End-to-end speaker segmentation for overlap-aware resegmentation

Hervé Bredin, Antoine Laurent

End-to-end speaker segmentation for overlap-aware resegmentation

Hervé Bredin[1] *& Antoine Laurent*[2]

[1]IRIT, Université de Toulouse, CNRS, Toulouse, France
[2]LIUM , Université du Mans, France

herve.bredin@irit.fr, antoine.laurent@univ-lemans.fr

Abstract

Speaker segmentation consists in partitioning a conversation between one or more speakers into speaker turns. Usually addressed as the late combination of three sub-tasks (voice activity detection, speaker change detection, and overlapped speech detection), we propose to train an end-to-end segmentation model that does it directly. Inspired by the original end-to-end neural speaker diarization approach (EEND), the task is modeled as a multi-label classification problem using permutation-invariant training. The main difference is that our model operates on short audio chunks (5 seconds) but at a much higher temporal resolution (every 16ms). Experiments on multiple speaker diarization datasets conclude that our model can be used with great success on both voice activity detection and overlapped speech detection. Our proposed model can also be used as a post-processing step, to detect and correctly assign overlapped speech regions. Relative diarization error rate improvement over the best considered baseline (VBx) reaches 17% on AMI, 13% on DIHARD 3, and 13% on VoxConverse.

Index Terms: speaker diarization, speaker segmentation, voice activity detection, overlapped speech detection, resegmentation.

1. Introduction

The speech processing community relies on term *segmentation* to describe a multitude of tasks: from classifying the audio signal into three classes {*speech*, *music*, *other*}, to detecting breath groups, localizing word boundaries, or even partitioning speech regions into phonetic units. On this coarse-to-fine time scale, speaker segmentation lies somewhere between {*speech*, *non-speech*} classification and breath groups detection. It consists in partitioning speech regions into smaller chunks containing speech from a single speaker. It has been addressed in the past as the combination of several sub-tasks. First, voice activity detection (VAD) removes any region that does not contain speech. Then, speaker change detection (SCD) partitions remaining speech regions into speaker turns, by looking for time instants where a change of speaker occurs [1]. From a distance, this definition of speaker segmentation may appear clear and unambiguous. However, when looking more carefully, lots of complex phenomena happen in real-life spontaneous conversations – overlapped speech, interruptions, and backchannels being the most prominent ones. Therefore, researchers have started working on the overlapped speech detection (OSD) task as well [2, 3, 4].

This work was granted access to the HPC resources of IDRIS under the allocation AD011012177 made by GENCI, and was partly funded by the French National Research Agency (ANR) through the PLUMCOT (ANR-16-CE92-0025) and the GEM (ANR-19-CE38-0012) projects.

End-to-end speaker segmentation. Instead of addressing voice activity detection, speaker change detection, and overlapped speech detection as three different tasks, our first contribution is to train a unique end-to-end speaker segmentation model whose output encompasses the aforementioned sub-tasks. This model is directly inspired by recent advances in end-to-end speaker diarization and, in particular, the growing *End-to-End Neural Diarization* (EEND) family of approaches developed by *Hitachi* [5, 6, 7]. The proposed segmentation model is better than (or at least on par with) several voice activity detection baselines, and sets a new state of the art for overlapped speech detection on all three considered datasets: AMI Mix-Headset [8], DIHARD 3 [9, 10], and VoxConverse [11]. We did not run speaker change detection experiments.

Overlap-aware resegmentation. Our second contribution relates to the problem of assigning detected overlapped speech regions to the right speakers. While a few attempts have been made in the past [4, 12], it remains a very difficult problem for which a simple heuristic baseline has yet to be beaten [13]. We show, through extensive experimentation, that our segmentation model consistently beats this heuristic when turned into an overlap-aware resegmentation module – setting a new state of the art on the AMI dataset when combined with the *VBx* approach.

Reproducible research. Last but not least, our final contribution consists in sharing the pretrained model and integrating it into *pyannote* open-source library for reproducibility purposes: `huggingface.co/pyannote/segmentation`. Expected outputs of the proposed approaches (VAD, OSD, and resegmentation) are also available at this address in `RTTM` format to facilitate future comparison.

