Odyssey 2020 The Speaker and Language Recognition Workshop
1-5 November 2020, Tokyo, Japan

# Optimal Mapping Loss: A Faster Loss for End-to-End Speaker Diarization

*Qingjian Lin*[1,2], *Tingle Li*[1], *Lin Yang*[4], *Junjie Wang*[4], *Ming Li*[1,3]

[1]Data Science Research Center, Duke Kunshan University, Kunshan, China
[2]School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China
[3]School of Computer Science, Wuhan University, Wuhan, China
[4]AI Lab of Lenovo Research, Beijing, China

ming.li369@dukekunshan.edu.cn

## Abstract

Recently, deep neural network based approaches have become more and more popular among modules of speaker diarization such as voice activity detection, speaker embedding extraction and clustering. However, end-to-end speaker diarization training still remains as a challenging task, partly due to the difficult loss design for the speaker-label ambiguity problem. The permutation-invariant training (PIT) loss could be a possible solution, but its time complexity is $\mathcal{O}(T \times N \times N!)$, where $T$ and $N$ denote the number of frames and speakers in each audio file, respectively. In this paper, we investigate the improvement on the PIT loss and further propose a novel optimal mapping loss, which directly computes the best matches between the output speaker sequence and the ground-truth speaker sequence through the Hungarian algorithm. Our proposed loss successfully reduces the time complexity to $\mathcal{O}(T \times N^2) + \mathcal{O}(N^3)$, meanwhile keeping the same performance as the PIT loss. The implement of three loss functions is available at https://github.com/lawlict/EEND_loss.

**Index Terms**: end-to-end speaker diarization, speaker-label ambiguity, permutation-invariant training loss, optimal mapping loss, Hungarian algorithm

## 1. Introduction

Speaker diarization is the task of partitioning multi-speaker audios into short segments and clustering them according to the speaker identities. It solves the problem of *"who spoke when"* [1, 2], which is essential in a variety of applications like telephone calls, meeting recordings and child care. Diarization systems can also serve as the frontend of automatic speech recognition (ASR) to enhance the transcription performance in multi-speaker conversations.

A standard speaker diarization framework usually consists of multiple modules, including voice activity detection (VAD), segmentation, speaker embedding extraction and clustering, as shown in Figure 1. First, VAD detects speech in the audio and removes non-speech regions. Typical VAD systems include generative models like Gaussian Mixture Model (GMM) [3] and Hidden Markov Model (HMM) [4], and discriminative models like linear discriminant analysis (LDA) [5], support vector machine (SVM) [6] and deep neural networks (DNN) [7, 8, 9]. Second, speaker changepoint detection (SCD) [10, 11] or uniform segmentation [12] splits speech into speaker-homogeneous short segments. Third, the short segments are mapped into the speaker-wise subspace and generate fixed-dimensional speaker embeddings like i-vector [13], x-vector [14] or other penultimate layer outputs from end-to-end speaker verification systems [15, 16]. Finally in the clustering stage, similarity measurement techniques such as cosine distance, probabilistic linear discriminant analysis (PLDA) [17, 18] and long short-term memory models (LSTM) [19] measure similarity scores between speaker embeddings, followed by clustering algorithms like K-means [20], agglomerative hierarchical clustering (AHC) [12, 21] or spectral clustering [19, 20].

Although such a framework has been widely adopted and achieved state-of-the-art performance, it cannot be optimized as a whole to reduce the diarization error rate. Therefore, the potential of deep neural networks has not been fully explored. When researchers turn to end-to-end speaker diarization, there arise a series of difficulties, one of which is speaker-label ambiguity. Given an audio with ground-truth labels "AAABBC" where A, B and C are speakers, we are required to encode the speakers into integers in the data preparation process. However, encoded labels like *"111223"* and *"222113"* are equally correct, making it hard to define unique training labels. To solve the problem, [22] and [23] propose supervised online clustering by defining speaker labels according to the first appearance (that is, *"111223"* should be the correct labels other than *"222113"*), and predict speakers of the current moment relying on the past moments. Although it is possible to migrate the online clustering idea to end-to-end speaker diarization, the limitation of predicting one speaker at a time prevents it from detecting overlapped speakers, another difficulty in the diarization task.

In [24] and [25], the authors come up with another solution by using the permutation-invariant training (PIT) loss function, which was first proposed in the speech separation field [26]. Given frame-wise diarization outputs and ground-truth speaker labels, PIT considers all permutations of labels, and computes losses between outputs and each permutation. Then the minimum loss is returned for network backpropagation. It is shown to be effective in handling problems of speaker-label ambiguity and overlapped speaker detection. However, as the number of speakers increase, PIT begins to compute losses in factorial time. The weakness was initially not considered by the PIT designer because there are only two mixed speakers in most of speech separation cases. As for our diarization task, it is common that conversations arise between three or more speakers, demanding expensive computation costs.

In this paper, we are motivated to improve the PIT loss and propose a novel optimal mapping (OPTM) loss for end-to-end speaker diarization. Instead of permutating all kinds of possible labels, we calculate the best matches between the sequences of output speakers and label speakers using the Hungarian algorithm. Our proposed OPTM loss function runs in polynomial time, meanwhile keeping the same performance as the PIT loss.



