Monaural Single-Source Speaker Separation

E511 - Machine Learning for Signal Processing - Final Project

Jonathan Branam, Jamie Israel, Jinju (Hellen) Jiang

***Abstract*—In this project we evaluated and implemented state of the art techniques to solve the monaural speech separation (diarization) problem using a variety of approached. We investigated classification, masking, and deep learning approaches to separate a generated set of mixed speech recordings.**

I. Introduction

There have been several recent developments in the use of deep learning to perform speaker diarization, or separation of multiple speech signals from a single source into separate homogeneous signals representing each individual speaker. This task is particularly challenging where the input is monophonic, there is no control over environmental conditions, the number of speakers is unknown and there are no samples of the speakers from which to train. Using the CSR-I(WSJ0) Complete dataset[**?**], we tested several algorithms for accomplishing monaural speech separation including classification (KMeans), various masking approaches (Nonnegative Matrix Factorization [NMF], Ideal Ratio Mask[IRM], and complex Ideal Ratio Mask [cIRM]), and permutation-based deep learning approaches (Permutation Invariant Training [PIT] and utterance-level Permutation Invariant Training [uPIT]).

The motivating application is to separate agent and customer utterances from call center recordings, which involve short duration speech with occasional cross talk and at least one unknown speaker.

II. Literature Review

A. Signal Approximation

There are several approaches to signal separation (e.g., denoising, diarization) based on the general concept that audio can be represented as the combination of a frequency matrix and a corresponding matrix temporal activations. As a supervised learning problem, a time-frequency representation can be used to emphasize the points at which a target signal dominates the mixed audio and deemphasize the points where an alternate speaker (or other noise source) predominates.[7]

Masking functions range from the Ideal Binary Mask (IBM), in which a masking matrix is used to turn certain frequencies on or off at given moments based on a binary (0/1) assessment of the dominant source at the given time and frequency.[7] IRM provides a more nuanced approach by representing the relative dominance of each source at a given time and frequency as a ratio of the source signals rather than a binary mask. And, cIRM is an Ideal Ratio Mask calculated in the complex domain, which allows for a full reconstruction of the audio signal. [7]

Each of the aforementioned masking models relies on a global representation of the frequency spectrum associated with a given audio sample. More recently, probabilistic approaches have been used to represent the signal as a collection of frequency spectrum (hidden states) over time. On such approach, Nonnegative Hidden Markov Model (N-HMM), models a magnitude spectrogram for each latent component (source) at a given time frame as one of many states; like a traditional HMM, movement between states - and the spectral representation governing each latent component - is learned and defined by a transition matrix. [**?**]

B. Deep Learning Approaches

A seminal paper in applying deep learning to the task of speaker diarization was the Deep Clustering (DPCL) [5] approach proposed by Hershey, et. al. They divided the task of separating source signals into two steps. The first step is to use a deep neural network to learn a set of embeddings that estimate a class-independent, low-rank pairwise affinity matrix. These embeddings are trained to minimize the distance between embeddings in the same partition while maximizing the distances between embeddings in different partitions. The partitions used in their technique do not include class labels so the model does not assign a particular class to each embedding. The resulting embeddings can then be separated using simple clustering algorithms such as k-means. This is in contrast to previous work that relies on *spectral clustering* for segmentation which uses local affinity measures to optimize an objective function using spectral decomposition. The approach of Hershey, et. al. is typical in current research for deep learning: rather than using a complicated set of specially designed features, the deep clustering approach uses a deep neural network to discover the best features for producing the desired partitions.

This work is also important because it introduced a new training dataset for the task of monaural speech separation. The authors found that existing datasets were too limited in size or were not well fitted to their task. The deep learning techniques also require a large training set and so the authors adapted the Wall Street Journal (WSJ0) [**?**] single-channel speaker corpus to produce a new dataset called WSJ0-2mix [3] that was used by the later literature for comparison with DPCL.

The Deep Clustering technique achieved Signal-to-Distortion (SDR) improvements of up to 6 dB which outperformed their baseline of an oracle NMF approach of 5 dB SDR which was trained on the clean speaker sources.

It is worth noting the these initial results were improved to 10.3 dB by further optimization of the model by using a deeper architecture, a longer temporal context, and improving the regularization of the network by [6]. This advancement shows the improvements possible by tuning hyper-parameters of a deep learning model when applied to the task of speaker diarization.

