Streaming Multi-talker Speech Recognition with Joint Speaker Identification

Liang Lu, Naoyuki Kanda, Jinyu Li and Yifan Gong

Microsoft Corp., USA

{liang.lu, naoyuki.kanda, jinyli, yifan.gong}@microsoft.com

Abstract

In multi-talker scenarios such as meetings and conversations, speech processing systems are usually required to transcribe the audio as well as identify the speakers for downstream applications. Since overlapped speech is common in this case, conventional approaches usually address this problem in a cascaded fashion that involves speech separation, speech recognition and speaker identification that are trained independently. In this paper, we propose Streaming Unmixing, Recognition and Identification Transducer (SURIT) – a new framework that deals with this problem in an end-to-end streaming fashion. SURIT employs the recurrent neural network transducer (RNN-T) as the backbone for both speech recognition and speaker identification. We validate our idea on the LibrispeechMix dataset – a multi-talker dataset derived from Librispeech, and present encouraging results.

Index Terms: Overlapped speech recognition, Streaming, Unmixing transducer, Joint recognition and identification

1. Introduction

In [1], Morgan *et al.* referred to the process of spoken language from meetings as a nearly "ASR-complete" problem. The authors used this metaphor to highlight that almost all the key problems in spoken language process can be explored in the context of meetings, such as conversational speech recognition, signal processing with far-field microphone arrays, overlapped speech detection and separation, utterance segmentation and speaker identification or diarization, disfluency detection, etc. Indeed, rich transcription of meetings and conversational speech is still a challenging problem to the speech community nowadays. Since the birth of the ICSI meeting corpus [2] and AMI dataset [3], the progress to deal with this challenge has been made mainly by following the divide-and-conquer strategy, namely, researchers usually attack each sub-problem independently, and put all the components together in a cascaded fashion for meeting transcription [4, 5]. Not surprisingly, such systems are difficult to be optimized end-to-end, and their computational complexities are usually very high. More importantly, such systems are usually not capable to be deployed in a streaming scenario with a low latency requirement.

Since deep learning reshaped the landscape of the speech field [6], recent years have witnessed the paradigm shift for almost every problem covered by the meeting transcription scenario, such as automatic speech recognition (ASR) [7, 8, 9, 10, 11], speaker verification [12, 13], speech separation [14, 15], and speaker diarization [16, 17, 18] etc. As a result, the systems for meeting transcription have evolved remarkably [19]. However, these systems mostly still operates in a cascaded fashion, and have inherited the drawbacks of their previous generation. Recently, an emerging trend in the speech community is to address the problems in meetings in an end-to-end fashion, and jointly optimize all the components. In particular,

there are series of research studies on end-to-end multi-talker ASR that deal with speech separation and speech recognition jointly [20, 21, 22, 23]. While those works are mostly constrained in an offline condition, in [24, 25], the authors investigated steaming multi-talker ASR based on recurrent neural network transducers (RNN-T), and reported promising results.

In this paper, we aim at a further stride along this direction, and explore joint speech separation, ASR, and speaker identification (SID) with a streaming, end-to-end model. Built upon our prior work on streaming multi-talker speech recognition [24], we propose a novel approach referred as Streaming Unmixing, Recognition and Identification Transducer (SURIT) for this problem. While streaming end-to-end speech recognition has achieved a remarkable progress, most works on speaker identification only concern offline scenarios. In order to perform speech recognition and speaker identification jointly in a streaming fashion, we introduce RNN-T based speaker identification. Ultimately, similar to [24], SURIT firstly applies an Unmixing module to extract single-speaker feature representations, followed by RNN-Ts for streaming transcription and speaker identification. To evaluate our proposed model, we performed experiments using the LibriSpeechMix dataset [22], which simulate the overlapped speech data from the LibriSpeech corpus [26] with an artificially defined speaker identification protocol. We show that SURIT can achieve strong results for both speech recognition and speaker identification.

