# Multi-latency look-ahead for streaming speaker segmentation

Bilal Rahou, Hervé Bredin

Interspeech 2024
1-5 September 2024, Kos, Greece

# Multi-latency look-ahead for streaming speaker segmentation

*Bilal Rahou, Hervé Bredin*

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France
herve.bredin@irit.fr

## Abstract

We address the task of streaming speaker diarization and propose several contributions to achieve a better trade-off between latency and accuracy. First, computational latency is reduced to its bare minimum by switching to a causal frame-wise speaker segmentation architecture. Then, a multi-latency look-ahead mechanism is used during training to support adaptive latency during inference at no additional computational cost. Finally, we detail the method used during inference to achieve the final frame-wise segmentation. We evaluate the impact of these contributions on the AMI meeting dataset with a focus on the speaker segmentation step, seen through the prism of voice activity detection, overlapped speech detection and speaker change detection.

**Index Terms**: speaker diarization, speaker segmentation, low latency, lookahead

## 1. Introduction

Speaker diarization is the task of partitioning the recording of a multi-speaker conversation into segments based on speaker identity. While it has been addressed as a series of simpler subtasks in the past [1], recent years have witnessed a strong interest in monolithic approaches where a single neural network is trained to ingest the audio and output diarization results directly [2, 3, 4]. Because multi-stage and end-to-end approaches both have their pros and cons, recent works have investigated their combination into hybrid approaches [5, 6], reaching state-of-the-art performance on several benchmarks [7].

More precisely, hybrid approaches usually consist of two main steps: end-to-end speaker diarization of a short (up to a minute) window, sliding over the whole recording; followed by a clustering step (based on speaker embedding) that stitches the diarization results of the short windows into a coherent whole. Both steps assume that their whole input (the window for the first step, or the recording for the second) is available at once, and therefore are not suitable for processing live recordings in a streaming fashion out of the box.

Addressing speaker diarization in a streaming setting brings its own set of challenges, mostly related to finding the best compromise between latency and accuracy (the lower the expected latency, the lower the accuracy) [8, 9, 10]. For instance, *Coria et al.* propose to switch from batch to incremental clustering but still expect 5s windows to be available at once [11]. Practically, the latency of this approach is the sum of two terms: the algorithmic latency which is controlled by the stride of the sliding window, and the computational latency which is the time needed to process a window. While the algorithmic latency can be reduced by using a smaller stride, the computational latency is more difficult to reduce as it depends on both the hardware and the window duration. In Section 2, we build on top of [11, 7] and focus on the architectural changes needed to turn the offline speaker segmentation model (whose computational latency depends on the duration of the window) into a causal frame-wise model with (very) low computational latency, and study their impact on the accuracy of the system.

Keeping computational latency constant, we then extend the speaker segmentation model with the built-in ability to control the trade-off between accuracy and algorithmic latency, through the use of look-ahead mechanism that greatly facilitates the real-time detection of speaker changes in particular. Look-ahead mechanisms have been implemented in very different ways in the literature: from selecting hypothetical new words from a proposal distribution [12] to changing the whole structure of the model to adapt to it [13]. As discussed in Section 3, our proposed approach differs from related diarization work such as [14] (that appends a 1-dimensional convolutional layer at the end of the encoder of their self-attention system) because it does not need any architectural changes of the segmentation model, everything happening at training time. We also further extend the model to support adaptive latency at inference time, at no extra computational cost.

Finally, we evaluate the impact of these contributions on the AMI meeting dataset in Section 6 with a focus on the speaker segmentation step, seen through the prism of voice activity detection, overlapped speech detection, and speaker change detection.

