INVESTIGATION OF SPECAUGMENT FOR DEEP SPEAKER EMBEDDING LEARNING

Shuai Wang[1,2]*, Johan Rohdin*[2]*, Oldřich Plchot*[2]*, Lukáš Burget*[2]*, Kai Yu*[1]*, Jan Černocký*[2]

[1] SpeechLab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence
Shanghai Jiao Tong University, China
[2]Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Brno, Czechia
{feixiang121976,kai.yu}@sjtu.edu.cn, {rohdin,iplchot,burget,cernocky}@fit.vutbr.cz

ABSTRACT

SpecAugment is a newly proposed data augmentation method for speech recognition. By randomly masking bands in the log Mel spectogram this method leads to impressive performance improvements. In this paper, we investigate the usage of SpecAugment for speaker verification tasks. Two different models, namely 1-D convolutional TDNN and 2-D convolutional ResNet34, trained with either Softmax or AAM-Softmax loss, are used to analyze SpecAugment's effectiveness. Experiments are carried out on the Voxceleb and NIST SRE 2016 dataset. By applying SpecAugment to the original clean data in an on-the-fly manner without complex off-line data augmentation methods, we obtained 3.72% and 11.49% EER for NIST SRE 2016 Cantonese and Tagalog, respectively. For Voxceleb1 evaluation set, we obtained 1.47% EER.

Index Terms— speaker embedding, on-the-fly data augmentation, speaker verification, specaugment

1. INTRODUCTION

In the last few years, the state-of-the-art embeddings for speaker recognition have shifted from the so-called *i*-vectors[1] produced by *generative shallow* models towards embeddings produced by discriminatively trained deep neural networks (DNN) [2, 3, 4, 5, 6, 7, 8, 9]. Such *deep speaker embeddings* models do not make as strong assumptions about the data distribution as i-vectors but require, on the other hand, much more data in order to be trained accurately.

One way to deal with the large training data requirements of DNNs is by *data augmentation*. Typically, data augmentation means that corrupted versions of the training data are created (for example by adding noise) and used as additional training data. By artificially increasing the amount and variability of the training data augmentation can reduce the risk of overfitting as well as mitigate the effects of domain mismatch between the training and evaluation data.

Data augmentation has proven to improve the training of DNNs in a large variety of tasks, for example, image recognition[10, 11] and speech recognition[12, 13]. For speaker recognition, data augmentation was shown to be very effective for training DNN based speaker embedding extractors in the Kaldi *x-vector* [8] recipe. In that work, *noise*, *reverberation*, *music* and *babble* were used to augment the original training data. This has become the most popular data augmentation strategy for deep speaker embedding learning.

Despite its effectiveness, this approach has disadvantages. First, it is unclear whether the choice of augmentations should depend on the application. Should we, for example, include babble noise in the training data if babble is not expected in the application? Second, this data augmentation strategy is troublesome (though not impossible) to do on-the-fly, i.e., to do during the training instead of before it. On-the-fly data augmentation greatly improves the flexibility of the training as well as saves disk space. Therefore, it is desirable to find simpler augmentation approaches that are not based on any specific corruption categories and that easily can be applied on-the-fly.

Recently, one approach for simple on-the-fly augmentation called *spectral augmentation* (SpecAugment) was proposed for speech recognition, where it showed very promising results[14]. In this paper, we apply this approach to speaker recognition. We investigate its effect on different architectures and loss functions for both 8K telephone data (NIST SRE16) and 16K video data (Voxceleb).

2. SPECAUGMENT

SpecAugment is a simple data augmentation method for speech recognition proposed by Google in [14]. The augmentation is directly applied to the feature inputs of a neural

The work is done during Shuai's internship at BUT speech group.

Shuai and Kai are supported by the Major Program of Science and Technology Commission of Shanghai Municipality (STCSM) 17JC1404104 and the China NSFC project (No. U1736202). Johan, Oldřich, Lukáš and Jan are supported by the Czech National Science Foundation (GACR) project "NEUREM3" No. 19-26934X, Czech Ministry of Interior project No. VI20152020025 "DRAPAK", and Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602".