10.21437/Odyssey.2020-18

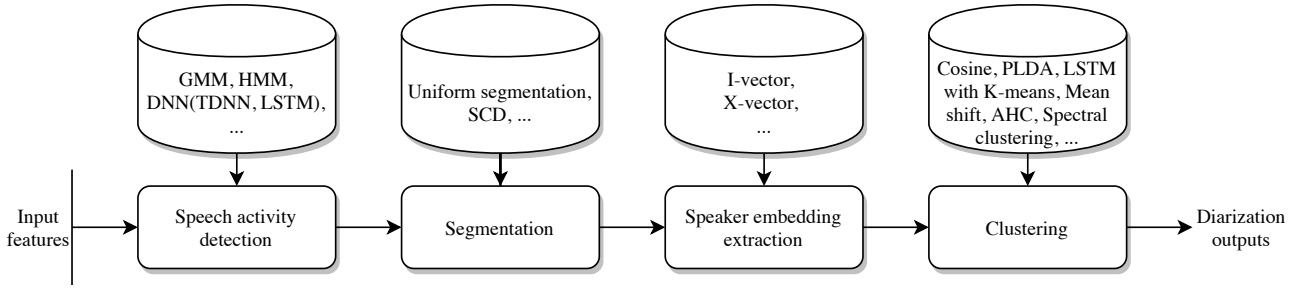



Figure 1: A standard speaker diarization pipeline.

The rest of this paper is organized as follows. Section 2 briefly presents the end-to-end diarization system in [25]. Section 3 describes two kinds of improvement on PIT loss and analyzes their time complexity mathematically. Experimental results and discussions are provided in Section 4, while conclusions are drawn in Section 5.

## 2. End-to-end system with the PIT loss

This section describes the self-attentive end-to-end speaker diarization (SA-EEND) proposed in [25]. SA-EEND employs the encoder of Speech Transformer [27] as the model, with position decoding removed. Given frame-wise inputs like log-mel-filterbank (fbank) features, it directly generates speaker posteriors through the model. In addition, the PIT loss is used to cope with the speaker-label ambiguity problem. Figure 2 shows an overview of SA-EEND for the two-speaker case.

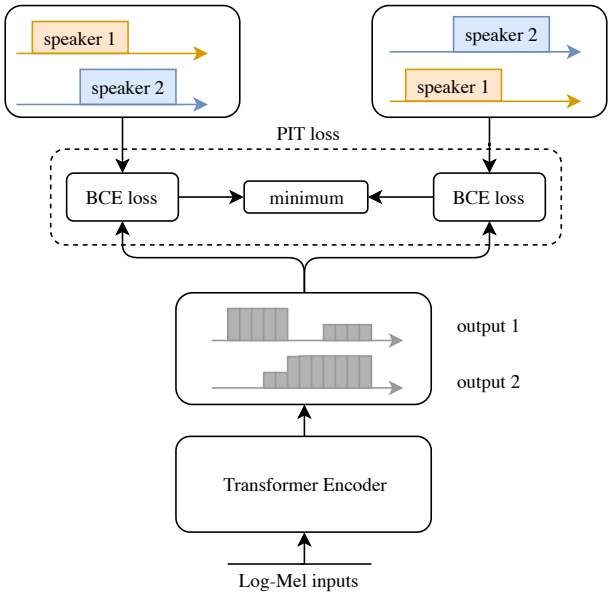



Figure 2: The SA-EEND structure for the two-speaker case.

### 2.1. Label encoding

Given a sequence of features $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T]^\top \in \mathbb{R}^{T\times D}$, SA-EEND encodes ground-truth labels as $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T]^\top \in \mathbb{R}^{T\times N}$, where $T$ and $N$ denote the duration and the number of speakers in the audio, respectively. Speaker label $\boldsymbol{y}_t = [y_{t,n} \in \{0,1\} | n = 1, 2, ..., N]^\top$ indicates a joint activity of speakers $1, 2, ...N$ at the $t$-th moment. $\boldsymbol{y}_t$ is zero-filled in non-speech regions, and $y_{t,m} = y_{t,n} = 1$ indicates overlapped speech of speakers $m$ and $n$ at moment $t$. Since labels are not encoded as one-hot vectors, SA-EEND is categorized as multi-label classfication rather than multi-class classficaton. The objective function is to estimate the most probable label sequence $\hat{\boldsymbol{Y}}$ among all possible sequences $\mathcal{Y}$:

$$\hat{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}\in\mathcal{Y}} P(\boldsymbol{Y}|\boldsymbol{X}). \tag{1}$$

With the assumption that frame-wise posterior is conditional independent on all inputs and speakers are presented independently, $P(\boldsymbol{Y}|\boldsymbol{X})$ can be factorized as:

$$\begin{aligned} P(\boldsymbol{Y}|\boldsymbol{X}) &= P(\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T|\boldsymbol{X}) \\ &\approx \prod_{t=1}^{T} P(\boldsymbol{y}_t|\boldsymbol{X}) \approx \prod_{t=1}^{T}\prod_{n=1}^{N} P(y_{t,n}|\boldsymbol{X}). \end{aligned} \tag{2}$$