2. End-to-end speaker segmentation

Like in the original EEND approach [5], the task is modeled as a multi-label classification problem using permutation-invariant training. As depicted in Figure 1, the main difference is that our model operates on short audio chunks (5 seconds) but at a much higher temporal resolution (around every 16ms). Processing short audio chunks also implies that the number of speakers is much smaller and less variable than with the original EEND approach (dealing with whole conversations) – making the problem easier to address. For instance, we found that 99% of every possible 5s chunks in the training set (later defined in Section 3) contained less than $K_{\text{max}} = 4$ speakers.

2.1. Permutation-invariant training

Given an audio chunk $\mathbf{X}$, its reference segmentation can be encoded into a sequence of K_{max}-dimensional binary frames

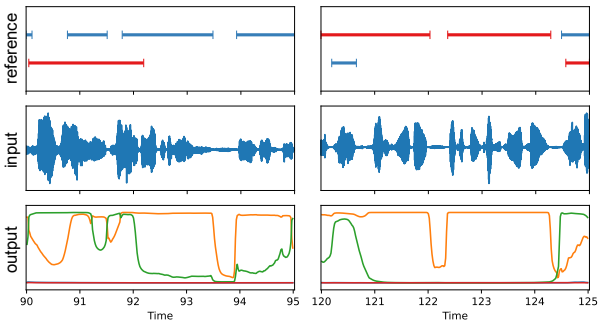

Figure 1: *Actual outputs of our model on two 5s excerpts from the same conversation between two speakers (source: file* `DH_EVAL_0035.flac` *in DIHARD3 dataset). Top row shows the reference annotation. Middle row is the audio chunk ingested by the model. Bottom row depicts the raw speaker activations, as returned by the model. Thanks to permutation-invariant training, notice how the blue speaker corresponds to the orange activation on the left and to the green one on the right.*

$\mathbf{y} = \{\mathbf{y_1}, \dots, \mathbf{y_T}\}$ where $\mathbf{y_t} \in \{0,1\}^{K_{\max}}$ and $y_t^k = 1$ if speaker k is active at frame t and $y_t^k = 0$ otherwise. We may arbitrarily sort speakers by chronological order of their first activity but any permutation of the $K_{\max}$ dimensions is a valid representation of the reference segmentation. Therefore, the binary cross entropy loss function $\mathcal{L}_{\text{BCE}}$ (classically used for such multi-label classification problems) has to be turned into a permutation-invariant loss function $\mathcal{L}$ by running over all possible permutations $\text{perm}(\mathbf{y})$ of $\mathbf{y}$ over its $K_{\max}$ dimensions:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \min_{\text{perm}(\mathbf{y})} \mathcal{L}_{\text{BCE}}(\text{perm}(\mathbf{y}), \hat{\mathbf{y}}) \quad (1)$$

with $\hat{\mathbf{y}} = f(\mathbf{X})$ where f is our segmentation model whose architecture is described later in the paper. In practice, for efficiency, we first compute the $K_{\max} \times K_{\max}$ binary cross entropy losses between all pairs of $\mathbf{y}$ and $\hat{\mathbf{y}}$ dimensions, and rely on the Hungarian algorithm to find the permutation that minimizes the overall binary cross entropy loss.

2.2. On-the-fly data augmentation

For training, 5s audio chunks (and their reference segmentation) are cropped randomly from the training set. To increase variability even more, we rely on on-the-fly random data augmentation. The first type of augmentation is additive background noise with random signal-to-noise ratio. Inspired by our previous work on overlapped speech detection [4], the second type of augmentation consists in artificially increasing the amount of overlapping speech. To do that, we sum two random 5s audio chunks with random signal-to-signal ratio (and merge their reference segmentation accordingly). Resulting chunks whose number of speakers is higher than $K_{\max}$ are not used for training.

2.3. Segmentation

Once trained, the model can be used for segmentation purposes or any sub-tasks by a simple post-processing of its output speaker activations:

- for **segmentation** or **speaker change detection**, a single $\theta = 0.5$ binarization threshold already gives decent results, but one can get even better performance by using a slightly more advanced post-processing borrowed from [14] and summarized in Figure 2.
- for **voice activity detection**, we start by computing the maximum activation over the $K_{\max}$ speakers:

$$\hat{y}_t^{\text{VAD}} = \max_k \hat{y}_t^k \quad (2)$$

and, then only, apply the aforementioned post-processing on resulting mono-dimensional $\hat{\mathbf{y}}^{\text{VAD}}$.
- for **overlapped speech detection**, since at least two speakers need to be active simultaneously to indicate overlapping speech, we compute the second highest (denoted $\max_{2\text{nd}}$) activation:

$$\hat{y}_t^{\text{OSD}} = \max_{k,\,2\text{nd}} \hat{y}_t^k \quad (3)$$

and post-process the resulting mono-dimensional $\hat{\mathbf{y}}^{\text{OSD}}$ with the same approach.

