The following publication Y. Tu and M. -W. Mak, "Aggregating Frame-Level Information in the Spectral Domain With Self-Attention for Speaker Embedding," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 30, pp. 944-957, 2022 is available at https://dx.doi.org/10.1109/TASLP.2022.3153267.

Aggregating Frame-Level Information in the Spectral Domain With Self-Attention for Speaker Embedding

Youzhi Tu and Man-Wai Mak, *Senior Member, IEEE*

***Abstract*—Most pooling methods in state-of-the-art speaker embedding networks are implemented in the temporal domain. However, due to the high non-stationarity in the feature maps produced from the last frame-level layer, it is not advantageous to use the global statistics (e.g., means and standard deviations) of the temporal feature maps as aggregated embeddings. This motivates us to explore stationary spectral representations and perform aggregation in the spectral domain. In this paper, we propose attentive short-time spectral pooling (attentive STSP) from a Fourier perspective to exploit the local stationarity of the feature maps. In attentive STSP, for each utterance, we compute the spectral representations through a weighted average of the windowed segments within each spectrogram by attention weights and aggregate their lowest spectral components to form the speaker embedding. Because most of the feature map energy is concentrated in the low-frequency region of the spectral domain, attentive STSP facilitates the information aggregation by retaining the low spectral components only. Attentive STSP is shown to consistently outperform attentive pooling on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval. This observation suggests that applying segment-level attention and leveraging low spectral components can produce discriminative speaker embeddings.**

***Index Terms*—Speaker verification, speaker embedding, short-time Fourier transform, self-attention, statistics pooling.**

I. Introduction

SPEAKER embedding plays a vital role in boosting the performance of speaker verification (SV) systems. Modern speaker embeddings mostly use deep neural networks (DNNs) to process frame-level acoustic feature vectors [1]–[4]. For instance, time delay neural networks (TDNNs), ResNets [5], DenseNets [6], and Res2Nets [7] have been widely used to extract the frame-level information in speaker embedding networks [3], [4], [8], [9]. Compared with the traditional i-vectors [10], DNN-based speaker embeddings are not only more speaker discriminative but also more robust to noise, reverberation, and domain mismatch [11]–[14]. In particular, the convolutional neural network (CNN) based speaker embedding has become the state-of-the-art due to its outstanding performance [15], [16].

The current speaker embedding extractors often share a similar structure: a CNN-based frame-level network, a pooling layer, and a fully-connected utterance-level network. Because the embedding network aims to produce fixed-dimensional embeddings from variable-length utterances, how to aggregate speaker information from frame-level representations into utterance-level embeddings is of significant importance.

One common aggregation strategy is to use channel-wise means and standard deviations of the last frame-level feature maps as the summarization of the whole utterance [3]. Due to the sharp dimensionality reduction of the frame-level features, some speaker information will inevitably be lost in the aggregation process, even though multiple heads [8], [17], [18] or higher-order statistics [19] are utilized. Another way to aggregate information is to enhance the mutual information between the frame-level features and the aggregated embeddings. In [20], a mutual information neural estimator (MINE) was introduced in the pooling layer so that more meaningful information can be preserved in the aggregated statistics. However, this method only shows marginal improvement over those without an MINE.

Besides using a limited number of statistics (e.g., means, standard deviations, etc.) for aggregation or explicitly regularizing the aggregated features for information preservation, we can perform aggregation from a Fourier perspective. This is also the objective of this paper.

A. Motivation

In [21], spectral pooling was proposed to replace max pooling for better information preservation in computer vision. This method involves three steps: 1) transforming the convolutional features from the spatial domain to the spectral domain by discrete Fourier transform (DFT), 2) cropping and retaining the low spectral components, and 3) performing inverse DFT on the cropped features to transform them back to the spatial domain. Because most of the spectral energy locates in the low-frequency region, spectral pooling is able to preserve most of the feature information by retaining the low spectral components.

However, because DFT can only be applied to deterministic or wide-sense stationary signals, it is not suitable for non-stationary speech signals [22]. To account for the non-stationarity of the convolutional feature maps in speaker embedding networks, short-time spectral pooling (STSP) was proposed in [23] by replacing DFT with short-time Fourier transform (STFT) [24]. It was shown in [23] that STSP is a generalized statistics pooling method. This is because

This work was supported by the RGC of Hong Kong SAR, Grant No. PolyU 152137/17E and National Natural Science Foundation of China (NSFC), Grant No. 61971371.

Y. Z. Tu and M. W. Mak are with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong SAR (email: 918tyz@gmail.com; enmwmak@polyu.edu.hk).

from a Fourier perspective, statistics pooling only exploits the DC (zero-frequency) components in the spectral domain, whereas STSP incorporates more spectral components besides the DC ones during aggregation and is able to retain richer speaker information. The experimental results on VoxCeleb1 also verify that STSP remarkably outperforms the statistics pooling method.

However, one limitation of STSP is that the brute average of the spectrograms along the temporal axis ignores the importance of individual windowed segments when computing the spectral representations. In other words, all segments in a specific spectrogram were treated with equal importance. In practice, however, this is not reasonable because phonetic information is rarely distributed uniformly across an utterance. As a result, different segments of an utterance have different speaker discriminative power. Therefore, it is unlikely that each segment contributes equally to the discrimination of speakers.

To address the above limitation of STSP, we propose applying self-attention [25] on the windowed segments for each spectrogram while computing the spectral representations. As a result, the discriminative segments can be emphasized during aggregation, which contributes to a more discriminative power for speaker embedding. We call the proposed method *attentive STSP* in this paper. Unlike the conventional attention mechanisms for speaker embedding that perform attention on temporal frames [8], [17], [18], [26], attentive STSP performs attention on individual windowed segments. Nevertheless, the rationale behind attentive STSP is the same as that of the conventional attentive pooling methods.

The intuition that exploiting the local stationarity in the convolutional feature maps is beneficial to utterance-level aggregation can be interpreted from a perspective of stochastic process. Specifically, if we consider a feature sequence at the final convolutional layer as a realization of a stationary stochastic process, its global statistics (e.g., mean, standard deviation, etc.) will not change with time. However, once the stationarity assumption is violated, which is common for the final convolutional feature maps, these global statistics will become unreliable for summarizing the process. This suggests that the performance of statistics pooling and its attentive variants would suffer more severely on long utterances because of the non-stationarity in the feature sequence. Therefore, the conventional pooling methods that operate in the temporal domain can be sensitive to the duration variations in the evaluation sets. On the other hand, attentive STSP is more robust to duration variations, attributed to its ability to handle the local stationarity in the feature maps.

In fact, it has been observed in portfolio optimization that exploiting only the mean and variance of a non-stationary sequence is not sufficient. In [27], the authors pointed out that although the mean-variance optimization (MVO) has long been an optimal strategy for investment, it presents poor out-of-sample performance due to the non-stationarity in the financial time series. To overcome the inherent time-varying property of the price series, a complex spectral portfolio method was proposed to model the cyclostationarity of the time series.

To make the contributions of attentive STSP clear, we summarized the novel parts of attentive STSP as follows:

1) Compared with spectral pooling in computer vision [21], attentive STSP has two advantages.
   - Attentive STSP uses STFT rather than DFT to transform the temporal feature maps to the spectral domain. The rationale of applying STFT is twofold. On the one hand, STFT exploits the local stationarity instead of the global stationarity (assumed by DFT) in the feature maps, which is reasonable for non-stationary speech signals. Exploring the local stationarity makes attentive STSP more resilient to the non-stationarity in the feature maps. On the other hand, with STFT, we can consider attentive STSP as a generalization of the conventional statistics pooling, which lays the foundation of attentive STSP for utterance-level aggregation. This generalization, however, cannot be done by applying DFT on the temporal feature maps.
   - Spectral pooling requires an inverse DFT to transform the truncated spectral components back to the spatial domain. In attentive STSP, however, the pooling operation is performed completely in the spectral domain and inverse STFT is not required, which reduces computation and facilitates the aggregation.
2) Compared with vanilla STSP [23], attentive STSP applies a self-attention mechanism to highlight the contribution of the discriminative windowed segments in the pooling operation. The attention mechanism endows attentive STSP with higher discriminative power, and it is a new part not covered by [23].

B. Related Work

Various pooling methods have been used for speaker embeddings. In [2], the channel-wise mean vectors of frame-level features were exploited in temporal pooling. In the x-vector extractor, both the means and the standard deviations are computed through a statistics pooling layer [3]. Compared with temporal pooling, statistics pooling shows remarkably better performance and has become a baseline pooling strategy. By simultaneously pooling over the features from different frame-level layers, the authors of [28] increased the number of aggregated statistics by multiple times. Inspired by the NetVLAD architecture [29] in computer vision, the authors of [30] proposed the learnable dictionary encoding (LDE) where the encoded vectors act like the means of a Gaussian mixture model (GMM). In [4], a NetVLAD layer was directly applied for utterance-level aggregation.

Another popular category is the attention-based pooling. For instance, an attention mechanism was introduced to weight the temporal frames so that the attended frames can substantially contribute to speaker discrimination [26]. To increase the representation capacity of the aggregated embeddings, multi-head attentive pooling was proposed to attend the convolutional features from multiple perspectives [17]. The authors

of [18] further extended this multi-head idea and diversified the attention heads by allowing different resolutions in the multiple heads. Different from [17] where each head attends the frame-level features across all channels, the authors of [31] applied each head to a subset of the channels. By integrating the attention mechanism and GMM clustering, the authors of [8] proposed a mixture of attentive pooling from a probabilistic perspective. Instead of performing attention across the frames, channel- and context-dependent statistics pooling [9] extends attention along the channel dimension to highlight the contribution of individual channels.

In [32], a joint time-frequency pooling was introduced for utterance-level aggregation. However, because frequency pooling along the frequency axis uses the same strategy as temporal pooling, it is different from our proposed method, where pooling is operated in the *Fourier transformed* domain obtained by applying STFT to the temporal feature maps.

This paper is organized as follows. In Section II, we briefly introduce the architecture of the speaker embedding network and several existing pooling methods. Section III details the principle of the proposed attentive STSP and clarifies its relationship with the previous work. The experimental settings and results are provided in Section IV and Section V, respectively. We then give conclusions in Section VI.

II. Speaker Embedding

In this paper, we investigate the proposed pooling method on the modified x-vector architecture. Statistics pooling [3], multi-head attentive pooling [17], and channel- and context-dependent statistics pooling [9] are used as the baseline pooling strategies for utterance-level aggregation.

