Speaker Diarization with 1D convolution using Deep learning

Garima Chaudhary(23PGAI0046), AhmedRaza Sharif(23PGAI0120)

Abstract

Speech diarization is the task of automatically segmenting an audio recording into homogeneous regions based on speaker identity. One of the most effective approaches for speech diarization is to use deep learning techniques. In recent years, 1D-Convolutional Neural Networks (1D-Conv) have gained popularity due to their ability to capture temporal features efficiently

Introduction

This project proposes a speech diarization approach that utilizes 1D-Conv neural networks for feature extraction and clustering. The objective is to improve the accuracy of speech diarization while reducing computational costs. The proposed approach involves two stages: feature extraction and clustering. The feature extraction stage involves using 1D-Conv layers to extract high-level features from the audio signal, while the clustering stage groups these features into clusters corresponding to different speakers.

The project aims to compare the proposed method with other popular methods such as i-vector and x-vector and demonstrate its superior performance in terms of clustering accuracy and computational efficiency. Additionally, the project aims to explore the effectiveness of the proposed method on different datasets and investigate the impact of various parameters on its performance.

The rest of the project is organized as follows. Section 2 provides an overview of related work in the field of speech diarization. Section 3 describes the proposed method in detail, including the feature extraction and clustering stages. Section 4 presents experimental results and compares the proposed approach with other methods.

Motivation & Background

The motivation for this project is to improve the accuracy of speech diarization while reducing computational costs. The proposed approach aims to leverage the power of 1D-Conv neural networks to extract high-level features from the audio signal, which can then be used to cluster the audio segments into different speakers. The proposed approach is expected to outperform traditional clustering algorithms while reducing the need for hand-crafted features and domain expertise.
Its crucial in field of speech processing, with applications such as automatic speech recognition, speaker identification, and content-based multimedia retrieval

Traditional approaches to speech diarization relied on hand-crafted features and clustering algorithms. However, these approaches often require significant domain expertise and may not perform well under challenging conditions. In recent years, deep learning techniques have shown remarkable performance in speech processing tasks, including speech diarization. Specifically, 1D-Convolutional Neural Networks (1D-Conv) have gained popularity due to their ability to capture temporal features efficiently.

Description of DataSet

This dataset we have taken from kaggle contains speeches of five prominent leaders namely; Benjamin Netanyahu, Jens Stoltenberg, Julia Gillard, Margaret Tacher and Nelson Mandela which also represents the folder names. Each audio in the folder is a one-second 16000 sample rate PCM encoded.

Originally, the speech for each speaker was a one lengthy audio, I chunked them into one-second each for easier workability. If you combine the chunked audios from 0.wav to 1500.wav, it forms a complete speech of the respective speaker.

A folder called background_noise contains audios that are not speeches but can be found inside and around the speaker environment e.g audience laughing or clapping. It can be mixed with the speech while training.

Implementation

Libraries used:

TensorFlow
Keras
Librosa
Joblib
Soundfile

Methodology:

We prepare a dataset of speech samples from different speakers, with the speaker as label.

We add background noise to these samples to augment our data.

We take the FFT of these samples.

We train a 1D convnet to predict the correct speaker given a noisy FFT speech sample

- Speech samples, with 5 folders for 5 different speakers. Each folder contains 1500 audio files, each 1 second long and sampled at 16000 Hz.
- Background noise samples, with 2 folders and a total of 6 files. These files are longer than 1 second (and originally not sampled at 16000 Hz, but we will resample them to 16000 Hz). We will use those 6 files to create 354 1-second-long noise samples to be used for training.