## 2. From offline to causal frame-wise model

Table 1: *Impact of architectural changes in the speaker segmentation model. Average diarization error rates computed on 5s chunks sampled from the AMI test set. A is the offline segmentation model and D is our proposed frame-wise online model.*

| | LSTM direction | Instance norm. | Gain augm. | 5s chunk DER% |
|---|---|---|---|---|
| A | $\longleftrightarrow$ | ✓ | | 17.3 |
| B | $\longrightarrow$ | ✓ | | 20.3 |
| C | $\longrightarrow$ | | | 22.2 |
| D | $\longrightarrow$ | | ✓ | 21.1 |

We base our segmentation model on the architecture and loss of [7], which consists of SincNet convolutional layers [15] followed by bi-directional LSTMs. To switch from this offline block-wise segmentation model to a causal frame-wise one, two changes in the architecture are needed, summarized in Table 1: bi-directional LSTMs must be replaced by uni-directional ones


10.21437/Interspeech.2024-923

(from A to B), and the instance normalization layers within the SincNet architecture must be removed (from B to C) as they cannot be used in an online fashion, as they take the whole waveform into account when calculating their internal mean and standard deviation. This enables us to achieve a causal frame-wise model with a computational latency of around 100$\mu$s. However, it also significantly worsens the accuracy of the segmentation. To compensate for the lack of normalization, we use training time gain augmentation (from C to D), which makes the model more robust to variability in the amplitude of the input signal, practically addressing the same issue as normalization. Practically, a random gain sampled from the interval [-30, 30] is added to each input.

Looking at Table 1 in more details, we can observe that gain augmentation allows to close half of the gap between a model with normalization layers and a model without. Yet, there is still a significant difference between the offline model (DER=17.3%) and the frame-wise one (DER=21.1%), mostly caused by the change in the LSTM direction.

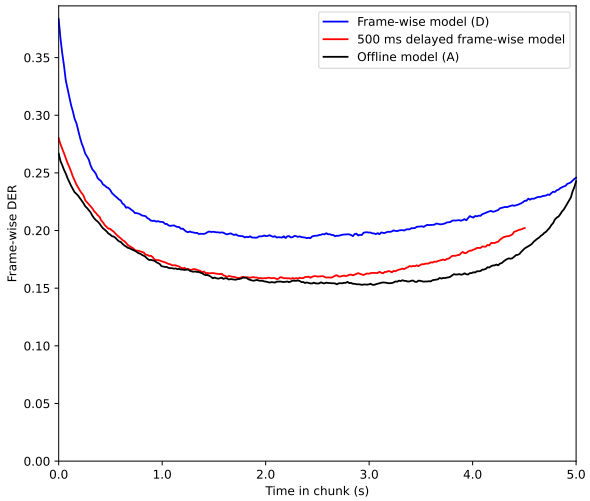


Figure 1: *Frame-wise DER within a 5s chunk, as a function of the frame position within the chunk. Blue curve is the model D, black curve is the original offline model A, and red curve is our proposed streaming model with a 500ms latency look-ahead.*

Figure 1 dives deeper into the impact of the architectural changes on the behavior and accuracy of the model within a 5s chunk. Focusing on the black (model A) and blue (model D) curves for now, we notice that the bidirectional offline model A is, on average, more accurate in the middle of 5s chunks than on both sides for which its performance degrades symmetrically. As expected, its streaming counterpart (model D) is significantly worse at the beginning of each chunk – which can be explained by the total absence of context at the beginning of the chunk.

## 3. Multi-latency training with look-ahead

The segmentation model ingests a 5s audio waveform, processes it into a sequence of frames, and operates on each frame in a causal manner. We propose to rely on a look-ahead mechanism to give the model information on several frames ahead of the current one. The model will simply *"wait"* for a small but constant number frames (which we denote $\lambda$) before giving its prediction. Figure 2 depicts this principle: the output frame $t + \lambda$ actually corresponds to the prediction for the input frame $t$. This allows the model to access more context about what happens right after the frame of interest $t$, *i.e.* frames between $t$ and $t + \lambda$.

