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

On Parameter Adaptation in Softmax-based Cross-Entropy Loss for Improved Convergence Speed and Accuracy in DNN-based Speaker Recognition

Magdalena Rybicka and Konrad Kowalczyk

AGH University of Science and Technology
Department of Electronics
30-059 Krakow, Poland
mrybicka@agh.edu.pl, konrad.kowalczyk@agh.edu.pl

Abstract

In various classification tasks the major challenge is in generating discriminative representation of classes. By proper selection of deep neural network (DNN) loss function we can encourage it to produce embeddings with increased inter-class separation and smaller intra-class distances. In this paper, we develop softmax-based cross-entropy loss function which adapts its parameters to the current training phase. The proposed solution improves accuracy up to 24% in terms of Equal Error Rate (EER) and minimum Detection Cost Function (minDCF). In addition, our proposal also accelerates network convergence compared with other state-of-the-art softmax-based losses. As an additional contribution of this paper, we adopt and subsequently modify the ResNet DNN structure for the speaker recognition task. The proposed ResNet network achieves relative gains of up to 32% and 15% in terms of EER and minDCF respectively, compared with the well-established Time Delay Neural Network (TDNN) architecture for x-vector extraction.

Index Terms: speaker recognition, deep neural networks, softmax activation functions, speaker embedding, ResNet

1. Introduction

Recently we have seen a rapid increase in popularity of speaker modeling using deep neural networks (DNNs) [1, 2, 3]. State-of-the-art solution consist in extracting a speaker embedding from the Time Delay Neural Network (TDNN), which is commonly referred to as the x-vector [2]. Extensions of the basic TDNN structure have been presented e.g. in [4, 5], while different network structures for the extraction of speaker embeddings have very recently been proposed e.g. in [1, 3, 6, 7, 8, 9]. Finding an appropriate network structure which facilitates notable improvement in speaker recognition performance is thus still an on-going research topic.

In speaker recognition, it is common to use a cross-entropy loss function with softmax activations in the last DNN layer [1, 4]. This loss function is also widely used in other tasks such as image recognition [10] or speech emotion recognition [11]. Recent modifications to such loss functions include increasing inter-class separation by an introduction of various types of margins in the angular functions [12, 13, 14]. They have been successfully applied in speaker recognition e.g. in [15, 16]. The convergence of the training process and the resulting model performance strongly depend on the selection of hyperparameters of the modified loss function, which often need to be tuned by repeating the training with different hyperparameter values. To address this issue, in [17] a cosine-based activation function called AdaCos has been proposed for the face recognition application, which adapts the scale parameter in angular softmax representation to improve the training effectiveness.

In this paper, we aim to develop a softmax-based cross-entropy loss function which adapts its hyperparameters so that they strengthen supervision at different neural network training phases. The proposed approach allows to adapt the scale and additive angular margin parameters in joint softmax-based cross-entropy loss function, resulting in a notable improvement in convergence speed of the network training and in speaker recognition accuracy. In addition, we propose a modification of the Residual Network (ResNet) [18] architecture which shows significant improvements in accuracy over the standard TDNN architecture. In evaluations, we compare the proposed parameter adaptation (ParAda) in softmax-based loss function in terms of convergence speed and accuracy for two speaker recognition systems based on TDNN and the proposed modified ResNet.

2. Softmax-based cross-entropy loss functions

In this section, we present an overview of several recently proposed modifications of a standard cross-entropy loss with softmax activation functions for DNN training, and next we propose a softmax-based loss function with adaptive parameters (ParAda) which improves discriminative capabilities of the network and accelerates its convergence. Let us first observe that the dot product between the softmax layer input vector and the weight vector can be written as $\mathbf{w}_k^T \mathbf{x}_i = \|\mathbf{w}_k\| \, \|\mathbf{x}_i\| \, \cos(\theta_{i,k})$, where $\|\mathbf{w}_k\|$ denotes the norm of the weight vector for the kth class, $k = 1, 2, ..., K$ with K being the number of speakers in the entire training set, $\|\mathbf{x}_i\|$ denotes the norm of the input vector for the ith minibatch example, $i = 1, 2, ..., N$ with N denoting the minibatch size, and $\cos(\theta_{i,k})$ denotes the cosine angular distance between vectors $\mathbf{w}_k$ and $\mathbf{x}_i$. Next, we can express the general equation for the softmax-based cross-entropy loss function as

