INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Self-Attention Encoding and Pooling for Speaker Recognition

Pooyan Safari, Miquel India, Javier Hernando

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

Abstract

The computing power of mobile devices limits the end-user applications in terms of storage size, processing, memory and energy consumption. These limitations motivate researchers for the design of more efficient deep models. On the other hand, self-attention networks based on *Transformer* architecture have attracted remarkable interests due to their high parallelization capabilities and strong performance on a variety of Natural Language Processing (NLP) applications. Inspired by the *Transformer*, we propose a tandem Self-Attention Encoding and Pooling (SAEP) mechanism to obtain a discriminative speaker embedding given non-fixed length speech utterances. SAEP is a stack of identical blocks solely relied on self-attention and position-wise feed-forward networks to create vector representation of speakers. This approach encodes short-term speaker spectral features into speaker embeddings to be used in text-independent speaker verification. We have evaluated this approach on both *VoxCeleb1 & 2* datasets. The proposed architecture is able to outperform the baseline x-vector, and shows competitive performance to some other benchmarks based on convolutions, with a significant reduction in model size. It employs 94%, 95%, and 73% less parameters compared to ResNet-34, ResNet-50, and x-vector, respectively. This indicates that the proposed fully attention based architecture is more efficient in extracting time-invariant features from speaker utterances.

Index Terms: Self-Attention Encoding, Self-Attention Pooling, Speaker Verification, Speaker Embedding

1. Introduction

Recently, there have been several attempts to apply Deep Learning (DL) in order to build speaker embeddings [1, 2, 3, 4, 5]. Speaker embedding is often referred to a single low dimensional vector representation of the speaker characteristics from a speech signal. It is extracted using a Neural Network (NN) model. There are different architectures proposed to extract these speaker representations either in a supervised (e.g., [1, 3]) or in an unsupervised way (e.g., [2, 6]). In the supervised methods, usually a deep architecture is trained for classification purposes using speaker labels on the development set. Then in the testing phase, the output layer is discarded, the feature vectors of an unknown speaker are given through the network, and the pooled representation from one or more hidden layers are considered as the speaker embedding [3, 5].

On the other hand, attention mechanisms are one of the main reasons of the success of sequence-to-sequence (seq2seq) models in tasks like Neural Machine Translation (NMT) or Automatic Speech Recognition (ASR) [7, 8, 9]. In seq2seq models, these algorithms are applied over the encoded sequence in order to help the decoder to decide which region of the sequence must be either translated or recognized. Self-attention, as a specific type of attention, is the process of applying the attention mechanism to every position of the input itself. Self-attention has shown promising results in a variety of NLP tasks, such as NMT [8], semantic role labelling [10], and language representations [11]. The popularity of the self-attention as employed in networks such as *Transformer* [8], lies in its high parallelization capabilities in computation, and its flexibility in modeling dependencies regardless of distance by explicitly attending to the whole signal. Multi-head attention is a newly emerging attention mechanism, which is originally proposed in [8] for a *Transformer* architecture and appeared very effective in many seq2seq models such as [8, 12, 13]. People in [14, 15] employ multi-head attention mechanism in the pooling layer to aggregate the information over multiple input segments and output a single fixed-dimensional vector which is further used in the Deep Neural Network (DNN) classifier.

For speaker recognition, attention mechanism has been studied mostly in the pooling layer as an alternative to statistical pooling. In [4, 16, 17], attention is applied over the hidden states of a Recurrent Neural Network (RNN) in order to pool these states into speaker embeddings for text-dependent speaker verification. In [18], a unified framework is introduced for both speaker and language recognition using attention. In [14], a multi attention mechanism based on [19] is proposed to use more than one attention model to capture different kinds of information from the encoder features. Other works like [15] have explored the use of multi-head attention for the pooling layer.

Computational limitations of the mobile devices oblige the end-user applications to use models which comply with certain constraints such as energy consumption, memory and storage size. However, these constraints are not usually met by deep learning approaches. These obstacles motivate researchers for the design of more efficient deep models. *MobileNets* were proposed in [20, 21] and showed promising results for image classification. They are built based on depth-wise separable convolutions [22] and further used in *Inception* models [23] to reduce the computation in the first few layers. Architectures based on factorized convolution were also found to be successful in [24] and [25] for image classification tasks. In another attempt, a bottleneck strategy was utilized in [26] to design a very small network. There are other reduced computation networks such as structured transform networks [27] and deep fried Convolutional Neural Networks (CNNs) [28].

