



# Efficient Transformers for End-to-End Neural Speaker Diarization

*Sergio Izquierdo, Beltran Labrador, Alicia Lozano-Diez, Doroteo T. Toledano*

AUDIAS (Audio, Data Intelligence and Speech), Universidad Autónoma de Madrid, Spain

ser.izquierdo@estudiante.uam.es, {beltran.labrador, alicia.lozano}@uam.es

## Abstract

The recently proposed End-to-End Neural speaker Diarization framework (EEND) handles speech overlap and speech activity detection natively. While extensions of this work have reported remarkable results in both two-speaker and multi-speaker diarization scenarios, these come at the cost of a long training process that requires considerable memory and computational power. In this work, we explore the integration of efficient transformer variants into the Self-Attentive EEND with Encoder-Decoder based Attractors (SA-EEND EDA) architecture. Since it is based on Transformers, the cost of training SA-EEND EDA is driven by the quadratic time and memory complexity of their self-attention mechanism. We verify that the use of a linear attention mechanism in SA-EEND EDA decreases GPU memory usage by 22%. We conduct experiments to measure how the increased efficiency of the training process translates into the two-speaker diarization error rate on CALLHOME, quantifying the impact of increasing the size of the batch, the model or the sequence length on training time and diarization performance. In addition, we propose an architecture combining linear and softmax attention that achieves an acceleration of 12% with a small relative DER degradation of 2%, while using the same GPU memory as the softmax attention baseline.

**Index Terms:** speaker diarization, end-to-end neural diarization, efficient transformers

## 1. Introduction

Speaker diarization systems address the problem of “who spoke when” in a multi-speaker recording. Without any prior knowledge about the speakers or their identity, the task of such systems is to annotate the speaker turns [1].

Traditionally, modular systems have been designed to address the speaker diarization task. A typical modular system, such as [2], divides speech recordings into smaller chunks and computes a speaker embedding for each segment. Next, a clustering algorithm is executed on these representations, aiming to group all segments spoken by the same speaker. Last, a certain degree of post-processing is applied in order to obtain the final diarization output. The result is an annotated list of speech segments, describing the start and end timestamps of each segment and an identifier (e.g. speaker 1, 2, 3...) linking each detected speaker to each of their interventions.

A more recent, yet promising line of research is the end-to-end neural speaker diarization framework (EEND) [3, 4, 5]. In contrast to the modular approach, which requires each module to be trained separately, EEND systems are trained as a whole. As a consequence, the EEND approach offers three major advantages with respect to its modular counterpart. Firstly, EEND systems are specifically optimized to minimize diarization errors. Secondly, because they can handle speech overlap between multiple speakers, they have the potential to

reach a better performance on real-world applications. Thirdly, EEND systems do not require an external VAD (Voice Activity Detector), given they perform this role natively.

Posterior works have proposed architecture enhancements to the EEND framework [4, 5, 6], extensions to more complex scenarios of the diarization task [7, 8] and enhancements to the crucial data preparation stage [9, 10]. However, no work has delved into the computational complexity of the EEND architecture. Besides the evident advantages of accelerating the lengthy and computationally expensive EEND training process, works such as [11] have highlighted the necessity of shifting towards more efficient methods to cope with the non-negligible economic and environmental impact of deep learning techniques.

In particular, SA-EEND EDA (Self-Attentive EEND with Encoder-Decoder based Attractors) [5] obtains remarkable results in both two-speaker and multi-speaker diarization scenarios. Because SA-EEND EDA is based on Transformers, its training cost is driven by the quadratic time and memory complexity of their self-attention mechanism. Aiming to limit the impact of this shortcoming of the Transformer architecture, which is prevalent across many machine learning tasks, the community has proposed a wide range of efficient transformer variants that reduce their computational cost by the use of different techniques [12]. In particular, Linear Transformers [13] propose a simplification of the attention mechanism that achieves linear complexity, reporting impressive acceleration of speech recognition systems with a reduced penalty in error rates.

In this work, we leverage linear transformers to perform the speaker diarization task. We verify that integrating the linear attention mechanism into the SA-EEND EDA architecture leads to a sizable decrease in GPU memory usage, which allows fitting bigger batches in memory and subsequently accelerating the training process by up to 17%. In addition to increasing the batch size, it is possible to allocate the released memory to feed longer sequences or train bigger models, working around GPU memory limitations. We present the experimental results, in which we measure the impact of each design choice on both training speed and Diarization Error Rate (DER) on CALLHOME[14].

