18 May 2021

ECE 9123 Deep Learning

Final Report

Group member: Chengfeng Luo, Jiaxi Liu

Code link: https://github.com/JiaxiLiu/DeepLearning_Project_VoiceClassification

Voice Classification

- **Problem Statement**:

  Imagine a meeting is taking place in a conference room. There are several people, they take turns speaking. We want to train a model that can accurately predict and classify who is speaking only by listening to their voice and more information like the gender of speakers. If we can implement this, it will be great if voice classification can be applied to the zoom meeting we used and to identify each speaker in the meeting. For this, we decided to go deep into this topic.

- **Literature Survey**:

  Our voice classification 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. So we want to use this other way around. For the speaker encoder which is used to adjust the synthesis network based on the reference voice signal from the desired target speaker, we will refer to Ye jia's paper. The loss function we are going to use is called generalized end-to-end (GE2E) loss, which makes the training of speaker verification models more efficient. [2] Speaker encoders 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.

- **Technical details (dataset, model, algorithms, etc):**
  - **Dataset:**

    The dataset we are using 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.[4] Following is the model details.
  - **Input:**
    - 40-channels log-mel spectrograms with 25ms window width, 10ms step.
  - **Model:**
    - 3-layer LSTM with 768 (256 per layer) hidden nodes
  - **Loss function:**
    - Generalized End-to-End loss (GE2E)[2]
  - **Output:**
    - L2-normalized hidden state of the last layer (256 elements vector)
    - Will implement a logistic regression for classification

    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.

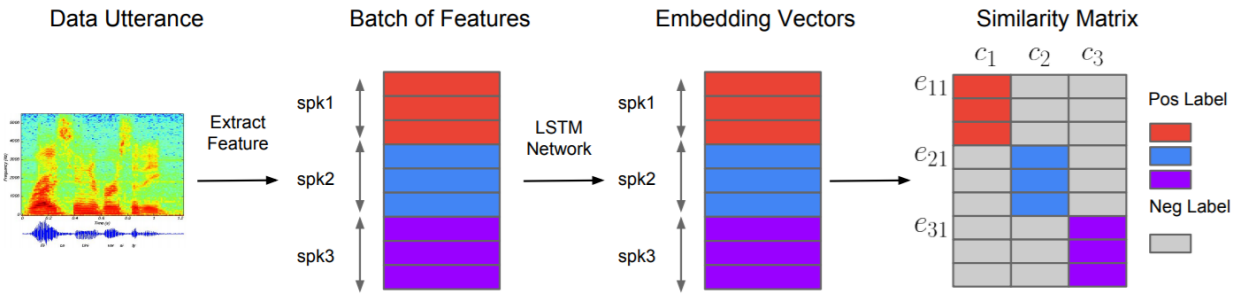

Figure 1. System overview. Different colors indicate utterances/embeddings from different speakers [2]

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, [2]

$$L(\mathbf{e}_{ji}) = -\mathbf{S}_{ji,j} + \log \sum_{k=1}^{N} \exp(\mathbf{S}_{ji,k})$$

When computing the loss each e_{ji} is 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]

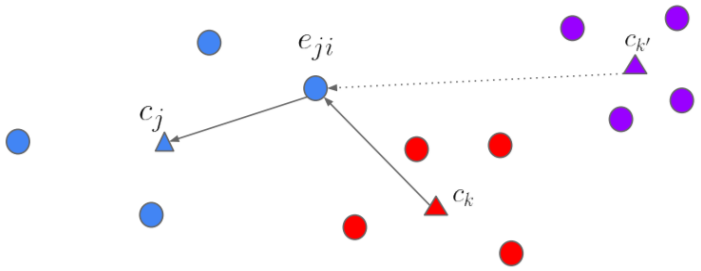

Figure 2. GE2E loss pushes the embedding towards the centroid of the true speaker, and away from the centroid of the most similar different speaker.[2]

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.

