Double Multi-Head Attention for Speaker Verification

Miquel India, Pooyan Safari, Javier Hernando

TALP Research Center
Universitat Politecnica de Catalunya
{miquel.angel.india,javier.hernando}@upc.edu,pooyan.safari@tsc.upc.edu

Abstract

Most state-of-the-art Deep Learning systems for speaker verification are based on speaker embedding extractors. These architectures are commonly composed of a feature extractor front-end together with a pooling layer to encode variable-length utterances into fixed-length speaker vectors. In this paper we present Double Multi-Head Attention pooling, which extends our previous approach based on Self Multi-Head Attention. An additional self attention layer is added to the pooling layer that summarizes the context vectors produced by Multi-Head Attention into a unique speaker representation. This method enhances the pooling mechanism by giving weights to the information captured for each head and it results in creating more discriminative speaker embeddings. We have evaluated our approach with the VoxCeleb2 dataset. Our results show 9.19% and 4.29% relative improvement in terms of EER compared to Self Attention pooling and Self Multi-Head Attention, respectively. According to the obtained results, Double Multi-Head Attention has shown to be an excellent approach to efficiently select the most relevant features captured by the CNN-based front-ends from the speech signal.

Index Terms: self multi-head attention, speaker recognition, speaker verification

1. Introduction

Speaker verification aims to determine whether a pair of audios corresponds to the same speaker. Given speech signals, speaker verification systems are able to extract speaker identity patterns from the characteristics of the voice. These patterns can be both statistically modelled or encoded into discriminative speaker representations. Over the last few years, researchers have put huge effort on encoding these traits into more discriminative speaker vectors. Current state-of-the-art speaker verification systems are based on Deep Learning (DL) approaches. These architectures are commonly trained as speaker classifiers in order to be used as speaker embedding extractors. Speaker embeddings are fixed-length vectors extracted from some of the last layers of these Deep Neural Networks (DNNs) [1]. The most known representation is the x-vector [2], which has become state-of-the-art for speaker recognition and has also been used for other tasks such as language and emotion recognition [3, 4].

Most of the recent network architectures used for speaker embedding extraction are composed of a front-end feature extractor, a pooling layer, and a set of Fully Connected (FC) layers. Lately, there have been several architectures proposed to encode audio utterances into speaker embeddings for different choices of network inputs. Using Mel-Frequency Cepstral Coefficient (MFCC) features, Time Delay Neural Network (TDNN) [5, 6] is the most currently used architecture. TDNN is the x-vector front-end and consists of a stack of 1-D dilated Convolutional Neural Networks (CNNs). The idea behind the use of TDNNs is to encode a sequence of MFCC into a more discriminative sequence of vectors by capturing long-term feature relations. 2-D CNNs have also shown competitive results for speaker verification. There are Computer Vision architectures such as VGG [7, 8, 9] and ResNet [10, 11, 12] that have been adapted to capture speaker discriminative information from the Mel-Spectrogram. In fact, Resnet34 has shown a better performance than TDNN in the most recent speaker verification challenges [13, 14]. Finally, there are also some other attempts to work directly on the raw signal instead of using hand-crafted features [15, 16, 17].

Given the encoded sequence from the front-end, a pooling layer is adopted to obtain an utterance-level representation. During the last few years, there are several studies addressing different types of pooling strategies. X-vector originally uses statistical pooling [6] or the Self Attentive pooling method proposed in [18]. A wide set of pooling layers based on self attention have been proposed improving this vanilla self attention mechanism. In [18] several attentions are applied over the same encoded sequence, producing multiple context vectors. In our previous work [9], the encoded sequence is split into different heads and a different attention model is applied over each head sub-sequence. Attention mechanisms have also been used to improve statistical pooling. In works like [19], attention is used to extract better order features statistics. Finally there are also works with competitive results such as [20, 21, 22] which proposed pooling methods independent from self attention models.