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log P_{y_i} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{f_{y_i}}}{e^{f_{y_i}} + \sum\limits_{k=1, k \neq y_i}^{K} e^{f_{y_i,k}}} , \quad (1)$$

where y_i is the ground truth label of a training example, P_{y_i} denotes the predicted classification probability of all samples in the minibatch, while f_{y_i} and $f_{y_i,k}$ denote the target and non-target logits given respectively by

$$f_{y_i} = s(\theta_{y_i}) \, \psi(\theta_{y_i}) \, , \quad (2)$$

$$f_{y_i,k} = s(\theta_{y_i}) \, \cos(\theta_{y_i,k}) \, . \quad (3)$$

This research received financial support from the Foundation for Polish Science under grant number First TEAM/2017-3/23 (POIR.04.04.00-00-3FC4/17-00) which is co-financed by the European Union and was supported in part by PLGrid Infrastructure.

http://dx.doi.org/10.21437/Interspeech.2020-2264

In standard softmax-based cross-entropy loss, $\psi(\theta_{y_i})$ is defined as the cosine of the angle between the ith minibatch input vector and the weight vector corresponding to its ground truth label. Note that for the convenience of comparing various softmax modifications, in (2) and (3) we normalized the weight vectors for all classes such that $\|\mathbf{w}_k\| = 1$ and replaced the norm of an input vector for the true class $\|\mathbf{x}_i\|$ with a new scale variable $s(\theta_{y_i})$.

2.1. State-of-the-art fixed angular and scale functions

There are three types of modifications of the standard angular function $\psi(\theta_{y_i})$ in (2), namely the so-called Angular Softmax (AS) [12], Additive Angular Softmax (AAS) [13], and Additive Margin Softmax (AMS) [14], which can all be presented in a general form

$$\psi(\theta_{y_i}) = \cos(m_{\mathrm{AS}}\,\theta_{y_i} + m_{\mathrm{AAS}}) - m_{\mathrm{AMS}}, \tag{4}$$

where m_{AS}, m_{AAS} and m_{AMS} are the real numbers for each modification. Although proper setting of these parameters has been shown to improve the accuracy of DNN-based speaker recognition [15, 16], the disadvantage of these approaches is that parameter tuning requires time-consuming repetitions of the network training. Another approach is taken in [17] in which a fixed scale function $s(\theta_{y_i}) = s_{\mathrm{Fix}}$ is given by a constant

$$s_{\mathrm{Fix}} = \sqrt{2}\,\log(K-1), \tag{5}$$

which depends on the number of speakers K in the training, which allows to avoid scale parameter tuning.

2.2. Adaptation of the scaling parameter

In this section, we discuss a method to adapt the scale parameter $s(\theta_{y_i})$ depending on network convergence at current training iteration. This method is based on the recently proposed Adaptively Scaling Cosine Logits (AdaCos) introduced in [17] in the context of face recognition, which relies on adapting the scale function $s(\theta_{y_i})$ during the network training. As derived in [17], the scale adaptation (SAda) is given by

$$s_{\mathrm{Ada}}(\theta_{y_i}) = \begin{cases} \sqrt{2}\log(K-1) & \mathrm{iter} = 0 \\ \frac{\log(B_{\mathrm{SAda}})}{\cos(\min(\frac{\pi}{4},\Theta))} & \mathrm{iter} \geq 1 \end{cases} \tag{6}$$

where $\Theta = \mathrm{median}(\theta_{y_1}, \theta_{y_2}, ..., \theta_{y_N})$ denotes the median of angles θ_{y_i} over the entire minibatch of length N, and iter denotes the iteration index. B_{SAda} denotes the summation of exponential functions of logits for all non-corresponding classes, averaged over the entire minibatch of size N, which is given by

$$B_{\mathrm{SAda}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1,k\neq y_i}^{K} e^{\widetilde{s}_{\mathrm{Ada}}\cos(\theta_{i,k})}, \tag{7}$$

where $\widetilde{s}_{\mathrm{Ada}} = s_{\mathrm{Ada}}(\mathrm{iter}-1)$ denotes the scale parameter value calculated according to (6) in the previous iteration.

