Attention Training: A New Training Framework for Talker-independent Speaker Extraction
Ashutosh Pandey and DeLiang Wang

1Department of Computer Science and Engineering, The Ohio State University, USA
{pandey.99, wang.77}@osu.edu

Abstract

Listening in a multitalker scenario, we typically attend to a single talker through auditory selective attention. Inspired by human selective attention, we propose attentive training: a new training framework for talker-independent speaker extraction with an intrinsic selection mechanism. In the real world, multiple talkers very unlikely start speaking at the same time. Based on this observation, we train a deep neural network to create a representation for the first speaker and utilize it to extract or track that speaker from a multitalker noisy mixture. Experimental results demonstrate the superiority of attentive training over widely used permutation invariant training for talker-independent speaker extraction, especially in mismatched conditions in terms of the number of speakers, speaker interaction patterns, and the amount of speaker overlaps.

Index Terms: speaker extraction, speaker separation, talker-independent, attentive training

1. Introduction

The cocktail party effect refers to the amazing ability of auditory perception attending to (hence extracting) a single speaker in a multitalker noisy scenario [1]. This effect has influenced the development of the perceptual theory of selective attention [2]. Separating all speakers or extracting a single one from a multitalker mixture is considered very challenging for machines, however, the introduction of deep learning to such tasks has led to dramatic advances in recent years [3].

Currently, there are two approaches to speaker extraction: speaker separation and target speaker extraction. Speaker separation aims at separating all speakers from a mixture. Early speaker separation work is extended from deep neural network (DNN) based speech enhancement, and such separation is talker-dependent. When applied to talker-independent speaker separation, these models suffer from the well-known permutation ambiguity problem, i.e., an underlying model cannot consistently assign DNN output streams to different speakers during training. Deep clustering [4] and permutation invariant training (PIT) [5] are two representative approaches to resolving the permutation ambiguity problem. In particular, the simplicity of PIT has led to many subsequent models for speaker separation [6, 7, 8, 9, 10].

Target speaker extraction aims at extracting a single speaker from a multitalker mixture, where the target speaker is cued with the help of some additional information in the form of audio [11, 12, 13, 14, 15, 16] or images [17, 18, 19]. Other kinds of cues include spatial [20, 21], speech activity [22], or onsets [23]. Target speaker extraction is closer to auditory selective attention, but it is not intrinsic to model training.

In this work, we propose a new training framework, which we name attentive training, for talker-independent speaker extraction (or tracking). In the real-world environments, it is very unlikely that multiple talkers start speaking at the same time; such a case would lead to their grouping into the same auditory stream on the basis of common onsets [24]. Therefore, we can assume that a given multitalker mixture has nonoverlapping speech in the beginning. We provide this important cue to a DNN to create a representation for the starting speaker in the early part of processing and utilize this representation for attending to this speaker throughout training. In other words, network selects the first speaker and then attends to it for the rest of the multitalker mixture. This, in a way, resembles speech enhancement in the sense that we treat the speech signals of the first speaker as speech, and the utterances of other speakers plus environmental sounds as background noise.

The attentive training framework is consistent with the dominant feature integration theory of attention [25]. According to this theory, attention serves to integrate perceptual features extracted in separate analyses into an object. Learning and attention are integral parts of perception.

A similar idea of using speaker onsets as a cue was proposed in serialized output training (SOT) [26]. The SOT uses onset order of speakers to determine the transcription order at the output of an automatic speech recognition (ASR) system. The proposed attentive training, on the other hand, aims at dealing with speaker interference in a speech enhancement system, and hence, is fundamentally different from SOT. The SOT outputs transcriptions of all the speakers in a mixture, whereas the attentive training outputs the enhanced speech corresponding to the first speaker.

To evaluate the efficacy of attentive training, we create a multitalker dataset by setting the first speaker to start slightly ahead of the rest of speakers in an utterance. Next, we train an end-to-end time-domain model to predict the first speaker in a given mixture. We demonstrate that a DNN model trained with attentive training generalizes well to different test conditions, such as an untrained number of speakers, utterances with larger gaps between consecutive segments of the target speaker, and smaller speaker overlaps.