In this paper, we present an end-to-end speaker embedding extractor for speaker verification inspired by self-attention networks. It is shown that self-attention mechanisms are able to capture time-invariant features from one's speech, which can result in a discriminative representation of the speaker. The end-to-end architecture comprises an encoder, a pooling layer, and a DNN classifier. The main contribution of this work is to use an encoder and pooling layer solely relied on self-attention

http://dx.doi.org/10.21437/Interspeech.2020-1446

and feed-forward networks to create discriminative speaker embeddings. In order to show the effectiveness of the proposed approach, these embeddings are evaluated on *Voxceleb1* [29], *Voxceleb2* [30], and *VoxCeleb1-E* [30] protocols. The preliminary results obtained on this attention-based system show superior performance compared to the x-vector baseline. Its performance is also competitive to some other CNN-based benchmarks while having much less parameters. More precisely, our proposed system employs more than 98%, 94%, 95%, and 73% less parameters compared to VGG-M, ResNet-34, ResNet-50, and x-vector, respectively. This is a substantial reduction in model size which makes it a great candidate for resource-constrained devices. To the best of our knowledge, there is only one study to address concise deep models for speaker identification on mobile devices [31], which is evaluated on TIMIT [32] and MIT [33] datasets.

2. Proposed Architecture

The system presented in this work can be divided into three main stages, namely an encoder, a pooling layer, and a DNN classifier (Figure 1). The encoder is a stack of N identical blocks each of which has two sub-layers. The first sub-layer is a single-head self-attention mechanism, and the second is a position-wise feed-forward network. A residual connection is employed around each of the two sub-layers, and it is followed by a layer normalization [34]. The output of the encoder network is a sequence of feature vectors. The pooling mechanism is then used to transform these encoded features into an overall speaker representation. Finally, the third stage of the network is a DNN classifier, which outputs the speaker posteriors. The activations of one or more of these hidden layers can be considered as the speaker embeddings. These embeddings can be further used in speaker verification using cosine scoring or more sophisticated back-ends.

2.1. Self-Attention Encoder

The input to each of the encoder blocks is passed through a self-attention network to produce representations by applying attention to every position of the input sequence. This is done using a set of linear projections. Consider an input sequence $X = [x_1, x_2, \ldots, x_T]^{tr}$ of length T, with $x_t \in \mathbb{R}^{d_m}$, and a set of trainable parameters $\{W_Q, W_K\} \in \mathbb{R}^{d_m \times d_k}, W_V \in \mathbb{R}^{d_m \times d_v}$, the model transforms the input into namely queries $Q \in \mathbb{R}^{T \times d_k}$, keys $K \in \mathbb{R}^{T \times d_k}$, and values $V \in \mathbb{R}^{T \times d_v}$:

$$Q = XW_Q,\ K = XW_K,\ V = XW_V \tag{1}$$

The output for each time instance o_t of the attention layer is computed as:

$$o_t = Attn(q_t, K)V \tag{2}$$

where $Attn(\cdot)$ is an attention function that retrieves the keys K with the query q_t. There are two most commonly used attention functions namely additive attention [7], and dot-product attention [35]. Here the attention is a scaled version of the dot-product attention mechanism, which is originally proposed in [8]. It is much faster and more space-efficient in practice, compared to the additive attention. Once the dimension d_k grows, the additive attention outperforms the dot-product attention [35]. Therefore the scaling factor comes in handy to fill the performance gap between the two while keeping its advantages.