### 2.2. Transformer encoder

The transformer structure with self-attention mechanism was originally proposed in language translation [28]. Due to the outstanding performance of capturing long-term sequence information, it is also widely applied in other fields. Here the encoder part of Speech Transformer is employed, with the position decoding part removed. The encoder is stacked with multiple layers, and input features are handled as follows:

$$\boldsymbol{e}_t^{(0)} = \text{Linear}(\boldsymbol{x}_t), \tag{3}$$

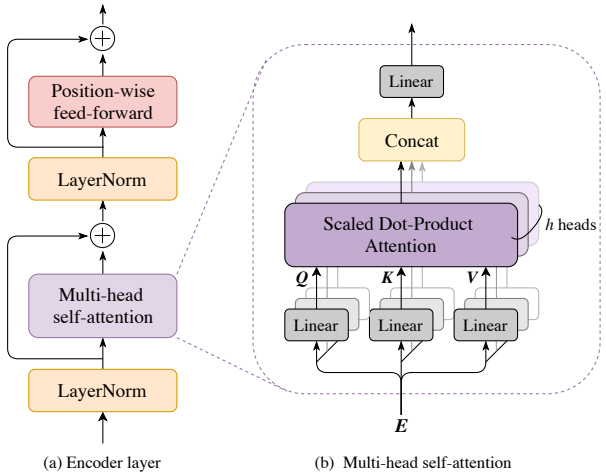
$$\boldsymbol{e}_t^{(p)} = \text{EncLayer}_t^{(p)}(\boldsymbol{e}_1^{(p-1)}, ..., \boldsymbol{e}_T^{(p-1)}), (1 \le p \le P), \tag{4}$$

$$\boldsymbol{z}_t = \sigma(\text{Linear}(\text{LayerNorm}(\boldsymbol{e}_t^{(P)}))), \tag{5}$$

where $\text{Linear}(\cdot)$ indicates the linear layer and $\text{EncLayer}^{(p)}(\cdot)$ is the $p$-th encoder layer. After $P$ encoder layers, features are normalized using layer normalization [29] $\text{LayerNorm}(\cdot)$, followed by the linear layer and the sigmoid function $\sigma(\cdot)$ sequentially. Outputs $\boldsymbol{z}_t$ are speech posteriors of $N$ speakers at the $t$-th moment.

The structure of one encoder layer is shown in Figure 3(a). Inputs first apply layer normalization and then go through the multi-head self-attention module with residual connection. The module consists of $h$-head parallelized self-attention blocks. For each head, feature mappings are converted into query matrix $\boldsymbol{Q} \in \mathbb{R}^{T\times d_q}$, key matrix $\boldsymbol{K} \in \mathbb{R}^{T\times d_k}$ and value matrix $\boldsymbol{V} \in \mathbb{R}^{T\times d_v}$ by different linear layers. Then Scaled Dot-Production Attention copes with these three matrices as follows:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}})\boldsymbol{V}. \tag{6}$$




(a) Encoder layer (b) Multi-head self-attention

Figure 3: (a) The structure of one encoder layer. (b) The structure of multi-head self-attention module.

Usuallly we set $d_q = d_k = d_v$ and the softmax function is performed row-wise. Results from each head are concatenated together over the last dimension and fed into a linear layer, as is shown in Figure 3(b). Outputs from the multi-head attention module again perform layer normalization and pass through the position-wise feed-forward layer [27] with residual connetction.

### 2.3. PIT loss

Viewing system outputs $\boldsymbol{Z}$ and ground-truth labels $\boldsymbol{Y}$ column-wise, we get $\boldsymbol{Z} = [\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, ..., \tilde{\boldsymbol{z}}_N]$ and $\boldsymbol{Y} = [\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{y}}_2, ..., \tilde{\boldsymbol{y}}_N]$, where $\tilde{\boldsymbol{z}}_n \in \mathbb{R}^{T\times 1}$ and $\tilde{\boldsymbol{y}}_n \in \mathbb{R}^{T\times 1}$ denote posteriors and labels of speaker $n$ across time, respectively. Speaker-label ambiguity arises that no matter how we shuffle $\boldsymbol{Y}$ by column, it can still be effective label expression. To cope with the problem, PIT considers all column-wise permutations of $\boldsymbol{Y}$ and computes the binary cross entropy (BCE) loss between $\boldsymbol{Z}$ and each kind of permutation $\boldsymbol{Y}^\phi = [\tilde{\boldsymbol{y}}_{a_1}, \tilde{\boldsymbol{y}}_{a_2}, ..., \tilde{\boldsymbol{y}}_{a_N}]$. $a_1, a_2, ..., a_N$ is the shuffled indices of $1, 2, ..., N$. Then the minimum loss is returned for network backpropagation. In brief, the PIT loss function can be written as

$$J^{\text{PIT}} = \frac{1}{TN} \min_{\phi\in\text{perm}(N)} \text{BCE}(\boldsymbol{Z}, \boldsymbol{Y}^\phi), \tag{7}$$

where perm($N$) denotes all possible permutations of (1,2,..., $N$).

## 3. Improvement on the PIT loss

Although the PIT loss has the potential to solve the speaker-label ambiguity problem, it imposes expensive computation costs when the number of speakers $N$ is large. Since the BCE loss processes $\boldsymbol{Z}$ and $\boldsymbol{Y}^\phi$ in $\mathcal{O}(T \times N)$ time and PIT generates $N!$ permutations of labels for a $N$-speaker audio, the time complexity of the PIT loss is $\mathcal{O}(T \times N \times N!)$ in total. Note that the PIT loss was initially proposed in speech separation, where audios are shorter with fewer speakers involved. In comparison, duration of diarization audios ranges from minutes to hours and more speakers are involved. Table 1 shows the duration and the number of speakers in three common diarization datasets including AMI [30], CallHome [31] and DIHARD2018 [32].