Further, we compare attentive training with PIT, and observe significant improvements in many conditions, especially in mismatched conditions. We demonstrate that attentive training can overcome the shortcomings of PIT, such as sensitivity to the number of speakers, interaction patterns and amount of speaker overlaps.

Additionally, we introduce a novel data generation technique for mixing an arbitrary number of speakers in a controlled way. Given a set of speakers, their corresponding utterances,
2. Attentive Training Mechanism

2.1. Problem Definition

A multitalker mixture \( y \) with \( N \) samples is modeled as

\[
y = \sum_{i=1}^{N} s_i + n
\]

(1)

where \( \{ y, s_i, n \} \in \mathbb{R}^{1 \times N} \), \( s_i \) is the \( i \)th speaker, \( n \) is the background noise, and \( C \) is the total number of speakers. Let \( m_i \) represent the sample when the \( i \)th speaker starts to speak. Without the loss of generality, we can assume that indices \( i = 1, 2, \ldots, C \) are sorted in the order of \( m_i < m_{i+1} \). The goal of attentive training is to get a close estimate \( \hat{s}_1 \) of the first speaker \( s_1 \) from \( y \).

2.2. Data Generation

In this section, we describe our technique to generate multitalker mixtures. We assume that we are given a set \( S = \{ S_1, \ldots, S_M \} \) of speakers, their corresponding utterances \( U^S = \{ s_1, \ldots, s_N \} \), and a set of noise segments \( N = \{ n_1, \ldots, n_N \} \). We create a multitalker mixture by adding together speech segments of multiple speakers and a noise segment.

First, we sort a given set of speech segments in an increasing order of their onset times, and based on this, define a concept called interaction pattern representing the order of speakers in a mixture. For example, an interaction pattern of 1212 represents a mixture created by adding 4 segments sorted in increasing order of their onset times, where the first and the third segments are from the first speaker and the second and the fourth segments are from the second speaker. We also define two parameters \( A \) and \( B \), where \( A \) is the minimum allowed gap between the onset of the first and the second speaker, and \( B \) is the gap between two adjacent nonoverlapping segments (regardless of speakers). We illustrate two interaction patterns in Fig. 1. For data generation, we use interaction patterns from a predefined set \( P = \{ p_1, \ldots, p_P \} \).

Similar to the LibriCSS dataset [28], we generate mixtures in a way that a given mixture can have an arbitrary number of speakers, but at a given time instant, only a maximum of two speakers can overlap. Algo. 1 describes the steps used in generating a sample mixture from \( S, U, N \), and \( P \).

Algorithm 1 A pseudo code for generating a random multitalker mixture.