In this paper we present a Double Multi-Head Attention (MHA) pooling layer for speaker verification. The use of this layer is inspired by [23], where Double MHA is presented as a double attention block which captures feature statistics and makes adaptive feature assignment over images. In this work this mechanism is used as a combination of two self attention pooling layers to create utterance-level speaker embeddings. Given a sequence of encoded representations from a CNN, Self MHA first concatenates the context vector from K head attentions applied over a K sub-embedding sequences. An additional self attention mechanism is then applied over the multi-head context vector. This attention based pooling summarizes the set of head context vectors into a global speaker representation. This representation is pooled through a weighted average of the head context vectors, where the head weights are produced with the self attention mechanism. On one hand, this approach allows the model to attend to different parts of the sequence, capturing at the same time different subsets of encoded representations. On the other the hand, the pooling layer allows to select which head context vectors are the most relevant to produce the global context vector. In comparison with [23], the second pooling layer operates over the head context vectors produced by a MHA instead of the global descriptors produced by a self multi attention mechanism applied over an image.

2. Proposed Architecture

Our proposed system architecture is illustrated in Figure 1. It utilizes a CNN-based front-end which takes in a set of variable length mel-spectrogram features and outputs a sequence of speaker representations. These speaker representations are further subject to a Double MHA pooling which is the main contribution of this work. The Double MHA layer comprises a Self MHA pooling and an additional Self Attention layer that summarizes the information of each head context vector into an unique speaker embedding. The combination of Self MHA pooling together with this Self Head Attention layer provides us with a deeper self-attention pooling mechanism (Figure 2). The speaker embedding obtained from the pooling layer is sent through a set of FC layers to predict the speaker posteriors. This network architecture is trained with Additive Margin Softmax (AMS) loss [24] as a speaker classifier so as to have a speaker embedding extractor.

2.1. Front-End Feature Extractor

Our feature extractor network is a larger version of the adapted VGG proposed in [9]. This CNN comprises four convolution blocks, each of which contains two concatenated convolutional layers followed by a max pooling with a 2×2 stride. Hence given a spectrogram of N frames, the VGG performs a downsampling reducing its output into a sequence of $N/16$ representations. The output of the VGG $h \in \mathbb{R}^{M \times N/16 \times D'}$ is a set of M feature maps with $N/16 \times D'$ dimension. These feature maps are concatenated into a unique vector sequence. This reshaped sequence of hidden states can now be defined as $h \in \mathbb{R}^{N/16 \times MD'}$, where $D = MD'$ corresponds to the hidden state dimension.

2.2. Self Multi-Head Attention Pooling

The sequence of hidden states output from the front-end feature extractor can be expressed as $h = [h_1 h_2 ... h_N]$ with $h_t \in \mathbb{R}^D$. If we consider a number of K heads for the MHA pooling, now we can define the hidden state as $h_t = [h_{t1} h_{t2} ... h_{tK}]$ where $h_{tj} \in \mathbb{R}^{D/K}$. Hence each feature vector is split into a set of sub-feature vectors of size D/K. In the same way we have also a trainable parameter $u = [u_1 u_2 ... u_K]$ where $u_j \in \mathbb{R}^{D/K}$. A self attention operation is then applied over each head of the encoded sequences. The weights of each head alignment are defined as:

$$w_{tj} = \frac{\exp\left(\frac{h_{tj}^T u_j}{\sqrt{d_h}}\right)}{\sum_{l=1}^{N} \exp\left(\frac{h_{lj}^T u_j}{\sqrt{d_h}}\right)} \tag{1}$$

where w_{tj} corresponds to the attention weight of the head j on the step t of the sequence and d_h corresponds to hidden state dimension D/K. If each head corresponds to a subspace of the hidden state, the weight sequence of that head can be considered as a probability density function (pdf) from that subspace features over the sequence. We then compute a new pooled representation for each head in the same way than vanilla self attention:

$$c_j = \sum_{t=1}^{N} h_{tj}^T w_{tj} \tag{2}$$

where $c_j \in \mathbb{R}^{D/K}$ corresponds to the utterance level representation from head j. The final utterance level representation is then obtained with the concatenation of the utterance level vectors from all the heads $c = [c_1 c_2 ... c_k]$. This method allows the network to extract different kinds of information over different regions of the network.

