Interspeech 2022
18-22 September 2022, Incheon, Korea

DF-ResNet: Boosting Speaker Verification Performance with Depth-First Design

Bei Liu, Zhengyang Chen, Shuai Wang, Haoyu Wang, Bing Han, Yanmin Qian[†]

MoE Key Lab of Artificial Intelligence, AI Institute
X-LANCE Lab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{beiliu, zhengyang.chen, feixiang121976, fayuge, hanbing97, yanminqian}@sjtu.edu.cn

Abstract

Embeddings extracted by deep neural networks have become the state-of-the-art utterance representation in speaker verification (SV). Despite the various network architectures that have been investigated in previous works, how to design and scale up networks to achieve a better trade-off on performance and complexity in a principled manner has been rarely discussed in the SV field. In this paper, we first systematically study model scaling from the perspective of the depth and width of networks and empirically discover that *depth is more important than the width of networks for speaker verification task*. Based on this observation, we design a new backbone constructed entirely from standard convolutional network modules by significantly increasing the number of layers while maintaining the network complexity following the depth-first rule and scale it up to obtain a family of much deeper models dubbed DF-ResNets. Comprehensive comparisons with other state-of-the-art systems on the Voxceleb dataset demonstrate that DF-ResNets achieve a much better trade-off than previous SV systems in terms of performance and complexity.

Index Terms: speaker verification, model scaling, performance and complexity

1. Introduction

Speaker verification aims to verify a person's identity according to his or her voice characteristics. In recent years, the thriving of deep learning has led to roaring success in the SV field [1, 2]. To further improve the performance and robustness of SV systems, researchers have made great efforts in different aspects, including network backbones [3, 4, 5, 6, 7, 8], pooling mechanisms [9, 10, 11, 12] and loss functions [13, 14].

Concerning network backbones, diverse architectures have proliferated, such as 1-D and 2-D convolution neural networks, self-attention networks. However, most design choices in previous works are ad-hoc and heuristic. Whether a principled method peculiar to the SV task exists to design architectures with better performance-complexity trade-off has been rarely discussed. For ResNet-based SV systems, people generally follow the same scaling-up rules as [15]. [5] proposes two models, ECAPA-TDNN (C=512) and ECAPA-TDNN (C=1024) by simply doubling the number of channels. Still, in terms of the depth and width of a network, which dimension plays a more critical role in the SV task is not well understood. We argue that the current scaling-up rules used in DNN-based SV systems are not optimal, there should be a more principled and exclusive way to design and scale up networks so that better performance and network complexity can be achieved for the SV task.

[†] corresponding author

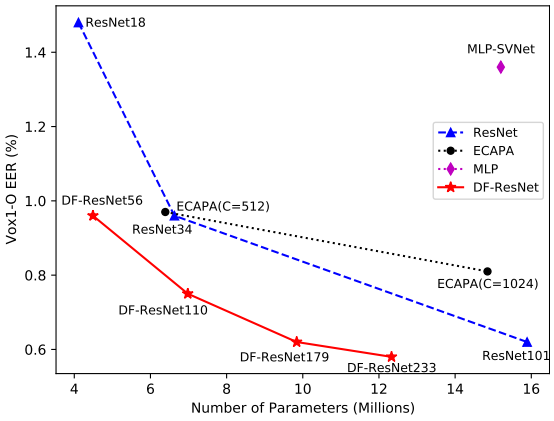

Figure 1: *Vox1-O EER vs. Model Size*

In this paper, we first systematically study the effect of depth and width on the performance of SV systems based on the family of ResNets in [15]. The empirical results show that *depth is more important than the width of networks for speaker verification task*. Based on this observation, we propose a depth-first rule and present a new baseline model by significantly deepening ResNet while maintaining its complexity. Then a family of much deeper models, namely DF-ResNets, is constructed by fixing the width and increasing depth in a specific ratio. Figure 1 summarizes the Voxceleb performance where our proposed DF-ResNets achieve the best trade-off on performance and model size than previous SV systems including two strong baselines (ResNet [4] and ECAPA-TDNN [5]).