2. Related Work

There were limited studies on the joint modeling of end-to-end ASR and SID. One simple approach is inserting a speaker-role tag [27] or a speaker-identity tag [28] at the end of each utterance. However, [27] and [28] did not cope with overlapping speech and also their approach cannot identify a speaker who is not included in the training data. Recently, Kanda et al. [29] proposed an offline sequence-to-sequence (S2S) based approach that can transcribe and identify the speakers from overlapping speech given a set of speaker profiles. Our work differs from [29] mainly in that SURIT applies the RNN-T for both ASR and SID, which can run in a streaming fashion. The idea of using an RNN-T for SID is inspired by [30], which proposed RNN-T based language identification. Our speaker verification module depends on a speaker inventory as an auxiliary information as same with [29]. Overall, our work is the first study on RNN-T based SID that is trained jointly with multi-talker ASR in a *streaming* fashion.

3. RNN-T: Review

RNN-T is the backbone in our model, which directly computes the sequence-level conditional probability of the label sequence given the acoustic features by marginalizing all the possible alignments. Given an acoustic feature sequence $X = \{x_1, \cdots, x_T\}$ and its corresponding label sequence $Y =$
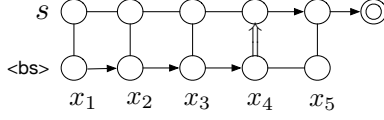
Figure 1: *An illustration of using RNN-T for streaming speaker identification. The arrowed path corresponds to $\{\langle bs \rangle \langle bs \rangle \langle bs \rangle s \langle bs \rangle\}$, where $\langle bs \rangle$ denotes the blank speaker token. The probability of the arc $\rightarrow$ is $b_{t,u}$, and the probability of the arc $\Rightarrow$ is $(1-b_{t,u})P(s \mid z_{t,u})$ in Eq. (6).*

$\{y_1, \cdots, y_U\}$, where T is the length of the acoustic sequence, and U is the length of the label sequence, RNN-T defines the conditional probability

$$P(Y \mid X) = \sum_{\tilde{Y} \in \mathcal{B}^{-1}(Y)} P(\tilde{Y} \mid X), \quad (1)$$

where $\tilde{Y}$ is a path that contains the blank token $\varnothing$, and the function $\mathcal{B}$ denotes mapping the path to Y by removing the blank tokens in $\tilde{Y}$. The probability can be efficiently computed by the forward-backward algorithm, which requires to compute the probability of each time step. Let $f(\cdot)$ denote the audio encoder, and $g(\cdot)$ the label encoder, the feature representations can be obtained as

$$\boldsymbol{h}_{1:t}^f = f(x_{1:t}), \quad \boldsymbol{h}_{1:u}^g = g(y_{0:u-1}),$$

where y_0 is a blank token. RNN-T fuses the two feature representations by a joint network $j(\cdot)$ as

$$\boldsymbol{z}_{t,u} = j(\boldsymbol{h}_t^f, \boldsymbol{h}_u^g), \quad (2)$$

and the conditional probability of the next token is obtained by a Softmax function, i.e.,

$$P(y_u \mid \boldsymbol{z}_{t,u}) = \text{Softmax}(\boldsymbol{W}\boldsymbol{z}_{t,u} + \boldsymbol{b}), \quad (3)$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and bias. Given $P(y_u \mid \boldsymbol{z}_{t,u})$, the sequence-level conditional probability can be computed by dynamic programming, and the RNN-T loss can be defined as the negative log-likelihood as:

$$\mathcal{L}_{\text{rnnt}}(Y, X) = -\log P(Y \mid X) \quad (4)$$

4. Streaming SID with Transducers

In this paper, we propose to apply transducers for streaming SID. Here, we assume a single-talker audio to introduce the idea of transducer-based SID. It will be extended for the joint modeling with multi-talker ASR with speech overlap in Section 5.

