Interspeech 2022
18-22 September 2022, Incheon, Korea

Training speaker recognition systems with limited data

Nik Vaessen[1], *David A. van Leeuwen*[1]

[1]Institute for Computing and Information Sciences, Radboud University
nvaessen@science.ru.nl, dvanleeuwen@science.ru.nl

Abstract

This work considers training neural networks for speaker recognition with a much smaller dataset size compared to contemporary work. We artificially restrict the amount of data by proposing three subsets of the popular VoxCeleb2 dataset. These subsets are restricted to 50 k audio files (versus over 1 M files available), and vary on the axis of number of speakers and session variability. We train three speaker recognition systems on these subsets; the X-vector, ECAPA-TDNN, and wav2vec2 network architectures. We show that the self-supervised, pre-trained weights of wav2vec2 substantially improve performance when training data is limited. Code and data subsets are available at `https://github.com/nikvaessen/w2v2-speaker-few-samples`.

Index Terms: speaker recognition, few-shot learning, wav2vec2

1. Introduction

Recently, the wav2vec2 framework [1] proposed a self-supervised pre-training, and consecutive fine-tuning approach for automatic speech recognition with a transformer network. Such a procedure has become the de facto standard in NLP with models like BERT [2]. One of the benefits of pre-training is the possibility to use large, unlabeled, and thus relatively inexpensive datasets. Another benefit is that these pre-trained networks are flexible, and can be fine-tuned to a variety of related tasks. This has been shown to be the case for wav2vec2 as well, which, while originally designed for speech recognition [1], has also been used for tasks like speaker recognition [3–5] and emotion recognition [5–7]. One property of fine-tuning a pre-trained network is that it requires less labeled data than training from scratch. For example, the authors of wav2vec2 pre-train on 53 k hours of unlabeled speech data, fine-tune on 10 minutes of labeled speech data, and achieve a word-error-rate of 4.8% on the clean test set of LibriSpeech [8]. For comparison, in 2016 the DeepSpeech2 system [9] achieved a 5.3% word-error-rate with 3600 hours of labeled training data.

In this work, we want to study the behaviour of wav2vec2 under similar low-resource data conditions, but for speaker recognition instead of speech recognition. We are interested in the following research questions:

1. How well does the self-supervised, pre-trained wav2vec2 network perform when fine-tuned for speaker recognition with little labeled data?
2. What is the most effective way to structure a limited training dataset? Is there a trade-off to be made between speaker variability and session variability?

We hypothesize that the pre-trained wav2vec2 network will also be beneficial for the limited-data speaker recognition scenario, as it has learnt to model speech representations. This was shown useful as a basis for speech recognition, and it seems plausible that speaker recognition can benefit from representations of speech. Although we compare wav2vec2 against non-self-supervised neural networks designed specifically for speaker recognition [10, 11], we imagine that these (or similar) networks can also benefit from self-supervision. There has been work for speaker recognition on self-supervised learning [12, 13], and consecutive fine-tuning [14], but to the extent of our knowledge, not for common speaker recognition networks [10, 11]. Also, note that a frequent solution to limited data is data augmentation [15]. In this work, we explicitly skip data augmentation in order to observe the effects of self-supervised weights. The second research question is focused on data collection. There might be scenarios, related to e.g., licensing, or the domain, where one needs to construct a dataset for fine-tuning. In this scenario, we hypothesize that maximizing the number of speakers in your dataset is paramount.

2. Related work

Earlier work [16–18] interprets limited data availability not in the size of the training dataset, but in the length of the utterances. However, since the advent of neural approaches for speaker recognition, it has become the norm to train with short audio segments, often between 0.5 and 3 seconds [10, 11, 19]. The contemporary field of few-shot learning [20] considers low resource scenarios where models need to adapt to new classes ("N-way") with only a few samples ("K-shot"). In [21], the LibriSpeech dataset [8] is used to study low resource conditions for speaker identification. They vary the total training data length per speaker between 20, 40 or 60 segments of 3 seconds, and show only minor degradation in test accuracies when using a prototypical loss [22], or their proposed adversarial few-shot learning-based speaker identification framework. In [23], speaker verification is considered within the few-shot learning paradigm with a subset of VoxCeleb2 [24] containing 71 train speakers and 30 test speakers. They compare a prototypical loss [22] against a triplet loss [25], and train with 200 segments of 2 seconds. They show prototypical loss achieves better equal-error-rates than the triplet loss in this scenario.