Table 1: Duration and speaker information about datasets. N_audios is the number of audios in datasets and n_spks is the number of speakers in audio files. Avg_dur is the average duration of the whole dataset. Note that CallHome refers NIST 2000 CallHome (Disk 8) dataset, and DIHARD2018 includes both dev and eval sets.

| Datasets | n_audios | avg_dur | n_spks |
|---|---|---|---|
| AMI | 170 | 35.05 min | $3 \sim 5$ |
| CallHome | 500 | 2.07 min | $2 \sim 7$ |
| DIHARD2018 | 336 | 7.03 min | $1 \sim 10$ |

### 3.1. FastPIT loss

Redundant computation exists in the process of Eq .7. To point it out, we push forward Eq. 7 as follows:

$$\begin{aligned} J^{\text{PIT}} &= \frac{1}{TN} \min_{\phi\in\text{perm}(N)} \text{BCE}([\tilde{\boldsymbol{z}}_1, ..., \tilde{\boldsymbol{z}}_N], [\tilde{\boldsymbol{y}}_{a_1}, ..., \tilde{\boldsymbol{y}}_{a_N}]) \\ &= \frac{1}{TN} \min_{\phi\in\text{perm}(N)} \sum_{n=1}^{N} \text{BCE}(\tilde{\boldsymbol{z}}_n, \tilde{\boldsymbol{y}}_{a_n}). \end{aligned} \tag{8}$$

Since $n$ ranges from 1 to $N$ and $a_n$ can be arbitrary integer in $1, 2, ..., N$, only $N^2$ pairs of $(\tilde{\boldsymbol{z}}_n, \tilde{\boldsymbol{y}}_{a_n})$ require actual computation of the BCE function. However, the function is called for $N \times N!$ times in Eq. 8, indicating that each $(\tilde{\boldsymbol{z}}_n, \tilde{\boldsymbol{y}}_{a_n})$ pair is repeatedly computed for $(N-1)!$ times. The redundant computation therefore appears when $N \geq 3$.

Our proposed idea is to compute BCE losses of all possible pairs $(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{y}}_j)$ $(i, j \in 1, 2, ...N)$ first and store them in the loss matrix $\boldsymbol{L} \in \mathbb{R}^{N\times N}$. Then during the permutation process, given $(\tilde{\boldsymbol{z}}_n, \tilde{\boldsymbol{y}}_{a_n})$ we just index and return the $L_{n,a_n}$ element. Here we name it as the FastPIT loss. The details are illustrated in Algorithm 1:

**Algorithm 1** FastPIT loss

**Input:** $\boldsymbol{Z} = [\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, ..., \tilde{\boldsymbol{z}}_N]$ and $\boldsymbol{Y} = [\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{y}}_2, ..., \tilde{\boldsymbol{y}}_N]$
**Output:** minloss

```
L ← zeros(N, N)
for i := 1 to N do
    for j := 1 to N do
        L_{i,j} ← BCEloss(z̃_i, ỹ_j)
    end for
end for
minloss ← INF
for a_1, a_2, ..., a_N in permutation(1,2,...,N) do
    loss ← (1/TN) Σ_{n=1}^{N} L_{n,a_n}
    minloss ← min(loss, minloss)
end for
```

We construct the loss matrix $\boldsymbol{L}$ in $\mathcal{O}(T \times N^2)$ time, and the permutation process costs $\mathcal{O}(N \times N!)$. Therefore, the time complexity of our FastPIT loss is $\mathcal{O}(T \times N^2) + \mathcal{O}(N \times N!)$ in total.

### 3.2. OPTM loss

Through the construction of loss matrix $\boldsymbol{L}$, we have achieved preliminary improvement on the PIT loss. However, the computation costs still increase in factorial time when $N$ is large. To deal with the problem, we must remove the permutation process. Recall that relationship between $\tilde{\boldsymbol{z}}_n$ and $\tilde{\boldsymbol{y}}_{a_n}$ is described

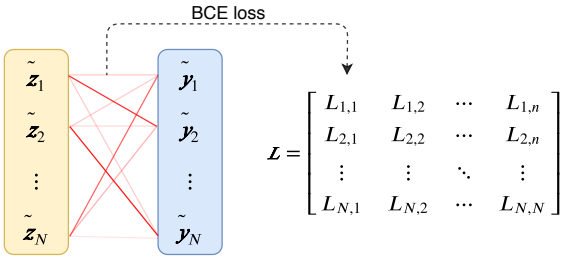



Figure 4: Definition of Task-Assignment problem. $\tilde{\boldsymbol{z}}_i$ and $\tilde{\boldsymbol{y}}_j$ belongs to diffrent groups and the BCE loss function describes the cost of assigning $\tilde{\boldsymbol{y}}_j$ to $\tilde{\boldsymbol{z}}_i$. The target is to find the best assignment with the least costs.

by BCE($\tilde{\boldsymbol{z}}_n$, $\tilde{\boldsymbol{y}}_{a_n}$), and each $\tilde{\boldsymbol{z}}_n$ must be assigned to one and only one optimal $\tilde{\boldsymbol{y}}_{a_n}$ in the final outputs (as is shown in Figure 4). This is a typical Task-Assignment problem. We thus propose the optimal mapping (OPTM) loss, which employs the Hungarian algorithm to find out the best matches between $\tilde{\boldsymbol{z}}_n$ and $\tilde{\boldsymbol{y}}_{a_n}$.