One of the biggest problems plaguing deep learning techniques in source separation is the *label permutation problem* which is that most separation algorithms must provide the reference magnitudes to the output layers of the network to compute the error. The ordering of these source streams is ambiguous and imposes an outside and unnecessary constraint on the training process. Note that DPCL, discussed above, avoids this problem due to its treatment of the source separation problem as a partitioning problem. Although DPCL was successful, it made a simplifying assumption that each time-frequency bin was assigned to a single speaker (determined by the results of clustering). In [8] Yu, et. al. proposed a new technique called permutation invariant training (PIT) that avoids the label ambiguity problem by treating the reference source streams as a set rather than an ordered list. One key innovation of PIT is to directly train on all possible permutations of sources ($S!$ possible assignments) and to compute the combined pairwise MSE for every assignment between each reference and the estimated source. PIT then directly minimized the MSE of the assignment with the lowest total MSE of every permutation. This means that the model is directly trained on label assignment and error evaluation.

PIT operates on a sequence of N input frames termed a *meta-frame* in order to make use of the available contextual information in the sequence. It also produces a window of frames as an output and minimizes the separation error at this meta-frame level. Since the input is shifted one (or more) frames and a new meta-frame is computed and the assignment is computed for each meta-frame, the output of PIT may produce different speaker assignments for each frame. This means that, although the label permutation problem at training time has been resolved, the output of PIT can still be improved by tracing the speaker assignments for each frame as a post-processing step.

The results of PIT are impressive and exceed 10 dB SDR improvements for the DNN and CNN approaches with optimal speaker assignment. The ideal ratio mask (IRM) results indicate the best possible improvement and are only slightly better than PIT with 12.3 dB SDR. Optimal speaker assignment means that as a post-processing step the speaker assignment from PIT is improved by tracing the speakers across meta-frames and determining the optimal speaker assignment for each frame. This improves performance because the PIT method assumes that the output-speaker does not switch across frames. Without the optimal assignment the performance of PIT degrades from a high of 10.9 dB to 5.6 - 7.8 dB.

The authors of PIT improved their technique by proposing uPIT which is an enhancement to PIT that includes an utterance-level cost function which solves the speaker tracing problem by causing the Recurrent Neural Network (RNN) to align frames from a particular speaker to the same output stream. uPIT (like PIT) is also a much simpler model than other proposed deep learning algorithms and the trained networks have strong generalization performance to unseen data. In the original PIT algorithm the optimal permutation is computed for each meta-frame which can result in the assignment of speakers to output streams to switch during inference. With uPIT the optimal permutation is calculated across the entire utterance assuming the same assignment for all output frames. This works in practice by using recurrent neural networks such as deep LSTM (long short-term memory) RNNs and bi-directional LSTM RNNs. Since a recurrent neural network has access to data from past frames (and in a BLSTM information both the past and future frames are available) uPIT does not need to be trained on a meta-frame of N stacked frames as PIT did.

The results of uPIT are similar to the performance of PIT with optimal speaker tracing. Using a BLSTM and a phase sensitive mask (PSM, also referred to as a cIRM - a complex ideal ratio mask) uPIT achieves results near 11 dB SDR. The maximum performance from the ideal PSM is about 15 dB SDR (an improvement over the 12.5 dB SDR possible with a standard IRM). Based on its performance and (relative) simplicity We chose uPIT as the deep learning solution that we evaluated in our project.

III. Dataset Preparation

A. CSR-I (WSJ0) Complete

For purposes of comparison with existing scholarship, each approach was developed based on a mixture of audio extracted from the WSJ0 dataset, a corpus of read speech texts drawn from articles in the Wall Street Journal. WSJ0 [?] consists of over 35,000 audio samples organized into training and evaluation data sets organized by speaker and gender. The audio files are stored in a format called Sphere using the Shorten compression algorithm. The Linguistic Data Consortium website provides a small utility called sph2pipe [?] which we used to convert the files from the Sphere format to waveform format. We wrote a small script to automate this process and also to merge the separate disks of the WSJ0 dataset into a single folder.

