02 December 2020

ECE 6143 Machine Learning

Project report

Group member: Chengfeng Luo, Jiaxi Liu

Voice Encoder

INTRODUCTION

Deep learning models have become dominant in many areas of applied machine learning. There are no exceptions for Text-to-speech (TTS) which is the process of synthesizing speech from text prompt. The ability to generate speech from any voice is attractive to a range of applications for their usefulness like translating speakers' speech to any language in their voice. Our voice cloning idea came out with the inspiration of Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis. [1] This paper describes a neural network-based text-to-speech (TTS) synthesis system, which can generate speech audio from different speakers. The final framework is able to run in a zero-shot setting, that is, for speakers who cannot be seen during training. In just a few seconds of reference speech, it can incorporate the speaker's voice. This system included three separate trained components: speaker encoder network, synthesizer network and vocoder network. Our final goal is to train a system that will recognize users voice features and output an audio that is similar to users voice. As the project goes on we focus on speaker encoder which is to capture users' voice characteristics first.

The structure of this report goes as follows. We first go through the concepts and training method about the encoder as Jemine Corentin mentions in his paper.[6] Follow with our implementation and conclude with a user interface we designed to demonstrate our output. Our project code will build based on Resemblyzer in github. [4]

MODEL ARCHITECTURE

The model is a 3-layer LSTM with 768 hidden nodes followed by a projection layer of 256 units. [1]Which is a layer where the 256 outputs of each LSTM are fully connected and can be repeatedly applied to each output of the LSTM. The input of the model is a 40-channel logarithmic mel spectrogram with a window width of 25ms and a step length of 10ms.[1] The output is the L2 normalized hidden state of the last layer, which is a vector of 256 elements.[1] In order to make the embedding sparse and therefore easier to interpret, there is a ReLU layer before normalization.

GENERALIZED END-TO-END LOSS

The model Ye Jia proposes consists of three parts, each of which has been individually trained. [1] This allows independent data training for each part, thereby reducing the need to obtain high-quality multi-speaker data.

First part is the speaker encoder. The job of the speaker encoder is to get some input audio from a given speaker and output the embedded content that captures "how the speaker sounds. " Speaker encoder only care about the voice of the speakers, e.g., high/low pitched voice, accent, tone and so on. All of these features are combined into a low dimensional vector known as the speaker embedding. The Generalized End-to-End loss (GE2E) simulates this process to optimize the model.

General end-to-end (GE2E) training is based on processing a large number of utterances at a time. The batch contains N speakers, and each speaker has an average of M speeches. The embedding vector is defined as the L2 normalization of the network output.[2]

$$\mathbf{e}_{ji} = \frac{f(\mathbf{x}_{ji}; \mathbf{w})}{||f(\mathbf{x}_{ji}; \mathbf{w})||_2}$$

Each feature vector x_{ji} ($1 \leq j \leq N$ and $1 \leq i \leq M$) represents the features extracted from speaker j utterance i. e_{ji} represents the embedding vector of the *j* th speaker's *i* th utterance.[2] The centroid of the embedding vectors from the jth speaker is defined as c_j, $\mathbf{c}_i = \frac{1}{M}\sum_{j=1}^{M} \mathbf{e}_{ij}$ [2] The similarity matrix $S_{ji,k}$ is defined as the scaled cosine similarities between each embedding vector e_{ji} to all centroids c_k ($1 \leq j$, k $\leq N$, and $1 \leq i \leq M$) $\mathbf{S}_{ji,k} = w \cdot \cos(\mathbf{e}_{ji}, \mathbf{c}_k) + b.$ where w and b are learnable parameters.[2] Entire process is illustrated in Figure 1.