Hungarian algorithm [33] is proposed as a combinatorial optimization algorithm to solve the assignment problem in polynomial time. For our case, the element in $i$-th row and $j$-th column of loss matrix $\boldsymbol{L}$ is explained as the cost of assigning ground-truth speaker $j$ to output speaker $i$. Our target is to find the optimal assigning indices $(1, a_1)$, $(2, a_2)$, ..., $(N, a_N)$ so that the overall cost $\sum_{n=1}^{N} L_{n,a_n}$ is the lowest. The Hungarian algorithm can be descsribed as follows:

a) **Subtract row minima**: For each row of $\boldsymbol{L}$, find the lowest element and subtract it from each element in that row.

b) **Subtract column minima**: For each column, find the lowest element and subtract it from each element in that column.

c) **Conver all zeros with a minimum number of lines**: Cover all zeros in the resulting matrix using a minimum number of horizontal and vertical lines. If $N$ lines are required, an optimal assignment exists among the zeros. Stop the algorithm and return indices $(1, a_1)$, $(2, a_2)$, ..., $(N, a_N)$ of the zeros.

d) **Create additional zeros**: Find the smallest element $k$ that is not covered by any line in Step c). Substract $k$ from all uncovered elements, and add $k$ to all elements that are covered twice. Return to Step c).

Then our OPTM loss sums up elements $L_{1,a_1}, L_{2,a_2}, ..., L_{n,a_n}$ as the minimum loss. An brief view of the whole process is shown in Algorithm 2. The Hungarian algorithm copes with the assignment problem in $\mathcal{O}(N^3)$ time, and the overall time complexity of our OPTM loss is $\mathcal{O}(T \times N^2) + \mathcal{O}(N^3)$.

### 3.3. The number of speakers mismatch

Usually, the number of ground-truth speakers $N_{ref}$ for the diarization task is unknown. Given the outputs $\boldsymbol{Z} = [\tilde{\boldsymbol{z}}_1, ..., \tilde{\boldsymbol{z}}_{N_{sys}}]$ and the ground truth $\boldsymbol{Y} = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_{ref}}]$, the number of output speakers $N_{sys}$ may be larger or smaller than $N_{ref}$ due to false alarm or miss detection of diarization systems. In this subsection we discuss how to deal with the speaker number mismatched cases.

**Algorithm 2** OPTM loss

**Input:** $\boldsymbol{Z} = [\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, ..., \tilde{\boldsymbol{z}}_N]$ and $\boldsymbol{Y} = [\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{y}}_2, ..., \tilde{\boldsymbol{y}}_N]$
**Output:** minloss

```
L ← zeros(N, N)
for i := 1 to N do
    for j := 1 to N do
        L_{i,j} ← BCEloss(z̃_i, ỹ_j)
    end for
end for
(1, a_1), (2, a_2), ..., (N, a_N) ← Hungarian(L)
minloss ← (1/TN) Σ_{n=1}^{N} L_{n,a_n}
```

#### 3.3.1. $N_{sys} > N_{ref}$

When $N_{sys}$ is larger than $N_{ref}$, we make assumptions that there are actually $N_{sys}$ ground-truth speakers, but $N_{sys} - N_{ref}$ of them keep silent over the whole audio, namely "silent speakers". We correspondingly pad $\boldsymbol{Y}$ with zero vectors so that $\boldsymbol{Y}' = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_{ref}}, \boldsymbol{0}, ..., \boldsymbol{0}] \in \mathbb{R}^{T \times N_{sys}}$. $\boldsymbol{0} \in \mathbb{R}^{T \times 1}$ is labels of the "silent speaker". Now the OPTM loss function computes loss between $\boldsymbol{Z}$ and $\boldsymbol{Y}'$ as usual. When $\boldsymbol{0}$ is assigned to $\tilde{\boldsymbol{z}}_i$ by the Hungarian algorithm, it indicates that the diarization system makes a false alarm error.

#### 3.3.2. $N_{sys} < N_{ref}$

Similarly, when $N_{sys}$ is smaller than $N_{ref}$, we assume there are actually $N_{ref}$ output speakers, but $N_{ref} - N_{sys}$ of them are "silent speakers". We correspondingly pad $\boldsymbol{Z}$ to $\boldsymbol{Z}' = [\tilde{\boldsymbol{z}}_1, ..., \tilde{\boldsymbol{z}}_{N_{sys}}, \boldsymbol{0}, ..., \boldsymbol{0}] \in \mathbb{R}^{T \times N_{ref}}$ and compute loss between $\boldsymbol{Z}'$ and $\boldsymbol{Y}$. When $\tilde{\boldsymbol{y}}_i$ is assigned to $\boldsymbol{0}$, it indicates the system fails to detect speaker $i$ in the audio.

