This is comparable to MOS 4.58 for professionally recorded voice. To validate the design decision, we present an ablation study of the key components of the system and evaluate the impact of using the Mel spectrogram as input to WaveNet instead of voice, duration, and F0 functions. In addition, a compact intermediate acoustic representation shows that the WaveNet architecture can be greatly simplified.

Paper VI: VOICE CLONING: A Multi-Speaker Text-To-Speech Synthesis:
Deep learning models are becoming mainstream in many areas of machine learning. Text-to-Speech, the process of synthesizing an artificial speech from text, is not an exception. For this purpose, deep neural networks are typically trained using a corpus of recorded audio from a single speaker for several hours. Attempting to generate a speaker voice that is different from the one you learned is costly and labor intensive, as you will have to acquire a new dataset and retrain the model. This is the main reason why TTS models are usually single speakers. The proposed approach aims to overcome these limitations by trying to obtain a system that can model a multi-speaker acoustic space. This allows you to generate sounds that resemble the sounds of different target speakers, even if they were not observed during the training phase.

PROPOSED SYSTEM

The basic components of the whole program can be expressed as follows.

The important elements are:

- Feature Extractor: -The role of the feature extractor is to provide data that better shows how the voice produced by the model sounds.
- Acoustic Model:- The relationship between the calculated features and the output acoustic features for the input text is learned by a statistical generative model called the acoustic model.
- Vocoder: A system that can reconstruct audio waveforms from the acoustic features generated by an acoustic model. Derivate the audio waveform from the spectrogram generated by the synthesizer.
- Speaker Encoder: Derives embedding from short utterances of a single speaker. Embedding is a meaningful expression of the speaker's voice, where similar voices approach each other in potential space.
- Synthesizer: The embedded speaker creates a spectogram from the text. This model is the popular Tacotron 2, which does not uses the WaveNet.