Suppose we have a speaker inventory $D = \{d_k | k = 1, ..., K\}$, where K is the total number of the profiles, d_k is a speaker embedding (e.g., d-vector [31]) of the k-th speaker in the speaker inventory. Given an input acoustic feature X, the goal of SID is estimating the speaker index k corresponding to X. To simulate the real scenario, the speaker inventory D can be *different* for each audio segment X, indicating that the participants of the meeting or conversation may vary each time. Standard SID approaches typically addresses this problem by extracting the vector representation of X, and measure its distance against the d-vectors in D according to a metric such as the cosine distance. However, those approaches usually operate in an offline fashion and require a voice activity detector (VAD) if the audio is not segmented. On the other hand, in this work, we focus on streaming SID that can emit the speaker label before the utterance is ended, and can handle both silent and voiced regions without a VAD.

4.1. Transducer-based SID

For the proposed transducer-based SID, we treat each speaker index $s \in \{1, ..., K\}$ as the token that should be predicted by the RNN-T, as illustrated in Fig. 1. To tailor the RNN-T model to fit in this scenario, we need to accommodate a few model structure changes. Since we do not capture the dependencies among the speaker labels, we omit the recurrent layers in the label encoder of RNN-T. To train such a model, we need to calculate the probabilities of emitting both the speaker label s and the blank token $\langle bs \rangle$. Given $\boldsymbol{z}_{t,u}$ from Eq.(2), the probability of emitting the speaker label s can be represented as

$$P(s = k \mid \boldsymbol{z}_{t,u}) = \frac{\exp(d_k^\top \boldsymbol{z}_{t,u})}{\sum_{k'=1}^{K} \exp(d_{k'}^\top \boldsymbol{z}_{t,u})}, \quad (5)$$

Where d_k is the d-vector of the target speaker. However, in the speaker inventory, there is no embedding vector for $\langle bs \rangle$, so we cannot apply Eq. (5) to compute the probability of $P(\langle bs \rangle \mid \boldsymbol{z}_{t,u})$. To deal with this problem, we apply a special type of RNN-T model, i.e., Hybrid Autoregressive Transducer (HAT) [32], which was originally proposed for external language model integration for RNN-T. The key idea of HAT is that it reformulates the token probability, and uses two distributions for blank and label tokens, i.e.,

$$P(v_u \mid \boldsymbol{z}_{t,u}) = \begin{cases} b_{t,u} & v_u = \langle bs \rangle \\ (1 - b_{t,u})P(s \mid \boldsymbol{z}_{t,u}) & \text{otherwise} \end{cases} \quad (6)$$

where $b_{t,u}$ is a Bernoulli distribution that can be estimated using a Sigmoid function, while $P(s \mid \boldsymbol{z}_{t,u})$ is defined as Eq. (5). Note that the value space of s is $[1, K]$, while the value space of v_u is $[1, K] \cup \langle bs \rangle$. With HAT, we can compute the emission probability of $\langle bs \rangle$ without its d-vector.

4.2. Optimizing the SID latency

Naively applying the RNN-T loss as Eq. (4) could cause the high latency of the HAT-based SID model. The reason is that the vanilla RNN-T loss does not encourage the model to emit the labels early. More critically, for each path in Fig. 1, the number of blank tokens is $T-1$ while there is only one speaker label for an audio input of T frames. Consequently, the gradient of $\langle bs \rangle$ is significantly larger than the gradient of speaker labels. This encourages the model to prefer emitting $\langle bs \rangle$ during inference. To counteract this effect, we scale the gradient of $\langle bs \rangle$ as

$$\frac{\partial \tilde{\mathcal{L}}}{\partial b_{t,u}} = \alpha \frac{\partial \mathcal{L}}{\partial b_{t,u}}, \quad \alpha \in (0, 1], \quad (7)$$

which is very similar to FastEmit [33], but we scale the gradient of $\langle bs \rangle$ since it is dominant. Inspired by the method to optimize the latency of end-pointer [34], another approach is to penalize the probability of emitting the speaker label late, i.e.,

$$\log P(s \mid z_{t,u}) -= \max(0, \beta(t - t_{\text{buffer}} - t_{\text{delay}})), \quad (8)$$

where β is a scaling parameter, t_{buffer} is the grace period, and t_{delay} means the delay when we simulate the overlapped audio. This equation means that for a path in Fig. 1, if the time step t that emits the speaker label is larger than $(t_{\text{buffer}} + t_{\text{delay}})$, then the probability of this path will be penalized.