#### 3.3.3. *Comparison of the two cases*

Although the aforementioned two cases seem similar, we prefer the $N_{sys} > N_{ref}$ case rather than the other one. For the former case where $\boldsymbol{Y}$ is zero-padded to $\boldsymbol{Y}' = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_{ref}}, \boldsymbol{0}, ..., \boldsymbol{0}]$, a well-trained end-to-end diarization system is able to make the perfect prediction $\boldsymbol{Z}$ and reduce loss to zero in condition that $\boldsymbol{Z}$ equals any column-wise permutation of $\boldsymbol{Y}'$. However, for the latter case where $\boldsymbol{Z}$ is zero-padded to $\boldsymbol{Z}' = [\tilde{\boldsymbol{z}}_1, ..., \tilde{\boldsymbol{z}}_{N_{sys}}, \boldsymbol{0}, ..., \boldsymbol{0}]$, mistatch always exists between $\boldsymbol{Z}'$ and permutations of $\boldsymbol{Y}$ because the $\boldsymbol{0}$ vectors in $\boldsymbol{Z}'$ cannot find perfect assignment from $\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{y}}_2, ..., \tilde{\boldsymbol{y}}_{N_{ref}}$. Assigning $\tilde{\boldsymbol{y}}_i$ to $\boldsymbol{0}$ indicates system miss detection, and there is no remedial action to recall the missed speakers. Basically, it is because $N_{sys}$ defines the maximum number of speakers that the diarization system is able to detect.

This is an important support for our OPTM loss function. Since $N_{sys}$ is the output size of the end-to-end diarization network, it should be determined in the early stage. Conversations in real life often involve multiple speakers. To avoid miss detection and enhance the robustness for different scenarios, we tend to set $N_{sys}$ as a large number. The application of our OPTM loss will efficiently reduce the time complexity in such conditions.

## 4. Experimental Results

In this section, we investigate the actual time costs of the PIT loss, the FastPIT loss and the OPTM loss. Also, by reproducing part of experiments in [25] using different losses, we validate

whether they result in the same performance.

### 4.1. Time cost experiment

In Section 3, we estimate the time complexity of different loss functions by mathematical formats, but it is not intuitive enough for us to understand how fast they run. In addition, matrix acceleration by software and hardware may further reduce the actual time costs. Therefore, recording time costs through experiments is more reliable.

Here we generate float outputs ranging from zero to one and binary ground-truth labels by random programs, with the same shape of $B \times T \times N$. The batch size $B$ is set to 128 and $T$ is fixed as 500. The number of speakers $N$ ranges from 2 to 10. For each $N$, we repeat the process of data generation and loss computation using different loss functions for 100 times. Their average time costs are recorded in Table 2. Experiments are carried out on both CPU and GPU platforms. The CPU platform is a single core of Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, and the GPU platform is a GeForce GTX 1080ti GPU card with 11GB memory.

Table 2: Time costs (second) of differnt loss functions.

| | CPU | | | GPU | | |
|---|---|---|---|---|---|---|
| n_spks | PIT | FastPIT | OPTM | PIT | FastPIT | OPTM |
| 2 | 0.0072 | **0.0048** | 0.0055 | **0.0004** | 0.0007 | 0.0012 |
| 3 | 0.0589 | **0.0212** | 0.0218 | **0.0011** | 0.0016 | 0.0022 |
| 4 | 0.2951 | **0.0368** | 0.0374 | 0.0046 | **0.0034** | 0.0036 |
| 5 | 1.7936 | 0.0618 | **0.0594** | 0.0201 | 0.0079 | **0.0050** |
| 6 | 12.874 | 0.1143 | **0.0883** | 0.1250 | 0.0326 | **0.0074** |
| 7 | 112.75 | 0.3447 | **0.1292** | 0.8819 | 0.2011 | **0.0107** |
| 8 | - | 2.0124 | **0.1727** | 7.2475 | 1.6444 | **0.0139** |
| 9 | - | 20.267 | **0.2352** | 68.288 | 15.008 | **0.0166** |
| 10 | - | 186.88 | **0.2590** | - | 155.24 | **0.0216** |

During the process, losses computed by three loss functions are equally the same. On the CPU platform, both of our two proposed methods are faster than the original PIT loss. To be specific, the FastPIT loss costs the least time for $N \leq 4$, while the OPTM loss runs fastest for the rest cases. As for the GPU platform, the PIT loss keeps the slight advantage for $N \leq 3$, but the OPTM loss is still the fastest when $N$ is larger than 4. On both platforms, the PIT loss and the FastPIT loss run in factorial time as $N$ increases, while the OPTM loss remains relatively stable. In the extreme case where $N$ equals 10, the OPTM loss costs only 0.259s on CPU and 0.0216s on GPU; meanwhile it takes hundreds of seconds for the other two loss functions.

### 4.2. SA-EEND experiment

#### *4.2.1. Data*

SA-EEND method requires audio mixtures of different speakers for training, which can be simulated using a combination of single-speaker utterances. Consider a diarization-style mixture: each simulated mixture should have multiple utterances from $N$ speakers with reasonable silence intervals between utterances.