1. **Input**: \( S, U, N, P \)
2. **Output**: \( y, s_1, \ldots, s_C, n \)
3. Sample a speaker pattern \( \pi \) from \( P \)
4. Set \( C = \text{Len}(\text{Unique}(\pi)) \)
5. Sample \( C \) speakers \( S_1, \ldots, S_C \) from \( S \)
6. Initialize List \( V = [ ] \), \( E = [ ] \), \( \text{Bool} \text{Overlap} = \text{False} \)
7. for \( k \) in \( p \) and \( i \) in \( \{ 1, 2, \ldots, \text{Len}(\pi) \} \) do
8. Sample an utterance \( s \) from \( U^S \)
9. Remove silences in the beginning and the end of \( s \)
10. Sample a value \( T \) for segment length
11. Extract a random segment \( \alpha \) of length \( T \) from \( s \)
12. Set Overlap = True with a probability \( \text{Overlap} \)
13. if \( i = 1 \) then
14. PadLeft = 0
15. else if \( i = 2 \) then
16. Sample a value for \( B \)
17. if \( k \) in \( E \) then
18. Overlap = False
19. end if
20. if no Overlap then
21. Set PadLeft = \( E[\ -1 \] + \( B \)
22. else
23. Sample PadLeft from \( [ A, E[\ -1 \] ] \)
24. end if
25. else
26. Sample a value for \( B \)
27. if \( k \) in \( E \) and \( E[i][k] = E[\ -1 \] then
28. Overlap = False
29. end if
30. if no Overlap then
31. Set PadLeft = \( E[\ -1 \] + \( B \)
32. else
33. Sample PadLeft from \( [ E[\ -2 \] + B, E[\ -1 \] ] \)
34. end if
35. end if
36. Apply a left padding of PadLeft to \( x \)
37. Set \( V[\ i \] = \( x \)
38. Set \( E[\ -1 \] = \text{Len}(\alpha) + \text{PadLeft} \)
39. Set \( E[i][k] = \text{Len}(\alpha) + \text{PadLeft} \)
40. Sort \( E \) in increasing order
41. Set \( E = [ E[\ -2 \], E[\ -1 \] ] \)
42. end for
43. Apply right padding to segments in \( V \) to match lengths
44. Sample a separate value of loudness for all segments
45. Scale all segments for loudness
46. Create a multitalker mixture by adding all segments together
47. Sample a noise segment from \( N \)
48. Sample a value for noise loudness
49. Scale the noise segment for loudness and add it to the multitalker mixture

Figure 1: Examples of interaction patterns with 2 and 3 speakers, and a minimum onset gap of \( A \) between the first and the second speaker. In pair \( (a, b) \) inside a box, \( a \) and \( b \) respectively represent the speaker order and the segment order.
segments from the same speaker do not overlap. We remove silences from all utterances and then pad zeroes in the beginning to shift a given segment. We use no padding for the first speaker, the second speaker has a minimum padding of \( A \), and the rest of the speakers use padding in a way that a maximum of two speakers overlap at a time.

2.3. DNN Model

We employ a recently proposed attentive recurrent network (ARN) for time-domain speech enhancement [29]. The model architecture is shown in Fig. 2. It comprises an input linear layer followed by four ARN layers and one output linear layer. An input mixture \( y \) is first converted to frames \( X \in \mathbb{R}^{T \times L} \), where \( T \) is the number of frames and \( L \) is the frame size. Next, frames in \( Y \) are projected to size \( D \), processed by a stack of four ARN layers, and projected back to size \( L \) using the output linear layer. Finally, an overlap-and-add (OLA) is used to get the enhanced waveform. A more detailed description of ARN can be found in [29].

2.4. Loss Function

For attentive training, we use an utterance level SNR loss between the first speaker \( s_1 \) and its estimate \( \hat{s}_1 \), defined as

\[
L(s_1, \hat{s}_1) = -10 \cdot \log_{10} \frac{|\|s_1\|^2|}{|\|s_1 - \hat{s}_1\|^2|} \tag{2}
\]

3. Experiments

3.1. Datasets

We create two datasets: one from the WSJ0 corpus [30] and the other from Librispeech [31]. For WSJ0, we use a random split of 90% and 10% from \( s_i \) to \( s_i \) speakers for training and validation. All speakers from \( s_i \) and \( s_i \) are used for evaluation. For Librispeech, we use speakers from train-clean-100, dev-clean, and test-clean for training, validation, and evaluation respectively. The Librispeech corpus is more challenging than WSJ0 as it has a larger number of speakers, and a wide variety of acoustic conditions [32].

We use noises from the WHAM! corpus [33]. First, we split training noises into 10-s chunks, and validation and test noises of acoustic conditions [32]. We use noises from the WHAM! corpus [33]. First, we split training noises into 10-s chunks, and validation and test noises of acoustic conditions [32].

We use scale-invariant SNR (SI-SNR), extended short-time objective intelligibility (eSTOI) [35], and perceptual evaluation of speech quality (PESQ) [36] as evaluation metrics. Objective scores are computed for the first speaker and eSTOI is reported in percentage.