5. Joint Multi-talker ASR and SID

In a multi-talker scenario, dealing with the overlapped speech is the key challenge. In [24], we proposed a streaming end-to-end
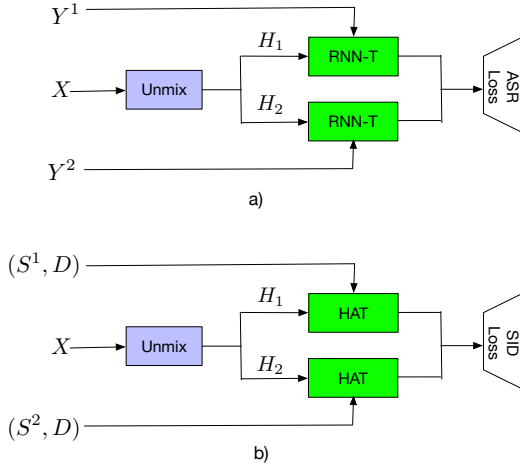
Figure 2: *a) The multi-talker speech recognition network. Y^1 and Y^2 are wordpiece labels, and X represents acoustic features. b) The multi-talker speaker identification network. S^1 and S^2 are speaker labels, and D refers to the speaker inventory. Note that, there are only one set of RNN-T and HAT model parameters, which are used to process the two feature streams from the Unmix module in the 2-speaker case. The Unmix module is shared by the speech recognition network and the speaker identification network.*

model for multi-talker speech recognition, referred as SURT. The idea of SURT is illustrated in Figure 2-a) for the 2-speaker case, in which, the Unmixing module emits two streams of audio representation where each stream contains the feature representations of one speaker. In this work, we use convolutional neural networks (CNNs) for the Unmixing module, in particular

$$M = \sigma(\text{CNN}_{\text{mask}}(X)), \quad H = \text{CNN}_{\text{enc}}(X),$$
$$H_1 = H * M, \quad\quad\quad H_2 = H - H_1, \quad\quad (9)$$

where σ denotes the Sigmoid function; CNN_{mask} and CNN_{enc} are two CNN encoders. The two feature representations are then fed into the same RNN-T network to compute the corresponding losses. It is well understood that label permutation is a key challenge for multi-talker speech processing [15]. Suppose the two ASR label sequences are Y^1 and Y^2, it is unclear if H_1 corresponds to Y^1 or Y^2, and similar for H_2. A widely used approach to address this problem is Permutation Invariant Training (PIT) [15], which considers all the possible label assignments. In [24], we evaluated another simple label assignment strategy, referred as Heuristic Error Assignment Training (HEAT) [23]. Unlike PIT, HEAT only considers one possible label assignment based on some heuristic information, for example, the order that they were spoken according to their starting times. Suppose Y^1 is from the speaker who spoken first, HEAT defines the loss function as

$$\mathcal{L}_{\text{asr}}(X, Y^1, Y^2) = \mathcal{L}_{\text{rnnt}}(Y^1, H_1) + \mathcal{L}_{\text{rnnt}}(Y^2, H_2), \quad (10)$$

which means that we always assign the feature representation H_1 to the first spoken speaker. The loss function can drive the network to learn the corresponding feature mapping. In [24], we show that HEAT works slightly better than PIT in terms of recognition accuracy, and it consumes less memory with lower computational cost.