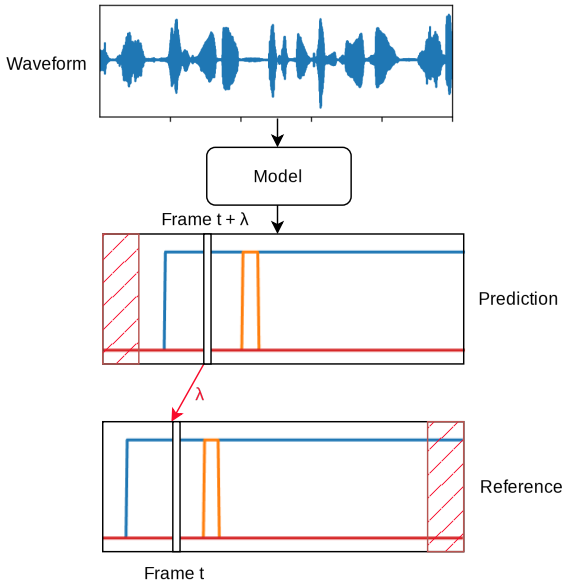


Figure 2: *Look-ahead mechanism (the left part of the prediction and the right part of the reference are never used)*

In practice, adding this look-ahead ability to the model does not need any additional architectural changes on the model side. It is achieved at training time, by shifting the prediction and the target before calculating the permutation-invariant powerset cross-entropy loss [7]. More precisely, without look-ahead, the loss function is defined as

$$\mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \min_{p \in \mathbb{P}} \mathcal{L}_{\text{CE}}\left(\mathrm{p}(\mathbf{y}), \hat{\mathbf{y}}\right) \tag{1}$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the targets and predictions in *powerset* space [7], and $\mathbb{P}$ is the set of speaker permutations in this space. Adding look-ahead is obtained by cropping out the last $\lambda$ frames of the targets and the first $\lambda$ frames of the predictions (depicted as hatched red rectangles in Figure 2):

$$\mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \min_{p \in \mathbb{P}} \mathcal{L}_{\text{CE}}\left(p(\mathbf{y}_{0 \to T-\lambda}), \hat{\mathbf{y}}_{\lambda \to T}\right) \tag{2}$$

The approach can easily be generalized to multiple latencies $(\lambda_1, \lambda_2, \ldots, \lambda_K)$, though that needs a slight modification of the final classifier layer. We denote $K_{\text{powerset}}$ the number of classes in the *powerset* space (typically $K_{\text{powerset}} = 7$ for chunks with a maximum number of 3 speakers per chunk and 2 simultaneous speakers per frame). We duplicate the final classifier layer $K$ times, so that the model now outputs $K$ predictions $\hat{\mathbf{y}}^{\mathbf{k}}$, one for each latency. The rest of the model is common to every latency. The training loss is computed as the sum of the aforementioned look-ahead training loss over each latency:

$$\mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \sum_{k=1}^{K} \min_{p \in \mathbb{P}} \mathcal{L}_{\text{CE}}\left(p(\mathbf{y}_{0 \to T-\lambda_k}), \hat{\mathbf{y}}^{k}_{\lambda_k \to T}\right) \tag{3}$$

Despite this slight change in architecture, this infers no additional cost at inference time as one simply goes through the