A. Network Architecture

As illustrated in Table I, the configuration of the speaker embedding network used in this paper is almost identical to that of [3]. One difference is that the former uses a nonlinear dense layer of 256 nodes for utterance-level processing. Each TDNN layer aggregates several contextual frames from the previous layer, resulting in a temporal context of 15 acoustic frames at Layer 3. Another difference is that the additive margin softmax (AMSoftmax) [33] is used in the output layer. For each utterance, its speaker embedding vector is the affine output at Layer 7.

B. Pooling Strategy

1) Statistics Pooling: The default pooling method for x-vector is statistics pooling. Denote $\mathbf{H} = \{\mathbf{h}_t\}_{t=0}^{T-1} \in \mathbb{R}^{C\times T}$ as a feature map at the last convolutional layer, where C is the number of channels in the feature map $\mathbf{H}$ and T is the number of frames. $\mathbf{H}$ comprises a sequence of frame-level vectors fed to the pooling layer. The aggregated representation $\mathbf{z}$ is expressed as

$$\mathbf{z} = (\boldsymbol{\mu}, \boldsymbol{\sigma}), \tag{1}$$

where

$$\boldsymbol{\mu} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{h}_t, \tag{2}$$

TABLE I
Architecture of the speaker embedding network used in this work. BN represents batch normalization [34]. T denotes the total number of frames in an utterance and N_{spk} is the number of training speakers.

Layer	Layer type	Layer context	Total context	Output dimension
1	TDNN-BN-ReLU	$[t-2, t+2]$	5	512
2	TDNN-BN-ReLU	$\{t-2, t, t+2\}$	9	512
3	TDNN-BN-ReLU	$\{t-3, t, t+3\}$	15	512
4	Dense-BN-ReLU	$\{t\}$	15	512
5	Dense-BN-ReLU	$\{t\}$	15	1,500
6	Pooling	$[0,\ T)$	T	3,000
7	Dense-BN	$[0,\ T)$	T	256
8	AMSoftmax	$[0,\ T)$	T	N_{spk}

and

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{T}\operatorname{diag}\left(\sum_{t=0}^{T-1}\mathbf{h}_t\mathbf{h}_t^\top - \boldsymbol{\mu}\boldsymbol{\mu}^\top\right)}. \tag{3}$$

In (3), $\operatorname{diag}(\cdot)$ means constructing a vector using the diagonal elements of a square matrix and the square root is operated element-wise. In short, the aggregated representation $\mathbf{z}$ is the concatenation of channel-wise means and standard deviations of the feature map.

2) Multi-head Attentive Pooling: In [17], an attention mechanism with multiple heads was introduced to attend frame-level features from various perspectives. Let us consider an H-head attention network with a *tanh* hidden layer of D nodes and a linear output layer. The attention weight matrix $\mathbf{A} = (a_{t,h}) \in \mathbb{R}^{T\times H}$ can be computed as

$$\mathbf{A} = \operatorname{Softmax}\left(\tanh\left(\mathbf{H}^\top\mathbf{W}_1\right)\mathbf{W}_2\right), \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{C\times D}$ and $\mathbf{W}_2 \in \mathbb{R}^{D\times H}$ are trainable weight matrices and the softmax function is operated column-wise. For the h-th head ($h \in \{1, \ldots, H\}$), the attended mean and standard deviation vectors are computed as follows:

$$\boldsymbol{\mu}_h = \sum_{t=0}^{T-1} a_{t,h}\mathbf{h}_t, \tag{5}$$

and

$$\boldsymbol{\sigma}_h = \sqrt{\operatorname{diag}\left(\sum_{t=0}^{T-1} a_{t,h}\mathbf{h}_t\mathbf{h}_t^\top - \boldsymbol{\mu}_h\boldsymbol{\mu}_h^\top\right)}. \tag{6}$$

Finally, we have the aggregated vector as follows:

$$\mathbf{z} = \left(\boldsymbol{\mu}_1, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\mu}_H, \boldsymbol{\sigma}_H\right). \tag{7}$$

A major difference between statistics pooling and attentive pooling is that the latter scales the feature maps by an attention weight vector $\{a_{t,h}\}_{t=0}^{T-1}$ for each head h during the pooling process (see (5) and (6)). The purpose of the attention weight vector is to emphasize discriminative frames for information aggregation. Because multi-head attentive pooling has H independent attention weight vectors in $\mathbf{A}$, its capacity for information preservation is larger than that of single-head attentive pooling.

3) Channel- and Context-Dependent Statistics Pooling: Multi-head attentive pooling applies attention weights independently on channel-wise feature sequences, assuming that each channel has equal importance to the discriminative power of speaker embeddings. In [9], channel- and context-dependent statistics pooling (CCDSP) was proposed to account for the contribution of individual channels. To enable the attention network to take the utterance's global properties (such as noise or recording conditions) into consideration, we concatenate $\mathbf{h}_t$ in (2) with the global non-weighted mean $\boldsymbol{\mu}$ (see (2)) and standard deviation $\boldsymbol{\sigma}$ (see (3)) along the channel axis, i.e., $\tilde{\mathbf{h}}_t = [\mathbf{h}_t^\top, \boldsymbol{\mu}^\top, \boldsymbol{\sigma}^\top]^\top$. Then, we compute the attention scores as

$$e_{t,c} = \boldsymbol{v}_c^\top f\left(\boldsymbol{W}_3 \tilde{\mathbf{h}}_t + \boldsymbol{b}\right) + k_c, \quad c = 1, \ldots, C, \tag{8}$$

where $\boldsymbol{W}_3 \in \mathbb{R}^{D' \times 3C}$ and $\boldsymbol{b} \in \mathbb{R}^{D'}$ are the channel independent weight matrix and bias vector to be learned, respectively, and $f(\cdot)$ is a non-linear activation function. $\boldsymbol{v}_c \in \mathbb{R}^{D'}$ and $k_c \in \mathbb{R}$ are the learned channel-dependent weight vector and bias of the c-th channel, respectively. $e_{t,c}$ is then normalized along the frame dimension via a softmax function to compute the channel-dependent attention weights

$$a_{t,c}^{\mathrm{CD}} = \frac{\exp\left(e_{t,c}\right)}{\sum_{\tau=0}^{T-1} \exp\left(e_{\tau,c}\right)}. \tag{9}$$

Finally, the weighted mean vector $\boldsymbol{\mu}^{\mathrm{CD}} = \left\{\mu_c^{\mathrm{CD}}\right\}_{c=1}^{C}$ and the standard deviation vector $\boldsymbol{\sigma}^{\mathrm{CD}} = \left\{\sigma_c^{\mathrm{CD}}\right\}_{c=1}^{C}$ are concatenated to form the aggregated embedding, where

$$\mu_c^{\mathrm{CD}} = \sum_{t=0}^{T-1} a_{t,c}^{\mathrm{CD}}\, h_{t,c}, \tag{10}$$

and

$$\sigma_c^{\mathrm{CD}} = \sqrt{\sum_{t=0}^{T-1} a_{t,c}^{\mathrm{CD}}\, h_{t,c}^2 - (\mu_c^{\mathrm{CD}})^2}. \tag{11}$$

C. Additive Margin Softmax Loss

We used additive margin softmax loss [33] to train the embedding network:

$$\begin{aligned}\mathcal{L} &= -\frac{1}{N_{\mathrm{trn}}} \sum_{i=1}^{N_{\mathrm{trn}}} \log \frac{e^{s\left(\cos\theta_{y_i} - m\right)}}{e^{s\left(\cos\theta_{y_i} - m\right)} + \sum_{j=1, j\neq y_i}^{N_{\mathrm{spk}}} e^{s\cos\theta_j}} \\ &= -\frac{1}{N_{\mathrm{trn}}} \sum_{i=1}^{N_{\mathrm{trn}}} \log \frac{e^{s\left(\tilde{\mathbf{w}}_{y_i}^\top \mathbf{f}^i - m\right)}}{e^{s\left(\tilde{\mathbf{w}}_{y_i}^\top \mathbf{f}^i - m\right)} + \sum_{j=1, j\neq y_i}^{N_{\mathrm{spk}}} e^{s\,\tilde{\mathbf{w}}_j^\top \mathbf{f}^i}},\end{aligned} \tag{12}$$

where m and s denote the cosine margin and the scaling factor, respectively. $\mathbf{f}^i$ is the affine output of the Dense-BN layer (Layer 7) in Table I of the i-th training sample, and y_i is the corresponding speaker label. $\tilde{\mathbf{w}}_j$, which corresponds to the j-th output node, is the j-th column of the weight matrix $\widetilde{\mathbf{W}}$, i.e., $\widetilde{\mathbf{W}} = \{\tilde{\mathbf{w}}_j\}_{j=1}^{N_{\mathrm{spk}}}$.

III. Attentive Short-time Spectral Pooling

In this section, we propose attentive short-time spectral pooling (attentive STSP) as an extension of the STSP in [23]. The principle and the rationale of attentive STSP are detailed. Its relationships to conventional pooling methods are explained.

A. Methodology

Fig. 1 (a) shows the process of attentive STSP. Because the pooling layer sits between the frame-level subnetwork and the utterance-level subnetwork, spectral analysis is performed on the output feature maps of the last convolutional layer, not on the MFCCs or filter-bank features. Given the c-th channel feature $\mathbf{x}_c = \{x_c(t)\}_{t=0}^{T-1}$ of a convolutional feature map $\{\mathbf{x}_c\}_{c=1}^{C} \in \mathbb{R}^{C\times T}$, its short-time Fourier transform (STFT) [24] is expressed as follows:

$$X_c(n,k) = \sum_{t=0}^{T-1} x_c(t) w(t - nS) e^{-\frac{j2\pi}{L}kt}, \tag{13}$$

where $w(\cdot)$ is a window function of length L, S denotes the sliding step of the window, n indexes the temporal segments (sliding windows), and $k = 0, \ldots, L-1$ indexes the frequency components. Note that in this paper, *we always make sure that the STFT length (the length of Fourier transform during STFT) is equal to the window length L*. Equation (13) suggests that by sliding the window, we may apply multiple STFTs on a 1-D sequence to produce a 2-D spectral feature map with a temporal index n and a frequency index k for each channel.