Similarly to SURT, we can represent the SID network for multi-talker case as shown in Fig. 2-b). We can also define the speaker identification loss with HEAT as

$$\mathcal{L}_{\text{sid}}(X, S^1, S^2) = \mathcal{L}_{\text{hat}}(S^1, H_1) + \mathcal{L}_{\text{hat}}(S^2, H_2), \quad (11)$$

where S^1 is the speaker label of the first spoken speaker, and $\mathcal{L}_{\text{hat}}(S^1, H_1)$ refers to the loss function defined by the HAT network. Given the two loss functions, we can train the two networks jointly by interpolating the two loss functions as

$$\mathcal{L} = \mathcal{L}_{\text{asr}}(X, Y^1, Y^2) + \lambda \mathcal{L}_{\text{sid}}(X, S^1, S^2), \quad (12)$$

where λ is the interpolation weight. Note that the Unmixing module is shared both for the ASR and SID networks.

6. Experiments and Results

6.1. Dataset

Our experiments were performed on the LibriSpeechMix[1] dataset [22], which is the simulated overlapped audio derived from the LibriSpeech corpus [26]. We used the same protocol to simulate the training and evaluation data as in [22], and we focus on the 2-speaker case. To generate the simulated training data, for each utterance in the original LibriSpeech train_960 set, we randomly picked another utterance from a *different* speaker, and mix the latter with the previous one with a random delay sampled from 0.5 seconds and the length of the first utterance. For each overlapped audio segments, the size of the speaker inventory varies within the range $K \in [2, 8]$ for training including the 2 speakers presented in the overlapped audio, while for evaluation, we fixed $K = 8$ and the minimum delay is 0 when mixing the two utterances. The dimension of the d-vectors in our experiments is 128. To generate the d-vectors in D, we randomly select 10 utterances for each speaker in Librispeech, and fed them into a pre-trained 17-layer convolutional network trained on VoxCelab [35, 36]. The model structure of the d-vector extractor is the modified version of the one in [37]. We used the same approach to generate the evaluation dataset.

6.2. Experimental Setup

In our experiments, we used the 80-dimensional log-mel filter-banks (FBANKs) as features, which were sampled the features at the 10 millisecond frame rate. We then spliced the features by a context window of 3 and finally downsampled super-frames by a factor of 3. Before feeding the features into the CNN blocks in the Unmixing module, we reshaped the input tensor to have 3 input channels, and consequently, the feature dimension became 80. For tokenization, we used 4,000 word-pieces as output tokens for RNN-T, which are generated by byte-pair encoding (BPE) [38]. We used a 4-layer 2-dimensional CNN network for both encoder and mask component in the Unmixing module. We used a 6-layer LSTM as the audio encoder and 2-layer LSTM as the label encoder for the ASR RNN-T network, and a 1-layer LSTM for the audio encoder in the SID HAT network. The number of hidden units was set to be 1024 for all LSTM layers unless specified otherwise. The number of model parameters without the SID network is around 81 million, while the number of parameters in the SID network is around 9 million. We set the dropout ratio as 0.2 for LSTM layers, and applied one layer of time-reduction to reduce the input sequence length by the factor of 2 [39, 9, 40]. We also applied speed perturbation for data augmentation by creating two additional versions of the acoustic features with the speed ratios as 0.9 and 1.1. All our experiments are targeting the 2-speaker scenario.

We report the word error rate (WER) results by considering all the possible label permutations as in [23, 22, 24, 25]. To measure the SID accuracy, we follow the similar protocol as we

---

[1] https://github.com/NaoyukiKanda/LibriSpeechMix

Table 1: *Multi-talker speech recognition results.*

System	Size	Feature	Streaming	WER
PIT-S2S [29]	160.7M	FBANK	✗	10.3
SOT-S2S-MBR [41]	135.6M	FBANK	✗	8.3†
SURT [24]	81M	STFT	✓	10.8
PIT-MS-RNN-T [25]	80.9M	FBANK	✓	10.2
SURT (this work)	81M	FBANK	✓	10.3
SURT (this work)	81M	FBANK	✗	7.2

† This model used speaker inventory as an auxiliary input.