B. WSJ0-mix

The WSJ0-mix dataset was developed by the authors of the DPCL [5] paper by mixing selections from the WSJ0 training set `si_tr_s` and mixing them together with different levels of signal-to-noise ratios (SNR) selected from 0 dB to 5dB. An additional cross validation was also generated from the training data and an evaluation set was generated with the same process but using the WSJ0 evaluation set `si_et_05` and development set `si_dt_05` which use different speakers than the training set. We found the definition of these mixtures and MATLABS scripts [3] to produce the dataset from the WSJ0 data. We made some small adjustments to the scripts so that they would run in Octave, a free, open-source alternative to MATLAB, and to skip the step of generating 16 kHz outputs and the 3-speaker mixtures since we would not be using them.

The scripts produce 20,000 training files, 5,000 cross validation files, and 3,000 test files. Each of these includes the mixed audio as well as the independent sources which have had the correct SNR adjustments made to them. We further selected 1,000 training examples and 100 cross-validation and testing examples since we our algorithm was not able to finish running on the full dataset.

IV. Speech Separation Algorithms

A. KMeans

The technique of speaker diarization relies on a big pipeline with following steps:

- Feature Extraction
- Speaker segmentation
- Speaker Clustering
- Evaluation

1) Feature Extraction: In this project, we used 3 approaches to extract features:

- A chroma vector (Wikipedia) that consists of 12 feature vectors indicating how much energy of each pitch class,In the signal we used the same notation as used i Western music notation, to present every pitch class. Then use mean of each element as one of input for clustering.
- MFCCs[Wikipedia] are commonly used as features in speech recognition systems, such as the systems which can automatically recognize numbers spoken into a telephone. In this experiment, mfcc computed 13 MFCCs.The first MFCC does not represent information related to the overall shape of the spectrum. It only contains a constant offset. Therefore, here I discard the first MFCC, then I scale the MFCCs such that each coefficient dimension has zero mean and unit variance.Then use mean of MFCCs as one part of input for clustering.
- We divide the audio signal into smaller frames. In smaller time scales audio signals are statistically unchanged. Then for each smaller frames we compute the power spectrum of the signal, and use it as the third part of input for clustering.

2) Speaker segmentation: [4] Speaker segmentation is to split an audio stream into smaller segments, so as every segment ideally contains only one speaker. So the key point for speech segment is to find out the change point where speakers changed at that point. For example, in our training data, we have conversation between customer service(speaker 0) and customer(speak 1), so we need to find out each change point speaker 0 to speaker 1 or vice versa. In this case, I skipped this step and used the features extracted from above steps as input for kmeans clustering.

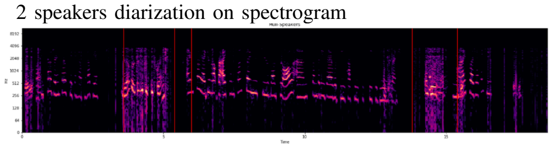
3) Kmeans Clustering: Since we knew that the audio from call center is the conversation between 2 speakers(customer service and customer), we set k=2 for kmeans clustering.
The results for kmeans speaker diarization are shown in Figure1 and Figure 2.

4) Evaluation[4]: The common metric to measure how good/bad for speaker diarization experiments is the Diarization Error Rate(DER), it contains 3 main parts:

Fig. 1. 2 speakers diarization output

```
TIME : 3.600000000000001 ---- SPEAKER : 0
TIME : 5.400000000000002 ---- SPEAKER : 1
TIME : 6.000000000000003 ---- SPEAKER : 0
TIME : 13.799999999999983 ---- SPEAKER : 1
TIME : 15.399999999999977 ---- SPEAKER : 0
```

Fig. 2. 2 speakers diarization on spectrogram



- The first one is called Missed Speech Rate(MSR), whichi is percentage of scored time that a hypothesized non-speech segment corresponds to a reference speaker segment.
- The second one is False alarm speech rate (FASR), which is percentage of scored time that a hypothesized speaker is labelled as a non-speech in the reference.
- The third one is Speaker error rate(SER), which is percentage of scored time that a speaker ID is assigned to the wrong speaker.
- The last one is Overlap speaker rate(OSR), which is percentage of scored time that some of the multiple speakers in a segment do not get assigned to any speaker.