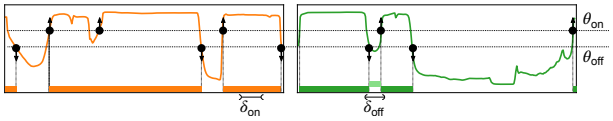

Figure 2: *To obtain the final binary segmentation, speaker activations are post-processed with θ_{on}/θ_{off} hysteresis thresholding, then filling gaps shorter than δ_{off} (light green region in right example) and finally removing active regions shorter than δ_{on} (does not happen in these examples).*

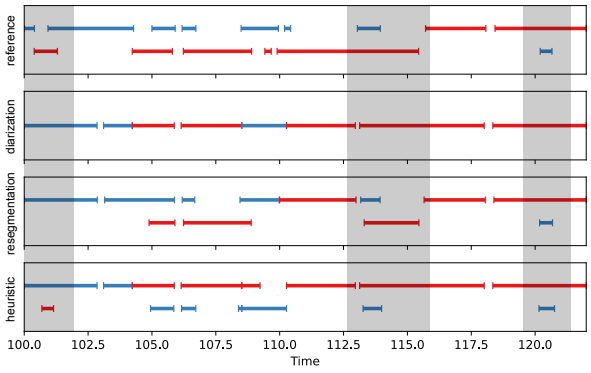

Figure 3: *Effect of the proposed overlap-aware resegmentation approach (third row) on the VBx diarization baseline (second row). We highlight three regions where the heuristic performs better ($t \approx 100s$), same ($t \approx 120s$), or worse ($t \approx 115s$) than the proposed approach (source: file* `DH_EVAL_0035.flac` *in DIHARD3 dataset).*

2.4. Overlap-aware resegmentation

While a growing number of diarization approaches do try and take overlapped speech into account [7], the most dependable ones (like the *VBx* approach [15] used in Figure 3) still assume internally that at most one speaker is active at any time. There is therefore a need for a post-processing step that assigns multiple speaker labels to overlapped speech regions [4, 17].

Given an existing speaker diarization output (with K speakers) encoded into a sequence of K-dimensional binary frames

y_t^{DIA}, we propose to use the segmentation model as a local, overlap-aware, resegmentation module. The segmentation model is applied on a 5s-long window sliding over the whole file. At each step, we find the permutation of the speaker activations $\hat{\mathbf{y}}$ that minimizes the binary cross entropy loss with respect to $\mathbf{y}^{\text{DIA}}$. Permutated sliding speaker activations are then aggregated over time and post-processed with the threshold-based approach introduced in Section 2.3.

3. Experiments

Datasets and partitions. We ran experiments and report results on three speaker diarization datasets, covering a wide range of domains:

DIHARD3 corpus [9, 10] does not provide a *training* set. Therefore, we split its *development* set into two parts: 192 files used as *training* set, and the remaining 62 files used as a smaller *development* set. The latter is simply referred to as *development* sets in the rest of the paper. When defining this split (shared at `huggingface.co/pyannote/segmentation`), we made sure that the 11 domains were equally distributed between both subsets. The *evaluation* set is kept unchanged.

VoxConverse does not provide a *training* set either [11]. Therefore, we also split its *development* set into two parts: first 144 files (`abjxc` to `qouur`, in alphabetical order) constitute the *training* set, leaving the remaining 72 files (`qppll` to `zyffh`) for the actual *development* set.

AMI provides an official {*training*, *development*, *evaluation*} partition of the Mix-Headset audio files [8]. While we kept the *development* and *evaluation* sets unchanged, we only used the first 10 minutes of each file of the *training* set, to end up with an actual *training* set similar in size (22 hours) to that of the DIHARD3 (25 hours) and VoxConverse (15 hours) *training* sets.

Experimental protocols. We trained a unique segmentation model using the *composite* training set (62 hours) made of the concatenation of all three *training* sets. The *composite* development set (24 hours) served as validation and was used to decrease the learning rate on plateau and eventually choose the best model checkpoint. At the end of this process, only one segmentation model is available (not one model per dataset) and used for all experiments.