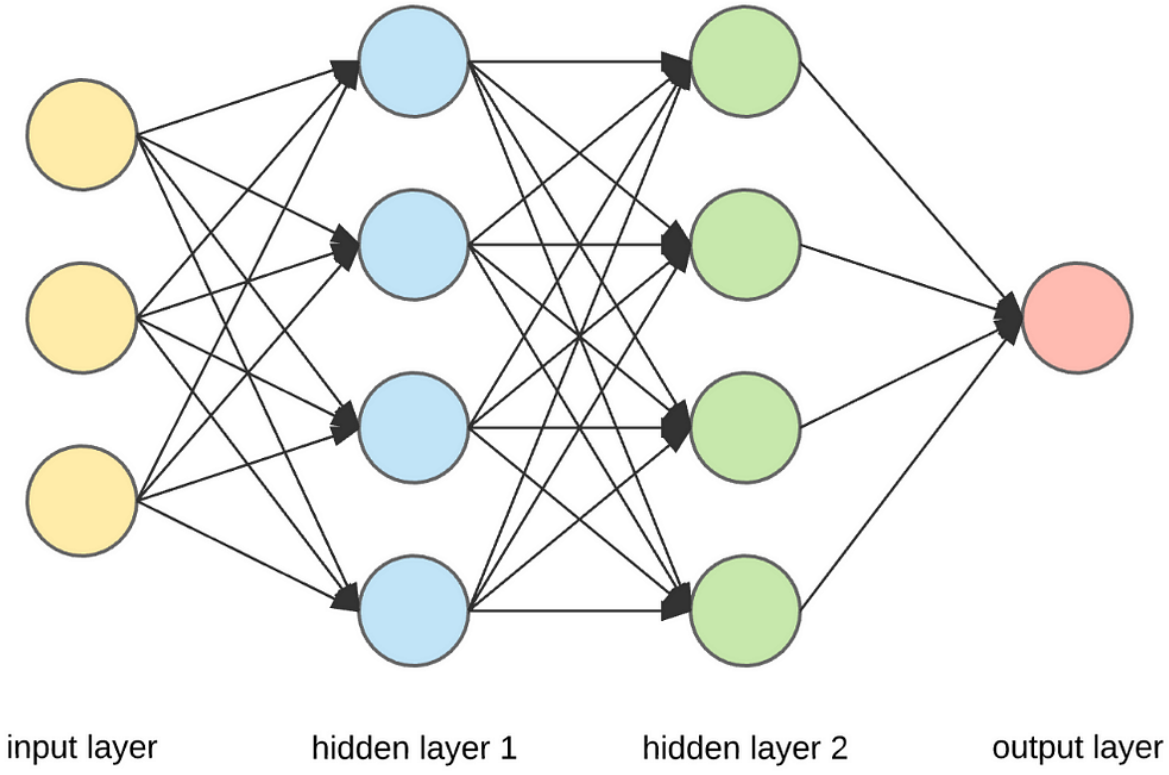
Let's sort these 2 categories into 2 folders:

- An audio folder which will contain all the per-speaker speech sample folders
- A noise folder which will contain all the noise samples

We used Conv 1D model

Data Modeling:

I built a sequential neural network model, which is the simplest method to build a model in Keras —it tells keras to stack all layers sequentially. Then I added four dense layers, which are fully connected layers in the model.



Model Architecture

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input (InputLayer)              [(None, 8000, 1)]    0           []

conv1d_33 (Conv1D)              (None, 8000, 128)    512         ['input[0][0]']

activation_23 (Activation)      (None, 8000, 128)    0           ['conv1d_33[0][0]']

conv1d_34 (Conv1D)              (None, 8000, 128)    49280       ['activation_23[0][0]']

activation_24 (Activation)      (None, 8000, 128)    0           ['conv1d_34[0][0]']

conv1d_35 (Conv1D)              (None, 8000, 128)    49280       ['activation_24[0][0]']

conv1d_32 (Conv1D)              (None, 8000, 128)    256         ['input[0][0]']

add_9 (Add)                     (None, 8000, 128)    0           ['conv1d_35[0][0]',
                                                                 'conv1d_32[0][0]']

activation_25 (Activation)      (None, 8000, 128)    0           ['add_9[0][0]']

max_pooling1d_9 (MaxPooling1D)  (None, 4000, 128)    0           ['activation_25[0][0]']

average_pooling1d_1 (AveragePo  (None, 1333, 128)    0           ['max_pooling1d_9[0][0]']
oling1D)

flatten_1 (Flatten)             (None, 170624)       0           ['average_pooling1d_1[0][0]']

dense_2 (Dense)                 (None, 256)          43680000    ['flatten_1[0][0]']

dense_3 (Dense)                 (None, 128)          32896       ['dense_2[0][0]']

output (Dense)                  (None, 5)            645         ['dense_3[0][0]']

==================================================================================================
```

Model Evaluation

I chose 15 as the number of epochs, which is the number of times the model will cycle through the data. After running around 15 epochs, the validation accuracy of the model is improved to 95-96%