So the Diarization Error Rate(DER) can be calculated as the sum of the above 4 rates(MSR+FASR+SER+OSR)
In this case, we don't have any labeled data to identify which segmentation is speaker 1/2 to calculate the training error, we just simply used human ears to justify whether the diarization was good or not. For some training data, it works well, but some training data, it performed really bad.

B. NMF, IRM and cIRM

Nonnegative matrix factorization (NMF) is used to represent high-dimensional data as the product of two matrices (typically referenced as W and H). In the case of an audio signal, these matrices can be viewed as a representation of the signal's frequency spectrum (W) and the corresponding activations (H).

To perform NMF, we transformed isolated audio samples of two speakers (speaker 1 and speaker 2) to a frequency-time representation using short-time Fourier transform (STFT) before decomposing each signal using the following iterative update rules:

$$W = W \odot \frac{\frac{X}{WH} H^T}{1_{FxT} H^T} \quad H = H \odot \frac{W^T \frac{X}{WH}}{W^T 1_{FxT}} \tag{1}$$

Using the frequency matrices associated with the sample from each speaker, a new set of activations (W) was generated from the mixed signal composed of both individual samples. A magnitude masking matrix, reflecting the magnitudes associated with the frequency representation for each speaker at each time frame, was generated using this new set of activations and the following formula:

$$\frac{W_{S1}H_{(1:30,:)}}{W_{S1}H_{(1:30,:)} + W_{S2}H_{(31:60,:)}} \tag{2}$$

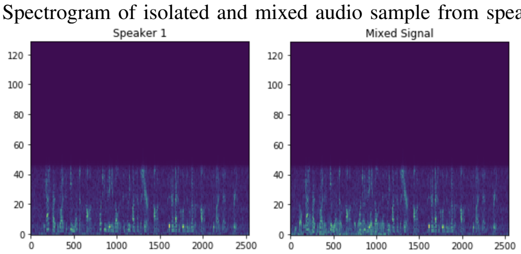
where S1 is the basis vector associated with the speaker 1 audio sample, S2 is the basis vector associated with the speaker 2 audio sample and H is the activation matrix generated from the combined speaker 1 and speaker 2 basis vectors.[2]

Using the STFT of a single speaker audio sample (speaker 1) and the STFT of the mixed audio sample that included speaker 1 (Figure 3), an IRM was generated based on the formula:

$$\frac{S^2_{(t,f)}}{S^2_{(t,f)} + N^2_{(t,f)}} \tag{3}$$

where S is the STFT generated from the isolated speaker 1 audio signal and N is the STFT generated from the mixed audio signal that included the same audio sample of speaker 1.[7]

Fig. 3. Spectrogram of isolated and mixed audio sample from speaker 1



Similarly, a cIRM was generated from the audio sample of speaker 1 and the same mixed audio sample using the formula:

$$\frac{Y_r S_r + Y_i S_i}{Y_r^2 + Y_i^2} + i\frac{Y_r S_i - Y_i S_r}{Y_r^2 + Y_i^2} \tag{4}$$

where S is the complex STFT representation generated from the isolated speaker 1 audio signal and Y is the complex STFT generated form the mixed audio signal that included the same audio sample of speaker 1.[7]

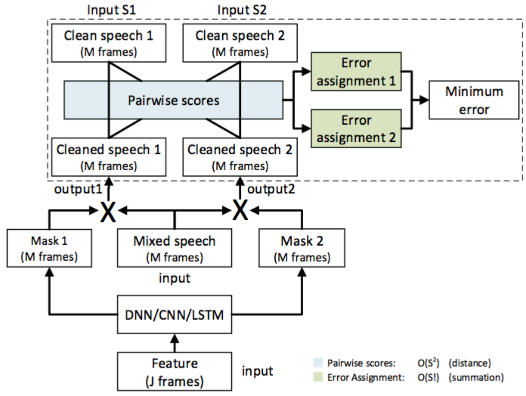
C. PIT and uPIT

Utterance-level Permutation Invariant Training (uPIT) is an advancement on the Permutation Invariant Training model depicted in Figure 4 where the reference signals are provided to the training model as a set rather than a list by calculating the error on all permutations of the output signals and then optimizing the model on the permutation with the lowest mean squared error (MSE). A traditional approach is to minimize the MSE between the estimated masks and the ideal masks such as in

$$J_m = \frac{1}{B}\sum_{s=1}^{S} ||\hat{\mathbf{M}}_S - \mathbf{M}_S||_F^2 \tag{5}$$

where $B = T \times N \times S$ is the count of all time-frequency units over all sources S and F refers to the Frobenius norm.