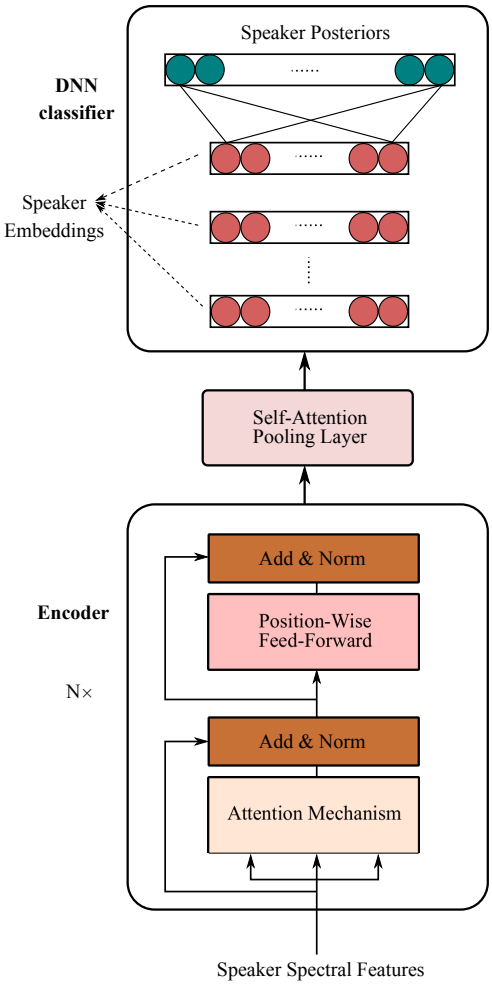

Figure 1: *The diagram of the proposed architecture. Encoder comprises a stack of N layers each of which is a series of attention mechanisms followed by position-wise feed-forward networks. A residual connection and layer normalization are applied to each of the sub-layers.*

The final output representation O of the attention mechanism is formulated as:

$$O = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

The output of the attention mechanism is sent to the other sublayer, which is a position-wise feed-forward network. It is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between:

$$FFN(h) = max(0, hW_1 + b_1)W_2 + b_2 \tag{4}$$

where h is the input and $FFN(h)$ is the output of the feed-forward network. These linear transformations are the same across different positions, however, they differ from layer to layer. $W_1 \in \mathbb{R}^{d_m \times d_{ff}}$ and $W_2 \in \mathbb{R}^{d_{ff} \times d_m}$ can be considered as two convolutions with kernel size of 1, while d_{ff} is the feed-forward dimension.

2.2. Self-Attention Pooling Layer

Self-attention based encoder outputs a sequence of short-term speaker representations. Given these frame-level features, it

is needed to convert them into an utterance level vector. In this work, we will use a self-attention pooling layer. This layer is an additive attention based mechanism, which computes the compatibility function using a feed-forward network with a single hidden layer. Given the sequence of encoded features $H = [h_1, h_2, \ldots, h_T]^{tr} \in \mathbb{R}^{T \times d_m}$, we compute the segment-level representation C as:

$$C = Softmax(W_c H^T) H \tag{5}$$

where $W_c \in \mathbb{R}^{d_m}$ is a trainable parameter. This self attention can be also understood as a dot product attention where the keys and the values correspond to the same representation and the query is only a trainable parameter. Therefore C is a weighted average of the encoder sequence of features. These weights correspond to the alignment learned by the self-attention mechanism. This kind of attention pooling is different from the ones proposed in [14, 36], where attention is applied to extract statistics necessary for statistical pooling. However, in this work, we replace the whole statistical pooling with a simple additive attention mechanism with less computational cost.

2.3. Speaker Embeddings

The fixed length vector obtained from the pooling layer is fed into a DNN classifier to output the speaker posteriors. This DNN classifier is based on three fully connected layers and a softmax layer to compute a multi-class cross entropy objective for training purposes.

Once the network is trained, embeddings are extracted from one of the DNN layers after the non-linearity. Instead of using the previous layer to the softmax as the speaker embedding, we use the second previous layer like in [37]. This is done because it has been shown by [3] that this layer generalizes better, so it contains speaker discriminative information less adapted to the training data.