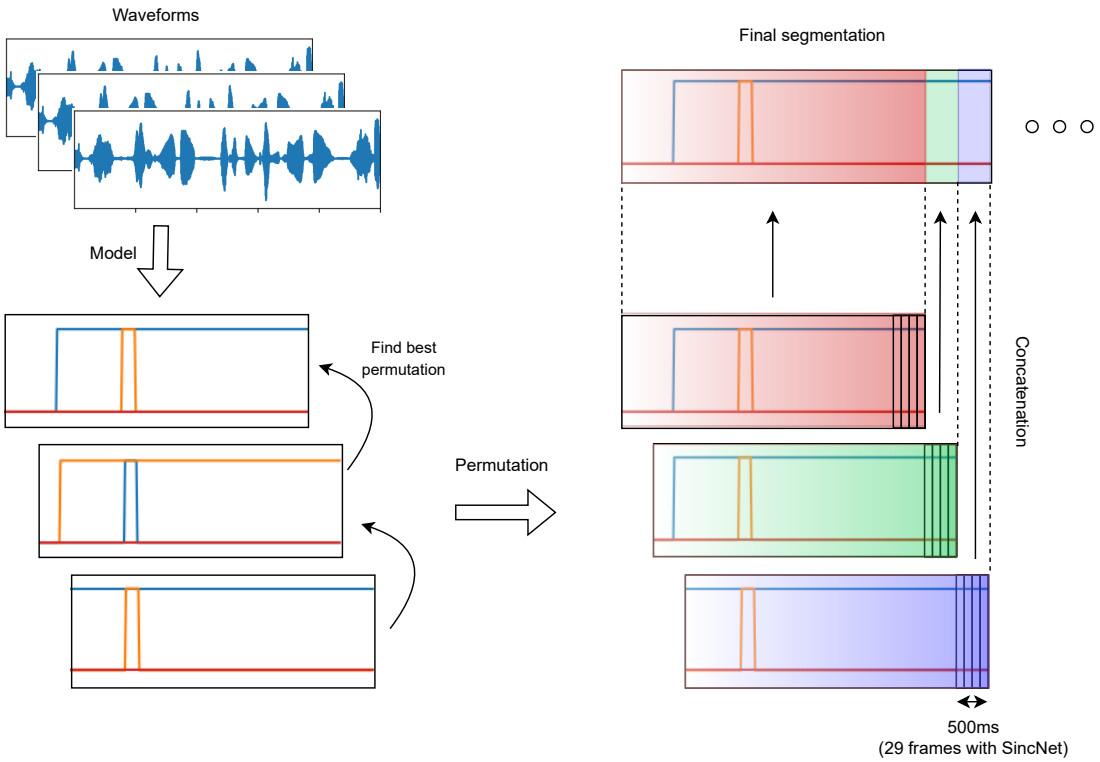


Figure 3: *Streaming inference. With the exception of the very first chunk that is used entirely, only the final* 500*ms of each subsequent chunk is used in the final concatenated output. Predictions of each chunk are permutated for consistency with previous chunks.*

branch of the requested latency. In other words, one can train a single model and use it for different use cases requiring different latencies. Going back to Figure 1, we can now focus our attention on the red curve which illustrates the performance at 500ms latency of such a multi-latency streaming model. Adding such a delay allows to compensate for most of the performance degradation initially witnessed when switching from offline to causal segmentation.

## 4. Streaming inference

Figure 3 depicts how this model is used in practice for streaming speaker segmentation of long form audio. To keep the advantages of hybrid speaker diarization approaches, we stick with an approach based on sliding windows (5s chunks with a 500ms stride). For instance, when working with 5s chunks and 500ms stride, 10 chunks are active at the same time, being processed in parallel, frame by frame. With the exception of the very first chunk that is used entirely, only the final 500ms of each subsequent chunk is used in the final concatenated output.

To ensure permutation consistency and not accidentally break speech turns when switching from chunk $c-1$ to chunk $c$, we find the permutation $p_c$ that minimizes the loss defined in Equation 1 between the prediction on chunk $c-1$ (used as targets $\mathbf{y}$) and chunk $c$, as soon as the part of chunk $c$ that will not end up in the final concatenation is available (here: frames between t=0 and t=4.5s, used as prediction $\hat{\mathbf{y}}$). The final 500ms of chunk $c$ are then concatenated, frame by frame, to the final output using this permutation $p_c$.

Note that this approach does not go all the way to speaker diarization but focuses on speaker segmentation (hence the title of the paper). Nevertheless it remains useful as several streaming tasks can be addressed with speaker segmentation only: streaming voice activity detection, streaming overlapped speech detection, streaming speaker change detection, or even streaming automatic speech recognition.

## 5. Experiments

We ran experiments and report results on the MixHeadset variant of the AMI meeting dataset [16], following the official train/development/test partition. It comprises 100 hours of recordings of meetings in English, divided into around 80h for training and 20h for the development and test sets.

Our segmentation model processes 5 second audio chunks sampled at 16kHz, corresponding to sequences of 80k audio samples. These sequences are fed into SincNet convolutional layers, following the original configuration [15], with the exception of the stride of the initial layer, set to 10 (we get one frame every 17ms), and the removal of the instance normalization layers. Four uni-directional Long Short-Term Memory (LSTM) recurrent layers (each with 128 units) 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 a logsoftmax activation function outputs either a $K_{\text{powerset}}$ dimensional speaker activation or a $K_{\text{powerset}} \times K$ dimensional activation if used in the multi-latency configuration (K is the number of latencies). Overall, our model contains 600k trainable parameters – most of which (500k) comes from the recurrent layers.