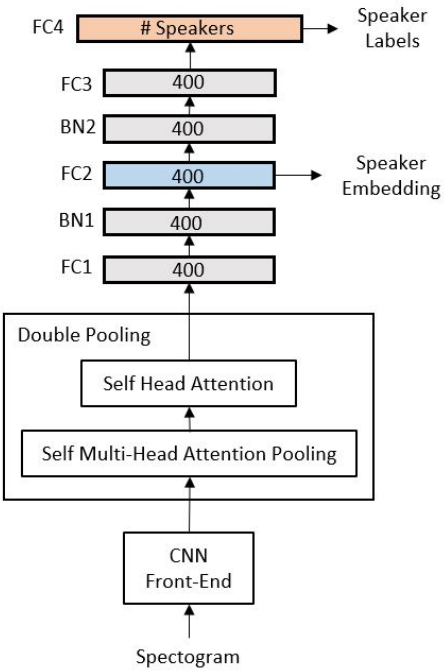

Figure 1: *System Architecture.*

2.3. Double Multi-Head Attention

The main disadvantage of Self MHA pooling is that it assumes uniform head relevance. The output context vector is the concatenation of all head context vectors and it is used as input of the following dense layers. Double MHA does not assume that. Therefore each utterance context vector is computed as a different linear combination of head context vectors. A summarized vector c is then defined as a weighted average over the set of head context vectors c_i. A self attention mechanism is used to pool the set of head context vectors c_i and obtain an overall context vector c.

$$w'_i = \frac{\exp\left(c_i^T u'\right)}{\sum_{l=1}^{K} \exp\left(c_l^T u'\right)} \tag{3}$$

$$c = \sum_{i=1}^{K} c_i^T w'_i \tag{4}$$

where w'_i corresponds to the aligned weight of each head and $u' \in \mathbb{R}^{D/K}$ is a trainable parameter. The context vector c is then computed as the weighted average of the context vectors among heads. With this method, each utterance context vector is created scaling the information of the most/least relevance heads. Considering the whole pooling layer, Double MHA allows to capture different kind of speaker patterns in different regions of the input, and at the same time allows to weight the relevance of each of these patterns for each utterance.

The number of heads used for this pooling defines both the context vector dimension and how the VGG feature maps

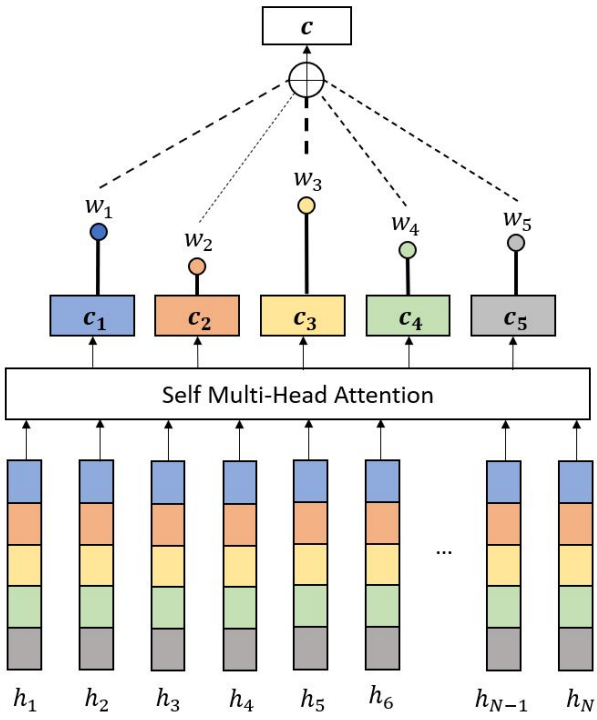

Figure 2: *An example of Double MHA Pooling with* 5 *heads.*

are grouped. Considering the number of M channels and K heads, for each head we would create a c_i context vector of $D'M/K$ dimension which contains a subset of M/K feature maps. Therefore, as the number of heads grows larger, it allows Double MHA to consider more subsets of features while decreases the dimension of the final utterance-level context vector. This implies a trade-off between the number of features subsets we can create and how much compressed are these features in the context vector subspace.