3. Experimental Setup

The performance of the proposed system is assessed using both *VoxCeleb1* [29], and *VoxCeleb2* [30] datasets with three different protocols. *VoxCeleb1* development set is used for training in the first set of experiments (referred to as Vox1 hereafter). It contains over $147,935$ utterances from $1,211$ speakers. The trained models are evaluated on the original *VoxCeleb1* test set, which contains $18,860$ verification pairs for each clients and impostors test. For the second set of experiments (referred to as Vox2 hereafter), *Voxceleb2* develpoment set is used for training. It comprises $1,092,009$ utterances among $5,994$ speakers. The *VoxCeleb1* test set is used for the assessment. In the third set of experiments (referred to as Vox1-E hereafter), we use *Voxceleb2* development set for training, and *VoxCeleb1-E* for the test. *VoxCeleb1-E* consists of $581,480$ random pairs sampled from the entire *VoxCeleb1* dataset (development+test), covering $1,251$ speakers.

For all the systems that we have trained, *librosa* toolkit [38] is employed to extract 30-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) with the standard $25ms$ window size and $10ms$ shift to represent the speech signal. The deltas and double deltas together are added to produce a 90 dimensional input for the networks. No data or test-time augmentation have been used during training or test. All features are subject to Cepstral Mean Variance Normalization (CMVN) prior to training. All models are then trained on chunks of speech features with 300 frames. During the test, embeddings are extracted from the whole speech signal. ReLU is used as the non-linearity for all the systems. Adam optimizer has been used to train all the models with standard values as proposed in [39] and learning rate of 1e-4.

The architecture of x-vector baseline is as proposed in [5]. Speaker embeddings are extracted from the affine component of the first layer after pooling. No dropout is used as suggested in [37] since it does not improve the results for the verification task. This statement is also confirmed by our experiments. All Time Delay Neural Network (TDNN) layers are followed by non-linearity and Batch Normalization [23]. For this baseline architecture we have also considered Probabilistic Linear Discriminant Analysis (PLDA) back-end. Given the speaker embeddings, these are centered, then subject to Linear Discriminant Analysis (LDA) dimension reduction to 200 and length-normalized. Additionally, we have trained a PLDA to score the trials of these post-processesed represenations for the speaker verification task.

For the SAEP configuration, we use a stack of $N = 2$ identical layers with $d_k = d_v = 512$, and position-wise feed-forward dimension of $d_{ff} = 2048$. For the encoder network a dropout of 0.1 is considered and for the rest of the network dropout is fixed to 0.2. The dimension of the first dense layer is equal to 90 and all other dense layers are equal to embedding dimension which is 400. The size of 400 for the embedding is chosen just to be similar as the i-vector system. No experiments have been conducted to explore the ideal size for the embedding. For SAEP, we have also tried Additive Margin Softmax (AMSoftmax) as originally proposed in [40] with scaling factor of 30 and 0.4 margin as the hyperparameters.

4. Results

Results for text-independent speaker verification task are summarized in Table 1 in three different sections for three different protocols. The cosine distance is employed for scoring unless otherwise stated. The performance of different systems have been evaluated using Equal Error Rate (EER). The proposed architecture is compared with i-vector, and also x-vector system, which is nowadays one of the most popular end-to-end speaker embedding extractors. Additionally, in order to analyse the trade-off between performance and network size, we have also included results reported by some other approaches such as VGG-M [29], ResNet-34, and ResNet-50 [30]. These models benefit from much larger amount of parameters.

For the Vox1 protocol we have compared SAEP with x-vector and the VGG based architecture proposed in [29]. It is the smallest protocol in terms of development data. SAEP with vanilla softmax shows a small relative improvement in terms of EER compared to x-vector with LDA/PLDA, and VGG-M. However, using AMSoftmax this improvement increases to a 8.93% in comparison with x-vector/LDA/PLDA and 7.99% compared to VGG-M.