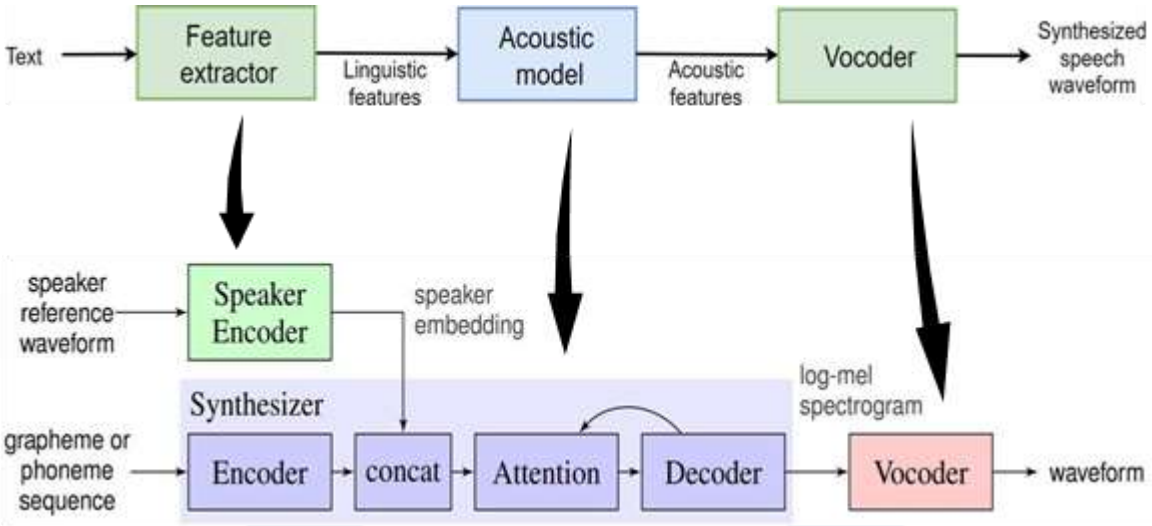
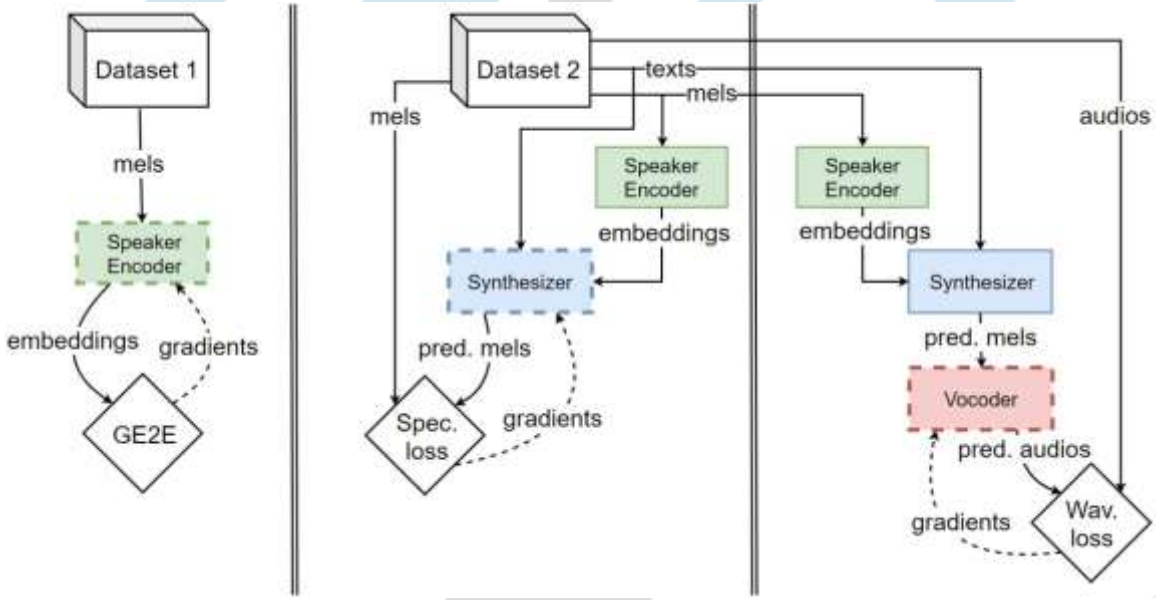


Figure 1: System Block Diagram

FLOW CHART

The above diagram elaborates the flow of the processes we plan to use for the execution of our project. The Encoder takes in the input audio and creates voice embeddings which have the characteristics of the unique speaker voice. The Synthesizer generates a grapheme or phoneme sequence from the input text using Machine learning algorithms. The outputs of the encoder and synthesizer generates a Mel-Spectrogram that is used by the vocoder to furnish the final cloned voice output in the most aspirated voice without the need to train the system again.

Figure 2: The System Flowchart



IMPLEMENTATION

V.I Implementation

The system-wide implementation includes processes such as installing prerequisites, configuring the environment for the project to function properly, retrieving datasets, encoding and implementing encoder modules, synthesizer modules, and vocoder modules. As mentioned above, all coding and required implementation was done in Python 3.

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 installation was done using the Python pip in Python shell. Once the installation is complete, you need to configure the new installation to create the appropriate environment for our work and accessing our project work. This is done using the demo_cli.py file. If the demo_cli.py file runs successfully, the requirements have been successfully installed and configured.

V.III Dataset used

Researchers have found two datasets on the SV2TTS to train both synthesizers and vocoders. These are the LibriSpeechClean mentioned above and the VCTK10, a panel of only 109 native English speakers recorded using professional equipment. During the test, VCTK audio is sampled at 48 kHz and downsampled to 24 kHz. This is still higher than LibriSpeech's 16kHz sampling. Synthesizers trained in LibriSpeech are more generalized in terms of similarity than VCTK, but at the expense of natural language. The record was public and a simple inquiry email to the University of Edinburgh (the original owner of the record) was sufficient to get the database download link.

