CS698 Machine Listening Final Project Proposal:

Conformer-based Speaker Recognition Model

Ethan Oh
eo238@njit.edu

Overview

The project objective is to build an automatic speaker verification (ASV) model based on the conformer model originally proposed in [1] for speech recognition. The paper received much attention for its novel solution in overcoming traditional speech recognition system's difficulty in learning relationships across data with far distances, while simultaneously learning local relationships. The Conformer model expands on the Transformer model by augmenting a Transformer with a convolution mechanism. The Transformer's Multi-head Attention (MHA) networks allow a model to learn relationships between data points without being constrained by distance. However, audio is time-bound and therefore, a data point's locality and its relationships' with its neighbors are significant. [1] shows us that we can address both these issues effectively.

The MFA-Conformer (MFAC) introduced in [2] adapts the original Conformer for ASV tasks. MFAC further expands on the Conformer model with its Multi-scale Feature Aggregator, which is an attentive statistic pooling mechanism introduced in [3] and is designed to incorporate low-level feature maps for increased accuracy .

The goal of this project is to replicate the MFA-Conformer, while experimenting with different loss functions than the Cross Entropy originally used in the paper.

Motivation

ASV has many applications. Smart devices like Amazon's Alexa employ it to distinguish user preferences as well as for security. As technology has become integral in our daily lives and plays an intimate role, it is very important that our ASV toolkits are highly performant.

Background

Reference papers:

[1] A. Gulati *et al.*, “Conformer: Convolution-augmented Transformer for Speech Recognition,” *arXiv:2005.08100 [cs, eess]*, May 2020, Available: https://arxiv.org/abs/2005.08100

[2] R. Jahangir, Y. W. Teh, H. F. Nweke, G. Mujtaba, M. A. Al-Garadi, and I. Ali, “Speaker identification through artificial intelligence techniques: A comprehensive review and research challenges,” *Expert Systems with Applications*, vol. 171, p. 114591, Jun. 2021, doi: https://doi.org/10.1016/j.eswa.2021.114591.

[3] Y. Zhang *et al.*, “MFA-Conformer: Multi-scale Feature Aggregation Conformer for Automatic Speaker Verification,” Mar. 2022, doi: https://doi.org/10.48550/arxiv.2203.15249.

[4] A. Gulati *et al.*, “Conformer: Convolution-augmented Transformer for Speech Recognition,” *arXiv:2005.08100 [cs, eess]*, May 2020, Available: https://arxiv.org/abs/2005.08100

[5] B. Desplanques, J. Thienpondt, and K. Demuynck, “ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification,” *Interspeech 2020*, pp. 3830–3834, Oct. 2020, doi: https://doi.org/10.21437/Interspeech.2020-2650.

Commercially available references:

Google offers a ASV Saas Product via their Google Cloud, which is used by Avaya and Genesys as part of their automated customer service platform. Microsoft has a similar offering via Azure.

Proposed Approach

A MFAC model will be built using the paper’s Github repository as reference (https://github.com/zyzisyz/mfa_conformer). The model will take speech audio .wav files from the VoxCeleb dataset for training, validating and testing. I will experiment with limiting the frequency range to 300 hz to 4k hz in an effort to reduce the data size and maximize compute resources. This is around the range used in telephony as it maintains speech coherency at minimum bandwidth. I will remove non-speech frames only if it is necessary that the files need further size reduction.

In terms of learning, the attention layers in the model will hopefully learn to disregard these non-speech frames as the objective is for the model to learn the relationships between the data points to correctly label the audio with the correct speaker ID. The model extracts features via convolution which will require processing the .wav into Mel spectrogram. The window size will be a hyper-parameter that will need to be experimented with to find an appropriate value but will begin with 25ms with 20ms hop. The model will be trained on labeled data using softmax and cross entropy for its loss function. Its performance will be evaluated using equal error rate (EER) as in the original paper, which will be compared to the baselines produced from the ECAPA-TDNN model [5] (as also performed in the paper). I will rely on the pretrained ECAPA-TDNN model available on HuggingFace (https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb).

Dataset

I have explored several candidate datasets of which Google's Fleura dataset and Carnegie Mellon University's Arctic dataset were considered. Fleura is 5gb of audio speech clips that are binary classified as male or female. Arctic is a 1gb multiclass set of 19 speaker id's. Fleura will require hand-labeling to convert it into a multiclass set, assuming there is a feasible number of speakers to give us the classifications. Arctic is ideal for its classification, but the dataset overall is small. Upon Professor Cartwright's suggestion, I will be using the VoxCeleb dataset, which consists of 2000+ hours of speech clips extracted from interview videos on YouTube and made available on HuggingFace (https://huggingface.co/datasets/ProgramComputer/voxceleb).

Evaluation

As mentioned in "Proposed Approach," I will use EER to evaluate the model's performance. EER is defined:

$$EER = \frac{FAR + FRR}{2}$$

FAR is the false acceptance rate and FRR is false recognition rate and they are defined:

$$FAR = \frac{number\ of\ false\ positives}{number\ of\ false\ positives + number\ of\ true\ negatives}\ x\ 100$$

$$FRR = \frac{number\ of\ false\ negatives}{number\ of\ false\ negatives + number\ of\ true\ positives}\ x\ 100$$

I will be looking to see if my MFAC model can obtain a lower EER than the pre-trained EPACA-TDNN model.

Challenges

Preprocessing and training on a large audio dataset may be compute intensive and the available computing resources may be an upper bound in how much performance I can achieve with the model.

Milestones

Phase 1: Data collection - 10/13
Phase 2: Data preprocessing - 11/1
Phase 3: Model building - 11/15
Phase 4: Model training - 12/1
Phase 5: Analyze and build report on model outputs - 12/15
Phase 6: Present report - 12/20