Table 2: *Joint ASR and SID results.*

System	Feature	Training	SpecAug	WER	SER
B1	FBANK	Joint	✗	**10.1**	**6.7**
B2	FBANK	Stepwise	✗	10.3	8.7
B3	FBANK	Joint	✓	10.8	7.9
B4	FBANK	Stepwise	✓	10.5	8.2

Table 3: *Speaker identification latency.* t_e *indicates the time step that emits the first speaker label, and T is the length of the overlapped audio. The frame rate is 60 milliseconds.*

System	α	β	SER	t_e	t_e/T
B4	1.0	0	8.2	181.7	0.95
C1	0.8	0	9.5	123.5	0.71
C2	0.6	0	9.9	103.7	0.62
C3	0.4	0	10.2	75.3	0.47
C4	1.0	1	8.5	173.1	0.92
C5	0.8	1	9.3	88.1	0.53
C6	0.8	3	10.6	62.4	0.37
C7	0.8	5	10.1	96.9	0.58
B1	1.0	0	6.7	153.8	0.85
D1	0.8	0	7.1	55.4	0.34
D2	0.6	0	7.6	52.7	0.32
D3	0.8	1	7.9	54.8	0.33

calculate the WERs. Suppose the SID outputs of the two feature streams (H_1, H_2) are (Q^1, Q^2), we compute the speaker error rate (SER) as follows

$$\text{SER} = \frac{1}{2N} \sum_{n=1}^{N} \min \Big(\mathcal{E}\left([Q_n^1, Q_n^2], [S_n^1, S_n^2]\right),$$
$$\mathcal{E}\left([Q_n^2, Q_n^1], [S_n^1, S_n^2]\right) \Big), \quad (13)$$

where n is the utterance index; N is the total number of utterances; $\mathcal{E}(\cdot)$ refers to the edit distance function, and $[S_n^1, S_n^2]$ are the reference speaker labels of the $n-$th utterance.

6.3. Multi-talker ASR Results

Table 1 shows the multi-talker speech recognition results on the `test-clean` evaluation set with the SURT model shown in Fig. 2-a). Previously, we achieved 10.8% WER with Short-time Fourier Transform (STFT) features in [24], and with FBANK features, we obtained slightly lower WER of 10.3%. Our streaming results compares favorably with the offline sequence-to-sequence (S2S) based models [29, 41] and the RNN-T-based streaming model proposed in [25]. We also evaluated the accuracy of an offline SURT model with bi-directional LSTMs (BLSTMs). For a fair comparison, we did increase the BLSTM model size by reducing the hidden size to be 600. The offline SURT model achieved 7.2% WER, demonstrating the large gap between the offline and streaming SURT models.

6.4. Joint ASR and SID Results

Table 2 shows the results of SURIT with or without joint training of the ASR and SID modules. For joint ASR and SID training, the SID loss is much smaller than ASR loss, and similarly for the ℓ_2 norms of the gradients. To compensate for the difference, we set $\lambda = 10$ and the results are given in Table 2. We also evaluated the stepwise training approach, in which we trained the model with the ASR loss first, and then froze the Unmix module before training the model with SID loss only. The stepwise training approach is not as effective as the joint training approach in terms of accuracy as Table 2 shows. The reason may be that the Unmixing module was not trained with SID loss, and thus the features from the Unmixing module are not very discriminative for speakers. However, this approach has the advantage that it is suitable for training the model with transcription-only or speaker label only data. We also evaluated adding the SpecAugment [42] as an additional regularization approach on top of dropout and speed perturbation. For SID, this approach is effective for stepwise training, reducing

the SER from 8.7% to 8.2%. However, for the joint training approach, additional regularization did not yield improvement, since multi-task learning itself is a kind of regularization.