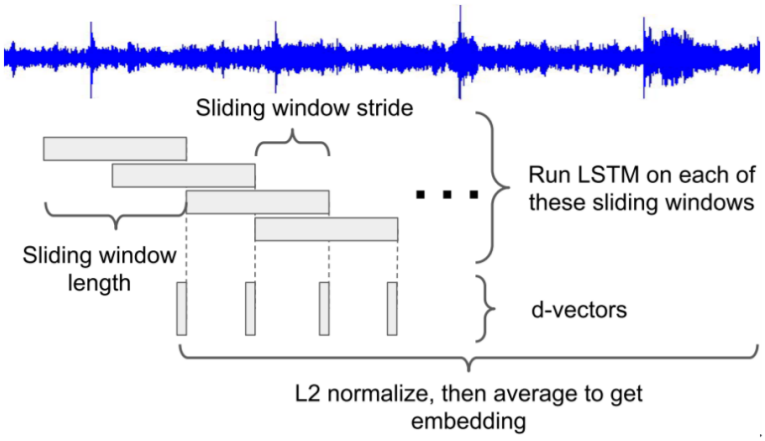

Figure 3. Computing the embedding of a complete utterance[2]

- **Preliminary results:**
  - **Data preprocessing:**

    When training the GE2E model, we set each batch to contain N = 64 speakers and M = 10 utterances per speaker. We followed the work from Real-Time Voice Cloning paper[6], using the webrtcvad python package to perform Voice Activity Detection (VAD)[5], in order to silent segments when sampling utterances from original utterances. After that we normalized the audio signal, to compensate for changes in speaker volume in the data set. Last, we generated the log-mel spectrograms for all the utterances samples, which is the input for our model.

  - **Model training**

    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 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] 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.[7] Selected UMAP projections are shown in Figure 4.

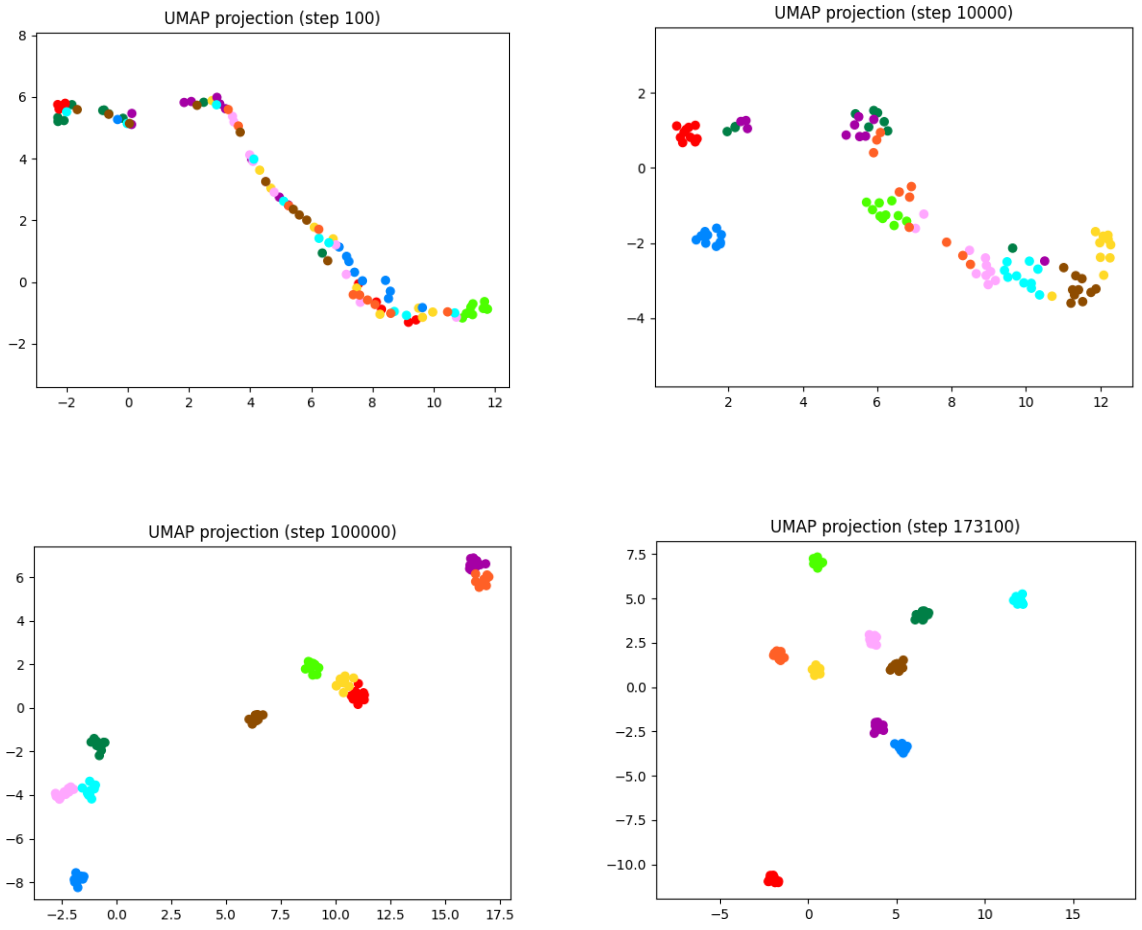

Figure 4. UMAP projections of utterance embeddings from randomly selected batches from the train set at different iterations of the model.