```
Epoch 1/15
53/53 [==============================] - 197s 4s/step - loss: 2.4096 - accuracy: 0.7000 - val_loss: 0.2334 - val_accuracy: 0.9267
Epoch 2/15
53/53 [==============================] - 195s 4s/step - loss: 0.1703 - accuracy: 0.9363 - val_loss: 0.1395 - val_accuracy: 0.9453
Epoch 3/15
53/53 [==============================] - 197s 4s/step - loss: 0.1484 - accuracy: 0.9412 - val_loss: 0.1194 - val_accuracy: 0.9587
Epoch 4/15
53/53 [==============================] - 194s 4s/step - loss: 0.0876 - accuracy: 0.9693 - val_loss: 0.0881 - val_accuracy: 0.9680
Epoch 5/15
53/53 [==============================] - 191s 4s/step - loss: 0.1203 - accuracy: 0.9570 - val_loss: 0.1179 - val_accuracy: 0.9627
Epoch 6/15
53/53 [==============================] - 189s 4s/step - loss: 0.0746 - accuracy: 0.9747 - val_loss: 0.0862 - val_accuracy: 0.9680
Epoch 7/15
53/53 [==============================] - 195s 4s/step - loss: 0.0518 - accuracy: 0.9815 - val_loss: 0.0784 - val_accuracy: 0.9747
Epoch 8/15
53/53 [==============================] - 198s 4s/step - loss: 0.0638 - accuracy: 0.9794 - val_loss: 0.0695 - val_accuracy: 0.9827
Epoch 9/15
53/53 [==============================] - 201s 4s/step - loss: 0.0717 - accuracy: 0.9748 - val_loss: 0.1113 - val_accuracy: 0.9653
Epoch 10/15
53/53 [==============================] - 201s 4s/step - loss: 0.0435 - accuracy: 0.9840 - val_loss: 0.0761 - val_accuracy: 0.9853
Epoch 11/15
53/53 [==============================] - 200s 4s/step - loss: 0.0612 - accuracy: 0.9809 - val_loss: 0.0922 - val_accuracy: 0.9760
Epoch 12/15
53/53 [==============================] - 199s 4s/step - loss: 0.0327 - accuracy: 0.9877 - val_loss: 0.0901 - val_accuracy: 0.9773
Epoch 13/15
...
Epoch 14/15
53/53 [==============================] - 192s 4s/step - loss: 0.0283 - accuracy: 0.9899 - val_loss: 0.0556 - val_accuracy: 0.9853
Epoch 15/15
53/53 [==============================] - 196s 4s/step - loss: 0.0359 - accuracy: 0.9874 - val_loss: 0.0527 - val_accuracy: 0.9787
```

.

Model Output:

```
Speaker: Julia_Gillard  Predicted: Julia_Gillard
Welcome
The speaker is Julia_Gillard
Speaker: Julia_Gillard  Predicted: Julia_Gillard
Welcome
The speaker is Julia_Gillard
Speaker: Jens_Stoltenberg       Predicted: Jens_Stoltenberg
Welcome
The speaker is Jens_Stoltenberg
Speaker: Nelson_Mandela Predicted: Nelson_Mandela
Welcome
The speaker is Nelson_Mandela
Speaker: Benjamin_Netanyau      Predicted: Benjamin_Netanyau
Welcome
The speaker is Benjamin_Netanyau
Speaker: Benjamin_Netanyau      Predicted: Benjamin_Netanyau
Welcome
The speaker is Benjamin_Netanyau
Speaker: Jens_Stoltenberg       Predicted: Jens_Stoltenberg
Welcome
The speaker is Jens_Stoltenberg
Speaker: Nelson_Mandela Predicted: Nelson_Mandela
Welcome
The speaker is Nelson_Mandela
...
The speaker is Benjamin_Netanyau
Speaker: Julia_Gillard  Predicted: Julia_Gillard
Welcome
The speaker is Julia_Gillard
```

Conclusion:

1. To accurately predict the voices with noises in the “test” audio file in Kaggle dataset, I need to process current training data by adding background noise.
2. Since there are unknown voices in the Kaggle “test” audio file, I also need to add “unknown” as one of my labels.
3. I can also use the combination of CNN and RNN to see if I can improve the model accuracy or no