For the Vox2 and Vox1-E protocol we have added ResNet-34 and ResNet-50 based models, which have shown recently very competitive results for the speaker verification task [30]. Here SAEP outperforms x-vector by a larger margin. The presented approach with softmax loss shows almost 20% relative improvement in terms of EER compared to x-vector with LDA/PLDA back-end in the Vox2 protocol and more than 15% in the Vox1-E protocol. Compared to the ResNet standard architectures, SAEP is able to perform similar to ResNet-34 with more than 94% less parameters. ResNet-34 and ResNet-50 have better results than our method and also x-vector, mainly due to

Table 1: *Evaluation results on VoxCeleb data sets. Cosine distance is employed for scoring unless otherwise stated.* ***Vox1****: train on Vox1-dev & test on Vox1-test.* ***Vox2****: train on Vox2-dev test on Vox1-test.* ***Vox1-E****: train on Vox2-dev test on extended VoxCeleb1 test set. The result reported for ResNet-50 on Vox1-E, takes advantage of the test time augmentation [30].*

	Method	Input	Loss	Dim.	# Params	**EER%**
Vox1	i-vector/PLDA [29]	MFCC	-	400	-	8.8
	VGG-M [29]	Spectrogram	Softmax	1024	67M	10.2
	VGG-M [29]	Spectrogram	Softmax+Contrastive	1024	67M	7.8
	x-vector	MFCC	Softmax	512	4.38M	12.62
	x-vector/LDA/PLDA	MFCC	Softmax	512	4.38M	7.83
	SAEP	MFCC	Softmax	400	1.16M	7.70
	SAEP	MFCC	AMSoftmax	400	1.16M	7.13
Vox2	ResNet-34 [30]	Spectrogram	Softmax+Contrastive	512	21.8M*	5.04
	ResNet-50 [30]	Spectrogram	Softmax+Contrastive	512	25.6M*	4.19
	x-vector	MFCC	Softmax	512	4.38M	11.64
	x-vector/LDA/PLDA	MFCC	Softmax	512	4.38M	7.87
	SAEP	MFCC	Softmax	400	1.16M	6.32
	SAEP	MFCC	AMSoftmax	400	1.16M	5.44
Vox1-E	ResNet-50 [30]	Spectrogram	Softmax+Contrastive	512	25.6M*	4.42
	x-vector	MFCC	Softmax	512	4.38M	11.62
	x-vector/LDA/PLDA	MFCC	Softmax	512	4.38M	8.18
	SAEP	MFCC	Softmax	400	1.16M	6.94
	SAEP	MFCC	AMSoftmax	400	1.16M	5.62

*Minimum estimation, actual number not provided by the reference.

the use of much larger number of parameters and more sophisticated training strategies such as contrastive loss. Furthermore, ResNet-50 takes advantage of test-time augmentation [30] for Vox1-E protocol.

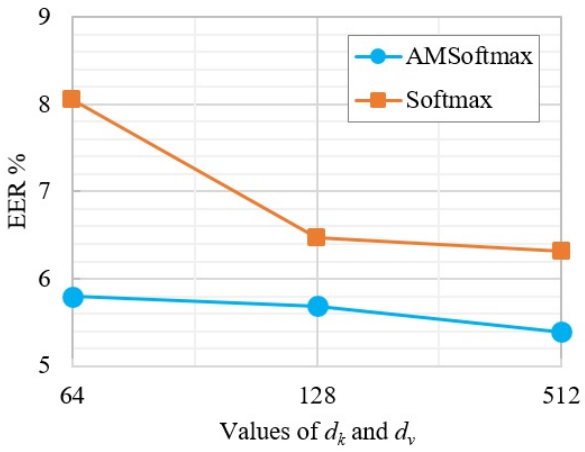

Figure 2: *Assessment of the impact of the key and value dimensions ($d_k = d_v$) on the performance of speaker embeddings in terms of EER% for SAEP architecture. The results reported for both Softmax and AMSoftmax training losses on Vox2 protocol.*

In Figure 2 we have presented the effect of key and value dimensions of $d_k = d_v$ on the performance of our proposed system for both Softmax and AMSoftmax losses on Vox2 protocol. We have tried three different dimensions 64, 128, and 512. It is notable to consider the total number of network parameters for each one of these systems. These dimensions correspond to 0.83M, 0.88M, and 1.16M total number of model parameters, respectively. In addition to these experiments, in order to show the efficiency of our architecture in terms of number of model parameters, we tried an alternative configuration with $d_{ff} = 1024$ and $d_v = d_k = 64$. This configuration achieved 7.83% EER on Vox2 protocol and Softmax loss, with only 0.45M parameters. Compared to x-vector, which offers similar performance, this alternative approach requires only almost one-tenth of the parameters required for the x-vector. This is a remarkable improvement, which justifies the superiority of our solution for resource-constraint devices, such as mobile phones.