Fig. 4. PIT 2 speaker separation model



However, to avoid issues predicting masks in silence segments and to directly calculate the error relative to the signal reconstruction, PIT and uPIT minimize the MSE between the actual signals by applying the mask to the original spectra and then calculating the error. For PIT using PSM with complex values the cost function is

$$J_{\phi^*} = \frac{1}{B}\sum_{s=1}^{S} ||\hat{\mathbf{M}}_s \odot \mathbf{R} - \mathbf{A}_{\phi^*(s)} \odot \cos(\theta_y - \theta_{\phi^*(s)})||_F^2 \tag{6}$$

where ϕ^* is the permutation with the lowest MSE:

$$\phi^* = \arg\min_{\phi \in \mathcal{P}} \frac{1}{B}\sum_{s=1}^{S} ||\hat{\mathbf{M}}_s \odot \mathbf{R} - \mathbf{A}_{\phi(s)} \odot \cos(\theta_y - \theta_{\phi(s)})||_F^2 \tag{7}$$

The major difference in uPIT is instead of calculating the permutation error for every meta-frame uPIT calculates the error *across the entire utterance*. uPIT relies on using recurrent neural networks (RNNs) to provide the context necessary for good separation and for selecting the best permutation across the entire utterance. The cost function for uPIT is the same as for PIT but is calculated for the entire utterance rather than for each meta-frame.

V. Results

A. Masking Methods

With the NMF method, we were unable to generate a substantial separation between the original isolated signal and the mixed audio. Both IMR approaches produced substantially better results with the cIMR method proving to be the best method for this type of separation on a consistent basis [Figure 5].

Fig. 5. Average SNR for Masking Methods

Method	SNR
NMF	1.924
IRM	6.380
cIRM	15.070

B. uPIT

Our implementation of uPIT uses a BLSTM with 3 hidden layers of size 896. We used the Adam optimizer and set a 0.5 rate for dropout. We trained using Google Colab free instances which allow using a Tesla K80 GPU with 12 Gb or RAM. We found that training with 1,000 samples for 60 epochs took about 5 hours in this setup. The Google Colab GPU will disconnect after 12 hours so we were unable to complete a full training run. We experimented with saving off the model parameters and restarting the training once the 12 hours had expired but were unable to complete the implementation with enough time to train again.

Our results with uPIT were not very good, probably due to the small training size that we used. We were able to find two open source implementations of uPIT one of them we used as a reference only [CITE 1] and the other [CITE 2] we were able to actually implement in order to verify our results. The author was able to achieve results comparable to those published in the uPIT paper and he agreed that we should not expect to match the paper's performance unless we are able to train on the full dataset.

VI. Conclusion

We got our best results from the cIRM with NMF separation. uPIT needed more training time and Kmeans...

References

[1] CSR-I (WSJ0) Complete. Web Download. Garofolo, John S., et al. CSR-I (WSJ0) Complete LDC93S6A. Philadelphia: Linguistic Data Consortium, 1993.

[2] Minje Kim, ENGR-E 511 Machine Learning for Signal Processing, Module 04: Dimension Reduction, Slides 34-37 (Fall 2018). Web Download. https://iu.instructure.com/courses/1737711/files/83333078?module_item_id=17726720.

[3] Mitsubishi Electric Research Laboratories (MERL). Web Download. http://www.merl.com/demos/deep-clustering/create-speaker-mixtures.zip.

[4] Unsupervised Speaker Diarization. Web Download. Akshay kumar,Anurendra Kumar,Unsupervised Speaker Diarization.

[5] John R. Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. *CoRR*, abs/1508.04306, 2015.

[6] Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R. Hershey. Single-channel multi-speaker separation using deep clustering. *CoRR*, abs/1607.02173, 2016.

[7] DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *CoRR*, abs/1708.07524, 2017.

[8] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multitalker speech separation. *CoRR*, abs/1607.00325, 2016.