However, detection thresholds (θ_{on}, θ_{off}, δ_{on}, and δ_{off}) were tuned specifically for each dataset using their own *development* set because the manual annotation guides differ from one dataset to another, especially regarding δ_{off} which controls whether to bridge small intra-speaker pauses. For the same reasons, detection thresholds were optimized specifically for each task addressed in the paper:

- voice activity detection thresholds are chosen to minimize the detection error rate (i.e. the sum of the false alarm and missed detection rates), with no forgiveness collar around speech turn boundaries;
- overlapped speech detection thresholds are chosen to maximize the detection F_1-score, with no forgiveness collar either;
- for resegmentation, detection thresholds are chosen to minimize the diarization error rate, without forgiveness collar but with overlapped speech regions. This is consistent with DIHARD3 evaluation plan [10] and AMI *Full* evaluation setup [15], but not with VoxConverse challenge rules that uses a 250ms collar [11].

All metrics were computed using *pyannote.metrics* [18] open source Python library.

Implementation details. Our segmentation model ingests 5s audio chunks with a sampling rate of 16kHz (*i.e.* sequences of 80000 samples). The input sequence is passed to *SincNet* convolutional layers using the original configuration [19] – except for the stride of the very first layer which is set to 10 (so that *SincNet* frames are extracted every 16ms). Four bidirectional Long Short-Term Memory (LSTM) recurrent layers (each with 128 units in both forward backward directions, and 50% dropout for the first three layers) are stacked on top of two additional fully connected layers (each with 128 units and leaky ReLU activation) which also operate at frame-level. A final fully connected classification layer with sigmoid activation outputs $K_{\max}$-dimensional speaker activations between 0 and 1 every 16ms. Overall, our model contains 1.5 million trainable parameters – most of which (1.4 million) comes from the recurrent layers.

As introduced in Section 2.2, 50% of the training samples are made out of the weighted sum of two chunks, with a signal-to-signal ratio sampled uniformly between 0 and 10dB. We also use additive background noise from the MUSAN dataset [20] with a signal-to-noise ratio sampled uniformly between 5 and 15dB.

We train the model with *Adam* optimizer with default *PyTorch* parameters and mini-batches of size 128. Learning rate is initialized at 10^{-3} and reduced by a factor of 2 every time its performance on the development set reaches a plateau. It took around 3 days using 4 V100 GPUs to reach peak performance. While we do share the pretrained model at `huggingface.co/pyannote/segmentation` for reproducing the results, the whole training process is also reproducible as everything has been integrated into version 2.0 of *pyannote.audio* open-source library [16].

Table 1: *Voice activity detection // FA = false alarm rate (%) / Miss. = missed detection rate (%)*

VAD	**AMI** [8, 15]			**DIHARD 3** [9]			**VoxConverse** [11]		
	FA	Miss.	FA+Miss.	FA	Miss.	FA+Miss.	FA	Miss.	FA+Miss.
silero_vad	9.4	1.7	11.0	17.0	4.0	21.0	3.0	1.1	4.2
dihard3 [9]	NA	NA	NA	4.0	4.2	8.2	NA	NA	NA
Landini et al. [12]	NA	NA	NA	NA	NA	NA	1.8	1.1	3.0
pyannote 1.1 [16]	6.5	1.7	8.2	4.1	3.8	7.9	4.5	0.3	4.8
Ours – *pyannote 2.0*	3.6	3.2	**6.8**	3.9	3.3	**7.3**	1.8	0.8	**2.5**

Table 2: *Overlapped speech detection // FA = false alarm rate (%) / Miss. = missed detection rate (%) / F_1 = F_1-score (%)*

OSD	AMI [8, 15]					DIHARD 3 [9]					VoxConverse [11]				
	FA	Miss.	Precision	Recall	F_1	FA	Miss.	Precision	Recall	F_1	FA	Miss.	Precision	Recall	F_1
Kunesova et al. [3]	NA	NA	71.5	46.1	56.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Landini et al. [12]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	10.4	71.8	73.0	28.2	40.7
pyannote 1.1 [16, 4]	51.1	12.1	63.2	87.9	73.5	48.2	45.2	53.2	54.8	54.0	130.4	17.7	38.7	82.3	52.6
Ours – *pyannote 2.0*	16.9	29.4	80.7	70.5	**75.3**	46.9	37.2	57.2	62.8	**59.9**	26.3	24.5	74.2	75.5	**74.8**

Table 3: *Resegmentation // FA = false alarm / Miss. = missed detection / Conf. = speaker confusion / DER = diarization error rate*