2.3. Adaptation of the margin-based angular function

In this section, we propose a method to adapt the margin parameter (MAda) in an angular function depending on network convergence state in the current iteration of the network training. The considered angular function is given by $\psi(\theta_{y_i}) = \cos(\theta_{y_i} + m_{\mathrm{Ada}})$, where m_{Ada} denotes the adaptive margin parameter. Since we aim to find a margin parameter which significantly changes the predicted classification probability P_{y_i} of all samples in the minibatch, similarly to [17], we calculate the point θ_0 where absolute gradient value of the predicted probability reaches the maximum value. It is found as point at which the second-order derivative of P_{y_i} is equal to zero, which yields the approximated relation for the angular function

$$\cos(\theta_0 + m_{\mathrm{Ada}}) = \frac{1}{s_m}\,\log\Big(\sum_{k=1,k\neq y_i}^{K} e^{s_m\cos(\theta_{i,k})}\Big), \tag{8}$$

where s_m denotes the fixed scale parameter in margin adaptation and $\theta_0 \in [0, \frac{\pi}{2}]$. In order to reflect the convergence state of the network in the current minibatch, we replace θ_0 with the median of angles θ_{y_i} over the entire minibatch, i.e., $\theta_0 = \Theta = \mathrm{median}(\theta_{y_1}, \theta_{y_2}, ..., \theta_{y_N})$, which yields the following update for margin adaptation (MAda)

$$m_{\mathrm{Ada}} = \arccos\Big(\frac{1}{s_m}\log(B_{\mathrm{MAda}})\Big) - \Theta\,, \tag{9}$$

$$B_{\mathrm{MAda}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1,k\neq y_i}^{K} e^{s_m\cos(\theta_{i,k})}. \tag{10}$$

2.3.1. Annealing strategy in margin-based angular function

Similarly to the procedure presented in [15] for the stabilization of the network training for fixed margin-based softmax, the annealing strategy can also be incorporated into the proposed softmax function with adaptive margin. The angular function with margin adaptation (MAda) then takes the form

$$\psi_{\mathrm{MAda}}(\theta_{y_i}) = \frac{1}{1+\gamma}\cos(\theta_{y_i}+m_{\mathrm{Ada}}) + \frac{\gamma}{1+\gamma}\cos(\theta_{y_i}) \tag{11}$$

where $\gamma = \max\{\gamma_{min}, \gamma_b(1+\beta\cdot\mathrm{iter})^{-\alpha}\}$ where iter is training iteration, γ_{min} is the minimum value, while γ_b, β and α are hyperparamters that control the annealing speed.

2.3.2. Lower bound on the scale parameter value

By noting that $\arccos(\cdot)$ function in (9) is only defined for arguments from the range $[-1, 1]$ and that $\theta_{i,k} \in [0, \frac{\pi}{2}]$, we can find the lower bound on the value of the scale parameter in the proposed MAda approach by solving inequality

$$-1 \leq \frac{1}{s_m}\,\log\Big(\sum_{k=1,k\neq y_i}^{K} e^{s_m\cos(\theta_{i,k})}\Big) \leq 1\,. \tag{12}$$

Assuming that all $\theta_{i,k}$ are approximately equal, we can set $\forall\, k, i\ \theta_{i,k} = \bar{\theta}$, and by replacing the values and solving (12) we obtain the lower bound on the scale parameter s_m value

$$s_m \geq [1-\cos(\bar{\theta})]^{-1}\ \log(K-1)\,. \tag{13}$$

In an ideal case, after network training, $\bar{\theta} \to \frac{\pi}{2}$ which would yield $s_m \to \log(K-1)$; while in the worst case, $\bar{\theta} \to 0$ which would impose $s_m \to \infty$. A reasonable assumption for the practical parameter setting is to assume that $\bar{\theta} = \frac{\pi}{4}$ and take a slightly higher value of s_m than the one obtained from (13).