6.5. SID Latency

Finally, we evaluated the SID latency by measuring the average time step t_e that HAT emitted the speaker label of the first utterance in an overlapped audio during inference. From the baseline system B4 in Table 3, we observe that HAT delays the output of the speaker label until almost at the end of the audio segment. While this is not unexpected, practical applications usually favor low SID latency. To this end, we evaluated the two approaches discussed in Section 4.2. To rule out the effect of the ASR module, we firstly used B4 in Table 2 as the baseline system, and tuned the two hyper-parameters α and β while freezing the Unmixing module. From the results in Table 3, we observed that a smaller α resulted in a significant latency reduction, albeit with the increase of SER.

We then evaluated the latency penalty approach as Eq. (8). We set $t_{\text{buffer}} = 3$ for all our experiments. However, this approach itself did not work well in our experiments (C4 vs. B4), as this approach further reduces the gradient of speaker labels during training, and consequently it increases the probability to emit $\langle bs \rangle$ during inference. To mitigate this effect, we scaled down the gradient of $\langle bs \rangle$ with a smaller α during training, and we can achieve further latency reduction (C5 vs. C1)). Finally, we performed joint training as Eq. (12) with different values of α and β. Again, we set $\lambda = 10$ for these experiments. We observed a similar trend (D1-3 vs. B1), though the latency reductions were much more significant since the Unmixing module was also updated with the SID loss during joint training.

7. Conclusions

In this paper, we aim at a simpler, end-to-end framework for rich transcription of meetings and conversational speech. In particular, we focused on joint ASR and SID from multi-talker audio in a streaming fashion. We proposed the SURIT framework, with the Umixing module as the feature extractor, while ASR and SID are both based on RNN-T and its variant. We proposed a transducer-based SID model, and evaluated two methods to reduce the SID latency. We studied the joint training approach for streaming ASR and SID, and demonstrated the effectiveness of the new framework with strong results. Our future work will focus on reducing the SID latency further as well as a large scale study of the proposed model.

8. References

[1] N. Morgan *et al.*, "Meetings about meetings: research at ICSI on speech in multiparty conversations," in *ICASSP*. IEEE, 2003.

[2] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke *et al.*, "The ICSI meeting corpus," in *Proc. ICASSP*. IEEE, 2003.

[3] S. Renals, T. Hain, and H. Bourlard, "Recognition and understanding of meetings the AMI and AMIDA projects," in *Proc. ASRU*. IEEE, 2007, pp. 238–247.

[4] A. Stolcke, X. Anguera, K. Boakye, Ö. Çetin *et al.*, "The SRI-ICSI Spring 2007 meeting and lecture recognition system," in *Multimodal Technologies for Perception of Humans*. Springer, 2007, pp. 450–463.

[5] T. Hain, L. Burget, J. Dines, P. N. Garner, F. Grézl *et al.*, "Transcribing meetings with the AMIDA systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 486–498, 2011.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[7] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*. Ieee, 2013, pp. 6645–6649.

[8] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. NeurIPS*, 2015, pp. 577–585.

[9] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 4960–4964.

[10] L. Lu, X. Zhang, and S. Renals, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 5060–5064.

[11] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*. IEEE, 2019, pp. 6381–6385.

[12] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. ICASSP*. IEEE, 2016, pp. 5115–5119.

[13] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Proc. SLT*. IEEE, 2016, pp. 165–170.

[14] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. ICASSP*. IEEE, 2016, pp. 31–35.

[15] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. ICASSP*. IEEE, 2017, pp. 241–245.

[16] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker diarization using deep neural network embeddings," in *Proc. ICASSP*. IEEE, 2017, pp. 4930–4934.

[17] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully supervised speaker diarization," in *Proc. ICASSP*. IEEE, 2019, pp. 6301–6305.

[18] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with permutation-free objectives," in *Proc. INTERSPEECH*, 2019, pp. 4300–4304.

[19] T. Yoshioka, H. Erdogan, Z. Chen, X. Xiao, and F. Alleva, "Recognizing overlapped speech in meetings: A multichannel separation approach using neural networks," in *Proc. INTERSPEECH*, 2018, pp. 3038–3042.