Baseline	Overlap-aware resegmentation	AMI [8, 15]				DIHARD 3 [9]				VoxConverse [11]			
		FA	Miss.	Conf.	DER	FA	Miss.	Conf.	DER	FA	Miss.	Conf.	DER
pyannote 1.1 [16]	–	5.0	16.2	8.5	29.7	3.4	13.2	12.6	29.2	2.0	10.1	9.5	21.5
	Heuristic [13] w/ **our OSD**	6.9	7.9	10.9	**25.7**	6.3	8.9	12.8	28.1	2.8	7.3	10.1	20.3
	Ours – *pyannote 2.0*	4.0	13.0	9.1	26.1	5.1	9.8	**10.3**	25.2	2.4	3.1	9.8	**15.4**
dihard3 [9]	–	NA	NA	NA	NA	3.6	13.3	8.4	25.4	NA	NA	NA	NA
	Heuristic [13] w/ **our OSD**	NA	NA	NA	NA	6.8	8.7	8.8	24.3	NA	NA	NA	NA
	Ours – *pyannote 2.0*	NA	NA	NA	NA	4.6	10.2	7.5	**22.2**	NA	NA	NA	NA
VBx [15] w/ **our VAD**	–	3.1	17.2	3.8	24.1	3.6	12.5	6.2	22.3	1.7	5.1	1.4	8.3
	Heuristic [13] w/ **our OSD**	5.1	8.7	6.1	**19.9**	7.0	7.8	6.4	21.2	2.7	2.1	2.0	**6.8**
	Ours – *pyannote 2.0*	4.3	10.9	4.7	**19.9**	4.7	9.7	4.9	**19.3**	2.7	2.6	1.8	7.1
Oracle	**Ours** – *pyannote 2.0*	4.7	10.0	1.4	16.1	4.6	9.8	1.8	16.2	2.6	2.5	0.6	5.7

4. Results and discussions

Voice activity detection. Table 1 compares the performance of the proposed voice activity detection approach with the official *dihard3* baseline [9], *Landini*'s submission to VoxConverse challenge [12], and *pyannote 1.1* VAD models [16]. The main conclusion is that, despite it being trained for segmentation, our model is better than other models trained specifically for voice activity detection. Note, however, that one should not draw hasty conclusions regarding the performance of *silero_vad* model [21] as it is an off-the-shelf model which was not trained specifically for these datasets.

Overlapped speech detection. Finding good and reproducible baselines for the overlapped speech detection task proved to be a difficult task. We thank *Kunesova et al.* [3] and *Landini et al.* [12] for sharing the output of their detection pipelines. Results are reported in Table 2 that shows that, like for voice activity detection, our segmentation model can be used successfully for overlapped speech detection, even though it was not initially trained for this particular task. It outperforms *pyannote 1.1* overlapped speech detection which we believe was the previous state of the art [4].

Overlap-aware resegmentation. While our segmentation model was found to be useful for both voice activity detection and overlapped speech detection, post-processing the output of existing speaker diarization pipelines is where it really shines. Table 3 summarizes the resegmentation experiments performed on top of three of them, ranked from worst to best baseline performance: *pyannote 1.1* pretrained pipelines [16], *dihard3* official baseline [9], and BUT's *VBx* approach [15]. The (admittedly wrong) criterion used for selecting those baselines was their ease of use and reproducibility. Because results reported in [15] for *VBx* baseline rely on oracle voice activity detection and the shared code base does not provide an official voice activity detection implementation, we used our own (marked as **Ours** in Table 1) and applied *VBx* on top of it. Our proposed resegmentation approach consistently improves the output of all baselines on all datasets. Relative diarization error rate improvement over the best baseline (*VBx*) reaches 17% on AMI, 13% on DIHARD, and 13% on VoxConverse.

For comparison purposes, we also implemented a heuristic that consists in assigning detected overlapped speech regions to the two nearest speakers in time [13]. Despite its simplicity, this heuristic happens to be a strong baseline, very difficult to beat in practice [12]. Yet, our proposed resegmentation approach outperforms it for all but two experimental conditions (for which the heuristic is better only by a small margin). A closer look at the speaker confusion error rates shows that our approach is significantly better at identifying overlapping speakers. This is confirmed by the low speaker confusion error rates obtained when we apply it on top of an oracle diarization (with $\mathbf{y}^{\text{DIA}} = \mathbf{y}$): only 1.4%, 1.8%, and 0.6% of speech are reassigned incorrectly on AMI, DIHARD and VoxConverse respectively. Figure 3 provides a qualitative sneak peak at their respective behavior on a short 20 seconds excerpt. In particular, it appears that the two (heuristic and proposed) approaches do behave differently and could complement each other.