3. Methodology

3.1. Subsetting VoxCeleb

In order to experiment with smaller dataset size conditions, we artificially limit ourselves to a subset of data from the so-called development set of VoxCeleb2 [24], which we use as *train* and *validation* set. This development set consists of nearly 6 k speakers distributed over 1 M speech utterances, with a mean length of 7.8 seconds and a standard deviation of 5.2 seconds. Each speaker has a number of associated video recordings (sessions), and from each recording one or more speech utterances are automatically extracted by using face tracking, face verification, and active speaker verification. This ensures each speech utterance is attributed to a single speaker, although some labeling noise is expected. Recordings were collected by [24] using the top 100 results from a YouTube search. The search query included the

Table 1: *Statistics on the vox2 training split, as well as the three tiny subsets we created, that we used for training.*

					sessions per speaker			utterances per sessions		
dataset	duration (h)	# speakers	# sessions	# utterances	mean	min	max	mean	min	max
vox2	2 314	5 994	136 632	1 068 871	22.8	4	89	7.8	1	264
tiny-few-speakers	113	100	5 066	49 400	50.7	22	87	9.8	1	264
tiny-few-sessions	100	5 994	6 275	47 952	1.0	1	4	7.6	1	8
tiny-many-sessions	97	5 994	46 813	47 952	7.8	4	8	1.0	1	3

name of the celebrity and the word “interview”. We limit each subset to 50 k utterances, but vary the amount of speakers, the amount of sessions per speaker, or the amount of utterances per session. Throughout this work, we will refer to the following datasets. Statistics of the datasets are shown in Table 1.

Vox2 refers to our train split of the original dataset. We also create a validation split for monitoring overfitting, and selecting the best checkpoint. For each speaker in the original VoxCeleb2 development set, we randomly move sessions to the validation dataset until less than 99% of all utterances from the respective speaker are remaining. The validation set is created *before* the tiny training subsets. To calculate a validation EER, we create a random trial list with 15 k positive and negative (same-gender) trials. The validation set is equal for *vox2*, *tiny-few-sessions* and *tiny-many-sessions*. For *tiny-few-speakers*, we modify the validation set such that it only contains the 100 speakers in the subset, and a different random trial list, with 2 k positive and 2 k negative (same-gender) trials is created.

Tiny-few-speakers is a subset with few speakers, but many sessions per speaker, and many utterances per session. We group the speakers from *vox2* by gender, and sort descendingly by the amount of recordings available. We select the first 50 female and male speakers.

Tiny-few-sessions is a subset with many speakers, few sessions per speaker, but relatively many utterances per session. This subset contains all speakers in *vox2*. For each speaker, we sort their sessions descendingly by the number of utterances. We then select 8 utterances from each speaker. We start sampling from the session with the most utterances. If a session is exhausted, we continue with the next session according to the sorted collection.

Tiny-many-sessions is a subset with many speakers, many sessions per speaker, but few utterances per session. Just as in *tiny-few-sessions*, we select 8 utterances from all speakers in *vox2*. However, the difference is that we sample only 1 utterance per session. When a speaker has fewer than 8 sessions available, we cycle through the sorted collection, selecting only 1 utterance per session per cycle, until 8 utterances are selected.

3.2. Speaker recognition networks

We train three different speaker recognition models on the datasets in §3.1. We use third-party library network implementations, but train and evaluate with our own PyTorch [26] code.

3.2.1. X-vector

The X-vector architecture [10] is a popular neural network for speaker recognition and diarization, initially from the Kaldi framework [27]. The X-vector network consists of 5 consecutive layers with 1-dimensional convolution, ReLU activation, and BatchNorm, followed by mean&std pooling, and 2 FC layers. During training, a third FC layer is used for computing the classification loss. We extract the speaker embeddings from the first fully-connected layer. We use the implementation by SpeechBrain [28], with the default settings, such that the speaker embeddings have a dimensionality of 1024.

3.2.2. ECAPA-TDNN

The ECAPA-TDNN architecture [11] is a more recent speaker recognition network that showed best performance in the VoxCeleb 2020 challenge [29]. It builds on top of the X-vector paradigm by adding global context through network architecture modifications. First, it makes use of three consecutive res2blocks [30], consisting of three 1-d convolutions, a squeeze-and-excitation layer [31] and a skip connection [32]. They also aggregate the output of each res2block, before using channel-wise attentive statistical pooling to compute a fixed-size speaker embedding. We use the SpeechBrain [28] implementation, with the default settings, such that the speaker embeddings have a dimensionality of 128.