## 2. Model description

### 2.1. EEND framework review

The EEND framework [3] laid the foundations for approaching speaker diarization as a supervised learning problem. After generating a vast amount of labelled synthetic conversations, the system is trained to estimate the probability of intervention of each speaker in every frame using a permutation invariant cost function. Then, a fine-tuning phase adapts the model weights to increase the diarization performance on real datasets.

Two extensions bring crucial improvements to the EEND

architecture. First, SA-EEND [4] replaces the original BLSTM by Transformer encoders, allowing the system to be conditioned on a wider range of input frames. Next, [5] includes the computation of Encoder-Decoder based Attractors (EDA) to allow speaker diarization on an unknown, variable number of speakers. The consolidation of these changes into the original EEND proposal results on the SA-EEND EDA architecture, which is portrayed in figure 1.

Effectively, the core of the self-attention architecture lies in the series of  $P$  transformer encoders [15]. The first encoder receives the result of a linear transformation of the sequence  $x$  formed by  $T$  frames of  $D$ -dimensional acoustic features:

$$e_0 = W_0 x + b_0 \quad (1)$$

The remaining encoders are connected sequentially. Encoder  $p$  receives the output  $e_{p-1}$  of its predecessor and applies layer normalization to obtain  $\bar{E}_{p-1}$ , which is passed to a multi-head attention mechanism of  $H$  heads. Each head  $h$  computes three linear projections to obtain the query, key and value, denoted as  $Q_{p-1,h}$ ,  $K_{p-1,h}$  and  $V_{p-1,h}$ . These are the input of the self-attention functions that we describe in section 2.2:

$$\alpha_{p,h} = \text{Attention}(Q_{p-1,h}, K_{p-1,h}, V_{p-1,h}) \quad (2)$$

The results of all  $H$  heads are concatenated, projected and added to a residual connection from the output of the previous encoder:

$$S_p = \text{Concat}(\alpha_{p,1}, \dots, \alpha_{p,H}) \cdot W_p + \bar{E}_{p-1} \quad (3)$$

$S_p$  is normalized and fed to a position-wise feedforward network, described in [4], supplemented by a residual connection. The output of the last encoder block is normalized and processed by the EDA module [5], which estimates the number of speakers in the given sequence. The result is  $A \in \mathbb{R}^{D \times S}$ , which identifies  $S$  speaker-wise attractors. Finally, the system outputs the frame-wise posterior probabilities of activity of each of the speakers using the sigmoid function  $\sigma(\cdot)$ :

$$\hat{Y} = \sigma(A^T E_P), \quad \hat{y} \in [0, 1]^{(S \times T)} \quad (4)$$

## 2.2. Attention mechanism

### 2.2.1. Softmax attention

The scaled dot-product attention [15], also referred to as *softmax attention* in the literature [13], is the standard self-attention function in SA-EEND EDA [4, 5]. Given the Query, Key and Value, of dimension  $D$ , it is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{Q^T K}{\sqrt{D/H}} \right) V \quad (5)$$

Hence, the similarity score between the query and the key is computed as the exponential of their dot product. For a sequence of size  $T$ , the computational cost of this attention mechanism is  $\mathcal{O}(T^2)$ .

### 2.2.2. Linear attention

Linear transformers, proposed in [13], were conceived to reduce the computational cost of the attention mechanism. Defining a kernel feature map  $\phi(\cdot)$  and leveraging the associativity

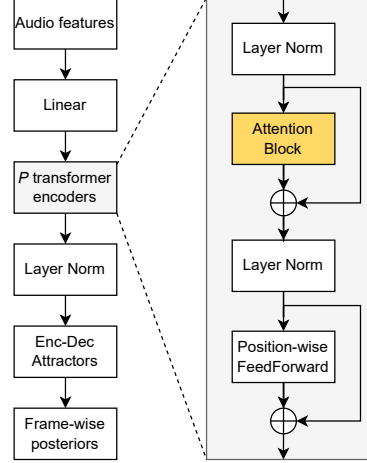


Figure 1: SA-EEND EDA [5] architecture.

property of matrix multiplication, it is possible to compute the attention as:

$$\text{Attention}(Q, K, V) = \phi(Q) U Z \quad (6)$$

where:

$$U = \phi(K) V$$

$$Z = \frac{1}{\sum_i \phi(K)^T} \quad (7)$$

Using this formulation and  $\phi(x) = \text{ELU}(x) + 1$ , the computational complexity decreases to  $\mathcal{O}(T)$ .