It is necessary to compute the spectral representation for each channel. Rather than brutely average the windowed segments within the spectrogram $|X_c(n,k)|$, we apply a weighted average of these segments by an H-head attention weight matrix and obtain a spectral sequence for each head as follows:

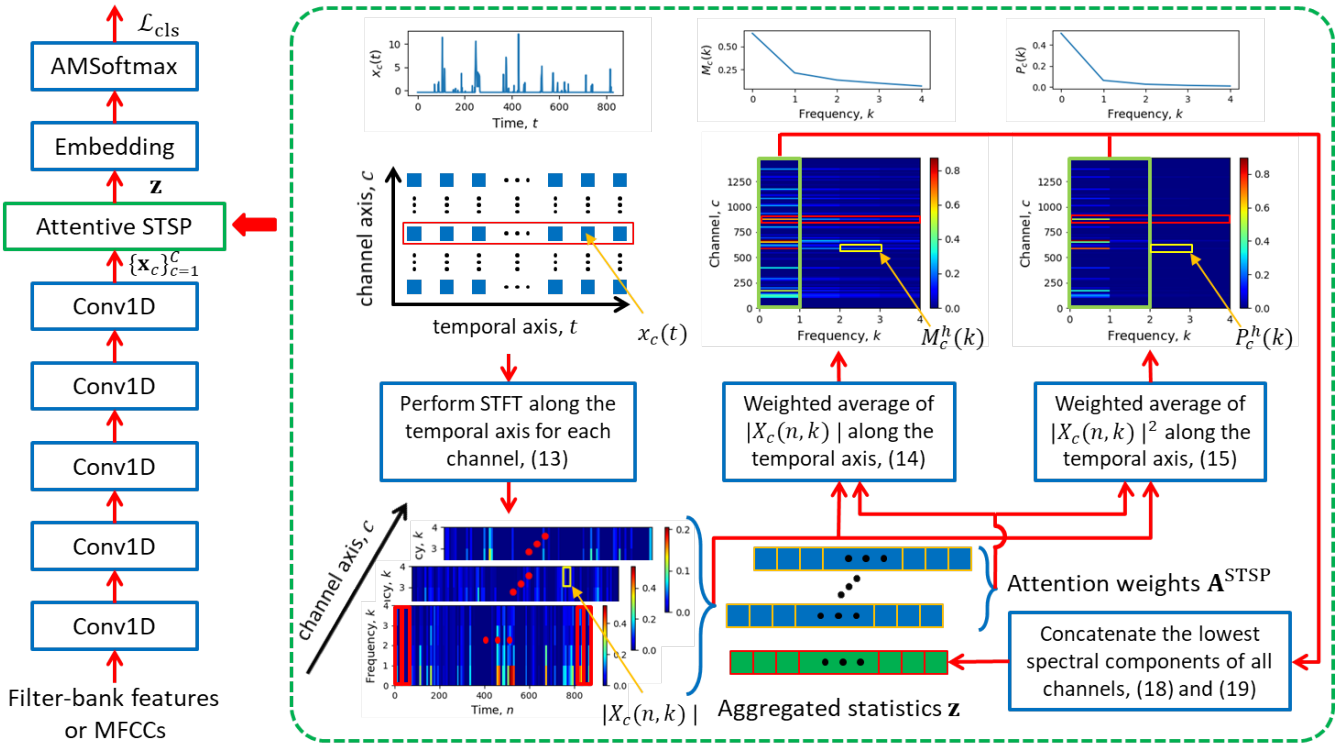
$$M_c^h(k) = \sum_{n=0}^{N-1} \alpha_n^h \left|X_c(n,k)\right|, \quad k = 0, \ldots, L-1 \tag{14}$$

where $N = \mathrm{floor}((T-L)/S) + 1$ is the number of windowed temporal segments, $h \in [1, H]$ indexes the attention heads, and $\boldsymbol{\alpha}^h = \{\alpha_n^h\}_{n=0}^{N-1}$ denotes the attention weight vector corresponding to head h. Similarly, if we attend to the square of the spectrogram, we obtain the second-order spectral statistics:

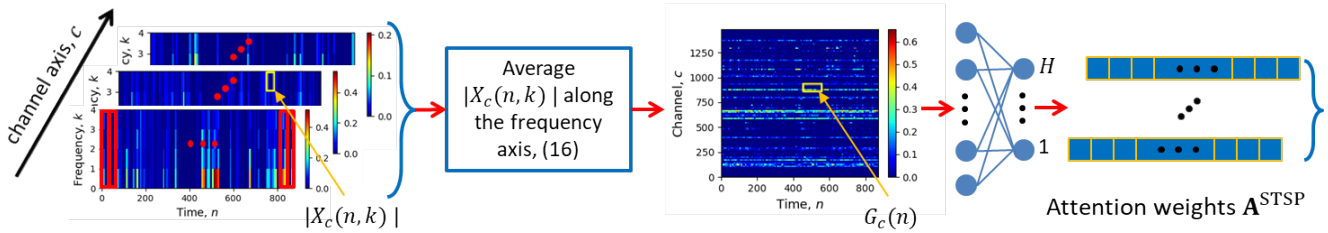
$$P_c^h(k) = \sum_{n=0}^{N-1} \alpha_n^h \left|X_c(n,k)\right|^2, \quad k = 0, \ldots, L-1. \tag{15}$$

The attention mechanism is shown in Fig. 1(b). Note that the attention process is operated on the windowed segments within each spectrogram. We first average the spectrogram for each channel along the frequency axis to make sure that all the spectral components within a specific segment share the same attention weights. The resulting feature map is denoted as $\mathbf{G} = \{G_c(n)\}_{c=1}^{C} \in \mathbb{R}^{C\times N}$, where

$$G_c(n) = \frac{1}{L} \sum_{k=0}^{L-1} \left|X_c(n,k)\right|, \quad n = 0, \ldots, N-1. \tag{16}$$



(a)



(b)

Fig. 1. (a) Schematic of attentive short-time spectral pooling. The left part depicts the signal flow within the embedding network, whereas the right part details the pipeline of attentive STSP. In the green dashed box, the left-most graph in the second row illustrates a temporal feature map extracted from the last convolutional layer. The bottom-left spectrograms were produced by STFT with length $L = 8$, and the vertical red boxes on top of the spectrogram denote spectral arrays to be averaged along the time axis by an attention weight matrix. The actual values of $\boldsymbol{M}_c^h$ and $\boldsymbol{P}_c^h$ (only $h = 1$ is considered here) after applying (14) and (15) are shown in the middle and the right-most maps in the second row, respectively. The top three plots correspond to the row vectors with elements $x_c(t)$, $M_c^1(k)$, and $P_c^1(k)$ in the red boxes, respectively. All the spectral features in the green boxes in the second row are concatenated to form the final utterance-level statistics (see (18) and (19) for details). (b) Schematic of the attention mechanism used in attentive STSP. The middle feature map denotes the actual value of $\mathbf{G}$ and the node graph illustrates an H-head attention network. The attention weight matrix $\mathbf{A}^{\text{STSP}}$ is computed as in (17).

Similar to (4), the attention weight matrix $\mathbf{A}^{\text{STSP}} = \{\boldsymbol{\alpha}^h\}_{h=1}^H$ is computed as follows:

$$\mathbf{A}^{\text{STSP}} = \text{Softmax}\left(\tanh\left(\mathbf{G}^\top \mathbf{W}_1^{\text{STSP}}\right)\mathbf{W}_2^{\text{STSP}}\right), \quad (17)$$

where $\mathbf{W}_1^{\text{STSP}} \in \mathbb{R}^{C\times D}$ and $\mathbf{W}_2^{\text{STSP}} \in \mathbb{R}^{D\times H}$ are trainable weight matrices.

During aggregation, we concatenate $M_c^h(0)$ and the square roots of the lowest R components of $P_c^h(k)$ to form the utterance-level representation of channel c for head h:

$$\mathbf{z}_c^h = \left(M_c^h(0), \sqrt{P_c^h(0)}, \ldots, \sqrt{P_c^h(R-1)}\right). \quad (18)$$

The final utterance-level feature is produced by concatenating the spectral statistics of all channels and all heads:

$$\mathbf{z} = \left(\mathbf{z}_1^1, \ldots, \mathbf{z}_c^h, \ldots, \mathbf{z}_C^H\right). \quad (19)$$

B. Rationale and Validity of Attentive STSP

As mentioned in Section I-A, to facilitate the aggregation process, spectral pooling requires that the energy of the features should concentrate in the low-frequency region. To demonstrate that attentive STSP also satisfies this requirement, we provide empirical evidences by plotting the statistics of the spectral representations computed from (14) and (15). The procedure for computing the spectral representations is as follows:

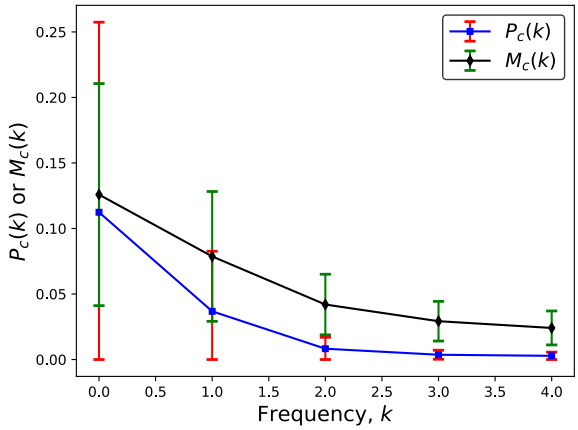


Fig. 2. Statistics of $M_c^h(k)$ in (14) and $P_c^h(k)$ in (15) of a randomly selected channel c with respect to the frequency components k's under $H = 1$. Black diamonds and blue squares denote the means of $M_c^1(k)$ and $P_c^1(k)$ over 24,220 utterances in the VoxCeleb1 development set, respectively. The error bars represent one standard deviation. The STFT length L for computing $X_c(n, k)$ in (13) was set to 8. Only the left half of $M_c^1(k)$ and $P_c^1(k)$ ($0 \leq k \leq 4$) are plotted due to the symmetry of spectrograms along the frequency axis.

1) Randomly select 20 utterances from each of the 1,211 speakers in the VoxCeleb1 development set.
2) Extract 40-dimensional filter-bank features from the selected utterances and perform mean normalization with a sliding window of 3 seconds.
3) Train an embedding system with a single-head attentive STSP layer ($H = 1$) using 5,984 speakers from the Voxceleb2 development set.
4) Extract the feature maps from the last convolutional layer of the embedding network.
5) Compute the spectral representations $M_c^1(k)$ and $P_c^1(k)$ according to (14) and (15), respectively.