2.4. Proposed softmax loss with parameter adaptation

This section presents the proposed method for parameter adaptation (ParAda) in softmax-based cross-entropy loss function. We focus on adapting the scale and margin parameters which affect the shape of the predicted classification probability P_{y_i} function, namely its range and the position of an inflection point. To facilitate the training process, we take the approach of gradual increasing the network training supervision. In the initial training phase, we aim to ease learning by shifting the inflection point in $\frac{\pi}{2}$ direction so that the probabilities would be high even for the relatively high values of θ_{y_i} angles. Along with further learning, the curve adaptively shifts towards the

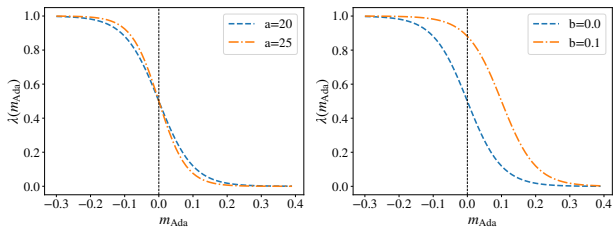

Figure 1: *The λ function with respect to the m_{Ada} margin value for $b = 0$ (left plot) and $a = 20$ (right plot).*

lower angles, thereby increasing the discriminative capability of the classification probability. We can realize it by emphasizing margin adaptation (which in an adaptive version starts from negative values), and as training progresses, more emphasis is put on the scale adaptation. To achieve this goal, we propose the following modifications of logits of the softmax-based loss function which facilitate the parameter adaptation (ParAda):

$$f_{y_i} = \lambda \cdot s_m \psi_{\mathrm{MAda}}(\theta_{y_i}) + (1-\lambda) \cdot s_{\mathrm{Ada}}(\theta_{y_i}) \cos(\theta_{y_i}), \quad (14)$$

$$f_{y_i,k} = \lambda \cdot s_m \cos(\theta_{y_i,k}) + (1-\lambda) \cdot s_{\mathrm{Ada}}(\theta_{y_i}) \cos(\theta_{y_i,k}). \quad (15)$$

A strongly desirable, integral property of ParAda is that the value of an adaptive parameter $\lambda(m_{\mathrm{Ada}})$ can be adjusted depending on the level of convergence of the trained network. The desired smooth transition between the margin adaptation (MAda) and the scale adaptation (SAda) is obtained when

$$\lambda(m_{\mathrm{Ada}}) = [1 + e^{a \cdot (m_{\mathrm{Ada}} - b)}]^{-1}, \quad (16)$$

where a and b are hyperparameters of $\lambda(m_{\mathrm{Ada}})$, whose shape for example parameter values is presented in Fig. 1. Parameter a controls the speed of switching between the two loss types, while parameter b shifts the inflection point of the function.

3. Deep neural network architectures

In this section, we outline the baseline TDNN architecture for x-vector extraction [2] and propose a set of modifications to adapt the original ResNet structure for speaker embedding extraction.

3.1. Time Delay Neural Network for x-vector extraction

The Time Delay Neural Network [2] consists of 5 delay-type layers which operate on 1D speech frames, extracting the temporal context of 15 frames. Next, the statistics pooling layer computes the mean and standard deviation of the aggregated outputs of the 5th layer for the entire utterance. The final part of the network consists of two fully connected layers followed by a softmax layer. During testing, the output of the first fully connected layer is used as the x-vector embedding.

3.2. Modified ResNet for speaker embedding extraction

In this work, we propose modifications of the Residual Network 18 (ResNet18) architecture for speaker embedding extraction with improved performance. The proposed architecture is composed of the following elements. A 2D feature vector is fed into a single 2D convolution layer with 7x7 filter size and stride of 2x2. Next, the network is composed of 4 large segments, each containing a different number of blocks which consists of 2 consecutive convolutional layers with the so-called identity shortcut connection that skips the entire block. The number of such 2-layer blocks is equal to $\{2, 2, 2, 2\}$ for all segments. The first convolutional layer of a segment downsamples the input along the feature dimension by 2, while the sizes of convolutional layer outputs for each segment are respectively given as $\{64, 128, 256, 512\}$. Then the statistics pooling layer computes the mean and standard deviation in time domain of the outputs from the convolutional segments. The pooling is followed by 2 fully connected layers with output size of 512 each, which are added before the softmax classification layer. Hereafter, we will refer to this structure as modified ResNet18 (mR18).