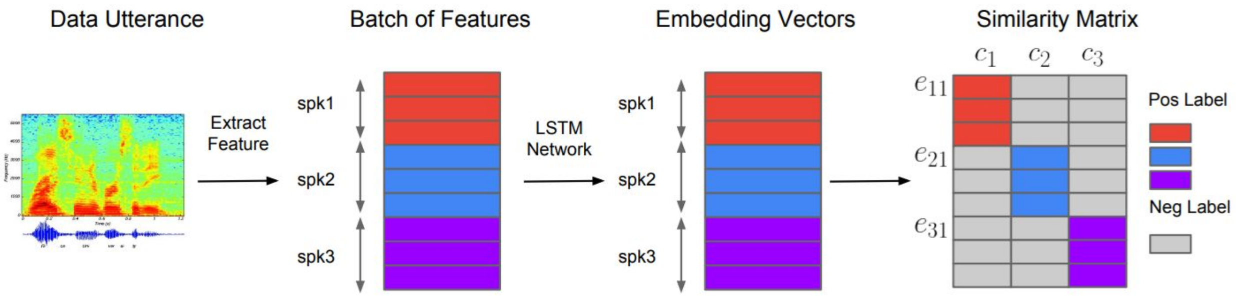


Figure 1. System overview. Different colors indicate utterances/embeddings from different speakers [2]

In order to make the embedding of each utterance to be similar to the centroid of all one speaker's embeddings, and far from other speaker's centroids at the same time, as shown in Figure 2, the author uses softmax loss function. Therefore the loss on each embedding vector e_{ji} could be defined as $L(\mathbf{e}_{ji}) = -\mathbf{S}_{ji,j} + \log \sum_{k=1}^{N} \exp(\mathbf{S}_{ji,k})$.[2] When computing the loss each e_{ji} is

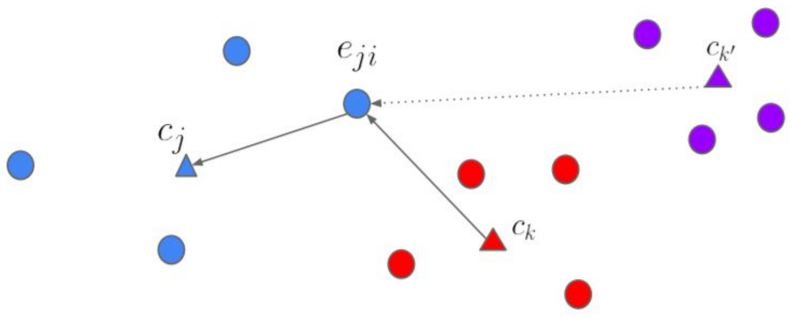


Figure 2. GE2E loss pushes the embedding towards the centroid of the true speaker, and away from the centroid of the most similar different speaker.[2]

included in the centroid c_i of the same speaker, this will lead to a bias against the correct speaker. Therefore, removing e_{ji} when calculating the centroid of the real speaker can stabilize the training and help avoid trivial solutions.[2] The similarity matrix $S_{ji,k}$ is then defined as

$$\mathbf{c}_j^{(-i)} = \frac{1}{M-1}\sum_{\substack{m=1\\m\neq i}}^{M}\mathbf{e}_{jm},$$

$$\mathbf{S}_{ji,k} = \begin{cases} w\cdot\cos(\mathbf{e}_{ji},\mathbf{c}_j^{(-i)})+b & \text{if}\quad k=j; \\ w\cdot\cos(\mathbf{e}_{ji},\mathbf{c}_k)+b & \text{otherwise.}\end{cases}$$

The utterances are partially sampled from the complete utterances in the dataset, so in the training batch the fixed duration of utterances is 1.6 seconds. At inference time, the utterance is divided into 1.6 second segments, overlapping by 50%, and the encoder forwards each segment separately.[2] Then result outputs will be averaged and normalized to produce the utterance embedding. As shown in Figure 3.

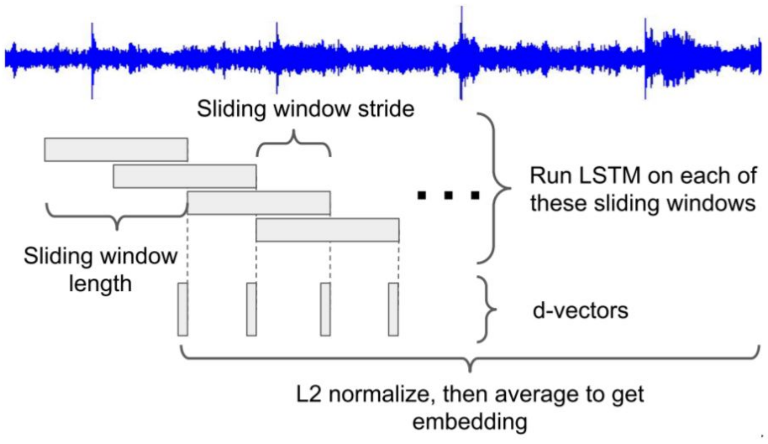

Sliding window stride
Run LSTM on each of these sliding windows
Sliding window length
d-vectors
L2 normalize, then average to get embedding