Here we employ the Swtchboard-2 (Phase II, III), Switchboard Cellular (Part 1, 2) and NIST Speaker Recognition Evualation datasets (2004, 2005, 2006, 2008) for data simulation. There are 5627 speakers in total, among which 5164 speakers are used for training while the rest 463 speakers are for testing. First we extract speech utterances from audios by using the time delay neural network (TDNN) based VAD model [1]. Then for each mixture we randomly sample $N$ speakers, each speaker with a nubmer of utterances from $U_{min}$ to $U_{max}$. Silent intervals between two speaker-homogeneous utterances are assumed to follow exponential distribution:

$$s(\delta) \sim \frac{1}{\beta} \exp(-\frac{\delta}{\beta}). \tag{9}$$

Note that larger $\beta$ generates diarization mixtures with less overlaps. After data generation, 37 recordings named "background" noises from MUSAN corpus [34] and 10,000 room impluse responses (RIRs) from Simulated Room Impulse Response Dataset [35] are employed for speech augmentation. Pseudocode about details can be view in [24]. Here we set $\beta = 2$, $N = 2$, $U_{min} = 10$ and $U_{max} = 20$. 100,000 simulated mixtures are generated for training and 500 for testing.

Moreover, 303 2-speaker audios are taken from NIST 2000 CallHome Disk 8 for real data evaluation. We split them into callhome1-spk2 with 148 audios and callhome2-spk2 with 153 audios. Callhome1-spk2 is used for model adaption and callhome2-spk2 is for testing.

#### *4.2.2. System configurations*

Input features are 23-dimensional fbank features with 25ms length and 10ms shift. Mean normalization is applied. Each feature frame is concatenated with the previous 7 frames and subsequent 7 frames, resulting in 23×15 input dimension. To deal with long audios and reduce computation, we use a subsampling factor of 10.

For the SA-EEND model, two encoder layers are stacked, each with 256 attention units and 4 heads. Dimension of the position-wise layer is set to 1024. We fix batch size $B = 64$ and length of input sequences $T = 500$. The model is trained on simulated training data by 100 epochs using Adam optimizer with learning rate scheduler [36]. During the process, the learning rate is set to $10^{-3}$. The training process terminates after 100 epochs, and model parameters over the last 10 epochs are averaged.

For domain adaption, the averaged model is retrained using callhome1-spk2. Configurations are almost the same as the last training stage, except that the learning rate is adjusted to $10^{-5}$.

In the evaluation stage, the SA-EEND model is tested on simulated testing data and callhome2-spk2 with or without adaption. We further apply an 11-frame filter on system outputs to get smooth results.

#### *4.2.3. Evaluation metric*

Following [25], we employ the diarization error rate (DER) [37] as our evaluation metric. Only short collars centered on each speech turn boundary is not evaluated (250ms on both sides). Note that this much stricter in comparison with [19] and [22], which assume an oracle VAD and ignore errors raised by overlapped speech. For our metric, false alarm and miss detection by VAD, speaker confusion errors and overlapped speech errors all account for DER.

#### *4.2.4. Results*

Experiments are carried out using three different loss functions. To avoid interference of the random process, we fix all random seeds to 777. Results are shown in Table 3.

[1]The VAD model: http://kaldi-asr.org/models/m4

Table 3: DERs of the simulated testing data (Simulated), callhome2-spk2 without adaption (CH) and with adaption (CH+adapt) using SA-EEND and different loss functions.

| | Simulated | CH | CH+adapt |
|---|---|---|---|
| SA-EEND + PIT | 9.19% | 18.35% | 13.00% |
| SA-EEND + FastPIT | 9.19% | 18.35% | 13.00% |
| SA-EEND + OPTM | 9.19% | 18.35% | 13.00% |
| i-vector et al. [25] | 33.74% | 12.10% | - |
| x-vector et al. [25] | 28.77% | 11.53% | - |

According to Table 3, combinations of SA-EEND and different loss functions result in the same DERs. It proves that the performance of the PIT loss and our two proposed methods is basically the same. The DER of the simulated testing data using SA-EEND is significantly lower than the two common baselines (i-vector and x-vector based methods in [25]) which take the same training databases, but SA-EEND performs worse than the baselines on the callhome2-spk2 dataset, whether model adaption is applied or not. The phenomenon has been reported in [25], indicating a large mismatch between simulated data and real-life data. In our future works, we will investigate how to improve the data simulation algorithms.

## 5. Conclusion

In this paper, we improve the PIT loss (FastPIT) by constructing loss matrices and further propose the OPTM loss function for end-to-end diarization training. In comparison with the PIT loss, our fastest loss reduces the time complexity from $\mathcal{O}(T\times N\times N!)$ to $\mathcal{O}(T\times N^2)+\mathcal{O}(N^3)$. Through experiments, we prove that time costs of three loss functions are similar when $N \leq 4$, while the OPTM loss runs faster for larger $N$ on both CPU and GPU platforms. We also validate that performance of the PIT loss, the FastPIT loss and the OPTM loss is equally the same by reproducing experiments with SA-EEND.

## 6. Acknowledgements

This research is funded in part by the National Natural Science Foundation of China (61773413), Key Research and Development Program of Jiangsu Province (BE2019054), Six talent peaks project in Jiangsu Province (JY-074), Guangzhou Municipal People's Livelihood Science and Technology Plan (201903010040), Science and Technology Program of Kunshan City. We acknowledge Hitachi for releasing the source code of SA-EEND[2].

[2]SA-EEND is available at https://github.com/hitachi-speech/EEND

## 7. References

[1] Sue E Tranter and Douglas A Reynolds, "An overview of automatic speaker diarization systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557–1565, 2006.

[2] Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals, "Speaker diarization: A review of recent research," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 356–370, 2012.