Authorized licensed use limited to: Brno University of Technology. Downloaded on May 13,2020 at 08:47:45 UTC from IEEE Xplore. Restrictions apply.

network, which means it can be easily used as an on-the-fly augmentation method. To increase the system's robustness to possible loss of frequency or small segments of speech, the following deformations on the log mel features are adopted:

- Frequency masking: f consecutive frequency channels $[f_0, f_0 + f)$ will be masked (i.e., set to 0), where f is first chosen from a uniform distribution from 0 to F, where F is a predefined hyper-parameter and f_0 is randomly chosen from $[0, \nu - f)$, ν is the number of frequency channels (the feature dimension)
- Time masking: t consecutive frames $[t_0, t_0 + t)$ will be masked, where t is first chosen from a uniform distribution from 0 to T, where T is a predefined hyper-parameter and t_0 is randomly chosen from $[0, \tau - t)$, τ is the length of the feature sequence (number of frames)

In [14], a third deformation method called "time warping" was proposed. In our initial experiments, this method was not effective and therefore will not be considered in this paper. It should be noted that with $F = 0$, frequency masking is equivalent to *drop-out*[15] of *one unit* in the input layer.

3. EXPERIMENTS

In this section, we investigate the effect of SpecAugment on different models with different training criteria and compare it with the standard *Kaldi style augmentation*. Experiments are carried out on Voxceleb and NIST SRE 16.

3.1. Models

To better show the impact of SpecAugment on the deep speaker embedding learning, two typical DNN architectures, TDNN and ResNet34 are used in our experiments. These two architectures are nowadays used by most of the researchers for speaker embedding learning[8, 7] to achieve state-of-the-art results. A second reason why we choose these two architectures is that the TDNN uses 1-D convolution and the ResNet34 uses 2-D convolution and it is interesting to examine the effect of the SpecAugment on these two different models.

TDNN: For the TDNN system, we follow the standard x-vector structure[8], which is shown in table 1. Five time delay layers are used for frame-level feature learning, followed by a statistics pooling layer, aggregating the frame-level features to segment-level representations. The output from the layer "segment1" is extracted as speaker embeddings.

ResNet34: For the ResNet34 system, we follow the standard 34-layer ResNet architecture[16]. This network uses 2-dimensional features as input and processes them using 2-dimensional convolution. Inspired by x-vector topology, both mean and standard deviation are used as statistics. The detailed topology of the used ResNet is shown in Table 2, the output of "Dense" layer is used as the speaker embedding.

Table 1. Architecture of TDNN based speaker embedding extractor, T denotes the sequence length, N is the number of speakers [8]

Layer	**Layer context**	**Input $\times$ output**
frame1	$[t-2, t+2]$	200×512
frame2	$\{t-2, t, t+2\}$	1536×512
frame3	$\{t-3, t, t+3\}$	1536×512
frame4	$\{t\}$	512×512
frame5	$\{t\}$	512×1500
stats pooling	$[0, T]$	1500×3000
segment1	$\{0\}$	3000×512
segment2	$\{0\}$	512×512
projection	$\{0\}$	$512 \times$ N

Table 2. Architecture of ResNet34 based speaker embedding extractor. T denotes the sequence length, N is the number of speakers.

Layer name	**Structure**	**Output**
Input	–	$40 \times T \times 1$
Conv2D-1	3×3, Stride 1	$40 \times T \times 32$
ResNetBlock-1	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$, Stride 1	$40 \times T \times 32$
ResNetBlock-2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$, Stride 2	$20 \times \frac{T}{2} \times 64$
ResNetBlock-3	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6$, Stride 2	$10 \times \frac{T}{4} \times 128$
ResNetBlock-4	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$, Stride 2	$5 \times \frac{T}{8} \times 256$
StatsPooling	–	10×256
Flatten	–	2560
Dense	–	256
Projection	–	N

3.2. Loss Functions

To train a good speaker embedding extractor, proper training criterion should be selected. In this paper, Softmax and Additive Angular Margin Softmax (AAM-Softmax) will be investigated.