The training procedure of the embedding network is detailed in Section IV-A.

Fig. 2 shows the statistics of $M_c^1(k)$ and $P_c^1(k)$ over 24,220 utterances. We observe that both the $M_c^1(k)$ and $P_c^1(k)$ of a randomly selected channel have most of their energy concentrated in the low-frequency region. This validates the feasibility of using attentive STSP for utterance-level aggregation. Attributed to the desirable statistics of $M_c^h(k)$ and $P_c^h(k)$ in the spectral domain, attentive STSP uses the lowest spectral components for aggregation.[1]

The property that most of the energy of the convolutional features concentrates in the low-frequency part in the spectral domain also reflects that the frame-level network is a low-pass filtering system. In [36], Rahaman *et al.* interpreted the generalization of DNNs [37], [38] from a Fourier perspective and revealed a learning bias of DNNs towards low-frequency functions (spectral bias). Although there is no exact clue that the low-pass characteristic of the speaker embedding network is completely attributed to the spectral bias of CNNs, we believe that this bias at least contributes partially to the low-pass property of the frame-level networks. On the other hand, in both temporal pooling [2] and statistics pooling [3], global averaging is used to extract the mean vector of the whole temporal features. In fact, global averaging can be seen as mean filtering with a global kernel [39], which is a low-pass filtering operation. Therefore, the pooling methods in [2] and [3] have already *implicitly* exploited the low-pass characteristic of the CNNs, although they only use the DC components of the spectral representations. Similar to the vanilla STSP, the proposed attentive STSP *explicitly* explores the low-pass filtering effect and improves these pooling strategies by accounting for more spectral components besides the DC ones. Thus, attentive STSP preserves more speaker information than the conventional statistics pooling during aggregation.

Note that for the k-th spectral component ($k > 1$), because $M_c^h(k)$ and $P_c^h(k)$ are both related to the k-th frequency, the information in $M_c^h(k)$ and $P_c^h(k)$ will be correlated. From Fig. 2, we observe that $P_c^1(k)$ decays faster than $M_c^1(k)$ and are more energy-concentrated towards the zero frequency. We hypothesize that using $P_c^h(k)$ can be more effective than using $M_c^h(k)$ alone (see Section V-E for details). Taking this observation into account and to avoid using repeated information, we only use $\sqrt{P_c^h(k)}$ ($k > 1$) in (18) for aggregation.

C. Relation to Previous Works

Attentive STSP is a generalized STSP in that if we apply equal attention weights produced from a single-head attention network on the windowed segments in (14) and (15), i.e., $\alpha_n^1 = 1/N$ for $n \in [1, N]$, attentive STSP reduces to the vanilla STSP in [23].

Similar to STSP, attentive STSP also generalizes statistics pooling. Under the condition where single-head attention is implemented and equal attention weights are applied, if we set $k = 0$ and use a rectangular window without any overlap between successive segments (i.e., $S = L$) in (14), the DC component $M_c^1(0) = \frac{1}{N}\sum_{t=0}^{NL-1} x_c(t)$ approximates the mean of $\mathbf{x}_c$ multiplied by a scaling factor L.[2] On the other hand, setting $k = 0$ in (15) resembles computing the power of $\mathbf{x}_c$. In the extreme case where $S = L = 1$, we have $P_c^1(0) = \frac{1}{T}\sum_{t=0}^{T-1} (x_c(t))^2$. This means that under these conditions, using the means and standard deviations in statistics pooling is an analogy to using the DC components ($k = 0$ in (14) and (15)) in attentive STSP. Therefore, attentive STSP can be seen as a generalized statistics pooling method.

Attentive STSP has a close relationship with multi-head attentive pooling [17] because they both apply an attention mechanism during aggregation. However, there are two major differences between these two methods. Firstly, as shown in (17), attentive STSP performs attention on a series of *windowed segments* in $\mathbf{G}$, whereas multi-head attentive pooling

[1]According to Parseval's theorem [35], the energy of a signal in the temporal domain is equal to that in the spectral domain.

[2]In fact, the DC component should be $M_c^1(0) = \frac{1}{N}|\sum_{t=0}^{NL-1} x_c(t)|$ under $S = L$ according to (14). However, in this paper, because each convolutional layer is followed by an ReLU layer in the speaker embedding network (see Table I), all the elements of the input feature $\mathbf{x}_c$ will be non-negative. Thus, we have $M_c^1(0) = \frac{1}{N}|\sum_{t=0}^{NL-1} x_c(t)| = \frac{1}{N}\sum_{t=0}^{NL-1} x_c(t) > 0$.

implements an attention network on a sequence of *frames* as in (4). Because the spectral components in each segment is (locally) stationary, segment-level attention can provide attenive STSP with better robustness against the non-stationarity in the feature maps than frame-level attention. Secondly, attentive STSP further preserves the speaker information by retaining the *informative* spectral components only. Note that not all the components in the spectral domain are beneficial for aggregation. Specifically, incorporating high-frequency components can cause detrimental effect to the speaker embeddings because these components are very noisy. In contrast, because multi-head attentive pooling takes all the temporal frames into account, it always includes all the spectral information during aggregation (due to the equivalence of information between the temporal domain and the spectral domain). Therefore, attentive STSP is advantageous to multi-head attentive pooling in information distillation.

Note that the windowed segment attention in attentive STSP is different from the sliding-window attention in [40] and [41], although both attention mechanisms involve the term "window." In particular, the segment-level attention in this paper is operated on the windowed segments to account for the local stationarity of the temporal feature maps. The attention mechanism aims to learn the *global* relationships across all of the windowed segments in an utterance. In contrast, the sliding-window attention takes a series of tokens (equivalent to frames in speaker verification) as input and only models the *local* relationships of the tokens within each sliding window. The objective is to reduce computation relative to the full attention [25]. Therefore, these two methods differ completely in their inputs, operating mechanisms, and objectives.

Interestingly, attentive STSP is also related to the modulation spectrum of speech [42], [43] because the spectral representations in attentive STSP and modulation spectrum are both produced from spectrograms. However, due to the differences in the input, the way to produce the spectrograms, and the strategy to compute the spectral representations, attentive STSP differs substantially from modulation spectrum. First, attentive STSP is operated on the output feature maps at the last convolutional layer of a speaker embedding network, whereas modulation spectrum takes speech signals as input. Second, attentive STSP applies STFT to perform time-frequency transformation, whereas filter-bank analysis is typically adopted for computing the spectrograms in modulation spectrum. Third, to compute modulation spectra, handcrafted bandpass filtering is often applied to the spectrograms, e.g., a linear filter is applied to the log-transformed spectrograms in RASTA processing [44]. In contrast, we compute $M_c^h(k)$'s and $P_c^h(k)$'s through a weighted average of the spectrogram and its square.

IV. Experimental Setup

Five pooling methods are compared in this paper, i.e., statistics pooling [3], multi-head attentive pooling [17], channel- and context-dependent statistics pooling [9], STSP [23], and the proposed attentive STSP. We evaluated the performance of these pooling methods on the VoxCeleb1 test set (clean) [15], the VOiCES 2019 evaluation set [45], the SRE16 evaluation set [46], and the SRE18-CMN2 evaluation set [47].

A. Training of Speaker Embedding Extractor

For the evaluation on VoxCeleb1, only the VoxCeleb2 development subset (approximate 2 million utterances from 5,984 speakers) was used for training. Whereas both VoxCeleb1 development and VoxCeleb2 development data were used as the training set for VOiCES 2019, which amounts to about 2.1 million utterances from 7,185 speakers. We followed the Kaldi's VoxCeleb recipe to prepare the training data, i.e., using 40-dimensional filter bank features, performing energy-based voice activity detection, implementing augmentation (by adding reverberation, noise, music and babble to the original speech files), applying cepstral mean normalization with a window of 3 seconds, and filtering out utterances with a duration less than 4 seconds.[3] Totally, we had approximately twice the number of clean utterances for training the embedding network.

For both SRE16 and SRE18-CMN2 evaluations, we followed the Kaldi's SRE16 recipe to prepare the training data.[4] Instead of using the 40-dimensional filter bank features, 23-dimensional MFCCs were used for training. The training set consists of SRE04–10, Mixer 6, Switchboard Cellular, and Switchboard 2 (all phases). Totally, we had 238,618 utterances from 5,402 speakers in the training set.

We used the architecture in Table I to implement the statistics pooling baseline. For systems that use multi-head attentive pooling, we used an attention network with 500 *tanh* hidden nodes and H linear output nodes, where H is the number of attention heads (see (7)). We used $D' = 256$ in CCDSP (see 8) and the non-linearity is *tanh*. For STSP, we used a rectangular window function with length $L = 8$ and $S = L$. The attention network in attentive STSP follows the structure of multi-head attentive pooling and uses various window functions with length L ranging from 4 to 16. The step size S of each windowed segment varies from $L/4$ to L.

The additive margin softmax loss [33] was used for training. The additive margin m and the scaling factor s in (12) were set to 0.25 and 30, respectively. The mini-batch size was set to 128 for all evaluation tasks. There are around 2,337 mini-batches in one epoch for VoxCeleb1 and VOiCES 2019, and 4,220 mini-batches for SRE16 and SRE18-CMN2. Each mini-batch was created by randomly selecting speech chunks of 2–4s from the training data. We used a stochastic gradient descent (SGD) optimizer with a momentum of 0.9. The initial learning rate was 0.02 and it was linearly increased to 0.05 at Epoch 20. After that, it was decayed by half at Epochs 50, 80 and 95, respectively. Totally, the networks were trained for 100 epochs. Once training was completed, the speaker embedding was extracted from the affine output at Layer 7 in Table I.

B. PLDA Training

We used a Gaussian PLDA backend [48] for all evaluations. For VoxCeleb1, the PLDA model was trained on the speaker

[3]https://github.com/kaldi-asr/kaldi/tree/master/egs/voxceleb/v2.