- **Conclusion**:

  From Figure 4 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. We also plot the average loss along with the steps shown in Figure 5. It is clear that the average loss decreases as steps increase which means the

more steps we train the better the outcome will be and more accurate of the result. The loss is actually below 0.1 after 100,000 steps. In Figure 6 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 perfectly or there might be other things we did not consider. Based on the limited time we have this result is really good.

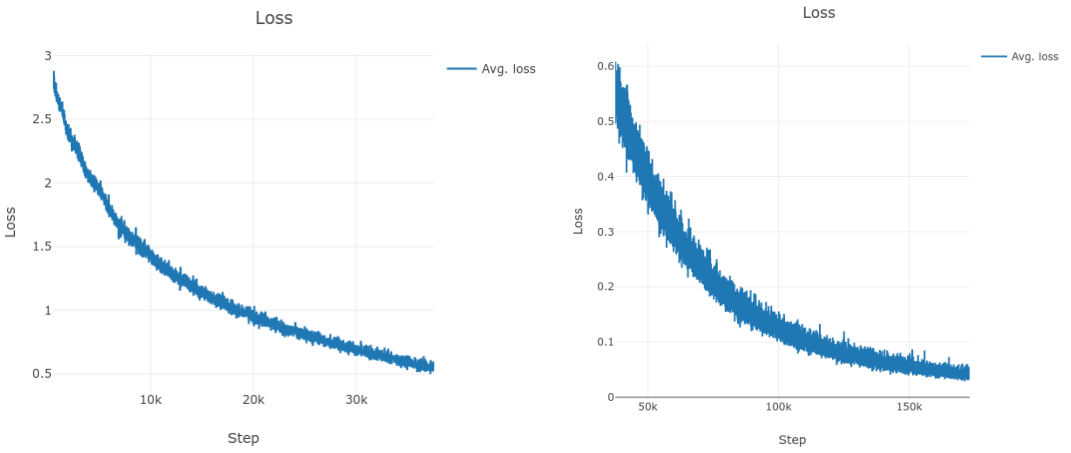

Figure 5. Average loss with steps increase

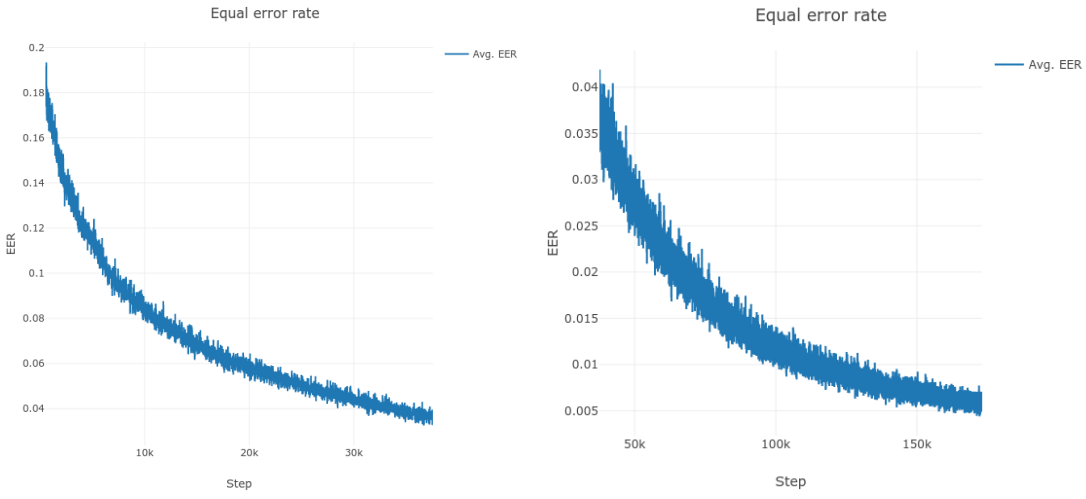

Figure 6. Average Equal Error Rate with steps increase

- **Model Implementation**:

  The user interface we built is shown in Figure 7. First users can select the input wav file or record their own voice as training samples with labels. 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. After loading training samples, users can start to input an unlabeled testing sample using the 'identify' button. In UMAP, the training samples will be displayed as 'dot' and the testing samples for identification will be marked with 'x'. Different colors represent different speakers' labels. 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. Moreover, the identified labed will print on the "Classified as Speakers" section.