V.IV Implementation of Speaker Encoder

The first module to be trained is the Speaker Encoder. It handles the audio input provided to the system and hence includes the preprocessing, audio training and visualization models. The Speaker Encoder is a LSTM3-layer comprised of 768 hidden nodes and a 256-unit projection layer. Since we did not find any reference to what a projection layer is in any of the articles, we believe that it is simply a closely networked layer of 256 outputs per LSTM that is iteratively 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. Here we get a 40-channel log-mel spectrograms as our output with a window width of 25ms and a stage of 10ms. The last layer's L2-normalized hidden state is the output (which is a 256-element vector).A pre-standardization ReLU layer with the goal of making embedding sparse is also featured in our implementation and thus easier to interpret.

V.VI Implementation of Synthesizer

The synthesizer used is the Google Tacotron 2 model used without Wavenet. Tacotron is an iterative intersequence system that predicts text-based mel spectrograms. Certain characters are first inserted as a vector from the text string. Followed by standard layers to increase the length of a single encoder block. These frames go through bidirectional LSTMs to create encoder output frames. Now, this is the point where SV2TTS makes changes to the architecture. The speaker embedding is tied to each frame created by the Tacotron encoder. To generate a decoder input frame, the attention function processes the encoder output frame. In our implementation, the pronunciation of the input text is not checked and the characters are provided as is. Still, there are some cleaning steps. Replace abbreviations and numbers with full-text format, move all letters to ASCII, normalize spaces, reduce all letters. Punctuation marks can be used, but they are not in the record.

V.VI Implementation of Vocoder

The modules in the sequence are expected to be trained by the Encoder Synthesizer Vocoder, so the Vocoder modules are trained last. WaveNet is a vocoder for SV2-TTS and Tacotron 2. The vocoder model used is the open source PyTorch implementation 15, based on WaveRNN, but with several different user fatchord design options.This architecture is called "Alternative WaveRNN". The mel spectrogram and its corresponding waveform are divided into the same number of segments in each training phase. The design inputs are segment t and segment t-1 of the simulated spectrogram. It should be designed to produce waveform segments t of the same length. The mel spectrogram goes through the upsampling network to match the length of the target waveform (the number of mel channels remains the same). Models like resnet use the spectrogram as an input to generate features that adjust the layers as the mel spectrogram is converted to a waveform. The resulting vector is repeated to adjust the length of the waveform segment. This adjustment vector is then evenly divided into channel dimensions in four ways, with the first part concatenated with the upsampling spectrogram and waveform segment of the previous time step. With a skip connection, the resulting vector undergoes some transformations. First two GRU layers, then a high density layer.

TESTING AND DEBUGGING

The model was tested in a Google provided online machine-learning environment known as Google collab to ensure the original produced good results before it adopted by a larger process. The initial issues in testing were related to audio input errors and environment configurations. Those were resolved after relevant problematic sections of code. The screenshots of the testing process are attached below:

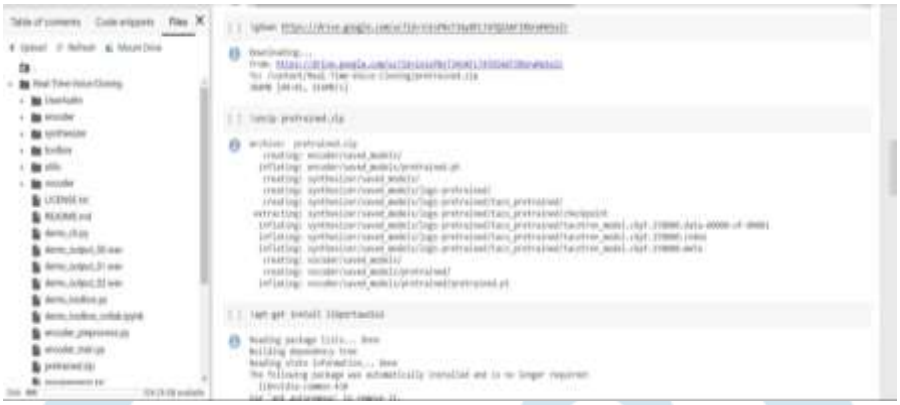


Figure 3: Uploading the project for testing on Google Collab

Figure 4: The output obtained on Google Collab while testing.
This signified successful testing of initial project