With regard to the existing ResNet-based speaker recognition systems, the original ResNet34 and ResNet50 network structures [18] with a fully connected layer added right before the global mean pooling layer are used in [1]. An analogous solution has been presented in [19] for ResNet18 and ResNet34. In [7] all TDNN layers have been replaced with the ResNet34 residual blocks with learnable dictionary encoding (LDE) layer [20] instead of the pooling layer. In [8] a fully connected layer has been inserted after pooling operation in ResNet50 structure.

4. Experimental results and evaluation

4.1. Experiments, datasets and evaluation measures

In this section, we present the system and datasets for 2 performed experiments. The first experiment is carried out on the VoxCeleb1 corpus [21] with training part used for the DNN, LDA and PLDA training, and test part of VoxCeleb1 used for system evaluation. In Experiment 2, the entire VoxCeleb2 corpus [1] is added to the training part of VoxCeleb1 for NN training. Each training dataset is extended by data augmentations of four types: convolution with reverberation (simulated RIRs [22] from small and medium sized rooms), augmented by adding music, ambient noises, and overlapping speech of randomly selected 3-7 speakers from the MUSAN corpus [23]. The final training dataset consists of the original training data and randomly selected subset of the augmented data, which results in 348 642 utterances (1 211 speakers) for Experiment 1 and 2 276 888 utterances (7 323 speakers) for Experiment 2.

Feature extraction, LDA, and PLDA training are performed in Kaldi toolkit [24], while DNN training and embedding extraction is performed using TensorFlow implementation [15, 25]. Input features are 64-band Mel filter bank coefficients computed using frames of 25 ms length with 10 ms overlap and mean-normalized over a 3 s window. The threshold in VAD of Kaldi is set to 3.5. System backend consists of length normalization, centering with mean of training data, LDA dimensionality reduction from 512 to 200, and PLDA scoring.

In two performed experiments, we compare the performance of the existing and the proposed softmax-based loss functions using two speaker recognition systems based on the existing TDNN and the proposed mR18 architectures. In particular, in Experiment 1 we evaluate the existing standard, Additive Angular Softmax (AAS) [13] with scale equal to 30 and margin set to 0.3, Fixed Scale parameter as given by (5) [17], and adaptive scale (SAda) that is equivalent to AdaCos [17], and compare their results with the proposed adaptive margin (MAda) in the angular function as described in Sec. 2.3 and the proposed parameter adaptation (ParAda) described in Sec. 2.4 for 4 different transitions in the loss function. In Experiment 2 we evaluate only the selected 4 methods namely the standard, AAS [13], AdaCos [17], and the proposed ParAda with $a = 20$ and $b = 0.0$ on the larger training dataset. The maximum number of epochs for network training in Experiment 1 is set to 4, while in Experiment 2 it is set to 8. In order to satisfy (13), s_m is set to 30 and 35 in Experiments 1 and 2, respectively, while in both experiments the annealing function parameters are set as $\gamma_{min} = 0$, $\gamma_b = 1000$, $\beta = 0.00001$ and $\alpha = 5$.

Table 1: *Accuracy in terms of EER and minDCF, and approximate network convergence time (in Epoch) for TDNN and mR18 based speaker recognition systems in Experiment 1.*

	Softmax	EER [%]	minDCF	Epoch
TDNN	Standard	5,23	0,479	1,94
	AAS [13]	4,61	0,432	3,52
	Fixed Scale [17]	4,75	0,512	3,11
	SAda (AdaCos [17])	4,75	0,453	2,72
	MAda	4,70	0,439	3,11
	ParAda (a=20,b=0)	**4,29**	0,422	**1,09**
	ParAda (a=25,b=0)	4,30	0,455	1,56
	ParAda (a=20,b=0.1)	4,32	**0,418**	2,14
	ParAda (a=25,b=0.1)	4,47	0,439	1,29
mR18	Standard	4,17	0,445	2,87
	AAS [13]	3,73	0,407	2,77
	Fixed Scale [17]	4,23	0,463	2,01
	SAda (AdaCos [17])	3,93	0,446	1,68
	MAda	3,83	0,389	3,16
	ParAda a=20,b=0	3,62	0,418	1,34
	ParAda a=25,b=0	3,69	0,408	1,85
	ParAda a=20,b=0.1	3,54	**0,377**	1,67
	ParAda a=25,b=0.1	**3,49**	0,395	**1,14**