Figure 3. Computing the embedding of a complete utterance[2]

EXPERIMENT

When training the GE2E model, each batch contains N = 64 speakers and M = 10 utterances per speaker.[2] Before training, it is necessary to modify the sample for better performance. The author uses the webrtcvad python package to perform Voice Activity Detection (VAD)[5], in order to silent segments when sampling utterances from original utterances. The author and his team found that a good choice for the maximum silence duration tolerated is 0.2s. Figure 4 shows the process. The last preprocessing step applied to the audio waveform is normalization, to compensate for changes in speaker volume in the data set.

The dataset we are use for training is LibriSpeech. [3] This resource consists of a combination of two "clean" training sets, including 436 hours of speech from 1,172 speakers, with a sampling frequency of 16 kHz.[7] We use train-other-500 as training set. When we implemented the code the instruction didn’t import the test set. We trained the speaker encoder for 173,001 steps. In order to monitor the training the process also reports the Equal Error Rate (EER) so we can observe the ability of the model to cluster speakers. EER is a measure

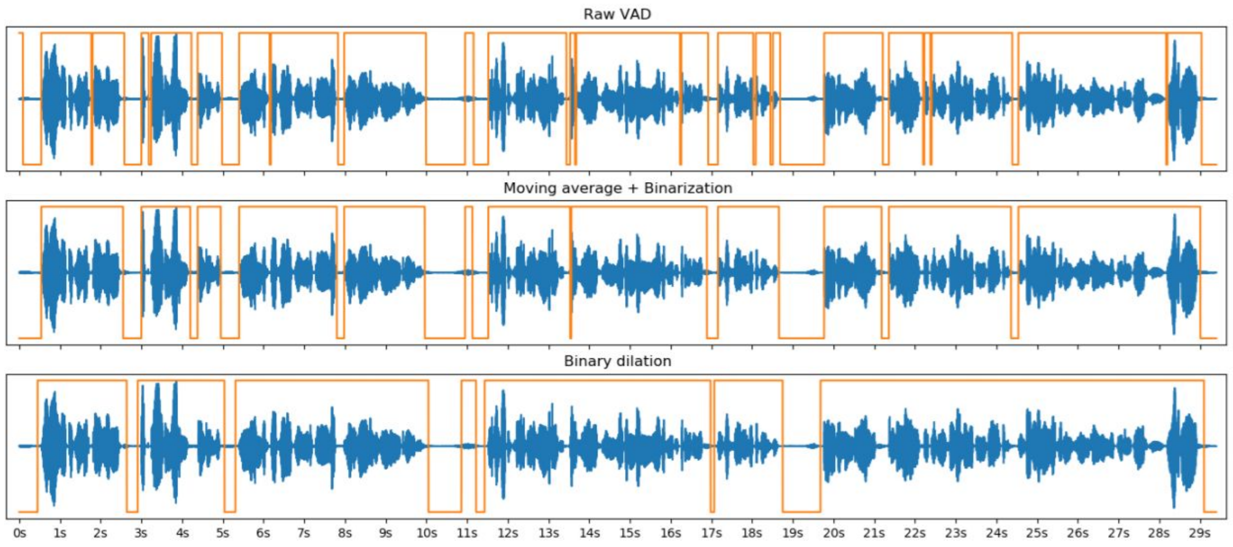
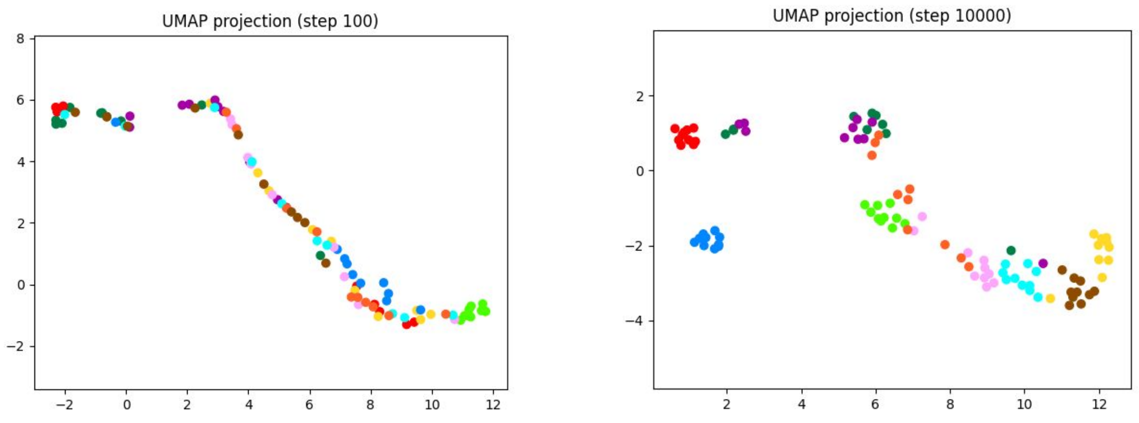