2. Motivation

In this section, we systematically study the effect of depth and width of ResNet on the SV system performance. Firstly, we define the depth of a network as the number of layers in a network, while the width of a network refers to the number of channels in feature maps. ResNet is one of the most popular architectures used in the SV field [4, 16, 17, 18, 19, 20, 21, 22, 23, 24], different variants of which have been explored. For example, [4] reduces the number of channels in each stage to half of the original ResNet [15] due to the memory limit. On the other hand, [17] adopts a thin-ResNet trunk architecture where the block number of four stages is decreased to [2, 3, 3, 3] and the channel expansion ratio is reduced from 4 to 2. In addition, [16] utilizes the standard ResNet in the experiments. These scaling-up choices are ad-hoc and heuristic. How the depth and width of ResNet affect SV systems' performance is still not well understood. In particular, it is not yet clear what role each dimension

10.21437/Interspeech.2022-484

Table 1: *The effect of depth and width of ResNet on the SV system performance. Equal error rate (EER) is reported on Voxceleb1.*

System	Depth (# Layers)	Width (# channels)	# Params	FLOPs	Vox1-O	Vox1-E	Vox1-H
ResNet34	34	[32, 64, 128, 256]	6.63M	4.63G	0.96	1.01	1.86
ResNet34 [16]	34	[64, 128, 256, 512]	21.54M	18.46G	0.85	1.05	1.82
ResNet101	101	[32, 64, 128, 256]	15.89M	10.07G	0.62	0.80	1.48

might play in the SV task.

In experiments, we implement ResNet34 and ResNet101 for comparison. As Table 1 shows, when doubling the number of channels in ResNet34, the number of parameters and FLOPs significantly increases by 3.2x and 3.9x, respectively. However, the performance gains are very limited. On the other hand, when fixing width and increasing the number of layers from 34 to 101, we can obtain the relative improvements in EER by 35.4%, 20.7%, 20.4% on Vox1-O, Vox1-E and Vox1-H, respectively. Moreover, ResNet101 surprisingly achieves much better performance than the standard ResNet34 with 26% fewer parameters and 45% fewer FLOPs, which reveals that more performance improvements can be achieved by deepening ResNet34 than widening it. These empirical results lead us to the following observation:

Observation - Depth is more important than the width of networks for speaker verification task.

3. Depth-First ResNets

Based on the above observation, we propose the depth-first design rule to deepen ResNet18 into DF-ResNet56 (Depth-First ResNet56) while maintaining the model complexity. Subsequently, a new family of DF-ResNets is constructed by scaling up DF-ResNet56 in a specific manner. Figure 2 schematically depicts the process of converting ResNet18 to DF-ResNet56. Table 2 presents the changes of parameter number, FLOPs and performance throughout the roadmap from ResNet18 to DF-ResNet56.

3.1. Depth-First Design Rule

As mentioned above, it is empirically observed that more performance gains can be obtained by deepening ResNet than widening it for the SV task. We assume that largely increasing the depth of ResNet should result in performance improvements. Therefore, we propose the depth-first design rule that significantly deepens ResNet while maintaining the network complexity. Note that deepening ResNet does not imply increasing the number of layers directly. The key is maintaining the complexity of a network while deepening it. In terms of the complexity of a network, parameter numbers and FLOPs are considered. We make several design choices to achieve this goal. And by applying them to ResNet18, a new backbone DF-ResNet56 is built. The trajectory is provided in the following section.

3.2. A Roadmap from ResNet18 to DF-ResNet56

In this section, we present a roadmap going from ResNet18 to DF-ResNet56 by adopting the proposed depth-first idea.

basicblock $\rightarrow$ bottleneckblock: Our starting point is a ResNet18 which consists of 4 stages where each contains 2 basic blocks. Firstly, we replace the basic block with the bottleneck block, as shown in Figure 2 (A). The number of channels is set to [32, 64, 128, 256] for 4 stages, respectively. The layer number of the resulting network is 26 ([2, 2, 2, 2]). Surprisingly, the performance worsens even though the parameter number doubles and FLOPs are increased to 2.93G, as Table 2 presents. It indicates that the original bottleneck block in ResNet is not computationally efficient, which urges the necessity of re-designing it.