## 3. Experimental setup

### 3.1. Data preparation

Similarly to [3, 9], because our objective is to diarize real, two-speaker phone conversations, we evaluate the system on CALLHOME [14]. We divide<sup>1</sup> the dataset into two partitions: CH1-2spk, that we use to extract conversational statistics and for fine-tuning; and CH2-2spk, that we use exclusively for evaluation.

In contrast, our training set is composed of simulated conversations, generated following the approach presented in [9]. We start by building a pool of single speaker utterances from datasets Switchboard-2 (phases II and III) [16, 17], Switchboard Cellular (parts 1 and 2) [18, 19] and NIST Speaker Recognition Evaluation (years 2004, 2005, 2006, 2008) [20, 21, 22, 23, 24, 25, 26]. Speech segments are extracted using a time delay neural network and statistical pooling VAD<sup>2</sup>. Then, we follow the approach described by [9], sampling the aforementioned pool to generate simulated conversations that respect the silence and overlap statistics from the real conversations in the CH1-2spk subset. We only augment the conversations with background noise from the MUSAN dataset [27] during the generation process. As a result, we obtain a labeled dataset containing 2,480 hours of two-speaker simulated conversations.

<sup>1</sup>Our data preparation process makes use of the scripts released by [3] and [9], available respectively at <https://github.com/hitachi-speech/EEND> and [https://github.com/BUTSpeechFIT/EEND\\_dataprep](https://github.com/BUTSpeechFIT/EEND_dataprep).

<sup>2</sup><https://kaldi-asr.org/models/m4>

### 3.2. Diarization models

#### 3.2.1. Baseline system

As a baseline system, we use the PyTorch implementation<sup>3</sup> of SA-EEND EDA [5]. To facilitate result comparison, we keep the training and adaptation configuration from [9]. Setting sliding windows of 25ms and a hop length of 10ms, we extract Log-Mel filterbank features from the simulated conversations. The actual input of the system is obtained after concatenating a context of the seven preceding and posterior frames to each frame, and temporally subsampling the result by a factor of 10. As a consequence, we obtain 345-dimensional features every 100ms.

Next, a stack of four transformer encoder blocks computes a sequence of embeddings of 256 dimensions. Not having positional encoding, each encoder consists of four heads that apply the standard softmax attention mechanism. Similarly to [5], the embeddings are shuffled chronologically and propagated to the LSTM encoder-decoder module. Since we focus on two-speaker diarization, we force EDA to always predict two attractors.

Finally, we use a linear classifier to obtain the frame-wise posterior probabilities of speech activity by each of the two speakers represented by the attractors. Following the same approach as [9], we apply a threshold of 0.5 to determine whether a speaker is active on a given frame. We do this for the sake of simplicity while comparing our experiments, but recognise that tweaking this parameter may result on better DERs.

#### 3.2.2. Linear transformers

To measure the effects of switching the attention module we equip the baseline system with a configurable attention block, which in turn incorporates an implementation<sup>4</sup> of the linear attention mechanism. For simplicity and comparability with the baseline system, we do not support causal masking.

We conduct two groups of experiments. Firstly, we evaluate the GPU memory and time that are required to train a model that is as similar as possible to the baseline system, both in terms of architecture and hyperparameters. Secondly, we investigate the best use for the GPU memory that the linear attention mechanism releases, compared to the baseline implementation. We measure how batch size, sequence length and number of transformer encoders impact memory consumption, training time and diarization performance.

In addition, we evaluate the combination of softmax and linear attention in the *sandwich* architecture, that is composed of four encoders: one softmax, two linear and one softmax.

### 3.3. Training and adaptation

We run the training and adaptation processes in a single NVIDIA GeForce RTX 2080 Ti GPU, with 11GB memory. We store the pre-extracted acoustic features in a local SSD drive to accelerate training and prevent other processes from impacting the performance.