Softmax: As a commonly used classification loss for training the speaker discriminative DNNs, Softmax loss can be formulated as:

$$L_{\text{softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\mathbf{W}_{y_i}^T\mathbf{x}_i+\mathbf{b}_{y_i}}}{\sum_{j=1}^{c}e^{\mathbf{W}_j^T\mathbf{x}_i+\mathbf{b}_j}} \tag{1}$$

where N is the batch size, c is the number of classes. $\mathbf{x}_i \in \mathbb{R}^d$ denotes the i-th input of samples to the projection layer and y_i is the corresponding label index. $\mathbf{W} \in \mathbb{R}^{d\times c}$ and $\mathbf{b} \in \mathbb{R}^c$ are the weight matrix and bias in the projection layer.

AAM-Softmax: To explicitly enforce the similarity for intra-class samples and the diversity for inter-class samples, several

Authorized licensed use limited to: Brno University of Technology. Downloaded on May 13,2020 at 08:47:45 UTC from IEEE Xplore. Restrictions apply.

variants based on the Softmax loss function have been proposed, including the Angular Softmax[17, 9], Additive Margin Softmax loss (AM-Softmax)[18] and Additive Angular Softmax (AAM-Softmax) loss[19]. A thorough comparison of different loss functions was carried out in our previous paper[20]. Here, we chose the best-performing of them, namely the AAM-Softmax, which is defined as

$$L_{\text{AAM-Softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i}+m))}}{Z} \qquad (2)$$

where $Z = e^{s(\cos(\theta_{y_i,i}+m))} + \sum_{j=1,j\neq i}^{c} e^{s(\cos(\theta_{j,i}))}$, $\theta_{j,i}$ is the angle between the column vector $\mathbf{W}_j$ and $\mathbf{x}_i$, where both $\mathbf{W}_j$ and $\mathbf{x}_i$ are normalized. s is a scaling factor and m is a hyperparameter controling the margin.

3.3. Data preparation

We follow the standard Kaldi recipe, which first removes all the silent frames using an energy-based voice activity detector (VAD), and then cut the obtained utterances to chunks with duration ranging from 2s to 4s. 40-dimensional Fbanks are extracted as features. For the kaldi augmentation baseline, besides the clean training utterances, we follow the kaldi recipe and consider the following different 4 corruption categories: 1) Reverberated using RIRs, 2) Augmented with Musan noise, 3) Augmented with Musan music, and 4) Augmented with Musan babel. Then a subset of the augmented data with similar size of the clean data is combined with the original dataset, resulting in a final training set which size is approximately doubled compared to the original data.

3.4. Experimental setups

Both the TDNN and ResNet models are trained using the stochastic gradient descent (SGD) optimizer with momentum set to $1e-4$. The learning rate is initially set to 0.1 and gradually reduced to $5e-5$ along the training process. For the AAM-Softmax used in the Voxceleb experiments, s is fixed to 40, the margin hyper-parameter m is initially set to 0.1 and gradually increased to 0.4 along the training process. The hyperparameters for the frequency masking, F, and for the time masking, T, are set to 25 and 5, respectively for all the experiments.

3.5. Experiments on NIST SRE

Following the standard settings in the literature, the training data consists of the SWBD portion and SRE portion, while the former includes Switchboard phases 2,3 and Switchboard Cellular 1,2, and the latter contains the NIST SRE 2004-2010. The *x*-vector extractors are trained on the SWBD and SRE pooled data, while the PLDA is trained only on the SRE portion. The standard SRE16 evaluation set is used to measure the performance of the proposed system, which consists of Tagalog and Cantonese subsets. The unlabeled SRE16 portion is used for the unsupervised PLDA adaption.

3.5.1. Results and Analysis

From our empirical results on NIST SRE 16, the best systems are always obtained using the Softmax loss, AAM-Softmax led to much worse results. Thus we only report the Softmax results for the SRE16 evaluation set.

Table 3. Results on SRE 2016 evaluation set, TM, FM and TFM denote time masking, frequency masking and doing both time and frequency masking, respectively. kaldi denotes the augmentation method in the kaldi recipe as described in Section 3.3.