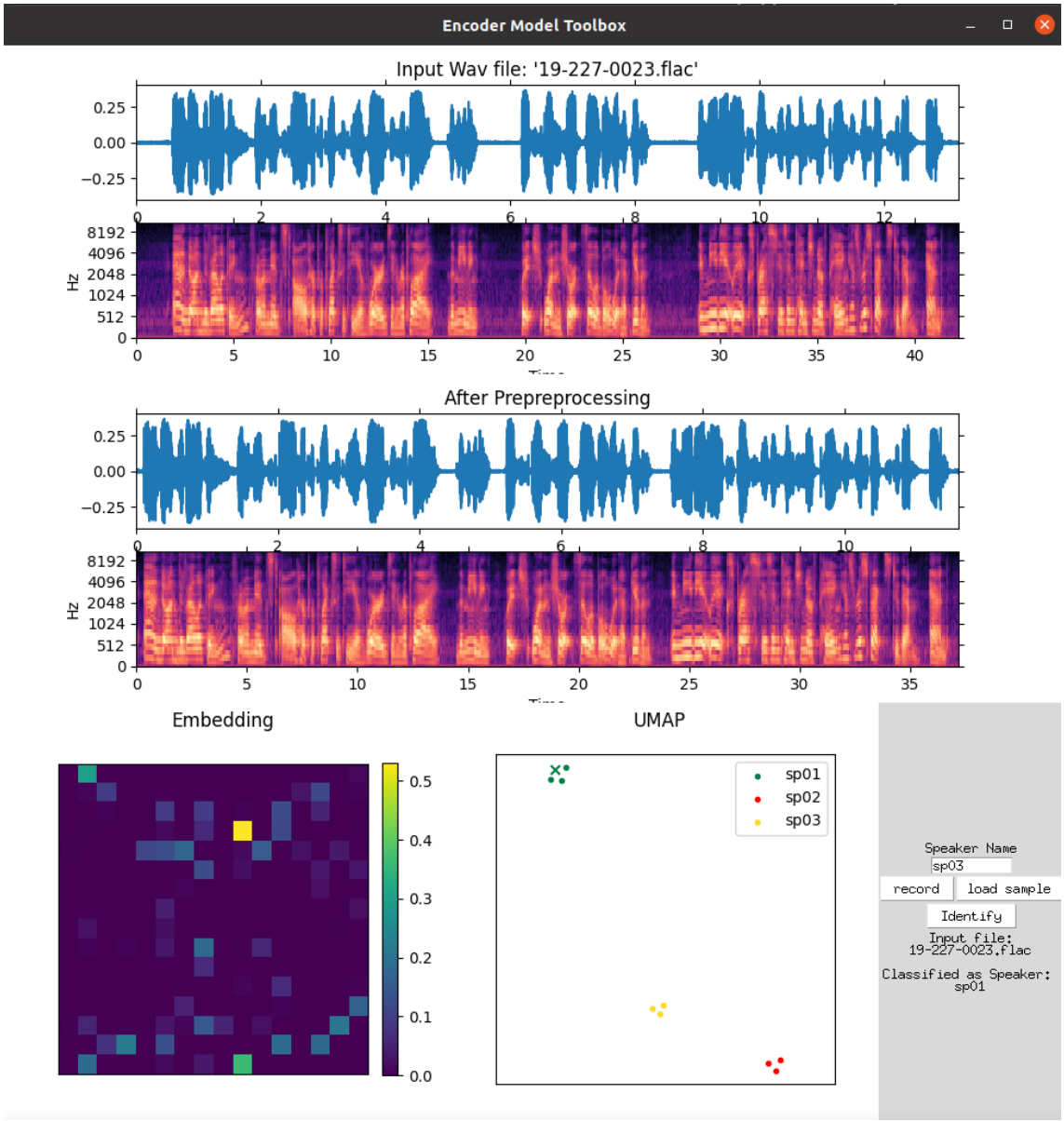

Figure 7. User Interface for Voice Encoder

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.

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

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

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