3.2. Experimental results

3.2. Experimental results

All the utterances are resampled to 16 kHz. A frame size of 16 ms, frame shift of 4 ms, and \( D = 512 \) is used for ARN. ARN uses BLSTMs with 256 hidden units in both directions. We train 2-speaker and 3-speaker models. All the models are trained on interaction patterns with 4 segments. A 2-speaker model is trained on 1 and 2 speakers. A 3-speaker model is trained on 1, 2, and 3 speakers. For a \( p \)-speaker PIT model, we use \( p \) linear layers at the output, and for an input with \( k \) speakers (\( k <= p \)), we select the minimum loss assignment from all possible \( C^p_k \) assignments.

All training samples are randomly and dynamically generated during training, and an episode of 100k samples is considered as one epoch. We use \( \rho_{overlap} = 0.75, A = 1 \) second. \( B \) is sampled from [0.25, 0.50] seconds. Segment length, \( T \), is sampled from [2, 4] seconds for training and from [2, 4] seconds for validation and test. We use interaction patterns 1211 and 1231 for validation of 2-speaker and 3-speaker models respectively. We provide our dataset generation script at https://github.com/asutosh620/AttentiveTraining.

All the training utterances longer than 10 seconds are trimmed to 10 seconds. All the models are trained for 100 epochs using the loss in (2) with a batch size of 26 utterances. The learning rate is initialized with 0.0004 and scaled by 0.98 after every two epochs.

All the models are evaluated on interaction patterns from \{1111, 1212, 1222, 122221, 1231, 123231, 12341, 123451\} and three overlap types: \{Max, Half, None\}. Following Algo. 1, Max uses maximum allowed overlap, Half uses half of the allowed regions for overlap, and None uses no overlap. We generate 3000 evaluation utterances for each combination of the interaction pattern and overlap type. The pattern 1212 is used to assess performance for an alternating pattern of the target and interfering speaker, 1221 is used to assess performance with a larger gap between two consecutive segments of the target speaker. The pattern 122211 is used to assess performance with an even larger gap not used during training. Similarly, patterns 12231 and 123231 are used to assess performance for 3 speakers with different gaps, where 123231 is not used during training. Patterns 12341 and 123451 are used to assess performance for untrained numbers of 4 and 5 speakers.

We use LKFS [34] based loudness for controlling the SNR. We sample loudness from \([-25, -30]\) dB for speaker segments and from \([-35, -40]\) dB for noise segments.

Table 1: Attentive training and PIT comparisons for single speaker test set with speaker pattern 1111.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>WSJ0</th>
<th>LibriSpeech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>SI-SNR</td>
<td>PESQ</td>
</tr>
<tr>
<td>Max</td>
<td>9.6</td>
<td>76.3</td>
</tr>
<tr>
<td>AT-2</td>
<td>18.3</td>
<td>92.8</td>
</tr>
<tr>
<td>PIT-3</td>
<td>17.7</td>
<td>91.7</td>
</tr>
<tr>
<td>AT-3</td>
<td>17.4</td>
<td>92.3</td>
</tr>
</tbody>
</table>
Table 2: Attentive training and PIT comparisons for 2 to 5 speakers. (a) number of speakers, (b) interaction pattern.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Type</th>
<th>Max</th>
<th>Min</th>
<th>WSJ1</th>
<th>LibriSpeech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Number of Speakers**: 2-5
- **Types of Interaction Patterns**: Mix.
- **Models Evaluated**: AT-2, AT-3, PIT-2, PIT-3
- **Performance Metrics**: SI-SNR, PESQ, ESTOI
- **Results**: AT-3 outperforms AT-2 and PIT in most cases for 2-5 speakers.

4. Conclusion

We have proposed a novel attentive training framework for talker-independent speaker separation. The proposed framework has an intrinsic mechanism for speaker selection. Experimental results have demonstrated the superiority of attentive training over permutation invariant training, especially in mismatched test conditions. Future work includes evaluating the proposed framework for multitalker speaker separation and speaker diarization.
5. References