[4]https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2.

embeddings extracted from the clean utterances in the training set for the embedding network. For VOiCES 2019, we trained the backend on the concatenated speech with the same video session and used utterances augmented with reverberation and noise. The PLDA training data for both SRE16 and SRE18-CMN2 were the embedding network's training set excluding the Switchboard part. Before PLDA training, the speaker embeddings were projected onto a 200-dimensional space by LDA for VoxCeleb1 and 150-dimensional space for VOiCES 2019, SRE16, and SRE18-CMN2, followed by whitening and length normalization. The LDA projection matrix was trained on the same dataset as for training the PLDA models. For VOiCES 2019, SRE16 and SRE18-CMN2, we also applied adaptive score normalization [49]. The cohort for VOiCES 2019 was selected from the longest two utterances of each speaker in the PLDA training data; whereas for SRE16 and SRE18-CMN2, the cohort was the respective unlabeled development set.

V. Results and Discussions

A. Performance on Various Evaluations

The performance was evaluated in terms of equal error rate (EER) and minimum detection cost function (minDCF) with $P_{\text{target}} = 0.01$. Table II shows the performance of different systems on VoxCeleb1 (clean), VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval. We can observe that all the pooling methods outperform the statistics pooling baseline. For attentive pooling, STSP and attentive STSP, we have the following analyses.

1) Multi-head attentive pooling: For all evaluation tasks, attentive pooling (Rows 2–5) achieves the best performance when the number of heads H was set to 2. When H further increases to 4, attentive pooling exhibits performance degradation, especially on SRE16 and SRE18-CMN2. Among many possibilities, the performance degradation can be caused by an increased number of non-stationary attention weight vectors produced by the attention network.

Take VoxCeleb1 for example, as shown in the first row of Fig. 3(a), the feature sequence ($\{h_{c,t}\}_{t=0}^{T-1}$ in (5)) presents a high non-stationarity along the temporal axis. To fit the drastic variations of the sequence, the attention network is trained to produce attention weights of large variations, as evident by the second row of Fig. 3(a). However, due to the substantial variations within the weight vectors, it is difficult for the attention network to generalize well on unseen utterances. Therefore, the non-stationarity of the attention weights could remarkably affect the performance of attentive pooling. On the other hand, a larger H does not necessarily indicate a larger degree of diversity in the attended feature sequences. For example, as shown in the third row of Fig. 3(a), the attended frames by Head 1 largely overlap those by Head 0. On the contrary, increasing the number of attention heads may introduce a larger degree of non-stationarity in the attention weights, causing poorer generalization to unseen data.

For SRE16-eval and SRE18-CMN2-eval, because the utterances in the evaluation sets are much longer than those in VoxCeleb1,[5] the degree of non-stationarity in the attention weights will be larger than that of the VoxCeleb1 test set. Thus, the performance degradation on SRE is severer than that on VoxCeleb1 and VOiCES 2019.

2) Channel- and Context-Dependent Statistics Pooling: We evaluated the performance of CCDSP with and without the global context vector ($\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, see Section II-B3). As observed in Rows 6–7, including the global statistics in CCDSP does not make a remarkable difference in performance. Although CCDSP achieves comparable performance with attentive pooling and outperforms the statistics pooling baseline, it cannot compete with STSP and attentive STSP on VOiCES 2019 and SREs.

3) STSP: From Rows 8–10 of Table II, we see that STSP achieves a consistent improvement in performance when the retained number of low-frequency components R in $P_c^h(k)$'s is increased from 1 to 3. However, further including the 4-th component will slightly degrade the performance, as can be seen in Row 11. This may be because there are more noises in the higher-frequency components. As demonstrated in [23], the magnitude of the spectral components $P_c(k)$'s in STSP approaches 0 when k becomes large. This suggests that we can hardly learn useful information from these vanishing components. Instead, the high frequency components can bring unwanted noise to the network during learning.

Comparing Rows 8–11 and Rows 2–5, we observe that STSP achieves similar performance as that of attentive pooling on VoxCeleb1 and VOiCES19-eval, but STSP remarkably outperforms attentive pooling on both the SRE evaluations. In fact, although both attentive pooling and STSP aim at preserving speaker information during aggregation, they fulfill the task from different perspectives. Specifically, attentive pooling emphasizes on discriminative *temporal frames* to enhance the information in the aggregated embeddings, whereas STSP emphasizes discriminative *spectral components*. Due to the duality of Fourier transform, the information in the temporal domain is equal to that in the spectral domain. This suggests that there should be little difference between attentive pooling and STSP, as can be verified by the comparison on VoxCeleb1. However, for long utterances, because of the high non-stationarity in the convolutional features (as analyzed in Section V-A1), it can be difficult to extract discriminative information in the temporal domain. In contrast, the spectral components in STSP are smoother, making STSP more robust against the non-stationary feature maps, especially for long utterances as in the SRE evaluations. Therefore, STSP can achieve larger performance gain over attentive pooling on both SRE tasks.

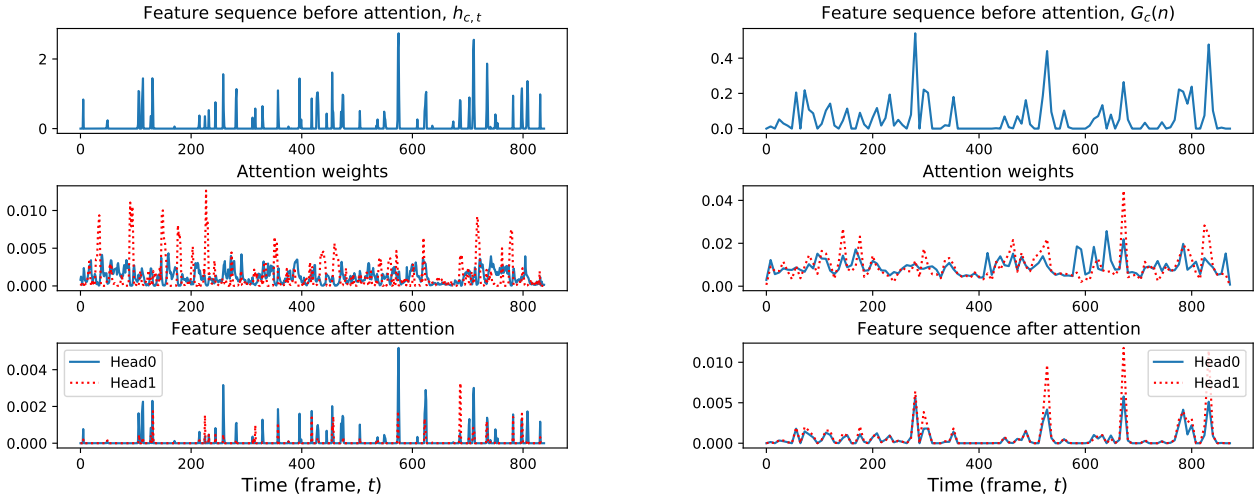
4) Attentive STSP: As shown in Rows 12–17 of Table II, the best performance of attentive STSP is achieved under $R = 2$ and $H = 1$. Compared with attentive pooling (Rows 2–5), a major advantage of attentive STSP is that because the attention mechanism is operated on the windowed segments instead of the frames, the produced attention weight vectors are much smoother than those by attentive pooling. This can be seen by

[5]The enrollment utterances of SRE16-eval and SRE18-CMN2-eval are approximate 60 seconds and the test utterances are 10–60s [46], [47], whereas the average duration of VoxCeleb1 is 8.2 seconds [15].

TABLE II

PERFORMANCE ON VOXCELEB1, VOICES19-EVAL, SRE16-EVAL, AND SRE18-CMN2-EVAL. CCDSP REFERS TO CHANNEL- AND CONTEXT-DEPENDENT STATISTICS POOLING (SEE SECTION II-B3). RECTANGULAR WINDOW FUNCTIONS WERE USED FOR STSP AND ATTENTIVE STSP UNDER $L = S = 8$. H DENOTES THE NUMBER OF HEADS IN MULTI-HEAD ATTENTIVE POOLING (SEE (7)) AND ATTENTIVE STSP (SEE (19)), AND R IS THE NUMBER OF THE LOWEST SECOND-ORDER SPECTRAL COMPONENTS IN STSP AND ATTENTIVE STSP (SEE (15).

Row	Pooling method	VoxCeleb1 EER	VoxCeleb1 minDCF	VOiCES19-eval EER	VOiCES19-eval minDCF	SRE16-eval EER	SRE16-eval minDCF	SRE18-CMN2-eval EER	SRE18-CMN2-eval minDCF
1	Statistics pooling	2.08	0.225	5.78	0.498	8.07	0.499	7.63	0.475
2	Multi-head attentive pooling (H=1)	1.97	0.224	5.69	0.472	8.11	0.480	7.32	0.467
3	Multi-head attentive pooling (H=2)	1.89	**0.206**	**5.41**	**0.454**	**7.61**	**0.472**	**6.89**	**0.454**
4	Multi-head attentive pooling (H=3)	**1.86**	0.219	5.53	0.461	8.02	0.475	6.90	0.457
5	Multi-head attentive pooling (H=4)	2.04	0.228	5.61	0.467	8.39	0.484	7.41	0.469
6	CCDSP w/o context	1.92	**0.209**	5.51	0.450	7.78	**0.475**	7.11	0.458
7	CCDSP w/ context	**1.88**	0.211	**5.46**	**0.441**	**7.69**	0.479	**7.02**	**0.457**
8	Vanilla STSP (R=1)	2.12	0.229	5.66	0.467	7.11	0.474	6.79	0.465
9	Vanilla STSP (R=2)	1.92	**0.210**	5.44	0.448	7.07	0.468	6.68	0.457
10	Vanilla STSP (R=3)	**1.87**	0.213	**5.30**	**0.443**	**6.77**	**0.460**	**6.65**	**0.443**
11	Vanilla STSP (R=4)	1.89	0.226	5.52	0.461	6.90	0.473	6.75	0.451
12	Attentive STSP (R=1, H=1)	1.89	0.211	5.40	0.453	6.65	0.469	6.32	0.453
13	Attentive STSP (R=2, H=1)	**1.76**	**0.193**	**5.03**	**0.396**	**6.30**	**0.441**	6.22	**0.434**
14	Attentive STSP (R=3, H=1)	1.81	0.195	5.13	0.415	6.40	0.450	**6.20**	0.437
15	Attentive STSP (R=4, H=1)	1.83	0.220	5.28	0.455	6.47	0.451	6.24	0.445
16	Attentive STSP (R=2, H=2)	1.89	0.207	5.34	0.444	6.50	0.458	6.39	0.441
17	Attentive STSP (R=2, H=3)	1.96	0.238	5.63	0.461	6.76	0.467	6.57	0.448



(a) (b)

Fig. 3. Illustration of the mechanism in (a) multi-head attentive pooling and (b) attentive STSP under H=2. We used a rectangular window function for attentive STSP under $L = S = 8$. Both the embedding networks were trained on the Voxceleb2 development data (see Section IV-A). For multi-head attentive pooling, the feature sequence ($\{h_{c,t}\}_{t=0}^{T-1}$ in (5)) in the first row corresponds to an utterance randomly selected from the VoxCeleb1 development set. For attentive STSP, the feature sequence is a random row vector in $\mathbf{G}$ of (16). Note that the unit in the horizontal axis is the frame index t in (5) and (13).

comparing the second rows of Fig. 3(a) and Fig. 3(b). In fact, it is natural to obtain smoother attention weights by segment-level attention because the coarse-grained attention mechanism has taken the local stationarity in the feature sequences into account. After all, exploiting the local stationarity in the frame-level features is a fundamental difference between the STSP-based pooling and the spectral pooling in [21]. On the other hand, as explained in Section V-A3, performing aggregation on long utterances in the spectral domain is superior to that in the temporal domain. Because of the segment-level attention and the spectral aggregation, attentive STSP obtains substantially better performance than attentive pooling, especially on the SRE16 and SRE18-CMN2.