3.2.3. Wav2vec2

The wav2vec2 architecture applies self-supervised pre-training to speech data, and has been used for multiple speech-related tasks. We only fine-tune the network. There is BASE and LARGE variant, we only consider the BASE network. The architecture consists of 3 components: First, raw audio is processed by a 7-layer CNN with 1-d convolutions, LayerNorm, and GELU activation [33]. Secondly, a linear projection with a FC layer and an additive relative positional embedding with a 1-layer CNN is applied. Lastly, the hidden state sequence is processed by 12 transformer blocks. We use the Transformers [34] implementation, with self-supervised weights provided by Fairseq[1]. The self-supervision was carried out (by Fairseq [1]) on the LibriSpeech [8] dataset. For the speaker recognition task, the final hidden state sequence is mean-pooled into a fixed-size speaker embedding of dimensionality 768 [3,4]. During training, a single FC layer is used for classification.

4. Experiments

The experiments consist of a hyperparamter search for the best learning rate for each network and dataset combination (§4.2), multiple runs with the best learning rate and varying amount of steps (§4.3), and an ablation study (§4.4).

4.1. Training and evaluation protocol

We base the following training protocol on the ECAPA-TDNN [11] and wav2vec2 [1] articles. Each network is trained for n_{steps} steps with the Adam [35] optimizer. We use a cyclic learning rate schedule [36] with a degrading maximum learning rate according to the *triangular2* policy. We always use 4 cycles, one cycle is therefore $n_{\text{steps}}/4$ iterations. The minimum LR each cycle is 10^{-8}. For the *vox2* dataset we validate every 5 k steps, for the tiny datasets we validate after each epoch. To create a batch, we randomly sample speech utterances, and select a random 2 second chunk from each utterance. We use a batch size of 100 chunks, matching the total batch size of 3.2 M audio samples used in [1]. For X-vector and ECAPA-TDNN the network input is a 80-dimensional MFCC with a window

[1]We downloaded the weights from `https://huggingface.co/facebook/wav2vec2-base`

Table 2: *The best-performing learning rate for each network and dataset combination, trained for 50 k steps. The respective EER is measured on the vox1-o development set.*

data		X-vector	ECAPA	wav2vec2
vox2	EER	6.30 %	2.91 %	2.40 %
	LR	3.16×10^{-3}	5.62×10^{-3}	1.78×10^{-4}
few	EER	12.91 %	12.19 %	7.46 %
speak.	LR	1.78×10^{-2}	1.78×10^{-2}	5.62×10^{-6}
few	EER	21.70 %	15.97 %	15.72 %
sess.	LR	1.00×10^{-4}	5.62×10^{-3}	1.78×10^{-4}
many	EER	9.75 %	6.04 %	3.60 %
sess.	LR	1.78×10^{-3}	5.62×10^{-3}	1.78×10^{-4}

length of 25 ms and a 12.5 ms shift. All three network are trained with angular additive margin softmax loss [37, 38]. We use a margin of 0.2 and a scale of 30. We do not use any weight decay in order to reduce the search space. For X-vector and ECAPA-TDNN, we use SpecAugment [39] with 5 to 10 masks of length 10 in the time axis, and 1 to 3 masks of length 4 in the channel axis. For wav2vec2, we use a LayerDrop [40, 41] of 10% in the transformer layers, and a dropout of 10% is applied after each fully-connected layer in the network. Wav2vec2 also applies masking before the relative positional embedding is added, similar to SpecAugment; 10% of the channel dimensions are randomly masked, and 50% of the time dimensions are randomly masked. We freeze the whole wav2vec2 network for the first 12.5 k steps, except for the last fully-connected layer used for the speaker classification. We also freeze the feature extractor CNN of wav2vec2 for the whole training run (n_{steps} iterations). Training is conducted on a RTX 3090 GPU for wav2vec2, and an GTX 2080Ti GPU for X-vector and ECAPA-TDNN. All experiments are capped to 32 GB RAM and 6 CPU cores. In total 209 days of GPU time was used for experiments.

We evaluate trials using a cosine score between speaker embeddings, without any other processing. We use the original VoxCeleb1 [42] test set (40 speakers, henceforth *vox1-o*) as a development set, and the VoxCeleb1 hard test set (1190 speakers, henceforth *vox1-h*) as the evaluation set. There is no overlap between VoxCeleb1 [42] and Voxceleb2 [24]. There is an overlap between the development and evaluation set, but we verified that the results are not significantly different when the trials from overlapping speakers are removed.