DEBUGGING

The main issues encountered during implementation and testing were environment and installation issues. Therefore, it was helpful to reinstall modules such as PyTorch and Webrtcvad with lower version specifications and customizations. Another bug we encountered was the toolbox used to open an instance before closing it immediately. Proper investigation revealed that this was happening because the root directory was not passed to the record path. After passing the required root directory command, I found that the system was working properly. The other errors I encountered were trivial and could be fixed by checking the code.

RESULT

With the error-free working and execution of the project, the system was able to successfully clone a given voice audio through TTS(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.

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 8.2. It is written with the graphical interface Qt4 in Python and is thus cross platform.

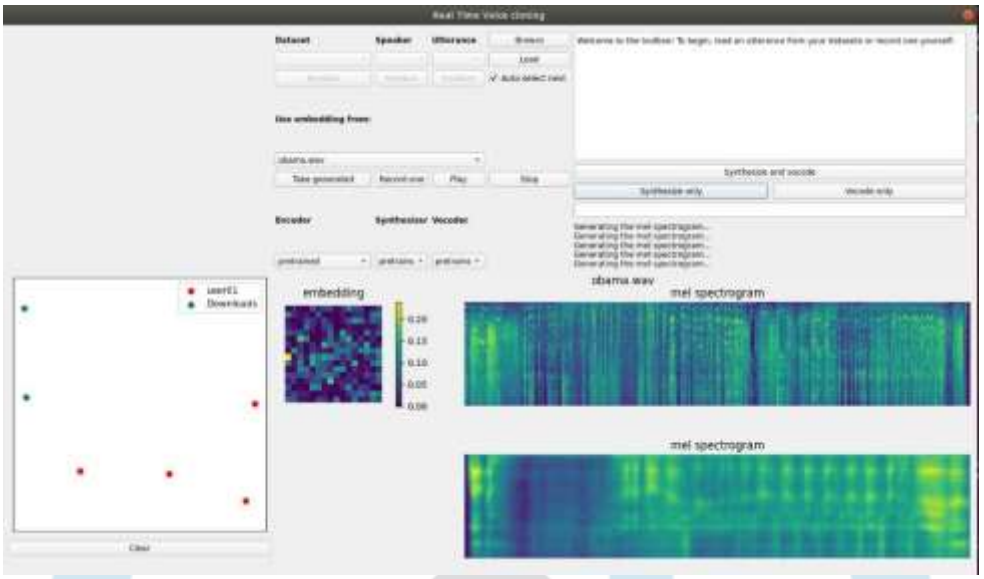


Figure 5: The SV2TTS Tool box used for graphical interfacing of the project.

The toolbox holds many common language records, and you can modify new records to include them. In addition, the client can record the utterance and duplicate his speech.

When the utterance is loaded, its embedding is calculated and the UMAP projection is automatically updated. A mel spectrogram of the utterance is drawn (center row on the right), but this is for comparison only as nothing has been measured. Note that the embedding is a one-dimensional vector, so the square shape has no structural meaning with respect to the embedding value. Drawing embeddings visually show the difference between the two embeddings.

The client can write any text to synthesize (upper right of the UI). As a reminder, the template does not support punctuation and is deprecated. The user needs to insert line breaks between the individually synthesized parts to adjust the rhythm of the generated utterance. The concatenation of these parts then becomes a complete spectrogram. By compositing the spectrogram, it will be displayed in the lower right corner of the interface.