Unless otherwise noted in section 4, we train the system for 100 epochs on the simulated conversations, with a batch size of

<sup>3</sup>As a baseline system we use the code released by [9], available at <https://github.com/BUTSpeechFIT/EEND>.

<sup>4</sup>Our implementation of the linear transformer borrows from the code released by [13] at <https://github.com/idiap/fast-transformers>

64. We use the Adam [28] optimizer in conjunction with Noam [15] learning rate schedule, setting a warm-up period of 100,000 minibatch updates. Then, we fine-tune the model to CH1-2spk for an additional 100 epochs. While we also use Adam, learning rate is fixed to  $2 \cdot 10^{-5}$ . In experiments with different batch sizes, we proportionally adapt the warm-up period and learning rate.

### 3.4. Evaluation

We measure diarization performance on the two-speaker subset of CALLHOME-2 (CH2-2spk) [14], which is unseen during the training and adaptation phases. As typically done on this dataset [29], we use a collar of 0.25s to avoid penalty from annotation inconsistencies. We use Kaldi toolkit [30] to compute the widely used Diarization Error Rate (DER) [1], which considers false alarms (FA), missed speech and speaker confusion. When comparing DERs in section 4, we indicate relative differences.

## 4. Results

### 4.1. Analysis of the base scenario

In this set of experiments, we train the baseline architecture and the linear transformer under the equivalent conditions described in section 3.3. GPU consumption and average time for a training epoch are reported in table 1, as well as the DER after fine-tuning.

While we notice that the inclusion of linear attention mechanism accelerates the training process by 7%, the most remarkable result is the 22% decrease in GPU memory usage. This increase in efficiency is accompanied by a 20% drop in diarization performance. These results contrast with those reported in [13], where Katharopoulos et al. reached an acceleration of 70% in their automatic speech recognition experiment, with an associated increase of 58% in phoneme error rate.

Table 1: Comparison of training requirements and resulting DER after adaptation, with the settings in section 3.3.

Attention	GPU Memory (MB)	Time/epoch (min)	DER
Softmax	8,829 (1x)	28.3 (1x)	<b>7.64%</b>
Linear	<b>6,897</b> (0.78x)	<b>26.1</b> (0.93x)	9.18% (+20%)

### 4.2. Fitting more data on the GPU

#### 4.2.1. Effect of the batch size

Multiple works have delved into the optimal choice of batch sizes during training. On one hand, works such as [31] suggest that smaller batch sizes are bound to achieve more accurate models. On the other hand, [32] finds that bigger batches are beneficial for transformer-based models. Because our objective is to accelerate the training process as much as possible while limiting DER degradation, we evaluate multiple batch sizes to find the optimal training strategy for linear transformers.

As seen in table 2, the superior GPU utilization of bigger batch sizes leads to sizable decreases in training time. In fact, the maximum batch size that fits in GPU memory (116) trains 28% faster than the baseline system, with a relative DER increase of 24%. This result suggests that the acceleration can become proportionally higher than the degradation in DER over a certain batch size.

In addition, our experiments suggest a certain stability in the convergence fine-tuning phase. Table 2 illustrates a bigger variation in DER before than after fine-tuning. Besides, smaller batch sizes and slower learning rates during the fine-tuning phase did not lead to significant variations in DER.

Table 2: *Performance of the system trained with multiple batch sizes. FT refers to the fine-tuning process.*

Attention	Batch size	Time/epoch (min)	DER (before/after adapt.)	
Softmax	64	28.3 (1x)	<b>10.00%</b>	$\xrightarrow{FT}$ <b>7.64%</b>
Linear	64	26.1 (0.92x)	11.60%	$\xrightarrow{FT}$ 9.18%
Linear	96	23.1 (0.82x)	13.13%	$\xrightarrow{FT}$ 9.09%
Linear	116	<b>20.3</b> (0.72x)	11.60%	$\xrightarrow{FT}$ 9.48%

#### 4.2.2. Bigger batches to maximize GPU memory usage

As seen in table 2, increasing the batch size is a suitable approach to accelerate training with a relatively small DER penalty. We find that the biggest batch size that our GPU supports when training the baseline model is 86. In contrast, we can reach a batch size of 116 when using linear attention, representing an increase of 34% with similar memory consumption.