Figure 4. The orange line is the binary voice flag where the upper value means that the segment is voiced, and unvoiced when lower. [6]

commonly used in biometric systems to evaluate system accuracy.[6] It is the value of the false positive rate when it is equal to the true negative rate.[6] In our understanding it is a way to measure the error as the encoder recognizes speakers from embedding. Following Jemine Corentin’s step the training process periodically samples a batch of 10 speakers with 10 utterances each, computes the utterance embeddings and projects them in a two-dimensional space with UMAP.[8] Selected UMAP projections are shown in Figure 5.



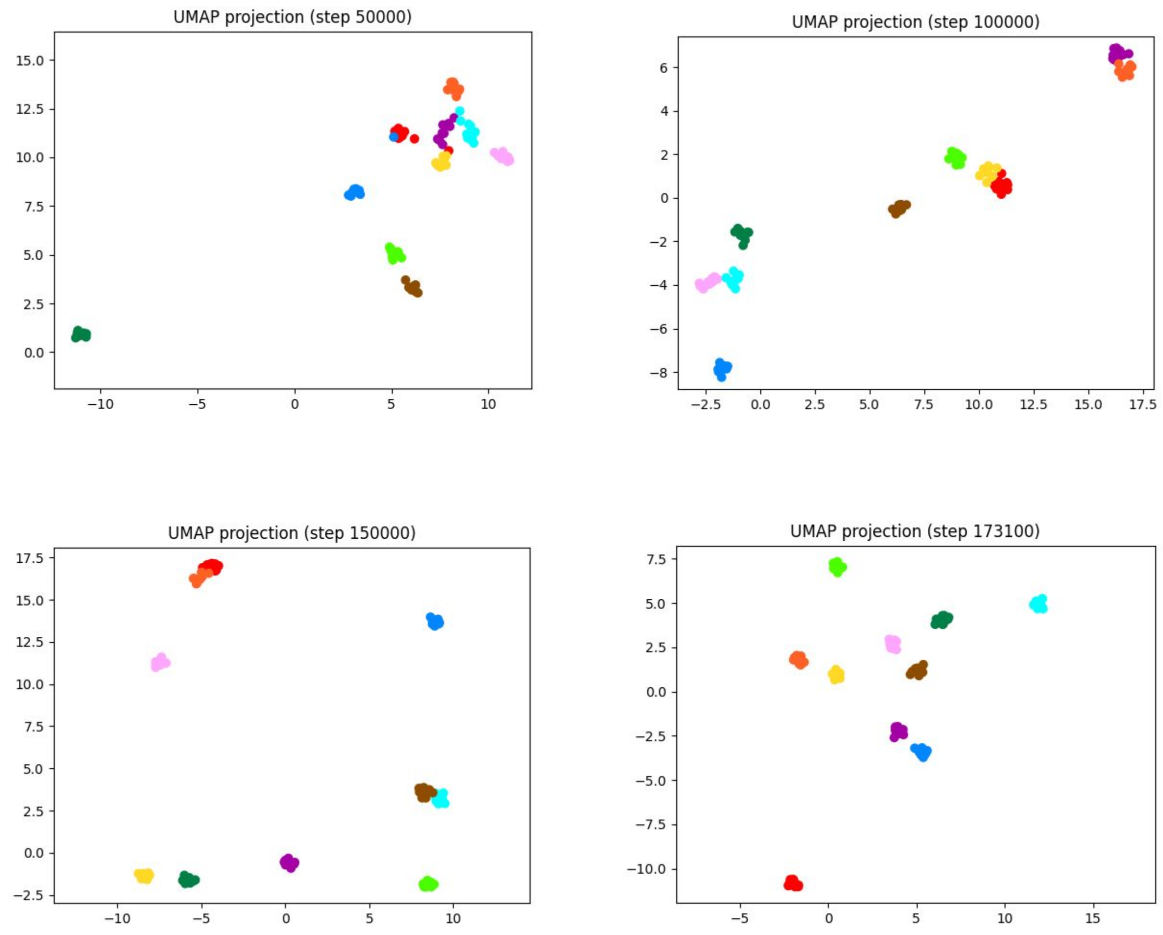


Figure 5. UMAP projections of utterance embeddings from randomly selected batches from the train set at different iterations of the model.

From Figure 5 we can tell the outcome is pretty good. Utterances from the same speaker are represented by a dot of the same color and all the utterances with the same color are grouped together as expected.

CONCLUSION

We also plot the average loss along with the steps shown in Figure 6. It is clear that the average loss decreases as steps increase which means the more steps we train better the outcome will be and more accurate of the result. The loss is actually below 0.1 after 100,000 steps. In Figure 7 we show the result of EER outcome. We computed the test set EER to be 0.5%. This is a surprisingly low value compared with the 4.5% of the authors for the same set with 5 times more steps. We are not sure it is because the model is performing perfect or there might be other things we did not consider. Based on the limited time we have this result is really good.