Table 2: *The roadmap from ResNet18 to DF-ResNet56 and the corresponding changes of parameter number, FLOPs and performance.*

System	# Params	FLOPs	Vox1-O
ResNet18	4.11M	2.22G	1.48
basicblock $\rightarrow$ bottleneckblock	8.74M	2.93G	1.68
conv2d $\rightarrow$ depthwise conv2d	7.18M	1.75G	1.96
invert dimension	2.89M	1.94G	2.20
separate downsampling	3.14M	1.41G	1.65
increase layer number	4.49M	2.66G	0.96

conv2d $\rightarrow$ depthwise conv2d: In the original bottleneck block of ResNet, the standard 3x3 2-dimensional convolution operator is adopted. In order to reduce the parameter number, we attempt to substitute the standard 3x3 convolution with depthwise convolution [25] (Figure 2 (B)), which is a special case of grouped convolution where the number of groups equals the number of channels. From Table 2, we can see that this change reduces the parameter number to 7.18M and FLOPs to 1.75G, resulting in further performance degradation to 1.96.

invert dimension: Although 1.6x reduction in FLOPs is achieved from previous step, the parameter number is still large (7.18M). Inspired by the inverted bottleneck design in [26], we firstly create an inverted bottleneck block by moving up 1x1 convolution with 128 channels to the first place and moving down 1x1 convolution with 32 channels to the third place (Figure 2 (B) to (C)). In addition, the channel number of depthwise convolution is increased from 32 to 128. This is a crucial step that significantly reduces the parameter number to 2.89M at a slight cost of FLOPs. The performance temporarily reaches the highest point 2.20.

separate downsampling: In the original ResNet, the spatial downsampling is achieved by using 3×3 convolution with stride 2 and 1×1 convolution with stride 2 at the shortcut connection in the residual block at the start of each stage (Figure 2 (A)). Inspired by [27], we utilize a separate downsampling layer which consists of a 3×3 convolution layer with stride 2 and padding 1 followed by a batchnorm (Figure 2 (D)). This separate layer is placed after each stage except for the last one to achieve the same spatial resolution downsampling as the original ResNet, as depicted in Table 3. This modification can significantly improve EER from 2.20 to 1.65 with a slight increase in parameter and a great decrease in FLOPs.

increase layer number: After the above changes, we reduce the parameter number from 4.11M (ResNet18) to 3.14M and FLOPs from 2.22G to 1.41G. It is ready to increase the number of layers. Different from the original stage compute ratio in ResNet, we follow the principle proposed in [27] and adjust the number of blocks in each stage from [2, 2, 2, 2] in

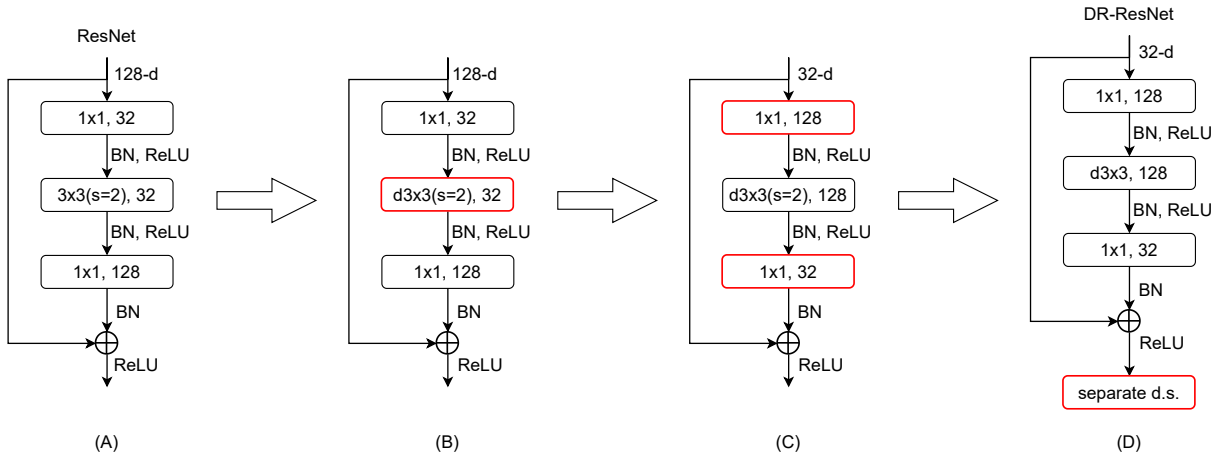

Figure 2: ***A-D:*** *The roadmap from ResNet18 to DF-ResNet56.* ***A:*** *The bottleneck block in the original ResNet.* ***B:*** *Replace the standard convolution with depthwise convolution.* ***C:*** *Swap the position of 1x1 convolution with 32 channels and 1x1 convolution with 128 channels. Meanwhile, increase the channel number of depthwise convolution from 32 to 128.* ***D:*** *Separate downsampling layer from the residual block. separate d.s. stands for separate downsampling.*

Table 3: *Detailed architecture and complexity comparison of our proposed DF-ResNet56 and ResNet18. Separate d.s. refers to the separate downsampling layer. GSP represents global statistical pooling.*