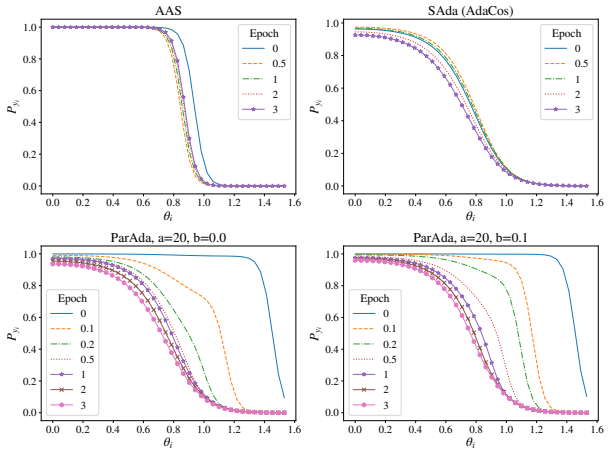

Figure 2: *Predicted probability curves $P_{y_i}(\theta_{y_i})$ at different training stages (in epochs) for TDNN system in Experiment 1.*

As evaluation metrics, we select the Equal Error Rate (*EER*) and minimum Detection Cost Function (*minDCF*) with parameters set as $C_{miss} = C_{fa} = 1$ and $P_{tar} = 0.01$. The phase of the neural network training is presented in epoch defined as one pass of all training data through the network.

4.2. Results and discussion

Table 1 presents the results of Experiment 1 for the TDNN and the proposed mR18 based speaker recognition systems. As can be clearly observed, existing modifications to the softmax-based loss function improve the EER and minDCF results, however, an increase in accuracy comes at a cost of a lower convergence speed. On the other hand, the proposed ParAda approach clearly outperforms the existing methods in terms of both the accuracy and convergence speed. In particular, the convergence speed can be increased by 2 times while the gain in EER and minDCF can also increase significantly when compared with the gains of the existing methods over the standard softmax function. Concerning the existing methods, the adaptive method known as AdaCos offers an increase in network convergence, however, its accuracy performance is lower than for the AAS method with a fixed margin value. Additional insight into the network convergence can be obtained from Fig. 2, which shows that AAS and AdaCos change the predicted probability curves only slightly during network training. In contrast, the proposed ParAda (for both considered cases) eases the network training at early training stages by shifting the probability curves to the right, and next it makes the logits more discriminative at the latter stages of NN training to enhance classification ability. Less strict supervision results from the negative margin value that controls λ parameter at the initial stage. Exceeding zero value by the margin brings about stricter classification requirements.

Table 2: *Accuracy in terms of EER and minDCF, and approximate network convergence time (in Epoch) for TDNN and mR18 based speaker recognition systems in Experiment 2.*

Network	Softmax	EER [%]	minDCF	Epoch
TDNN	Standard	3,06	0,338	3,10
	AAS [13]	2,57	0,289	7,83
	AdaCos [17]	2,44	0,276	7,53
	ParAda	**2,32**	**0,257**	5,76
mR18	Standard	2,07	0,286	2,82
	AAS [13]	2,12	**0,274**	6,77
	AdaCos [17]	1,94	0,335	5,11
	ParAda	**1,72**	**0,280**	3,51

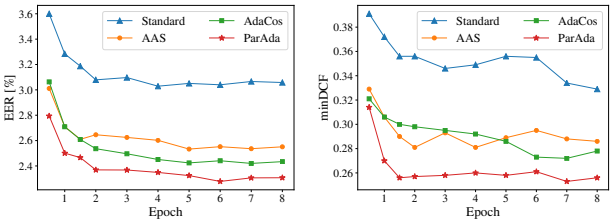

Figure 3: *EER and minDCF results at different stages of training (in epochs) for TDNN-based system in Experiment 2.*

Table 2 presents the results of Experiment 2 for four selected methods. In general, a similar trend can be observed for both network types, with ParAda clearly outperforming the other approaches in terms of accuracy and network convergence speed for the TDNN, and clearly outperforming the existing approaches in terms of the EER and convergence speed for the mR18 method (note that in case of this network, AAS achieved comparable minDCF result). For the methods studied in Experiment 2, in Fig. 3 we show speaker recognition accuracy at different stages of the network training. As can be observed, the proposed ParAda facilitates much faster network convergence at the very early part of the network training, which allows to reach high accuracy very quickly. In contrast, the compared existing approaches converge much more slowly, often not reaching the accuracy of the proposed ParAda approach.