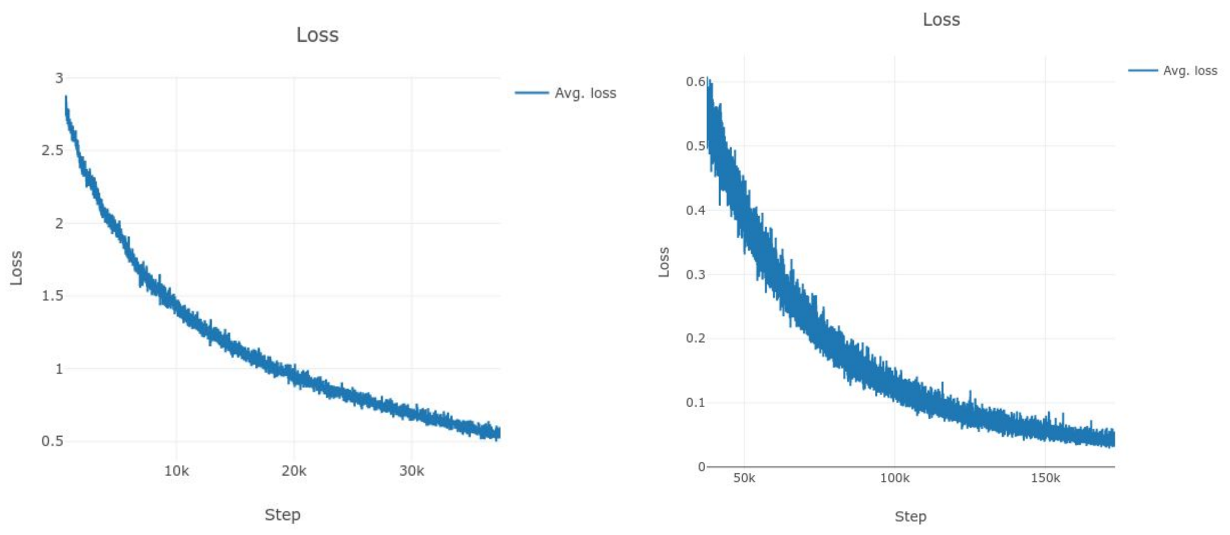


Figure 6. Average loss with steps increase

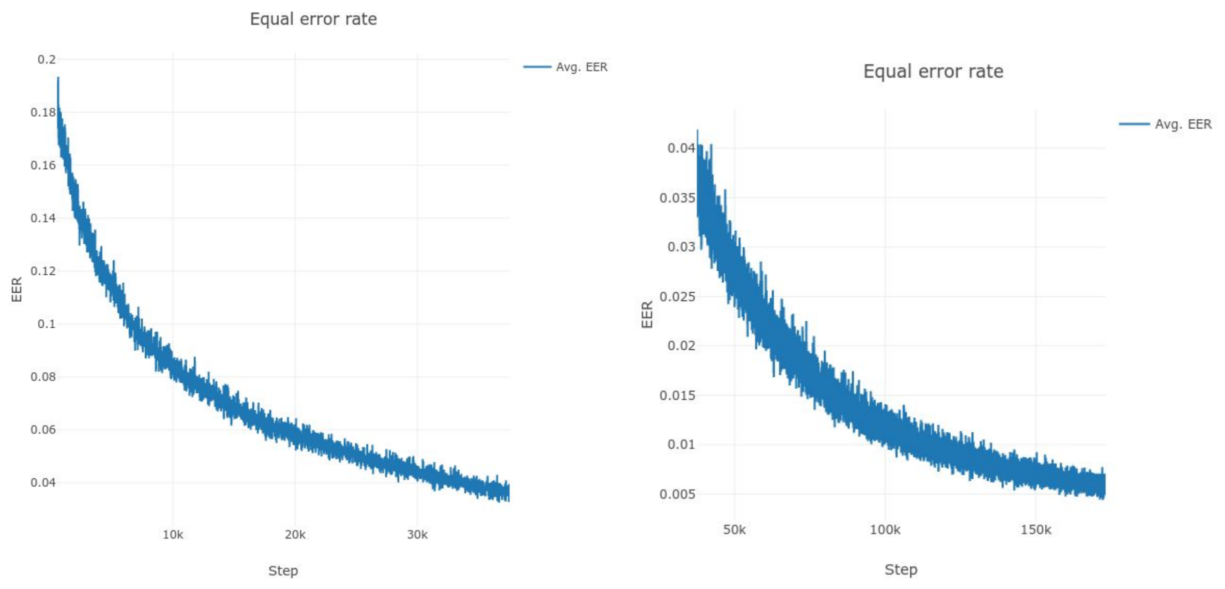


Figure 7. Average Equal Error Rate with steps increase

The user interface we build shown in Figure 8. First users can select the input wav file from train-clean-100 file or record their own voice. If the input file is selected the utterances' spectrogram and waveform will be shown along with the result of preprocessing step which is to silent segments when sampling utterances from original utterances. Moreover the embedding result and UMAP figure will be updated. For a specific one speaker it is better to input multiple utterances to get more clear UMAP projections. In Figure 8 we already loaded 3 different speaker utterances from the file, they are labeled as user 01, user 03, user 04. We also record our own voice which is labeled as rec02. Different colors represent different speakers. From the UMAP projections we can tell all the utterances with the same color are grouped together perfectly and different speakers separate away from each other clearly.

. Voice encoder have been implemented and this can be used for further application for voice cloning. After voice encoder there are synthesizers and vocoder that need to be implemented for the voice clone. And these are the directions for us to approach the final goal. Overall, we find the encoder results to be satisfying. At first, we are planning to train the model using chinese database but we do not have that much time to do the training again. So we stick with the database from the author. We might change the database later to see if the output is different.