[20] S. Settle, J. Le Roux, T. Hori, S. Watanabe, and J. R. Hershey, "End-to-end multi-speaker speech recognition," in *Proc. CASSP*. IEEE, 2018, pp. 4819–4823.

[21] X. Chang, Y. Qian, K. Yu, and S. Watanabe, "End-to-end monaural multi-speaker ASR system without pretraining," in *Proc. ICASSP*. IEEE, 2019, pp. 6256–6260.

[22] N. Kanda, Y. Gaur, X. Wang, Z. Meng, and T. Yoshioka, "Serialized output training for end-to-end overlapped speech recognition," in *Proc. INTERSPEECH*, 2020, pp. 2797–2801.

[23] A. Tripathi, H. Lu, and H. Sak, "End-to-end multi-talker overlapping speech recognition," in *Proc. ICASSP*. IEEE, 2020, pp. 6129–6133.

[24] L. Lu, N. Kanda, J. Li, and Y. Gong, "Streaming end-to-end multi-talker speech recognition," *IEEE Signal Processing Letters*, 2021.

[25] I. Sklyar, A. Piunova, and Y. Liu, "Streaming Multi-speaker ASR with RNN-T," in *Proc. ICASSP*, 2021.

[26] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*. IEEE, 2015, pp. 5206–5210.

[27] L. El Shafey, H. Soltau, and I. Shafran, "Joint speech recognition and speaker diarization via sequence transduction," in *Proc. INTERSPEECH*, 2019, pp. 396–400.

[28] H. H. Mao, S. Li, J. McAuley, and G. W. Cottrell, "Speech recognition and multi-speaker diarization of long conversations," in *Proc. INTERSPEECH*, 2020, pp. 691–695.

[29] N. Kanda, Y. Gaur *et al.*, "Joint speaker counting, speech recognition, and speaker identification for overlapped speech of any number of speakers," in *Proc. INTERSPEECH*, 2020.

[30] A. Waters, N. Gaur, P. Haghani, P. Moreno, and Z. Qu, "Leveraging language id in multilingual end-to-end speech recognition," in *Proc. ASRU*. IEEE, 2019, pp. 928–935.

[31] E. Variani, X. Lei, E. McDermott *et al.*, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP*. IEEE, 2014, pp. 4052–4056.

[32] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive transducer (HAT)," in *Proc. ICASSP*. IEEE, 2020, pp. 6139–6143.

[33] J. Yu, C.-C. Chiu, B. Li, S.-y. Chang, T. N. Sainath, Y. He, A. Narayanan, W. Han, A. Gulati, Y. Wu *et al.*, "FastEmit: Low-latency streaming ASR with sequence-level emission regularization," *arXiv preprint arXiv:2010.11148*, 2020.

[34] B. Li, S.-y. Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards fast and accurate streaming end-to-end ASR," in *Proc. ICASSP*. IEEE, 2020, pp. 6069–6073.

[35] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[36] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[37] T. Zhou, Y. Zhao, J. Li, Y. Gong, and J. Wu, "CNN with phonetic attention for text-independent speaker verification," in *Proc. ASRU*. IEEE, 2019, pp. 718–725.

[38] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[39] A. Graves, "Hierarchical subsampling networks," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, pp. 109–131.

[40] L. Lu, L. Kong, C. Dyer, N. A. Smith, and S. Renals, "Segmental recurrent neural networks for end-to-end speech recognition," in *Proc. INTERSPEECH*, 2016.

[41] N. Kanda, Z. Meng, L. Lu, Y. Gaur, X. Wang, Z. Chen, and T. Yoshioka, "Minimum Bayes risk training for end-to-end speaker-attributed ASR," in *Proc. ICASSP*, 2021.

[42] D. S. Park *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. INTERSPEECH*, 2019, pp. 2613–2617.