2.4. Fully-Connected Layers

The utterance-level speaker vector obtained from the pooling layer is fed into a set of four FC layers (Figure 1). Each of the first two FC layers is followed by a batch normalization layer [25] and Rectified Linear Unit (ReLU) activations. A dense layer is adopted for the third FC layer and the last FC corresponds to the speaker classification layer. Since AMS is used to train the network, the third layer is set up without activation and batch normalization as proposed in [24]. Once the network is trained, we can extract a speaker embedding from one of the intermediate FC layers. According to [26], we consider the second layer as the speaker embedding instead of the third one. The output of this FC layer then corresponds to the speaker representation that will be used for the speaker verification task.

3. Experimental Setup

The proposed system[1] in this work has been assessed by VoxCeleb dataset [27, 7]. VoxCeleb is a large multimedia database that contains more than 1 million utterances for more than 6K celebrities. These utterances are 16kHz audio chunks extracted from *Youtube* videos. VoxCeleb has two different versions with several evaluation conditions and protocols. For our experiments, VoxCeleb1 and VoxCeleb2 development partitions have been used to train both baseline and presented approaches. No data augmentation has been applied to increase the training data. On the other hand, the performance of these systems have been evaluated with the original Vox1 test set.

[1]Models are available at:
https://github.com/miquelindia90/DoubleAttentionSpeakerVerification

Table 1: *CNN Architecture. In and Out Dim. refers to the input and output feature maps of the layer. Feat Size refers to the dimension of each one of this output feature maps.*

Layer	Size	In Dim.	Out Dim.	Stride	Feat Size
conv11	3x3	1	128	1x1	Nx80
conv12	3x3	128	128	1x1	Nx80
mpool1	2x2	-	-	2x2	N/2x40
conv21	3x3	128	256	1x1	N/2x40
conv22	3x3	256	256	1x1	N/2x40
mpool2	2x2	-	-	2x2	N/4x20
conv31	3x3	256	512	1x1	N/4x20
conv32	3x3	512	512	1x1	N/4x20
mpool3	2x2	-	-	2x2	N/8x10
conv41	3x3	512	1024	1x1	N/8x10
conv42	3x3	1024	1024	1x1	N/8x10
mpool4	2x2	-	-	2x2	N/16x5
flatten	-	1024	1	-	N/16x5120

Two different baselines have been considered to compare with the presented approach. Double MHA pooling have been evaluated against two self attentive based pooling methods: vanilla Self Attention and Self MHA. In order to evaluate them, these mechanisms have replaced the pooling layer of the system (Figure 1) without modifying any other block or parameter from the network. The speaker embeddings used for the verification tests have been extracted from the same FC layer for each of the pooling methods. Cosine distance have been used to compute the scores between pairs of speaker embeddings.

The proposed network has been trained to classify variable-length speaker utterances. As input features we have used 80 dimension log Mel Spectrograms with 25ms length Hamming windows and 10ms window shift. The audios have not been filtered with any Voice Activity Detection (VAD) system and 0.97 coefficient pre-emphasis has been applied. The audio features have been only normalized with Cepstral Mean Normalization (CMN). The CNN encoder is then fed with $N \times 80$ Spectrograms to obtain a sequence of $N/16 \times 5120$ encoded hidden representations. For training we have used batches of N=350 frames audio chunks but for test the whole utterances have been encoded. The setup of the CNN feature extractor can be found on Table 1. For the pooling layer we have tuned the number of heads for both Self MHA and Double MHA. For the presented CNN setup we have considered 8,16, and 32 head number values, which implies a head context vector c_i of 640, 320, and 160, respectively. The last block of the system consists on four consecutive FC layers. The first three dense layers have 400 dimension. The last FC layer has 7205 dimension, which corresponds to the number of train speaker labels. Batch normalization has been applied only on the first two dense layers as mentioned in subsection 2.4. The network has been trained with AMS loss with $s = 30$ and $m = 0.4$ hyper-parameters. Batch size is set to 64 samples and Adam optimizer has been