System Configuration		Cantonese		Tagalog	
Arch.	Data Aug	EER	minDCF	EER	minDCF
TDNN	no	5.81	0.469	15.34	0.830
	kaldi	4.62	0.372	13.35	0.765
	spec_aug (TM)	5.84	0.448	14.9	0.817
	spec_aug (FM)	5.91	0.459	14.92	0.809
	spec_aug (TFM)	5.77	0.454	14.95	0.809
ResNet34	no	4.26	0.371	12.91	0.845
	kaldi	4.24	0.370	12.18	**0.742**
	spec_aug (TM)	4.10	0.361	12.72	0.845
	spec_aug (FM)	4.06	0.356	12.28	0.819
	spec_aug (TFM)	3.97	**0.352**	12.02	0.806
	2x epochs (TFM)	**3.72**	0.354	**11.49**	0.795

As shown in Table 3, systems based on ResNet34 outperform the TDNN systems by a large margin. For the TDNN model, although some improvements can be observed with SpecAugment compared to the baseline with no augmentation, there is still a performance gap between the systems using SpecAugment the Kaldi augmentation.

For the ResNet34 based systems, the results show that Kaldi augmentation only slightly improves the system's performance. We can observe that the SpecAugment is more suited to the ResNet34 architecture as here, we achieve further improvements compared to the Kaldi augmentation. It is now interesting to note that both systems based on kaldi augmentation and SpecAugment are trained with the same amount of epochs, but as the Kaldi style augmentation approximately doubles the size of the training data, it is not a completely fair comparison. We have equalized the amount of training examples for the system based on the on-the-fly SpecAugment by doubling the number of training epochs. By doing this, we have reached 3.72% and 11.49% EER on Cantonese and Tagalog, respectively via simple online data augmentation. Without PLDA adaptation, the results for Kaldi augmentation is better than for SpecAugment(e.g., for ResNet34, the average EER of Cantonese and Tagalog was 10.5% for Kaldi augmentation whereas for SpecAugment it was 12.3%). Probably Kaldi augmentation deals better with domain mismatch since it introduces larger variability to the training data.

Authorized licensed use limited to: Brno University of Technology. Downloaded on May 13,2020 at 08:47:45 UTC from IEEE Xplore. Restrictions apply.

Table 4. Results on Voxceleb Dataset, TM, FM and TFM denote time masking, frequency masking and doing both time and frequency masking, respectively. kaldi denotes the augmentation method in the kaldi recipe as described in Section 3.3, systems trained with AAM-Softmax use cosine scoring as the backend, the remaining ones use PLDA as the backend.

System Configuration			voxceleb1_O		voxceleb1_E		voxceleb1_H	
Arch.	Loss function	Data Aug	EER	minDCF	EER	minDCF	EER	minDCF
TDNN	Softmax	no	3.00	0.322	3.01	0.348	5.26	0.493
		kaldi	2.54	0.301	2.83	0.314	4.88	0.462
		spec_aug (TM)	2.89	0.311	2.85	0.327	4.92	0.472
		spec_aug (FM)	2.68	0.311	2.85	0.331	4.97	0.469
		spec_aug (TFM)	2.59	0.288	2.77	0.314	4.83	0.450
	AAM-Softmax	no	2.27	0.280	2.51	0.291	4.31	0.415
		kaldi	2.07	0.258	2.35	0.274	4.11	0.400
		spec_aug (TM)	2.31	0.261	2.39	0.272	4.13	0.406
		spec_aug (FM)	2.07	0.266	2.35	0.272	4.07	0.408
		spec_aug (TFM)	1.96	0.238	2.31	0.265	4.02	0.396
ResNet34	Softmax	no	1.94	0.219	1.89	0.235	3.24	0.323
		kaldi	1.74	0.216	1.78	0.209	3.07	0.306
		spec_aug (TM)	1.85	0.240	1.86	0.229	3.19	0.318
		spec_aug (FM)	1.62	0.215	1.82	0.218	3.14	0.310
		spec_aug (TFM)	1.68	0.236	1.800	0.217	3.08	0.309
	AAM-Softmax	no	1.76	0.198	1.79	0.196	3.04	0.275
		kaldi	1.54	**0.156**	**1.55**	**0.174**	2.73	**0.259**
		spec_aug (TM)	1.59	0.171	1.67	0.185	2.91	0.275
		spec_aug (FM)	**1.47**	0.188	1.64	0.187	2.84	0.257
		spec_aug (TFM)	1.52	0.201	1.58	0.183	**2.68**	0.274