Stage	**ResNet18**	**DF-ResNet56**
conv1	$3 \times 3, 32$, stride 1	$3 \times 3, 32$, stride 1
res2	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ \text{d}3 \times 3, 128 \\ 1 \times 1, 32 \end{bmatrix} \times 3$
separate d.s.	—	$3 \times 3, 64$, stride 2
res3	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ \text{d}3 \times 3, 256 \\ 1 \times 1, 64 \end{bmatrix} \times 3$
separate d.s.	—	$3 \times 3, 128$, stride 2
res4	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ \text{d}3 \times 3, 512 \\ 1 \times 1, 128 \end{bmatrix} \times 9$
separate d.s.	—	$3 \times 3, 256$, stride 2
res5	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 1024 \\ \text{d}3 \times 3, 1024 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
pooling	GSP	GSP
FC	(5120, 256)	(5120, 256)
# params	4.11×10^6	4.49×10^6
FLOPs	2.22×10^9	2.66×10^9

ResNet18 to [3, 3, 9, 3]. This step significantly improves from 1.65 to 0.96, exceeding the original ResNet18 by a large margin under similarly-sized parameters and FLOPs. This brings us to the final model, namely DF-ResNet56.

Summary: Throughout the above roadmap, we successfully deepen ResNet18 into DF-ResNet56 while maintaining the network complexity. The comparison of detailed structure and network complexity is presented in Table 3. It is noteworthy that we do not invent any new operators during this process. By merely adopting the standard convolutional modules discussed in previous literature, our DF-ResNet56 outperforms ResNet18 by a significant margin (**35%** relative improvement) under similar complexity. Next, we will scale up our DF-ResNet56 by further increasing depth in a specific ratio and obtaining a new family of much deeper models.

3.3. Construct A Family of DF-ResNets

In this part, we construct different DF-ResNet variants which align with ResNet18/34/101. The variants only differ in the layer number. During the scaling up process, the channel number C is fixed while the number of blocks B in each stage is increased gradually. Finally, we obtain the corresponding DF-ResNet56/110/179/233, a much deeper model family. The configurations are summarized below:

- DF-ResNet56: C=[32, 64, 128, 256], B=[3, 3, 9, 3]
- DF-ResNet110: C=[32, 64, 128, 256], B=[3, 3, 27, 3]
- DF-ResNet179: C=[32, 64, 128, 256], B=[3, 8, 45, 3]
- DF-ResNet233: C=[32, 64, 128, 256], B=[3, 8, 63, 3]

4. Experimental Setup

4.1. Dataset

Our experiments are conducted on Voxceleb1&2 [28, 29] datasets. Voxceleb2 development set used for training, which contains 1,092,009 utterances from 5994 speakers. The whole Voxceleb1 is used for testing. In addition, three data augmentation techniques are utilized: online data augmentation [30] with MUSAN [31] and RIR dataset [32], specaugment [33] and speed perturb [34] with 0.9 and 1.1 times speed changes to treble the number of speakers.

4.2. Implementation Details

The input features are 80-dimensional Fbank. The frame length is 25ms and the frame shift is 10ms. We randomly crop a 200-frame chunk from one utterance for training. Cosine similarity with (AS-Norm) [35, 36] is used for scoring. Performance is reported on EER and MinDCF with the settings of $P_{target} = 0.01$ and $C_{FA} = C_{Miss} = 1$. All systems are trained using the AdamW [37] optimizer with a weight decay of 0.05. AAM-softmax [14] with a margin of 0.2 and a scale of 32 is used as the loss function. We present a comprehensive comparison with previous works by re-implementing popular models in SV:

Table 4: *EER and MinDCF results of previous systems and our proposed DF-ResNets on the Voxceleb1 dataset.*