5. Conclusions

We have presented a tandem encoder and pooling layer solely based on self-attention mechanism and position-wise feed-forward networks. They are used in an end-to-end embedding extractor for text-independent speaker verification. Preliminary results show that this fully attention based system is able to extract the time-invariant speaker features and produce discriminative representation for speakers with much fewer number of parameters compared to some other CNN and TDNN-based benchmarks. SAEP uses more than 98%, 94%, 95%, and 73% less parameters compared to VGG-M, ResNet-34, ResNet-50, and x-vector, respectively. It reveals that SAEP is superior at capturing long-range dependencies compared to other models. One reason is that the attention mechanism looks into all time instances at once so it is more flexible to extract features, which are located at longer distances from one another. For the future work, we would like to study the effects of various data augmentation strategies since it is very effective on speaker embeddings such as x-vector. We also need to study the scale of the system with more encoding layers, larger dimensions, and the addition of an appropriate back-end.

6. Acknowledgements

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

7. References

[1] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *ICASSP*. IEEE, 2014, pp. 4052–4056.

[2] P. Safari, O. Ghahabi, and J. Hernando, "From features to speaker vectors by means of restricted boltzmann machine adaptation," in *ODYSSEY*, 2016, pp. 366–371.

[3] 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.

[4] G. Bhattacharya, M. J. Alam, and P. Kenny, "Deep speaker embeddings for short-duration speaker verification," in *Interspeech*, 2017, pp. 1517–1521.

[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *ICASSP*. IEEE, 2018, pp. 5329–5333.

[6] O. Ghahabi and J. Hernando, "Restricted boltzmann machines for vector representation of speech in speaker recognition," *Computer Speech & Language*, vol. 47, pp. 16 – 29, 2018.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[9] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[10] E. Strubell, P. Verga, D. Andor, D. Weiss, and A. McCallum, "Linguistically-informed self-attention for semantic role labeling," *arXiv preprint arXiv:1804.08199*, 2018.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[12] C.-C. Chiu, T. N. Sainath, Y. Wu, and *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *ICASSP*. IEEE, 2018, pp. 4774–4778.

[13] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, "Universal transformers," *arXiv preprint arXiv:1807.03819*, 2018.

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

[15] M. India, P. Safari, and J. Hernando, "Self Multi-Head Attention for Speaker Recognition," in *Proc. Interspeech 2019*.

[16] S.-X. Zhang, Z. Chen, Y. Zhao, J. Li, and Y. Gong, "End-to-end attention based text-dependent speaker verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 171–178.

[17] F. Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, "Attention-based models for text-dependent speaker verification," *arXiv preprint arXiv:1710.10470*, 2017.

[18] 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*, pp. 74–81.

[19] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[22] L. Sifre and S. Mallat, "Rigid-motion scattering for image classification," *Ph. D. thesis*, 2014.

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

[24] J. Jin, A. Dundar, and E. Culurciello, "Flattened convolutional neural networks for feedforward acceleration," *arXiv preprint arXiv:1412.5474*, 2014.

[25] M. Wang, B. Liu, and H. Foroosh, "Factorized convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 545–553.

[26] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[27] V. Sindhwani, T. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Advances in Neural Information Processing Systems*, 2015, pp. 3088–3096.

[28] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang, "Deep fried convnets," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1476–1483.

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

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

[31] J. A. C. Nunes, D. Macêdo, and C. Zanchettin, "Am-mobilenet1d: A portable model for speaker recognition," *arXiv preprint arXiv:2004.00132*, 2020.

[32] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[33] R. H. Woo, A. Park, and T. J. Hazen, "The mit mobile device speaker verification corpus: data collection and preliminary experiments," in *2006 IEEE Odyssey-The Speaker and Language Recognition Workshop*. IEEE, 2006, pp. 1–6.

[34] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[35] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

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

[37] 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, pp. 6141–6145.

[38] B. McFee, M. McVicar, S. Balke, and *et al.*, "librosa/librosa: 0.6.3," Feb. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2564164

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.