Table 3 summarizes the results of the experiment, in which we adapt the warm-up rate linearly, considering the batch sizes. Due to the larger batch sizes, both linear and *sandwich* attention reach considerable levels of acceleration. The *sandwich* offers a good balance between acceleration (12%) and DER (only a decrease of 2% with respect to softmax).

Table 3: *Comparison of training requirements and resulting DER after adaptation using the maximum batch size that fits in the GPU.*

Attention	Batch	GPU Mem.	Time/epoch (min)	DER
Softmax	86	10,209 MB	24.4 (1x)	<b>7.65%</b>
<i>Sandwich</i>	105	10,989 MB	21.6 (0.88x)	7.86%
Linear	<b>116</b>	10,125 MB	<b>20.3</b> ( <b>0.83x</b> )	9.48%

#### 4.2.3. Increasing sequence length

By default, the model is fed chunks of 500 frames during the training process. We hypothesize that feeding longer sequences might be beneficial for training, since it should expose the model to a larger amount of speaker changes. To verify this, we train the architecture with sequences of 1,000 frames. Results in table 4 suggest that while we can double the sequence length with a small increase in training time, the system requires more epochs to reach an equivalent DER.

Table 4: *Diarization performance after training on difference sequence lengths.*

Attention	No. Frames	Epochs	Time/epoch (min)	DER
Softmax	500	100	28.3	<b>7.65%</b>
Linear	500	100	<b>26.1</b>	9.18%
Linear	1,000	100	27.4	10.14%
Linear	1,000	125	27.4	9.19%

### 4.3. Effect of the number of encoders

Compared to softmax attention, a crucial advantage of the linear attention mechanism is its lower memory footprint. Besides accelerating training by using this released memory to process more data in parallel, it is possible to increase the number of parameters of the model. We measure the effect of training linear attention models with four, six and eight transformer encoder blocks, which is the maximum amount that fits the GPU memory with a batch size of 64. As expected, increasing model capacity leads to better diarization performance, at the cost of longer training times.

The results in table 5 indicate that the best DER with linear attention is obtained with 6 encoders. Because 6 linear attention encoders do not fill the GPU memory, we can increase batch size to 90, obtaining a DER of 8.17% in approximately the same time as the baseline system. This demonstrates how linear attention can help regulate the trade-off between memory consumption, training speed and diarization performance.

Table 5: *Results of increasing the number of encoders (enclosed between parenthesis next to the attention type). The batch size is always 64, except where indicated with †, where the batch size is 90.*

Attention	Time/epoch (min)	DER (before/after adapt.)	
Softmax (4 enc.)	28.3 (1x)	<b>10.00%</b>	$\xrightarrow{FT}$ <b>7.64%</b>
Linear (4 enc.)	<b>25.3</b> (0.89x)	11.60%	$\xrightarrow{FT}$ 9.18%
Linear (6 enc.)	31.7 (1.12x)	10.32%	$\xrightarrow{FT}$ 8.26%
Linear (6 enc.)†	27.9 (0.99x)	10.60%	$\xrightarrow{FT}$ 8.17%
Linear (8 enc.)	34.7 (1.23x)	10.61%	$\xrightarrow{FT}$ 8.50%

## 5. Conclusions

In this work, we have evaluated the effects of replacing the standard, softmax attention mechanism the SA-EEND EDA architecture by a more efficient linear attention function, both in terms of computational requirements and diarization performance. The smaller memory footprint of the linear attention mechanism opens multiple possibilities – from allowing the GPU to process bigger batches, therefore accelerating training by 17%; to increasing model sizes and processed sequence lengths in a way that would not be possible with softmax attention. In addition, we prove that the combination of both attention mechanisms achieves remarkable accelerations with a very small DER degradation. We believe these results, as well as the possibility of mixing linear and softmax attention in the architecture, have great potential as means of accelerating and reducing the cost of training multi-speaker SA-EEND EDA models, which we plan to address in future works.

## 6. Acknowledgements

This work was funded by the Spanish Ministerio de Ciencia, Innovacion y Universidades (MCIU) and Agencia Estatal de Investigacion (AEI), and also by the European Regional Development Fund (FEDER in Spanish, ERDF in English), by project RTI2018-098091-B-I00.