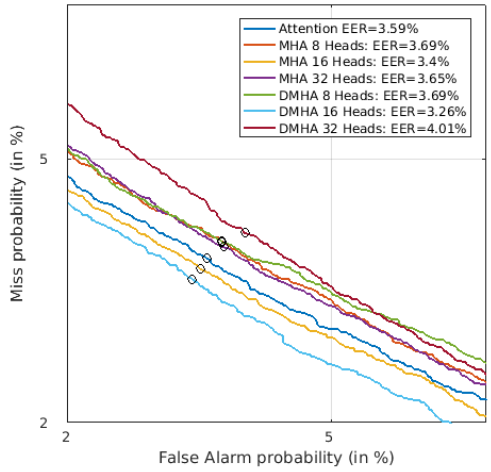

Figure 3: *DET curves for the experiments on VoxCeleb 1 test set verification task.*

used to train all the models with $1e-4$ learning rate and $1e-3$ weight decay. During the training we have used 15 patience early stopping criterion, where the models have been validated each $10,000$ batches.

4. Results

The proposed approach has been evaluated with different attention methods in the VoxCeleb text-independent verification task. Performance is evaluated using Equal Error Rate (EER) and Detection Cost Function (DCF) calculated using $C_{FA} = 1$, $C_M = 1$, and $P_T = 0.01$. The results of this task are presented in both Figure 3 and Table 2. DET curves are shown in Figure 3 and both EER and DCF metrics are presented in Table 2. Double MHA is referred as DMHA in both analysis.

Self Attention pooling has shown the worst results for this task compared to the best tuned approaches in both Self MHA and Double MHA. Compared to Self Attention, Self MHA has shown better results with 16 heads and worst results with both 8 and 32 heads. With 16 heads, Self MHA has shown a 5.29% EER relative improvement in comparison with Self Attention Pooling. Otherwise, DCF has only improved from 0.0029 to 0.0028. With 8 and 32 heads Self MHA performance in EER has decreased a 2.78% and a 1.67%, respectively. Double MHA have shown better results with 16 heads than both Self Attention and Self MHA approaches. Double MHA has shown a 9.19% EER relative improvement in comparison with Self Attention and 4.29% relative improvement compared with 16 heads Self MHA. In terms of DCF, Double MHA DCF has shown the best result with a 0.0027. If we compare Double MHA and Self MHA with 8 heads, Double MHA is better in terms of DCF but has not improved in terms of EER. Double MHA DCF has improved from 0.0036 to 0.0029 but EER has remain the same with a 3.65%. Double MHA with 32 heads has shown the worst results in comparison with both 32 heads Self MHA and Self Attention with a 4.01% EER and 0.0032 DCF.

As the results have shown, best performances in MHA based approaches are achieved with 16 heads. Besides verification metrics, Table 2 also indicates the head and global context vector dimensions. As it was discussed in subsection 2.3,

Table 2: *Evaluation results of the text-independent verification task on VoxCeleb 1.*

Approach	**Heads**	c_i **dim**	c **dim**	**EER**	**DCF**
Attention	1	5120	5120	3.59	0.0029
MHA	8	640	5120	3.69	0.0036
MHA	16	320	5120	3.4	0.0028
MHA	32	160	5120	3.65	0.0031
DMHA	8	640	640	3.69	0.0029
DMHA	16	320	320	**3.26**	**0.0027**
DMHA	32	160	160	4.01	0.0032

c_i in Self MHA and both c_i and c dimensions in Double MHA are inversely proportional to the number of heads. Therefore, there is a trade-off between number of heads and systems performance, which is related to context vector dimensions. Worst performance showed with Double MHA is achieved with 32 heads. This setup implies that both c_i and c dimensions are 160. This value can be considered small compared to current state-of-the-art speaker embeddings, whose dimension range is between 200 and 1500. Therefore, system performance with 32 heads is worst because the context vector subspace is not enough big to encode all the discriminative speaker information from the CNN output. On the other hand, as larger is the number of heads, more subsets of speaker features can be captured over the CNN encoded sequence. With 8 heads, 640 dimension head context vectors are extracted and with 16 heads, head context vectors have 320 dimension. Both Self MHA and Double MHA approaches have shown the best results with 16 heads, which implies 320 dimension context vectors. Therefore CNN output feature maps are more efficiently grouped in subsets of $M/K = 64$ channels, which correspond to sub-sequences of 320 dimension embeddings. Considering these sets of 16 context vectors pooled in that layer, these representations are efficiently averaged with Double MHA into unique 320 dimension utterance-level speaker representations.