Compared with STSP, the extra attention mechanism in attentive STSP can emphasize discriminative segments for information aggregation, leading to more discriminative speaker embeddings. This is why attentive STSP outperforms STSP on all the evaluation tasks, which verifies the motivation of this paper. An interesting observation is that different from STSP which achieves the best performance under $R = 3$, attentive STSP performs the best when $R = 2$. This indicates that the spectral energy of attentive STSP are more concentrated at the low-frequency region than the STSP, which further facilitates the aggregation process.

We also investigated the effect of the number of heads H on the performance of attentive pooling. Comparing Row 13 and Rows 16–17 in Table II, we see that increasing H does not offer any performance improvement on all evaluations. As illustrated in the third row of Fig. 3(b), the sequences after a two-head attention operation are almost the same. This suggests that more attention heads do not necessarily create richer diversity of the attended features. Instead, a larger H can introduce noises to the pooling operation because of the non-stationarity in the attention weights, as in attentive pooling in Section V-A1. If not stated otherwise, in the rest of the paper, we only used a single-head attention network for attentive STSP, i.e., $H = 1$.

B. Impact of Window Functions

In (13), a window function is applied to each temporal segment before performing DFT. To investigate the effect of the window function on performance, we implemented attentive STSP with the rectangular window, the Hanning window, and the Hamming window [50]. The performance is compared under $L = S = 8$ and $R = 2$.

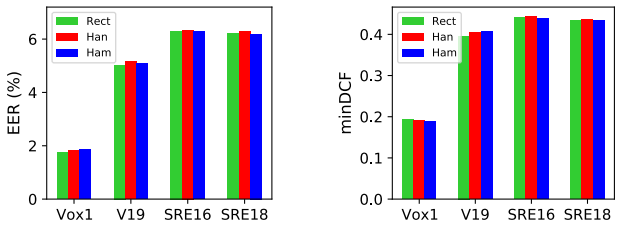
As shown in Fig. 4(a) and Fig. 4(b), there is no significant difference in the performance of the three windows. We have also tried other configurations by varying R and L, but the results are almost the same. These suggest that attentive STSP is not sensitive to the window function.

C. Impact of STFT Length

In Section III-A, we used STFT to exploit the local stationarity of temporal features for aggregation. Although each frame at the output of the last convolutional layer (Layer 5 in Table I) contains the information of 15 speech frames, we cannot guarantee that the CNN's outputs are locally stationary. Because it is difficult to quantify the degree of local stationarity in the convolutional feature maps, we varied the STFT length L to investigate its influence on the performance of STSP. In the following experiments, the step size S of the window function is equal to L.

As shown in Figs. 5(a)–5(h), attentive STSP consistently achieves the best performance when $L = 8$ on all evaluation tasks. When L further increases to 16, the performance degrades in most cases, especially on SRE16-eval and SRE18-CMN2-eval. We hypothesize that the performance degradation is caused by the violation of the local stationarity required by STFT. When the STFT length approaches 16, the local stationarity of STFT may not hold and thus it would be difficult to obtain effective local information. Another disadvantage of using $L = 16$ is that because there are more spectral components in the frequency domain than those under $L = 8$, we need a larger R to include sufficient speaker information in the aggregated embeddings. This is not favorable for aggregation. Therefore we did not account for the case where L is larger than 16.

Interestingly, the best results under $L = 4$ are comparable with those under $L = 8$ for VoxCeleb1, VOiCES19-eval, and SRE18-CMN2-eval. However, on SRE16-eval, the setting of $L = 8$ remarkably outperforms the case of $L = 4$. Although the local stationarity is largely satisfied under $L = 4$, there are insufficient components to hold speaker information in the spectral domain. Note that when $L = 4$, we can only have 3 spectral components in $P_c^h(k)$ because of the symmetry in STFT spectrograms.



(a) (b)

Fig. 4. (a) EER and (b) minDCF of attentive STSP on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval with respective to various window functions under $L = S = 8$, $R = 2$, and $H = 1$.

From the above analysis, the configuration of $L = 8$ makes a compromise between the local stationarity and the spectral resolution. This is the reason we used $L = 8$ in Section V-A and Section V-B.

D. Impact of Step Size

The step size S of windowed segments determines the degree of overlapping between successive segments and the number of segments in a temporal feature sequence. Because these factors can affect the results of STFT, which in turn affects the performance of STSP. To investigate the impact of step size on performance, we fixed the STFT length to 8 and varied S under $R = 2$.

As shown in Fig. 6(a) and Fig. 6(b), the step size does not have a substantial impact on the performance of attentive STSP across all the evaluations. This means that attentive STSP is not sensitive to the step size of the sliding window. However, given fixed L, because a larger S results in a smaller number of windowed segments for a fixed-length feature sequence, the subsequent computational load of the spectral representations will be reduced. Therefore, it is favorable to use $S = L$ in attentive STSP to reduce the computational cost. This is the reason why we used $S = L$ in the former sections.

E. Effect of $M_c^h(k)$ and $P_c^h(k)$

$M_c^h(k)$ in (14) and $P_c^h(k)$ in (15) denote the weighted average of the magnitude and energy of the spectrogram along the temporal axis, respectively. A noteworthy observation is that, as shown in Fig. 2, $P_c^h(k)$'s contain more energy in the low frequency components than $M_c^h(k)$'s do.[6] This phenomenon can also be observed from the rightmost two plots in the middle row of Fig. 1(a), where the number of salient components of $P_c^h(k)$ is smaller than that of $M_c^h(k)$ for all channels. Based on the observation from both figures, we may ask a question: *Is $P_c^h(k)$ more effective than $M_c^h(k)$ for attentive STSP due to its more energy-concentrated property?*

[6]The magnitude of $P_c^h(k)$'s presents a faster attenuation to zero than $M_c^h(k)$'s.

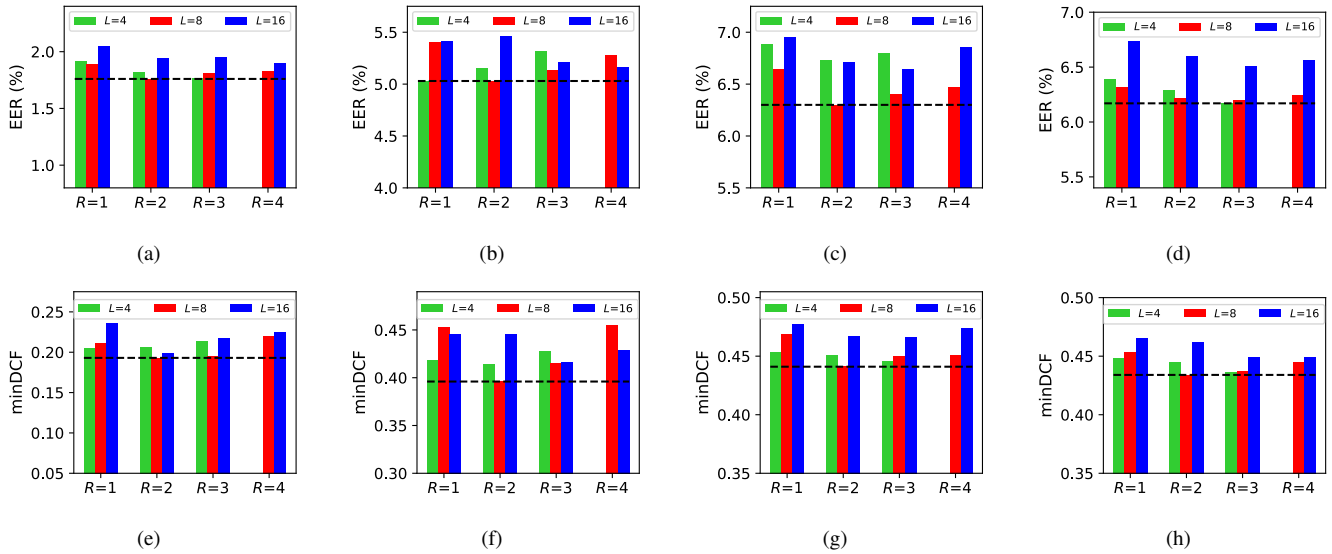


Fig. 5. Performance of attentive STSP on (a) and (e) VoxCeleb1, (b) and (f) VOiCES19-eval, (c) and (g) SRE16-eval, and (d) and (h) SRE18-CMN2-eval with various STFT lengths under the setting $H = 1$ and $L = S$, where L and S are the STFT length and the step size of the sliding window, respectively. The black dashed line indicates the best result in the individual subfigure.

TABLE III
COMPARISON OF ATTENTIVE STSP (ROWS 1–4) AND ITS MODIFIED VERSION (ROWS 5–9) WITH RESPECTIVE TO THE PERFORMANCE ON VOXCELEB1, VOICES19-EVAL, SRE16-EVAL, AND SRE18-CMN2-EVAL. R IS THE NUMBER OF THE LOWEST SECOND-ORDER SPECTRAL COMPONENTS ($P_c^h(k)$) IN (15). THE NUMBER OF ATTENTION HEADS WAS SET TO $H = 1$.