We train the model with Adam optimizer with default PyTorch parameters and mini-batches of size 32. Learning rate

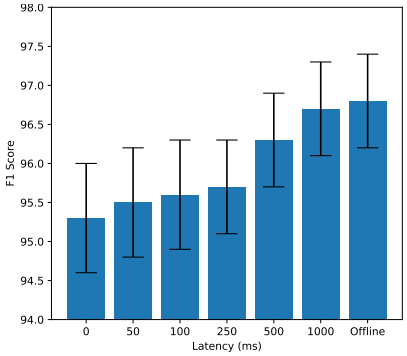


Figure 4: *Impact of latency on voice activity detection.*

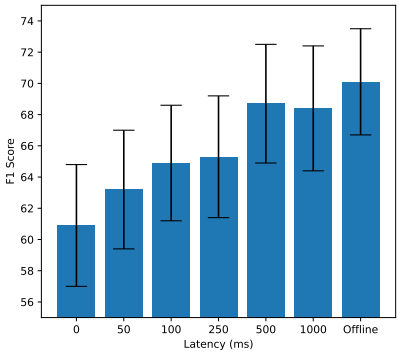


Figure 5: *Impact of latency on overlapping speech detection.*

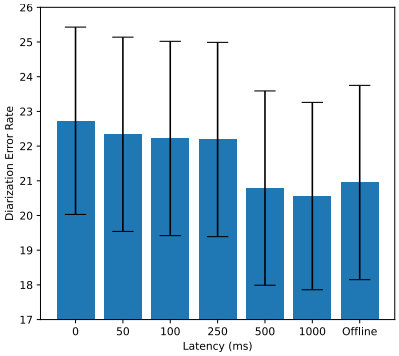


Figure 6: *Impact of latency on speaker change detection.*

is initialized at $10^{-3}$ and reduced by a factor of 2 every time its performance on the development set reaches a plateau for 30 epochs straight. The models are trained for at most one hundred hours. All metrics were computed using pyannote.metrics [17] open source Python library.

## 6. Results and discussion

We train a single multi-latency look-ahead speaker segmentation model on AMI training set, with the following target latencies: 0ms, 50ms, 100ms, 250ms, 500ms, and 1s. When studying the impact of latency in the rest of this section, keep in mind that there is only one single model behind, and we simply choose which latency branch to use at inference time.

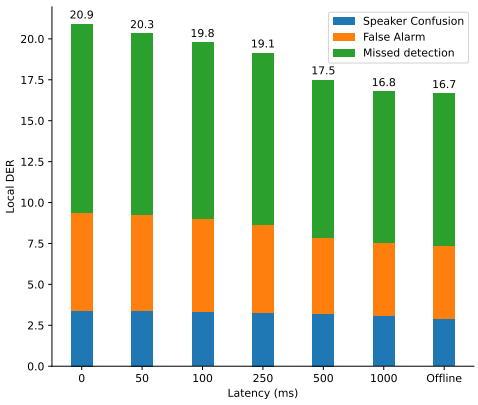


Figure 7: *Performance of the proposed streaming speaker segmentation model, as a function of the considered latency. Numbers are computed on the first 4 seconds of 5 seconds chunks sampled from AMI test set (for fair comparison with the* 1000*ms latency that does not generate output for last second).*

Before diving into long-form streaming experiments, Figure 7 focuses on the impact of latency within a 5s chunk. Reported numbers in Figure 7 shall therefore not be confused with standard diarization error rates on AMI test set in the literature. The first observation is that a 1s latency is almost as a good as the non-streaming bi-directional version of the model. We also report the actual value of the 3 components of diarization error rate (missed detection, false alarm and speaker confusion rates) to get a better understanding of the impact of the look-ahead mechanism. Most of the improvement comes from a significant decrease in missed detection and false alarm rates. The speaker confusion rate remains almost constant (and low) as we increase latency – which is somehow expected as there are only a few speakers in a 5s chunk.