5. Conclusion

In this paper we have implemented a Double Multi-Head Attention mechanism to obtain speaker embeddings at level utterance by pooling short-term representations. The proposed pooling layer is composed of a Self Multi-Head Attention pooling and a Self Attention mechanism that summarizes the context vectors of each head into a unique speaker vector. This pooling layer have been tested in a neural network based on a CNN that maps spectrograms into sequences of speaker vectors. These vectors are then input to the proposed pooling layer, which output activation is then connected to a set of dense layers. The network is trained as a speaker classifier and a bottleneck layer from these fully connected layers is used as speaker embedding. We have evaluated this approach with other pooling methods for the text-independent verification task using the speaker embeddings and applying cosine distance. The presented approach have outperformed both vanilla Self Attention and Self Multi-Head Attention poolings.

6. Acknowledgements

This work was supported in part by the Spanish Project DeepVoice (TEC2015-69266-P).

7. References

[1] O. Ghahabi, P. Safari, and J. Hernando, "Deep learning in speaker recognition," in *Development and Analysis of Deep Learning Architectures*. Springer, 2020, pp. 145–169.

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

[3] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors," in *Odyssey*, 2018, pp. 105–111.

[4] R. Pappagari, T. Wang, J. Villalba, N. Chen, and N. Dehak, "x-vectors meet emotions: A study on dependencies between emotion and speaker recognition," *arXiv preprint arXiv:2002.05039*, 2020.

[5] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 165–170.

[6] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Interspeech*, 2017, pp. 999–1003.

[7] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[8] ——, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[9] M. India, P. Safari, and J. Hernando, "Self Multi-Head Attention for Speaker Recognition."

[10] G. Bhattacharya, J. Alam, and P. Kenny, "Deep speaker recognition: Modular or monolithic?" in *Proc. Interspeech*, 2019, pp. 1143–1147.

[11] J. Zhou, T. Jiang, Z. Li, L. Li, and Q. Hong, "Deep speaker embedding extraction with channel-wise feature responses and additive supervision softmax loss function," *Proc. Interspeech 2019*, pp. 2883–2887, 2019.

[12] A. Hajavi and A. Etemad, "A deep neural network for short-segment speaker recognition," *arXiv preprint arXiv:1907.10420*, 2019.

[13] J. S. Chung, A. Nagrani, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, "Voxsrc 2019: The first voxceleb speaker recognition challenge," *arXiv preprint arXiv:1912.02522*, 2019.

[14] H. Zeinali, S. Wang, A. Silnova, P. Matějka, and O. Plchot, "But system description to voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[15] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," *arXiv preprint arXiv:1808.00158*, 2018.

[16] J.-W. Jung, H.-S. Heo, I.-H. Yang, H.-J. Shim, and H.-J. Yu, "Avoiding speaker overfitting in end-to-end dnns using raw waveform for text-independent speaker verification," *extraction*, vol. 8, no. 12, pp. 23–24, 2018.

[17] J.-w. Jung, H.-S. Heo, J.-h. Kim, H.-j. Shim, and H.-J. Yu, "Rawnet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," *arXiv preprint arXiv:1904.08104*, 2019.

[18] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification." in *Interspeech*, 2018, pp. 3573–3577.

[19] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv preprint arXiv:1803.10963*, 2018.

[20] 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 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

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

[22] Y. Jung, Y. Kim, H. Lim, Y. Choi, and H. Kim, "Spatial pyramid encoding with convex length normalization for text-independent speaker verification," *arXiv preprint arXiv:1906.08333*, 2019.

[23] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "Aˆ 2-nets: Double attention networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 352–361.

[24] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," *arXiv preprint arXiv:1904.03479*, 2019.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[26] H. Zeinali, L. Burget, J. Rohdin, T. Stafylakis, and J. H. Cernocky, "How to improve your speaker embeddings extractor in generic toolkits," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6141–6145.

[27] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.