5. Conclusions

The overall best pipeline reported in this paper is the combination of our voice activity detection, off-the-shelf VBx clustering, and our overlap-aware resegmentation approach, reaching DER $= 19.9\%$ on AMI Mix-Headset using the *full* evaluation setup introduced in [15], DER $= 19.3\%$ on DIHARD 3 evaluation set (*full* condition, 2.6% behind the winning submission), and DER $= 7.1\%$ (or DER $= 3.4\%$ with a 250ms forgiveness collar) on VoxConverse *development* set.

Even with a forgiveness collar, missed detection and false alarms are the main source of errors (twice as high as speaker confusion) for all three datasets – indicating that, despite progress, overlapped speech detection remains an unsolved (and sometimes ill-defined) problem.

6. References

[1] R. Yin, H. Bredin, and C. Barras, "Speaker Change Detection in Broadcast TV Using Bidirectional Long Short-Term Memory Networks," in *Proc. Interspeech 2017*, 2017.

[2] D. Charlet, C. Barras, and J. Liénard, "Impact of overlapping speech detection on speaker diarization for broadcast news and debates," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 7707–7711.

[3] M. Kunešová, M. Hrúz, Z. Zajíc, and V. Radová, "Detection of overlapping speech for the purposes of speaker diarization," in *Speech and Computer*, A. A. Salah, A. Karpov, and R. Potapova, Eds. Cham: Springer International Publishing, 2019, pp. 247–257.

[4] L. Bullock, H. Bredin, and L. P. Garcia-Perera, "Overlap-aware diarization: Resegmentation using neural end-to-end overlapped speech detection," in *Proc. ICASSP 2020*, 2020.

[5] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, "End-to-End Neural Speaker Diarization with Permutation-free Objectives," in *Interspeech*, 2019, pp. 4300–4304.

[6] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 296–303.

[7] Y. Takashima, Y. Fujita, S. Watanabe, S. Horiguchi, P. García, and K. Nagamatsu, "End-to-end speaker diarization conditioned on speech activity and overlap detection," in *2021 IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 849–856.

[8] J. Carletta, "Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus," *Language Resources and Evaluation*, vol. 41, no. 2, 2007.

[9] N. Ryant, P. Singh, V. Krishnamohan, R. Varma, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "The Third DIHARD Diarization Challenge," *arXiv preprint arXiv:2012.01477*, 2020.

[10] N. Ryant, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "Third DIHARD Challenge Evaluation Plan," *arXiv preprint arXiv:2006.05815*, 2020.

[11] J. S. Chung, J. Huh, A. Nagrani, T. Afouras, and A. Zisserman, "Spot the Conversation: Speaker Diarisation in the Wild," in *Proc. Interspeech 2020*, 2020, pp. 299–303. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-2337

[12] F. Landini, O. Glembek, P. Matějka, J. Rohdin, L. Burget, M. Diez, and A. Silnova, "Analysis of the BUT Diarization System for VoxConverse Challenge," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[13] S. Otterson and M. Ostendorf, "Efficient use of overlap information in speaker diarization," in *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*. IEEE, 2007, pp. 683–686.

[14] G. Gelly and J.-L. Gauvain, "Optimization of RNN-Based Speech Activity Detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 3, pp. 646–656, March 2018.

[15] F. Landini, J. Profant, M. Diez, and L. Burget, "Bayesian hmm clustering of x-vector sequences (vbx) in speaker diarization: theory, implementation and analysis on standard tasks," 2020.

[16] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "pyannote.audio: neural building blocks for speaker diarization," in *Proc. ICASSP 2020*, 2020.

[17] S. Horiguchi, P. Garcia, Y. Fujita, S. Watanabe, and K. Nagamatsu, "End-to-end speaker diarization as post-processing," 2020.

[18] H. Bredin, "pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems," in *Proc. Interspeech 2017*, Stockholm, Sweden, August 2017. [Online]. Available: http://pyannote.github.io/pyannote-metrics

[19] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *Proc. SLT 2018*, 2018.

[20] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015.

[21] Silero Team, "Silero VAD: pre-trained enterprise-grade Voice Activity Detector (VAD), Number Detector and Language Classifier," https://github.com/snakers4/silero-vad, 2021.