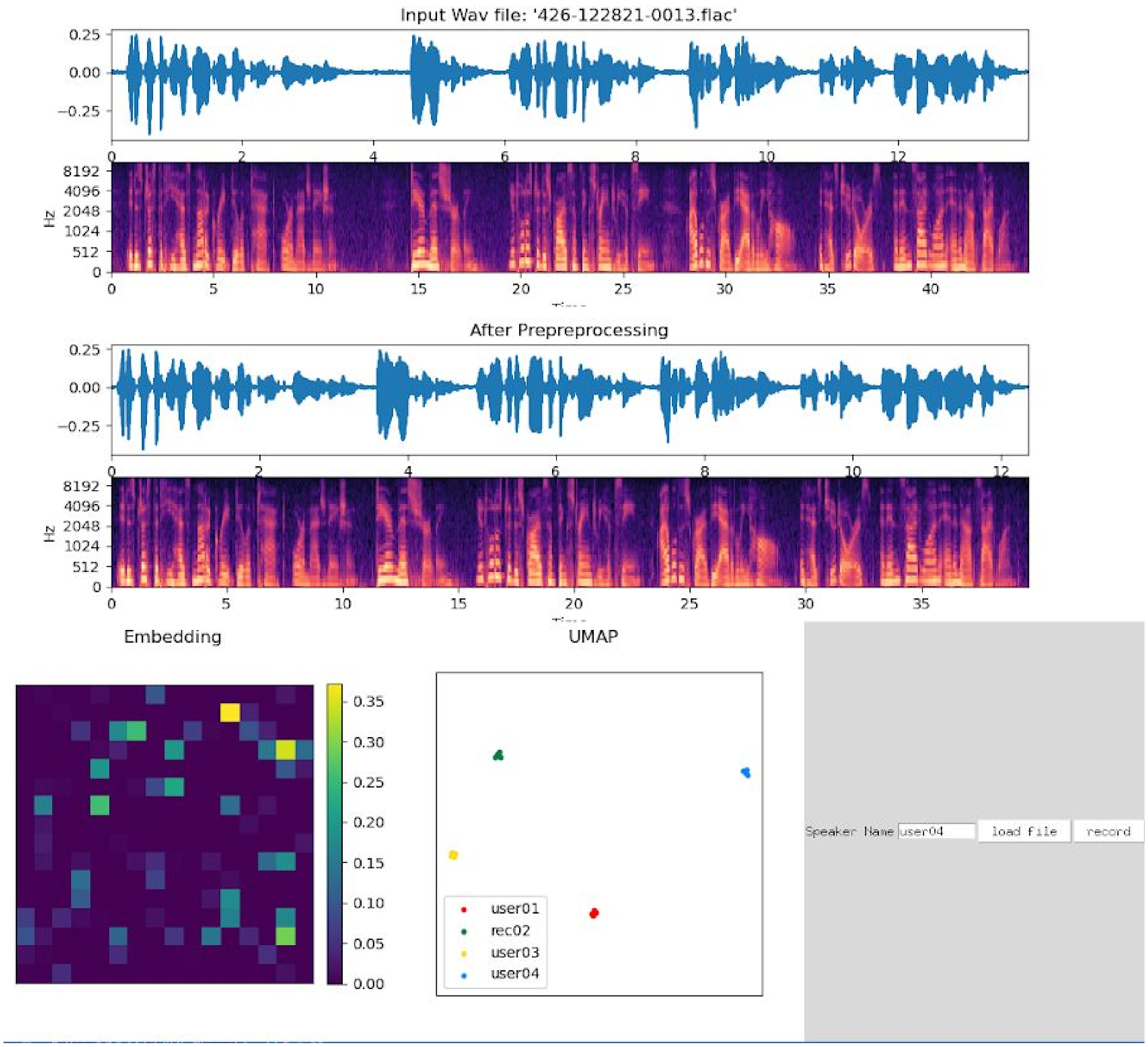


Figure 8. User Interface for Voice Encoder

Works Cited

Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu. *Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis.* In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018).

Li Wan, Quan Wang, Alan Papir, Ignacio Lopez Moreno. *Generalized End-to-End Loss for Speaker Verification.* In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '18).

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. *LibriSpeech: an ASR corpus based on public domain audio books.* In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pages 5206–5210. IEEE, 2015.

Resemblyzer https://github.com/resemble-ai/Resemblyzer

Py-webrtcvad https://github.com/wiseman/py-webrtcvad

Jemine, C. (2019). Master thesis : Real-Time Voice Cloning. (Unpublished master's thesis). Université de Liège, Liège, Belgique. Retrieved from https://matheo.uliege.be/handle/2268.2/6801

LibriSpeech ASR corpus http://www.openslr.org/12/

Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. 02 2018.