## 7. References

- [1] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, “A review of speaker diarization: Recent advances with deep learning,” *Computer Speech & Language*, vol. 72, p. 101317, 2022.
- [2] F. Landini, J. án 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*, vol. 71, p. 101254, 2022.
- [3] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, “End-to-end neural speaker diarization with permutation-free objectives,” *arXiv preprint arXiv:1909.05952*, 2019.
- [4] 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)*. IEEE, 2019, pp. 296–303.
- [5] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and K. Nagamatsu, “End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors,” *arXiv preprint arXiv:2005.09921*, 2020.
- [6] Y. C. Liu, E. Han, C. Lee, and A. Stolcke, “End-to-end neural diarization: From transformer to conformer,” *arXiv preprint arXiv:2106.07167*, 2021.
- [7] S. Horiguchi, Y. Takashima, P. Garcia, S. Watanabe, and Y. Kawaguchi, “Multi-channel end-to-end neural diarization with distributed microphones,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7332–7336.
- [8] Y. Xue, S. Horiguchi, Y. Fujita, S. Watanabe, P. García, and K. Nagamatsu, “Online end-to-end neural diarization with speaker-tracing buffer,” in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 841–848.
- [9] F. Landini, A. Lozano-Diez, M. Diez, and L. Burget, “From simulated mixtures to simulated conversations as training data for end-to-end neural diarization,” *arXiv preprint arXiv:2204.00890*, 2022.
- [10] N. Yamashita, S. Horiguchi, and T. Homma, “Improving the naturalness of simulated conversations for end-to-end neural diarization,” *arXiv preprint arXiv:2204.11232*, 2022.
- [11] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, “The computational limits of deep learning,” *arXiv preprint arXiv:2007.05558*, 2020.
- [12] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, “Efficient transformers: A survey,” *arXiv preprint arXiv:2009.06732*, 2020.
- [13] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, “Transformers are rnns: Fast autoregressive transformers with linear attention,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.
- [14] M. Przybicki and A. Martin, “2000 NIST Speaker Recognition Evaluation LDC2001S97,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2001.
- [15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [16] D. Graff, K. Walker, and A. Canavan, “Switchboard-2 phase II, LDC99S79,” *Web Download. Philadelphia: Linguistic Data Consortium*, 1999.
- [17] D. Graff, D. Miller, and K. Walker, “Switchboard-2 phase III, LDC2002S06,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2002.
- [18] D. Graff, K. Walker, and D. Miller, “Switchboard Cellular Part 1 audio LDC2001S13,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2001.
- [19] —, “Switchboard Cellular Part 2 audio LDC2004S07,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2004.
- [20] NIST Multimodal Information Group, “2004 NIST Speaker Recognition Evaluation LDC2006S44,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2006.
- [21] —, “2005 NIST Speaker Recognition Evaluation Training Data LDC2011S01,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2006.
- [22] —, “2005 NIST Speaker Recognition Evaluation Test Data LDC2011S04,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2011.
- [23] —, “2006 NIST Speaker Recognition Evaluation Evaluation Test Set Part 1 LDC2011S10,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2011.
- [24] —, “2006 NIST Speaker Recognition Evaluation Evaluation Test Set Part 2 LDC2012S01,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2012.
- [25] —, “2008 NIST Speaker Recognition Evaluation Training Set Part 1 LDC2011S05,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2011.
- [26] —, “2008 NIST Speaker Recognition Evaluation Test Set LDC2011S08,” *Web Download. Philadelphia: Linguistic Data Consortium*, 2011.
- [27] D. Snyder, G. Chen, and D. Povey, “Muson: A music, speech, and noise corpus,” *arXiv preprint arXiv:1510.08484*, 2015.
- [28] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [29] J. G. Fiscus, J. Ajot, M. Michel, and J. S. Garofolo, “The rich transcription 2006 spring meeting recognition evaluation,” in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2006, pp. 309–322.
- [30] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, “The kaldi speech recognition toolkit,” in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.
- [31] D. Masters and C. Lusch, “Revisiting small batch training for deep neural networks,” *arXiv preprint arXiv:1804.07612*, 2018.
- [32] M. Popel and O. Bojar, “Training tips for the transformer model,” *arXiv preprint arXiv:1804.00247*, 2018.