System	Architecture	# Params	FLOPs	Voxceleb-O		Voxceleb-E		Voxceleb-H	
				EER(%)	MinDCF	EER(%)	MinDCF	EER(%)	MinDCF
ResNet18	ResNet	4.11M	2.22G	1.48	0.1737	1.52	0.1751	2.72	0.2444
ResNet34		6.63M	4.63G	0.96	0.0885	1.01	0.1206	1.86	0.1769
ResNet101		15.89M	10.07G	0.62	0.0633	0.80	0.0880	1.48	0.1431
ECAPA(C=512)	TDNN	6.39M	1.05G	0.97	0.1358	1.22	0.1410	2.31	0.2187
ECAPA(C=1024)		14.85M	2.67G	0.81	0.1464	1.01	0.1155	2.04	0.2101
SAEP	Transformer	20.54M	5.92G	2.91	0.3486	2.87	0.3289	4.75	0.4459
GCSA		47.25M	13.47G	1.96	0.2654	2.07	0.2373	3.65	0.3687
MLP-SVNet	MLP	15.20M	4.47G	1.36	0.1461	1.46	0.1552	2.49	0.2287
DF-ResNet56	ResNet (ours)	4.49M	2.66G	0.96	0.1025	1.09	0.1219	1.99	0.1841
DF-ResNet110		6.98M	5.15G	0.75	0.0700	0.88	0.1002	1.64	0.1563
DF-ResNet179		9.84M	8.64G	0.62	0.0611	0.80	0.0899	1.51	0.1483
DF-ResNet233		12.33M	11.17G	0.58	0.0442	0.76	0.0831	1.44	0.1464

- ResNets: according to [4], we re-implement ResNet18, 34 and 101.
- ECAPA-TDNNs: following [5], ECAPA (C=512) and ECAPA (C=1024) are built.
- Transformers: plus, transformer-based model SAEP [6] and GCSA [7] are employed.
- MLP: also, a pure MLP model from [8] is included.

5. Results and Analysis

In this section, we first report the performance of the proposed DF-ResNet family on Voxceleb. Then, a comprehensive comparison of ResNet, ECAPA, Transformer, MLP and DF-ResNet is presented in terms of performance and model complexity.

5.1. Voxceleb Results for DF-ResNets

From Table 4, we can see that DF-ResNet56 can achieve the relative improvements in EER by 35.1%, 28.3%, 26.8% over ResNet18 system on Vox1-O, Vox1-E, Vox1-H respectively under similar complexity, which reveals that the proposed depth-first rule and the corresponding design choices described in section 3.2 is effective. Consistent with our empirical observation in section 2, it can be concluded that with deeper models comes better performance for the SV task, under similarly-sized parameters and FLOPs. Compared with the original basic blocks in ResNet18, our designed computation block is more efficient and superior, making it possible to increase the number of layers without swelling up networks. Additionally, DF-ResNets enjoy a trend of continuous performance improvements with the increase in the number of blocks in each stage, demonstrating the scalability and simplicity of our proposed architecture. In particular, our DF-ResNet233 achieves better performance than ResNet101 with 23% fewer parameters. These gains come from both better computation block and better scaling-up method.

5.2. Analysis of Performance and Complexity

As shown in Figure 1, DF-ResNet achieves the best EER-parameter trade-off among previous SV systems, including two strong baselines (ResNet [4] and ECAPA-TDNN [5]). We can see that DF-ResNet significantly outperforms ResNet and ECAPA-TDNN of similar complexity across the board. For example, DF-ResNet56 obtains 35% relative improvement over ResNet18. Compared with ResNet34 and ECAPA (C=512), DF-ResNet110 is also much better with a similar parameter size. Our DF-ResNet179 achieves approximately the same performance as ResNet110 with 38% fewer parameters. For transformer-based and MLP-based SV systems, they have much poorer performance in terms of EER-parameter trade-off. The above results demonstrate the superiority of our DF-ResNet over other architectures. As for FLOPs, it can be clearly observed that DF-ResNet exhibits a more favorable performance than ResNet in both low and high FLPOs regimes. Specifically, DF-ResNet56 achieves similar EER with 43% fewer FLOPs compared to ResNet34. In the high FLPOs regime, DF-ResNet179 contains 14% fewer FLOPs than ResNet101 while exhibiting the same EER performance. These results illustrate that our DF-ResNets achieve a much better trade-off on performance and network complexity than previous SV systems.

6. Conclusions

Although various models have been investigated in the SV field recently, rare works pay attention to the question: is there a principled method to design and scale up DNN-based SV systems that can achieve better performance and network complexity? In fact, this is an important problem for SV applications in real life. In this paper, we first study the effect of depth and width on the performance for the SV task and empirically observe that depth is more important than width. Based on this observation, we design a new baseline model according to the depth-first rule and construct a family of much deeper networks dubbed DF-ResNets. Experiments on the Voxceleb dataset show that our DF-ResNets can achieve a more favorable trade-off on performance and network complexity than previous SV systems. We hope this work can shed light on the direction towards architectural designs with better performance and complexity for the SV task.

