OSM: One-Shot Multi-Speaker Text-to-Speech

Nikolay Kozyrskiy [*1] **Balitskiy Gleb** [*1]

Abstract

One-Shot Multi-Speaker Text-to-Speech (OS MS TTS) systems are aimed to transform text into speech with voice determined by small single sample. The main problem here is to reproduce the new unseen voice without retraining the network. There is an approach with three main stages which is used to solve this problem. The unique for each voice speaker embeddings, which reveal the voice characteristics, are generated at the first stage (*Speaker Encoder*). At the second stage (*Synthesizer*) the text is transformed to mel-spectrogram using previously obtained embeddings. Finally, the speech is reproduced from the mel-spectrogram with the *Vocoder*. But there is lack of implementations with these three parts properly combined. So the goal of our project is to create a flexible framework to combine these parts and provide replaceable modules and methods in each part.

1. Main Challenges

By now we see the following main challenges:

- The solution to our problem consists of three subtasks, which already have a great solutions. Therefore, the existing solutions for OS MS TTS are essentially a compilation of solutions for these individual problems, for which there are many ready-made and well-implemented solutions. The main challenge is to made the framework flexible and ensure the compatibility of individual parts.

- The methods used in each subtask differ in the set of parameters and the nature of the algorithm. Therefore, it will be quite difficult to provide a single API.

[*]Equal contribution [1]Skolkovo Institute of Science and Technology, Russia. Correspondence to: Cieua Vvvvv <c.vvvvv@googol.com>, Eee Pppp <ep@eden.co.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

2. Baseline Solution

We choose solution proposed by the instructors as a baseline, which can be found here. It is the implementation of (Jia et al., 2018) made in Google in 2018. Here authors use the speaker encoder, presented in (Wan et al., 2018), which generates a fixed-dimensional embedding vector known as d-vector. As for Synthesizer they use model based on Tacotron 2 (Shen et al., 2018) while an auto-regressive WaveNet-based is used as the Vocoder (van den Oord et al., 2016).

2.1. Pros and Cons

The Real-Time-Voice-Cloning contains the realizations of encoder, Tacotron 2 and WaveRNN. The whole pipeline described in (Jia et al., 2018), including preprocessing steps, is also implemented in this repository. However, the project is not flexible enough. More specifically, in the current state it cannot be used as the framework for One-Shot Multi-Speaker Text-to-Speech system as there are no convenient mechanisms for manipulating with the three main modules. For example, the proposed multi-speaker TTS system in (Cooper et al., 2020) cannot be easily implemented with the help of Real-Time-Voice-Cloning as there are no extensibility points which allow to adjust the pipeline for the new method.

2.2. Our Improvement

Our plan is to use the Real-Time-Voice-Cloning as starting point with implemented baseline. We will introduce the flexible modular design of the framework. Such approach will help us to create the convenient API for external users who will be able to use our framework for incorporating the Multi-Speaker TTS system in their products. The API will also let the users customize modules and pipeline steps without changing the source code of the framework if needed. We will implement several Speaker Encoders (LDE, TDNN) and add them to our framework as well.

2.3. Project Structure Overview

From a high point, our project consists of 3 main elements: Speaker Encoder, Synthesizer, Vocoder. For each of them, a manager is implemented that allows one to access the parameters and perform standard actions such as inference.

Above them, the we implemented OS MS TTS manager, which brings together all three parts and allows one to make all pipeline and produce speech with needed voice. Each of these parts is also consist from elementary sub-parts typical for the corresponding elements. They can be described as follows:

1. **Speaker Encoder:** Here the base class is SpeakerEncoderManager, which allows to train(next update) and inference model.Also, we have already implemented the Wav Audio Preprocessing Interface. So, one can customize their own audio preprocessing functions, which can differ even for the same dataset. Also, the custom model can be used. We added standard preprocessing function and model presented in Real-Time-Voice-Cloning as a baseline method.
2. **Synthesizer:** Here the base class is SynthesizerManager, which allows to train and inference model. Also, the same sutiation with preprocessing functions, with one difference. In addition to the audio, one also need to process the text. For now, we only implemented text preprocessing function, as this only operation only needed during inference. Also, baseline from Real-Time-Voice-Cloning implemented.
3. **Vocoder:** Here the base class is VocoderManager, which allows to train(next update) and inference model. Also, baseline from Real-Time-Voice-Cloning implemented.

2.4. Evaluation Results

In our repository we added notebook, where one can download the voice audio, .txt file and produce speech with cloned voice.

3. Roles of the Participants

Nikolay will design the modular architecture, API for external usage and training pipeline. Gleb will implement working stack of models, write documentations and usage examples.

4. Link to the GitHub repository

All the materials are available at OSM-one-shot-multispeaker.

5. Project Structure

Project structure can be found here

6. Installation

Run **pip3 install .** from root directory.

7. Datasets

We have implemented complete processing for LibraSpeech Dataset for Speaker Encoder, Synthesizer and Vocoder . One can download LibraSpeech dataset via this link. Also, for Speaker Encoder we implemented interface to use custom dataset. One needs to implement **PreprocessDataset** interface functions, **WavPreprocessor** interface functions, or use implemented ones.

8. Configs

For baseline models the default configs will be loaded automatically. To change them one can use **update_config(...)** in **osms/common/configs/config.py**. To load default config one can use **get_default_"module_name"_config(...)**. Also, one can implement his own configs to use them for other models.

9. Managers

To work with each three modules we implemented its own manager: **SpeakerEncoderManager**, **SynthesizerManager**, **VocoderManager**. As main manager we implemented **MustiSpreakerManager** which give access to all three managers. One can use them to inference the whole TTS model and train each modules. The example of usage can be found in notebook.

10. Checkpoints

Baseline checkpoints are downloaded automatically in **checkpoints** directory with creation of **MultiSpeaker** object. Also, one can use other checkpoints by simple updating of config (change ...CHECKPOINT_DIR_PATH, CHECKPOINT_NAME).

References

Cooper, E., Lai, C.-I., Yasuda, Y., Fang, F., Wang, X., Chen, N., and Yamagishi, J. Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6184–6188, 2020. doi: 10.1109/ICASSP40776.2020.9054535.

Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Lopez-Moreno, I., and Wu, Y. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *NeurIPS*, 2018.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R., Saurous, R., Agiomyrgiannakis, Y., and Wu, Y. Natural tts

synthesis by conditioning wavenet on mel spectrogram predictions. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779–4783, 2018.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. In *SSW*, 2016.

Wan, L., Wang, Q., Papir, A., and Moreno, I. L. Generalized end-to-end loss for speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4879–4883, 2018. doi: 10.1109/ICASSP.2018.8462665.