Figures 4, 5 and 6 summarize results on long-form recordings, for voice activity detection, overlap speech detection, and speaker change detection. Because AMI test only contains 16 files, the confidence interval at 95% computed using `scipy`'s `bayes_mvs` is quite high [18]. Streaming voice activity detection is derived from the proposed segmentation by classifying frames as *"speech"* when at least one speaker is active, and *"non-speech"* otherwise. Streaming overlapped speech detection is achieved by classifying frames as *"overlapping speech"* when two or more speakers are active at the same time, and *"non-overlapping speech"* otherwise. Streaming speaker change detection is evaluated with oracle clustering of speech turns obtained thanks to the speaker segmentation, and computing the resulting diarization error rate. In all three cases, and as anticipated in Figure 7, a higher latency increases performance. For the three metrics, the improvement even seems almost linear with respect to the allowed latency, up to a latency of 1s that is almost on par with the performance of an offline model.

## 7. Conclusion

In this paper, we study the impact of a multi-latency look-ahead mechanism on the quality of a streaming speaker segmentation model. We test those approaches on the AMI dataset and compare the streaming model with its offline counterpart. We show that our proposed method is systematically capable of closing the gap between the streaming and the offline configuration. Next steps include validating the approach on other datasets and integrating this approach into a complete streaming speaker diarization pipeline using incremental online clustering of speaker embedding extracted from resulting speech turns.

## 8. Acknowledgements

This work was granted access to the HPC resources of IDRIS under the allocation AD011012177R3 made by GENCI, and supported by the Agence de l'Innovation Défense under the grant number 2022 65 0079.

# 9. References


[1] 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," *Computer Speech & Language*, 2022.

[2] 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.

[3] 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.

[4] H. Bredin and A. Laurent, "End-to-end speaker segmentation for overlap-aware resegmentation," in *Proc. Interspeech 2021*, 2021.

[5] K. Kinoshita, M. Delcroix, and N. Tawara, "Integrating End-to-End Neural and Clustering-Based Diarization: Getting the Best of Both Worlds," in *Proc. ICASSP 2021*, 2021.

[6] ——, "Advances in Integration of End-to-End Neural and Clustering-Based Diarization for Real Conversational Speech," in *Proc. Interspeech 2021*, 2021.

[7] A. Plaquet and H. Bredin, "Powerset multi-class cross entropy loss for neural speaker diarization," *Interspeech*, 2023.

[8] S. Horiguchi, S. Watanabe, P. Garcia, Y. Takashima, and Y. Kawaguchi, "Online neural diarization of unlimited numbers of speakers using global and local attractors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 706–720, 2023.

[9] Y. Xue, S. Horiguchi, Y. Fujita, S. Watanabe, and K. Nagamatsu, "Online end-to-end neural diarization with speaker-tracing buffer," *SLT 2021*, 2021.

[10] W. Wang and M. Li, "End-to-end online speaker diarization with target speaker tracking," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[11] J. M. Coria, H. Bredin, S. Ghannay, and S. Rosset, "Overlap-Aware Low-Latency Online Speaker Diarization Based on End-to-End Local Segmentation," in *Proc. ASRU 2021*, 2021.

[12] L. Du, H. Mei, and J. Eisner, "Autoregressive modeling with lookahead attention," *arXiv preprint arXiv:2305.12272*, May 2023, cS - Computation and Language. [Online]. Available: https://arxiv.org/abs/2305.12272

[13] L. Zhou, J. Zhang, and C. Zong, "Look-ahead attention for generation in neural machine translation," *Natural Language Processing and Chinese Computing*, 2017.

[14] D. Liang, S. Nian, and X. Li, "Frame-wise streaming end-to-end speaker diarization with non-autoregressive self-attention-based attractors," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

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

[16] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, and M. Kronenthal, "The AMI Meetings Corpus," in *Proc. Symposium on Annotating and Measuring Meeting Behavior*, 2005.

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

[18] T. E. Oliphant, "A bayesian perspective on estimating mean, variance, and standard-deviation from data," *Faculty Publications*, vol. 278, 2006, https://scholarsarchive.byu.edu/facpub/278.