The client lastly develops a section with the help of the vocoder that responds to the synthesized spectrogram. The import bar shows the progress of the generation. When complete, a composite utterance embedding is generated (on the left side of the composite spectrogram) and projected in UMAP. Clients can use embedding as a next-generation guide.

OpinionSet	Naturalness	Similarity	Source
VCTK	4.28 ±0.05	1.82 ± 0.08	Jiaet al 2018 paper
LibriSpeech	4.01 ± 0.06	2.77 ± 0.08	
ThisProject	4.10 ± 0.12	2.19 ± 0.20	Survey(25 people)

Table 1 : Final MOS Results obtained from a personal survey

The table above is a summary of the final results. It attempts to give the highest subjective measure that can be obtained to understand the quality of the audio produced by our project.

CONCLUSION

This project has successfully developed a framework for real-time voice cloning that has not been published. Despite some unnatural prosody, the results are satisfactory and the framework's ability to replicate speech is very good, but at the level of how to use more reference speech time. There is none. Beyond the scope of this project, there are still ways to improve certain frameworks and implement some of the recent advances in this area made at the time of writing. The above projects and research used advanced deep learning networks and improved the previously tested approach to generate speeches. While it has been agreed that our design and toolbox is one of the improved TTS prototype versions, we can also confirm the hypothesis that better and more advanced models for the same technical discipline will be developed in the future. Therefore, this approach proved to be an attempt to understand, implement and innovate the expertise gained during the company's research. We believe that more powerful forms of

voice clones will be available in the near future.

ACKNOWLEDGMENT
We would like to offer thanks to the project guide Dr. T.N. Anitha, thank you for guiding us through such great support and research. She gave the paper the insights and expertise needed to make it look good. Her advice, professional insight, and encouragement proved to be invaluable as a guide.

REFERENCES

[1] Gilles Loupe, Corentin Jemine. Master Thesis: Automatic Multispeaker Voice Cloning. Faculty of Science Applications, University of Liege. URL http://hdl.handle.net/2268.2/6801
[2] Sander Dieleman, Heiga Zen, Aaron van den Oord, Karen Simonyan, Nal Kalchbrenner, Andrew W. Senior, Oriol Vinyals, Alex Graves and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. CoRR, abs/1609.03499, 2016. URL http://arxiv.org/abs/1609.03499.
[3] Karen Simonyan, Seb Noury, Norman Casagrande, Nal Kalchbrenner, Erich Elsen, Edward Lockhart, Florian Stimberg,Sander Dieleman, and Koray Kavukcuoglu, Aaron van den Oord. Efficient neural audio synthesis, 2018.
[4] R. J. Skerry-Ryan, Rif A. Saurous, Jonathan Shen, Ruoming Pang, Ron J. Weiss,Yuxuan Wang, Yannis Agiomyrgiannakis, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, 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] G. Penn and S. Shirali-Shahreza. Mos-naturalness and the quest for human-like speech. In 2018 IEEE Spoken Language Technology Workshop (SLT), pages 346 to 352, Dec 2018. doi:1109/SLT.2018.8639599.
[6] Andrew Gibiansky,Jonathan Raiman, John Miller, Sercan Arik, Gregory Diamos, Kainan Peng, Wei Ping, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech, 2017.
[7] Giuseppe Ruggiero, Enrico Zovato, Luigi Di Caro, Vincent Pollet. Voice Cloning: a Multi-Speaker Text-to-Speech Synthesis Approach based on Transfer Learning, 2021. URL https://arxiv.org/abs/2102.05630.
[8] Corentin James. Real Time Voice Cloning, 2019.
[9] Sercan O Arik, Jitong Chen, Kainan Peng, Wei Ping,Yanqi Zhou. Neural Voice Cloning with a few Samples, 2018. URL https://arxiv.org/abs/1802.06006.
[10] Jian Cong, Shan Yang, Lei Xie, Guoqiao Yu, Guanglu Wan. Data Efficient Voice Cloning fom Noisy Samples with Domain Adversial Training, 2020.