7. Acknowledgement

This work was supported in part by China NSFC projects under Grants 62122050 and 62071288, and in part by Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102.

8. References

[1] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, "Deep feature for text-dependent speaker verification," *Speech Communication*, vol. 73, pp. 1–13, 2015.

[2] B. Liu, H. Wang, Z. Chen, S. Wang, and Y. Qian, "Self-knowledge distillation via feature enhancement for speaker verification," in *ICASSP*, 2022.

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

[4] 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.

[5] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: emphasized channel attention, propagation and aggregation in tdnn based speaker verification," in *INTERSPEECH*, 2020, pp. 3830–3834.

[6] P. Safari, M. India, and J. Hernando, "Self-attention encoding and pooling for speaker recognition," in *INTERSPEECH*, 2020, pp. 941–945.

[7] B. Han, Z. Chen, and Y. Qian, "Local information modeling with self-attention for speaker verification," in *ICASSP*, 2022.

[8] B. Han, Z. Chen, B. Liu, and Y. Qian, "Mlp-svnet : a multi-layer perceptrons based network for speaker verification," in *ICASSP*, 2022.

[9] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *INTERSPEECH*, 2018, pp. 2252–2256.

[10] 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.

[11] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," in *INTERSPEECH*, 2019, pp. 4305–4309.

[12] S. Wang, Y. Yang, Y. Qian, and K. Yu, "Revisiting the statistics pooling layer in deep speaker embedding learning," in *ISCSLP*, 2021, pp. 1–5.

[13] 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.

[14] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in *INTERSPEECH*, 2019, pp. 2873–2877.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[16] X. Qin, N. Li, C. Weng, D. Su, and M. Li, "Simple attention module based speaker verification with iterative noisy label detection," *arXiv preprint arXiv:2110.06534*, 2021.

[17] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP*, 2019, pp. 5791–5795.

[18] S. Kim and Y. Park, "Adaptive convolutional neural network for text-independent speaker recognition," in *INTERSPEECH*, 2021, pp. 66–70.

[19] J. Qi, W. Guo, and B. Gu, "Bidirectional multiscale feature aggregation for speaker verification," in *INTERSPEECH*, 2021, pp. 71–75.

[20] Y. Wu, J. Zhao, C. Guo, and J. Xu, "Improving deep cnn architectures with variable-length training samples for text-independent speaker verification," in *INTERSPEECH*, 2021, pp. 81–85.

[21] Y. Liu, Y. Song, I. McLoughlin, L. Liu, and L. Dai, "An effective deep embedding learning method based on dense-residual networks for speaker verification," in *ICASSP*, 2021, pp. 6668–6672.

[22] J. Thienpondt, B. Desplanques, and K. Demuynck, "Integrating frequency translational invariance in tdnns and frequency positional information in 2d resnets to enhance speaker verification," in *INTERSPEECH*, 2021, pp. 2302–2306.

[23] M. Zhao, Y. Ma, M. Liu, and M. Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[24] L. Zhang, Q. Wang, and L. Xie, "Duality temporal-channel-frequency attention enhanced speaker representation learning," *arXiv preprint arXiv:2110.06565*, 2021.

[25] 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.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[27] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *arXiv preprint arXiv:2201.03545*, 2022.

[28] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017, pp. 2616–2620.

[29] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: deep speaker recognition," in *INTERSPEECH*, 2018, pp. 1086–1090.

[30] W. Cai, J. Chen, J. Zhang, and M. Li, "On-the-fly data loader and utterance-level aggregation for speaker and language recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 28, pp. 1038–1051, 2020.

[31] D. Snyder, G. Chen, and D. Povey, "Musan: a music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[32] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *ICASSP*, 2017, pp. 5220–5224.

[33] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: a simple data augmentation method for automatic speech recognition," in *INTERSPEECH*, 2019, pp. 2613–2617.

[34] W. Wang, D. Cai, X. Qin, and M. Li, "The dku-dukeece systems for voxceleb speaker recognition challenge 2020," *arXiv preprint arXiv:2010.12731*, 2020.

[35] Z. N. Karam, W. M. Campbell, and N. Dehak, "Towards reduced false-alarms using cohorts," in *ICASSP*, 2011, pp. 4512–4515.

[36] S. Cumani, P. Batzu, D. Colibro, C. Vair, P. Laface, and V. Vasilakakis, "Comparison of speaker recognition approaches for real applications," in *INTERSPEECH*, 2011, pp. 2365–2368.

[37] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.