Row	Pooling method	VoxCeleb1 EER	VoxCeleb1 minDCF	VOiCES19-eval EER	VOiCES19-eval minDCF	SRE16-eval EER	SRE16-eval minDCF	SRE18-CMN2-eval EER	SRE18-CMN2-eval minDCF
1	Attentive STSP (R=1)	1.89	0.211	5.40	0.453	6.65	0.469	6.32	0.453
2	Attentive STSP (R=2)	**1.76**	**0.193**	**5.03**	**0.396**	**6.30**	**0.441**	6.22	**0.434**
3	Attentive STSP (R=3)	1.81	0.195	5.13	0.415	6.40	0.450	**6.20**	0.437
4	Attentive STSP (R=4)	1.83	0.220	5.28	0.455	6.47	0.451	6.24	0.445
5	Attentive STSP w/ only $M_c(0)$ (R=0)	2.46	0.274	6.00	0.478	8.47	0.523	8.17	0.528
6	Attentive STSP w/o $M_c(0)$ (R=1)	1.87	0.201	5.38	0.462	6.54	0.472	6.40	0.458
7	Attentive STSP w/o $M_c(0)$ (R=2)	**1.84**	**0.176**	**5.27**	0.432	**6.35**	**0.455**	**6.16**	0.441
8	Attentive STSP w/o $M_c(0)$ (R=3)	1.88	0.187	5.33	**0.417**	6.42	0.458	6.23	**0.438**
9	Attentive STSP w/o $M_c(0)$ (R=4)	1.87	0.194	5.37	0.434	6.51	0.461	6.27	0.442

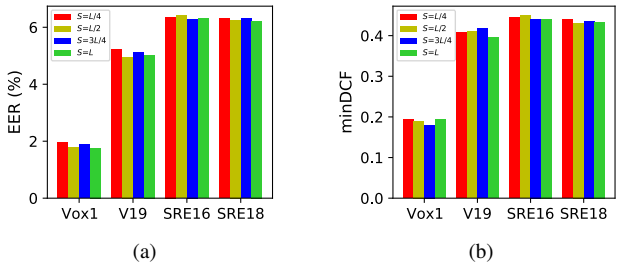


Fig. 6. (a) EER and (b) minDCF of attentive STSP on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval with respective to the step sizes of the sliding window under $L = 8$, $R = 2$, and $H = 1$.

To answer the above question, we modified the procedure of attentive STSP in Section III-A slightly by either excluding $M_c^h(0)$ or only including $M_c^h(0)$ in (18) and (19). The results of the modification are shown in Rows 5–9 of Table III. Comparing Rows 1–4 and Rows 6–9, we observe that the attentive STSP without $M_c^h(0)$ obtains comparable results with the standard attentive STSP consistently across all the evaluations under various R's. This observation suggests that once $P_c^h(k)$'s are used in the aggregation process, $M_c^h(0)$ does not offer any effective performance gain. This argument can be further demonstrated by the comparison between Row 5 and Row 6. For example, when $M_c^h(0)$'s are used alone as the aggregated statistics, the performance of attentive STSP degrades substantially, as can be seen in Row 5. In contrast, using $P_c^h(0)$'s alone (Row 6) remarkably outperforms that using $M_c^h(0)$'s alone (Row 5) across all the evaluations. Therefore, using $P_c^h(0)$ alone is much more effective than using $M_c^h(0)$ only for aggregation.

However, as verified in Section III-C, attentive STSP is a generalized statistics pooling method in that using the DC components of the spectral representations is an analogy to using the means and standard deviations in statistics pooling. Therefore, to make attentive STSP complete and compatible with the historical statistics pooling method, we still keep $M_c^h(0)$ in attentive STSP.

TABLE IV
PERFORMANCE OF VARIOUS POOLING METHODS ON TRUNCATED TEST UTTERANCES. STATS AND MHAP REFER TO STATISTICS POOLING AND MULTI-HEAD ATTENTIVE POOLING (H=2), RESPECTIVELY. *Full* MEANS THAT WE USED THE ORIGINAL DURATION OF THE TEST UTTERANCES WITHOUT TRUNCATION. THE *medium* DURATION DENOTES 5S FOR VOXCELEB1 AND VOICES19, AND 20S FOR SRE16 AND SRE18-CMN2. SIMILARLY, THE *short* DURATION REPRESENTS 2S FOR VOXCELEB1 AND VOICES19, AND 5S FOR BOTH SRES.

Row	Test Utt. Dur.	Pooling method	VoxCeleb1		VOiCES19-eval		SRE16-eval		SRE18-CMN2-eval	
			EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF
1	Full	Stats	2.08	0.225	5.78	0.498	8.07	0.499	7.63	0.475
2		MHAP	1.89	0.206	5.41	0.454	7.61	0.472	6.89	0.454
3		CCDSP	1.88	0.211	5.46	0.441	7.69	0.479	7.02	0.457
4		STSP	1.87	0.213	5.30	0.443	6.77	0.460	6.65	0.443
5		Att-STSP	**1.76**	**0.193**	**5.03**	**0.396**	**6.30**	**0.441**	**6.22**	**0.434**
6	Medium	Stats	2.35	0.264	8.59	0.697	9.52	0.580	9.21	0.571
7		MHAP	2.27	0.265	8.36	0.642	9.27	0.563	8.74	0.554
8		CCDSP	2.38	0.267	8.28	0.680	9.37	0.571	8.79	0.561
9		STSP	2.19	0.251	8.23	0.653	8.55	0.557	8.45	**0.542**
10		Att-STSP	**2.17**	**0.242**	**8.04**	**0.614**	**8.25**	**0.545**	**8.33**	0.544
11	Short	Stats	5.81	0.564	13.80	0.902	16.33	0.760	14.48	0.777
12		MHAP	5.72	0.542	13.67	0.861	16.01	0.755	14.53	0.769
13		CCDSP	5.78	0.548	13.69	0.862	20.52	0.828	15.70	0.834
14		STSP	**5.63**	0.538	13.46	0.858	15.82	0.745	**13.60**	0.768
15		Att-STSP	5.65	**0.535**	**13.44**	**0.843**	**15.75**	**0.742**	13.79	**0.765**

F. Effect of Test Utterance Duration

From Table II, we observe that compared with the baseline, the performance improvement of attentive STSP on SREs is much larger than that on VoxCeleb1 and VOiCES 2019. This observation suggests that attentive STSP can be more effective on long utterances (SREs) than on short ones (Voxeleb1 and VOiCES 2019). To investigate whether the superior performance gain of attentive STSP is related to the utterance duration or the dataset, we compared the performance of various pooling methods by truncating the test utterances.

In the experiments, we kept the duration of the enrollment utterances unchanged on all datasets. For VoxCeleb1 and VOiCES 2019, we randomly truncated the test utterances into 2s and 5s; whereas the test utterances are truncated to 5s and 20s for SRE16 and SRE18-CMN2. Note that if the duration of the original utterances is less than the target duration, we used the full-length utterances. The results are shown in Table IV. For each setting, the performance is an average of 5 runs.

From Table IV, we observe that the performance of all pooling methods degrades severely when the test utterances were truncated. Generally, attentive STSP outperforms the other pooling strategies consistently across different test durations. The performance improvement of attentive pooling, CCDSP, STSP, and attentive STSP with respective to statistics pooling is shown in Fig. 7. We observe that the performance gain of attentive STSP becomes larger when the test utterances are longer. Besides, the performance improvement of attentive STSP consistently exceeds that of the other pooling methods. For example, in SRE16, when the duration of test utterances increases from short (5s) to medium (20s), the EER improvement ($\mathrm{EER}_{\mathrm{Stats}} - \mathrm{EER}_{\mathrm{Att-STSP}}$) increases from 0.58% to 1.27%. This observation suggests that attentive STSP favors long utterances and that attentive STSP is more resilient to the duration variations in the test utterances.

G. Relationship Between Attention Weights and Phonemes

An interesting observation from Fig. 3 is that the attention weight vectors could indicate which frames or windowed segments in a feature map are discriminative. With the help of a speech recognizer, we can conduct a statistical analysis on the relationship between the attention weights and the phonemes.

For each speaker in the Voxceleb1 development set, we randomly selected 20 utterances, which amount to 24,220 utterances in the analysis. We followed the Kaldi's Librispeech recipe[7] and used the available pre-trained speech recognizer[8] to produce the phoneme posteriors. 39 phonemes (excluding SIL and SPN) in the ARPAbet encoding were considered. For each frame, a phoneme posterior vector was produced and the phoneme with the highest value was considered the recognized phoneme. Also, we extracted the attention weights in attentive pooling and attentive STSP after the training of the embedding extractors (see Section IV-A). Attention weights of individual frames were averaged according to the recognized phonemes. The relationship between the attention weights and the phonemes is shown in Fig. 8. We observe that the attention weights of attentive pooling and attentive STSP exhibit a similar trend with respect to the phonemes. In particular, both pooling methods emphasize the phonemes IH, OW, Z, S, and T. However, there are also phonemes for which the emphasis of the two methods is different. For example, attentive STSP pays more attention to UW, V, W, IY, and N than attentive pooling. This analysis suggests that when phoneme labels are available, better discriminative embeddings can be obtained by highlighting the discriminative phonemes such as OW, IH, S, Z, N, T, etc.