[3] Andreas Tsiartas, Theodora Chaspari, Nassos Katsamanis, Prasanta Kumar Ghosh, Ming Li, Maarten Van Segbroeck, Alexandros Potamianos, and Shrikanth Narayanan, "Multi-band long-term signal variability features for robust voice activity detection.," in *Interspeech*, 2013, pp. 718–722.

[4] Chuck Wooters, James Fung, Barbara Peskin, and Xavier Anguera, "Towards robust speaker segmentation: The icsi-sri fall 2004 diarization system," in *RT-04F Workshop*, 2004.

[5] Elias Rentzeperis, Andreas Stergiou, Christos Boukis, Aristodemos Pnevmatikakis, and Lazaros C Polymenakos, "The 2006 athens information technology speech activity detection and speaker diarization systems," in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2006, pp. 385–395.

[6] Andrey Temko, Dusan Macho, and Climent Nadeu, "Enhanced svm training for robust speech activity detection," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007, pp. 1025–1028.

[7] Rubén Zazo Candil, Tara N Sainath, Gabor Simko, and Carolina Parada, "Feature learning with raw-waveform cldnns for voice activity detection," in *Interspeech*, 2016, pp. 3668–3672.

[8] Florian Eyben, Felix Weninger, Stefano Squartini, and Björn Schuller, "Real-life voice activity detection with lstm recurrent neural networks and an application to hollywood movies," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 483–487.

[9] Shuo-Yiin Chang, Bo Li, Gabor Simko, Tara N Sainath, Anshuman Tripathi, Aäron van den Oord, and Oriol Vinyals, "Temporal modeling using dilated convolution and gating for voice-activity-detection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5549–5553.

[10] Marek Hrúz and Zbyněk Zajíc, "Convolutional neural network for speaker change detection in telephone speaker diarization system," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 4945–4949.

[11] Ruiqing Yin, Hervé Bredin, and Claude Barras, "Speaker change detection in broadcast tv using bidirectional long short-term memory networks," in *Interspeech*, 2017, pp. 3827–3831.

[12] Gregory Sell and Daniel Garcia-Romero, "Speaker diarization with plda i-vector scoring and unsupervised calibration," in *2014 IEEE Spoken Language Technology Workshop*, 2014, pp. 413–417.

[13] Stephen Shum, Najim Dehak, Reda Dehak, and James R Glass, "Unsupervised speaker adaptation based on the cosine similarity for text-independent speaker verification.," in *Odyssey 2010 The Speaker and Language Recognition Workshop*, 2010.

[14] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5329–5333.

[15] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

[16] Weicheng Cai, Jinkun Chen, and Ming Li, "Analysis of length normalization in end-to-end speaker verification system," in *Interspeech*, 2018, pp. 3618–3622.

[17] Simon JD Prince and James H Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[18] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Md Jahangir Alam, and Pierre Dumouchel, "Plda for speaker verification with utterances of arbitrary duration," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 7649–7653.

[19] Qingjian Lin, Ruiqing Yin, Ming Li, Hervé Bredin, and Claude Barras, "Lstm based similarity measurement with spectral clustering for speaker diarization," in *Interspeech*, 2019, pp. 366–370.

[20] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopz Moreno, "Speaker diarization with lstm," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5239–5243.

[21] Gregory Sell, David Snyder, Alan McCree, Daniel Garcia-Romero, Jesús Villalba, Matthew Maciejewski, Vimal Manohar, Najim Dehak, Daniel Povey, Shinji Watanabe, et al., "Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge.," in *Interspeech*, 2018, pp. 2808–2812.

[22] Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang, "Fully supervised speaker diarization," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 6301–6305.

[23] Qiujia Li, Florian L Kreyssig, Chao Zhang, and Philip C Woodland, "Discriminative neural clustering for speaker diarisation," in *arXiv preprint arXiv:1910.09703*, 2019.

[24] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Kenji Nagamatsu, and Shinji Watanabe, "End-to-end neural speaker diarization with permutation-free objectives," in *Interspeech*, 2019, pp. 4300–4304.

[25] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe, "End-to-end neural speaker diarization with self-attention," in *arXiv preprint arXiv:1909.06247*, 2019.

[26] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 241–245.

[27] Linhao Dong, Shuang Xu, and Bo Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5884–5888.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[29] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[30] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al., "The ami meeting corpus: A pre-announcement," in *International workshop on machine learning for multimodal interaction*. Springer, 2005, pp. 28–39.

[31] NIST, "2000 nist speaker recognition evaluation," 2000.

[32] Neville Ryant, Kenneth Church, Christopher Cieri, Alejandrina Cristia, Jun Du, Sriram Ganapathy, and Mark Liberman, "First dihard challenge evaluation plan," in *2018 tech. Rep.*, 2018.

[33] Roy Jonker and Ton Volgenant, "Improving the hungarian assignment algorithm," *Operations Research Letters*, vol. 5, no. 4, pp. 171–175, 1986.

[34] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," in *arXiv preprint arXiv:1510.08484*, 2015.

[35] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 5220–5224.

[36] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2014.

[37] NIST, "The 2009 (rt-09) rich transcription meeting recognition evaluation plan," in *http://www.itl.nist.gov/iad/mig/tests/rt/2009/docs/rt09-meeting-eval-plan-v2.pdf*, 2009.