3.6. Experiments on Voxceleb

Our systems (both NN and PLDA) are trained on the Voxceleb2 development set. This set contains 5994 speakers with 1092009 utterances. For the evaluation, we use the standard Voxceleb1 dataset and report results on cleaned trial lists which can be found online.[1]

3.6.1. Results and Analysis

For the experiments on the voxceleb dataset, besides different model architectures, different loss functions introduced in Section 3.2 are also considered. As shown in Table 4, and as for the SRE16 evaluation set, ResNet34 consistently outperforms the TDNN systems. On Voxceleb2, we can further observe that training with AAM-Softmax consistently outperforms the standard Softmax. We speculate that the reason for such a good performance of the AAM-Softmax can be a consistency between the training and evaluation data as there is most likely no domain mismatch between the Voxceleb1 and Voxceleb2 datasets. We have not observed this gain from AAM-Softmax during the experiments with NIST SRE2016 and neither during NIST SRE2019, both of which impose challenging domain mismatch between the available training data and the actual evaluation set. Finally, when focusing on the effect of the SpecAugment, we can observe that now there is not a performance gap between SpecAugment and Kaldi augmentation for the TDNN system. With the systems based on ResNet34, we also achieve similar performance to the Kaldi style augmentation both for the SoftMax and AAM-Softmax training criterions.

4. CONCLUSION

In this paper, we investigated the usage of the SpecAugment method for speaker recognition. We provided the experimental analysis with two state-of-the-art NN architectures for speaker embedding extraction: the TDNN with 1-D convolution and Resnet34 with 2-D convolution. We experimented with different loss functions such as Softmax and AAM-Softmax and carried out the experiments on two different datasets of different domains, the Voxceleb1 and the NIST SRE16. We show that this simple online data augmentation method can achieve state-of-the-art results: for NIST SRE 2016, we obtained 3.72% and 11.49% EER for Cantonese and Tagalog, respectively. For the Voxceleb1 evaluation set, we achieved 1.47% EER.

[1] `http://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox2.html`

Authorized licensed use limited to: Brno University of Technology. Downloaded on May 13,2020 at 08:47:45 UTC from IEEE Xplore. Restrictions apply.

5. REFERENCES

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

[2] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014*. IEEE, 2014, pp. 4052–4056.

[3] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5115–5119.

[4] Shi-Xiong Zhang, Zhuo Chen, Yong Zhao, Jinyu Li, and Yifan Gong, "End-to-end attention based text-dependent speaker verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 171–178.

[5] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev 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] G. Bhattacharya, J. Alam, and P. Kenny, "Deep Speaker Embeddings for Short-Duration Speaker Verification," in *Interspeech 2017*, 08 2017, pp. 1517–1521.

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

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

[9] Zili Huang, Shuai Wang, and Kai Yu, "Angular softmax for short-duration text-independent speaker verification," *Proc. Interspeech, Hyderabad*, 2018.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

[12] J. Villalba and E. Lleida, "Unsupervised adaptation of plda by using variational bayes methods," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 744–748.

[13] Hu Hu, Tian Tan, and Yanmin Qian, "Generative adversarial networks based data augmentation for noise robust speech recognition," in *ICASSP*, Calgary, Canada, April 2018, pp. 5044–5048.

[14] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[18] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[19] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[20] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," *arXiv preprint arXiv:1906.07317*, 2019.

Authorized licensed use limited to: Brno University of Technology. Downloaded on May 13,2020 at 08:47:45 UTC from IEEE Xplore. Restrictions apply.