[7]https://github.com/kaldi-asr/kaldi/tree/master/egs/librispeech/s5
[8]http://kaldi-asr.org/models/m13

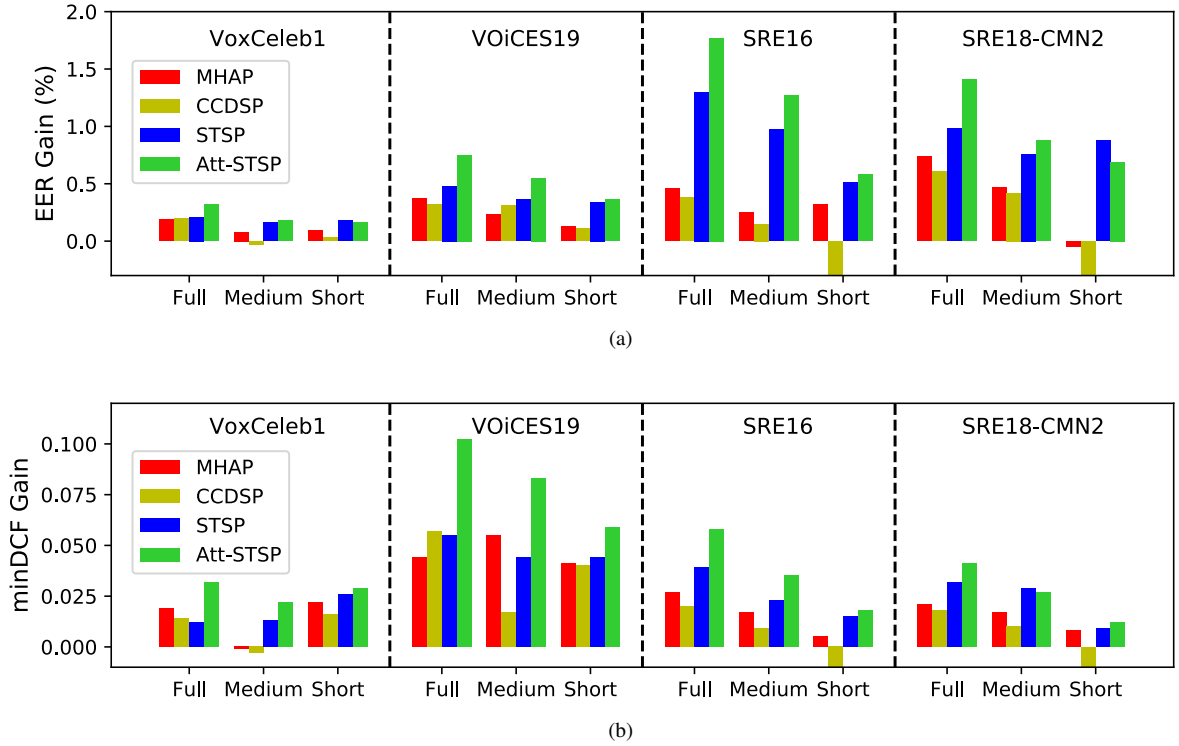


Fig. 7. Improvement in (a) EER and (b) minDCF with respective to statistics pooling. *MHAP*: multi-head attentive pooling; *CCDSP*: channel- and context-dependent statistics pooling; *STSP*: short-time spectral pooling; *Att-STSP*: attentive short-time spectral pooling (proposed).

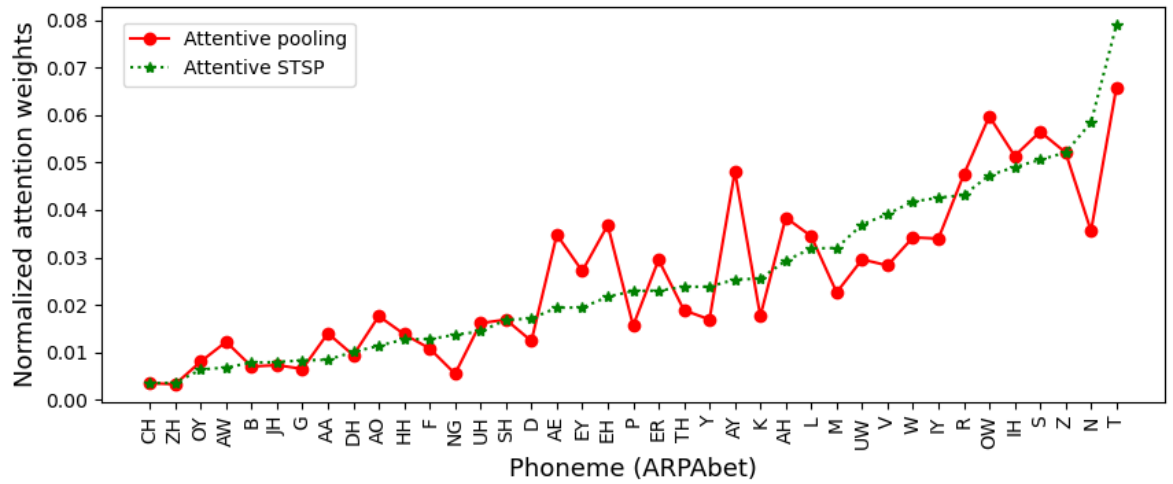


Fig. 8. Relationship between the attention weights and the phonemes on VoxCeleb1 development data. The attention weights (illustrated in the vertical axis) were normalized across the phonemes.

VI. Conclusions

In this paper, we proposed a novel attentive STSP for speaker embedding from a Fourier perspective. Attentive STSP exploits two levels of information enhancement strategies during the aggregation process: 1) applying self-attention on the windowed segments of STFT to emphasize on the discriminative information and 2) retaining the low-frequency components in the spectral domain to eliminate the effect of the noisy high-frequency information. Evaluation results on VoxCeleb1, VOiCES19-eval, SRE16-eval, and SRE18-CMN2-eval show that attentive STSP consistently outperforms multi-head attentive pooling and the vanilla STSP, suggesting that it is beneficial to apply segment-level attention and perform aggregation in the spectral domain for SV.

REFERENCES

[1] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2016, pp. 5115–5119.

[2] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: An end-to-end neural speaker embedding system," in *arXiv preprint arXiv:1705.02304*, 2017.

[3] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2018, pp. 5329–5333.

[4] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 5791–5795.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[7] S. Gao, M. Cheng, K. Zhao, X. Zhang, M. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 652–662, 2021.

[8] W. W. Lin, M. W. Mak, and L. Yi, "Learning mixture representation for deep speaker embedding using attention," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2020, pp. 210–214.

[9] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2020, pp. 3830–3834.

[10] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[11] D. Snyder, J. Villalba, N. Chen, D. Povey, G. Sell, N. Dehak, and S. Khudanpur, "The JHU speaker recognition system for the VOiCES 2019 challenge," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 2468–2472.

[12] P. Matějka, O. Plchot, H. Zeinali, L. Mošner, A. Silnova, L. Burget, O. Novotný, and O. Glembek, "Analysis of BUT submission in far-field scenarios of VOiCES 2019 challenge," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 2448–2452.

[13] W. W. Lin, M. W. Mak, N. Li, D. Su, and D. Yu, "Multi-level deep neural network adaptation for speaker verification using MMD and consistency regularization," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2020, pp. 6839–6843.

[14] Y. Z. Tu, M. W. Mak, and J. T. Chien, "Variational domain adversarial learning with mutual information maximization for speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2013–2024, 2020.

[15] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, 2020.

[16] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCreeb, G. Sellb, J. Borgstromc, L. García-Perera, F. Richardson, R. Dehak, P. Torres-Carrasquillo, and N. Dehak, "State-of-the-art speaker recognition with neural network embeddings in NIST SRE18 and Speakers in the Wild evaluations," *Computer Speech & Language*, vol. 60, 2020.

[17] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2018, pp. 3573–3577.

[18] Z. Wang, K. Yao, X. Li, and S. Fang, "Multi-resolution multi-head attention in deep speaker embedding," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2020, pp. 6464–6468.

[19] L. You, W. Guo, L. Dai, and J. Du, "Multitask learning with high-order statistics for x-vector based text-independent speaker verification," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 1158–1162.

[20] M. Han, W. Kang, S. Mun, and N. Kim, "Information preservation pooling for speaker embedding," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2020, pp. 60–66.

[21] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.

[22] T. F. Quatieri, *Discrete-Time Speech Signal Processing: Principles and Practice*. Prentice Hall, 2002.

[23] Y. Z. Tu and M. W. Mak, "Short-time spectral aggregation for speaker embedding," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2021, pp. 6708–6712.

[24] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 235–238, 1977.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[26] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Proc. Annual Conference of the International Speech Communication Association*, 2018, pp. 2252–2256.

[27] B. Scalzo, A. Arroyo, L. Stankovic, and D. Mandic, "Nonstationary portfolios: Diversification in the spectral domain," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2021, pp. 5155–5159.

[28] Y. Tang, G. Ding, J. Huang, X. He, and B. Zhou, "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6116–6120.

[29] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[30] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

[31] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," in *Proc. Annual Conference of the International Speech Communication Association*, 2019, pp. 4305–4309.

[32] S. Yadav and A. Rai, "Frequency and temporal convolutional attention for text-independent speaker recognition," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019, pp. 6794–6798.

[33] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 235–238, 2018.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning*, 2015, pp. 448–456.

[35] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing (2nd Edition)*. Prentice Hall, 1999.

[36] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *Proc. International Conference on Machine Learning*, 2019, pp. 5301–5310.

[37] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations*, 2017.

[38] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. Kanwal, T. Maharaj, A. Fischer, A. Courville, and Y. Bengio, "A closer look at memorization in deep networks," in *Proc. International Conference on Machine Learning*, 2017, pp. 233–242.

[39] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Prentice-Hall, 2006.

[40] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage BERT: A globally normalized BERT model for open-domain question answering," in *Proc. Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5878–5882.

[41] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," in *Advances in Neural Information Processing Systems*, 2020, pp. 17 283–17 297.

[42] H. Hermansky, "Modulation spectrum in speech processing," in *Signal Analysis and Prediction*, A. Procházka, J. Uhlíř, P. W. J. Rayner, and N. G. Kingsbury, Eds. Birkhäuser, Boston, 1998, ch. 27, pp. 395–406.

[43] M. Elhilali, "Modulation representations for speech and music," in *Timbre: Acoustics, Perception, and Cognition*, K. Siedenburg, C. Saitis, S. McAdams, A. Popper, and R. Fay, Eds. Springer, Cham, 2019, ch. 12, pp. 335–359.

[44] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589, 1994.

[45] M. K. Nandwana, J. Van Hout, M. McLaren, C. Richey, M. Lawson, and A. Barrios, "The VOiCES from a distance challenge 2019 evaluation plan," in *arXiv preprint arXiv:1902.10828*, 2019.

[46] NIST, "NIST 2016 speaker recognition evaluation plan," https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2016, 2016.

[47] NIST, "NIST 2018 speaker recognition evaluation plan," https://www.nist.gov/itl/iad/mig/nist-2018-speaker-recognition-evaluation, 2018.

[48] S. Ioffe, "Probabilistic linear discriminant analysis," in *Proc. European Conference on Computer Vision*, 2006, pp. 531–542.

[49] P. Matějka, O. Novotný, O. Plchot, L. Burget, M. Sáchez, and J. Černocký, "Analysis of score normalization in multilingual speaker recognition," in *Proc. Annual Conference of the International Speech Communication Association*, 2017, pp. 1567–1571.

[50] F. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.