In both experiments, the mR18 significantly outperforms the TDNN. Therefore, we can assess the proposed structure as an interesting alternative for DNN-based speaker recognition.

5. Conclusions

In this paper, we have proposed a softmax-based cross-entropy loss function with adaptive parameters (ParAda) which significantly improves speaker recognition accuracy and neural network training convergence speed compared with state-of-the-art alternatives. In addition, we have shown that the proposed modified ResNet-based architecture brings about large improvement in speaker recognition over the TDNN-based x-vector system.

6. References

[1] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep Speaker Recognition," *Proc. Interspeech 2018*, pp. 1086–1090, 2018.

[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] M. K. Nandwana, J. van Hout, C. Richey, M. McLaren, M. A. Barrios, and A. Lawson, "The VOiCES from a Distance Challenge 2019," in *Proc. Interspeech 2019*, 2019, pp. 2438–2442.

[4] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker Recognition for Multi-speaker Conversations Using X-vectors," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5796–5800.

[5] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks," in *Proc. Interspeech 2018*, 2018, pp. 3743–3747.

[6] S. Novoselov, A. Gusev, A. Ivanov, T. Pekhovsky, A. Shulipa, G. Lavrentyeva, V. Volokhov, and A. Kozlov, "STC Speaker Recognition Systems for the VOiCES from a Distance Challenge," in *Proc. Interspeech 2019*, 2019, pp. 2443–2447.

[7] 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. Interspeech 2019*, 2019, pp. 2468–2472.

[8] J. Huang and T. Bocklet, "Intel Far-Field Speaker Recognition System for VOiCES Challenge 2019," in *Proc. Interspeech 2019*, 2019, pp. 2473–2477.

[9] D. Cai, X. Qin, W. Cai, and M. Li, "The DKU System for the Speaker Recognition Task of the 2019 VOiCES from a Distance Challenge," in *Proc. Interspeech 2019*, 2019, pp. 2493–2497.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[11] K. Han, D. Yu, and I. Tashev, "Speech Emotion Recognition Using Deep Neural Network and Extreme Learning Machine," in *INTERSPEECH*, 2014, pp. 223–227.

[12] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "SphereFace: Deep Hypersphere Embedding for Face Recognition," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[13] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive Margin Softmax for Face Verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, p. 926–930, Jul 2018.

[15] Y. Liu, L. He, and J. Liu, "Large Margin Softmax Loss for Speaker Verification," in *Proc. Interspeech 2019*, 2019, pp. 2873–2877.

[16] Y. Li, F. Gao, Z. Ou, and J. Sun, "Angular Softmax Loss for End-to-end Speaker Verification," *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 190–194, 2018.

[17] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[19] Z. Chen, Z. Ren, and S. Xu, "A Study on Angular Based Embedding Learning for Text-independent Speaker Verification," *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Nov 2019.

[20] W. Cai, J. Chen, and M. Li, "Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System," *Odyssey 2018 The Speaker and Language Recognition Workshop*, Jun 2018.

[21] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A Large-Scale Speaker Identification Dataset," *Proc. Interspeech 2017*, pp. 2616–2620, 2017.

[22] *Room Impulse Response and Noise Database*, accessed March 9, 2020. [Online]. Available: http://www.openslr.org/28/

[23] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," IEEE Signal Processing Society, Tech. Rep., 2011.

[25] H. Zeinali, L. Burget, J. Rohdin, T. Stafylakis, and J. H. Cernocky, "How to Improve Your Speaker Embeddings Extractor in Generic Toolkits," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6141–6145, 2018.