4.2. Learning rate search

We conduct a learning rate search for the maximum LR in the cyclic schedule. This is done for all three networks and all four datasets. We conduct this search in two phases. In the first phase we scan over a large magnitude: we consider 10^{-i}, with $i \in \{2, 3, 4, 5, 6, 7\}$. Based on the development set, we select the LR 10^{-j} with the lowest EER. In the second phase, we scan around this LR: we consider $\{1.78, 3.16, 5.62\} \times \{10^{-j-1}, 10^{-j}\}$. After the second phase, the LR with the lowest EER is used for the remaining experiments. The random seed is kept constant across all training runs in the grid search, and thus every learning rate is attempted only once. Each run has $n_{\text{steps}} = 50\,\text{k}$.

The best LR, and the respective EER on the development set *vox1-o*, are shown in Table 2. We see that wav2vec2 performs best on all datasets. However, performance on *tiny-few-sessions* seems poor for all three networks. In Figure 1 we plot the learning rate against the EER. In general, we can see that wav2vec2 requires a lower learning rate. Moreover, for all three networks, the optimal learning rate is dependent on the dataset.

Table 3: *Each network and dataset combination is trained for 25 k, 50 k, 100 k and 400 k steps with the learning rate from Table 2. The EER values are measured on the* vox1-h *evaluation set. Each experiment was run 3 times.*

	EER (mean, std in %) on vox1-hard					
steps	X-vector		ECAPA		wav2vec2	
vox2						
25k	16.30	0.64	6.80	0.06	7.76	0.07
50k	11.21	0.33	5.46	0.07	4.66	0.15
100k	7.21	0.10	4.61	0.13	**4.20**	0.16
400k	**5.01**	0.04	**3.93**	0.07	5.90	0.77
tiny-few-speakers						
25k	18.93	0.15	18.27	0.03	22.48	0.16
50k	18.02	0.08	17.59	0.40	**15.19**	0.24
100k	**18.00**	0.22	17.48	0.07	15.60	0.20
400k	18.04	0.31	**16.95**	0.15	18.50	0.43
tiny-few-sessions						
25k	28.46	0.12	22.67	0.23	23.78	0.10
50k	27.23	0.14	21.25	0.44	**21.88**	0.16
100k	26.29	0.30	20.58	0.25	22.08	0.29
400k	**24.00**	0.19	**19.52**	0.12	23.31	0.10
tiny-many-sessions						
25k	18.00	0.11	11.51	0.39	10.41	0.52
50k	16.05	0.31	9.78	0.06	**6.72**	0.04
100k	13.53	0.73	**9.12**	0.11	7.66	0.83
400k	**10.58**	0.25	9.36	0.08	7.93	0.37

4.3. Varying number of steps with found LR

In the following experiments, we vary n_{steps} to 25 k, 50 k, 100 k, and 400 k for each network and dataset combination. Additionally, we run each experiment with 3 different random seeds, and we use the optimal LR found in the learning rate search. The networks are evaluated on the *vox1-h* evaluation set.

The results are shown in Table 3. First, we see that ECAPA-TDNN has the best performance on *vox2*, while wav2vec2 is slightly worse than ECAPA-TDNN but better than the X-vector network. We observe the same on the *tiny-few-sessions* dataset, although the EER values are much higher compared to the other three datasets. On *tiny-few-speakers* and *tiny-many-sessions*, we see that wav2vec2 has the best performance, while ECAPA-TDNN is slightly worse, but better than the X-vector network. Moreover, we see that on all three tiny datasets the wav2vec2 network achieved the best performance with 50 k steps, while ECAPA-TDNN and X-vector almost always have the best performance after 400 k steps.

4.4. Ablation study

For the last set of experiments we perform an ablation study on the baseline training protocol described in §4.1. We perform the ablations on the *tiny-few-speakers* and *tiny-many-sessions* datasets with the wav2vec2 network. All ablations use 50 k steps, and the best LR found in the grid search. In the first set of ablations, we vary the learning rate schedule, and instead use either 1) a constant schedule, 2) an exponentially decaying schedule, or 3) a cyclic schedule with one cycle instead of four cycles. In the second set of ablations, we focus on the weights, and therefore we 1) randomly initialize the wav2vec2 network, 2) use self-supervised pre-trained weights but without any freezing, 3) use the pre-trained weights and freeze the feature extractor CNN for all 50 k steps, or 4) use the pre-trained weights and freeze the network for the 1st learning rate cycle (12.5 k steps). The third set of ablations consider regularisation. We 1) disable

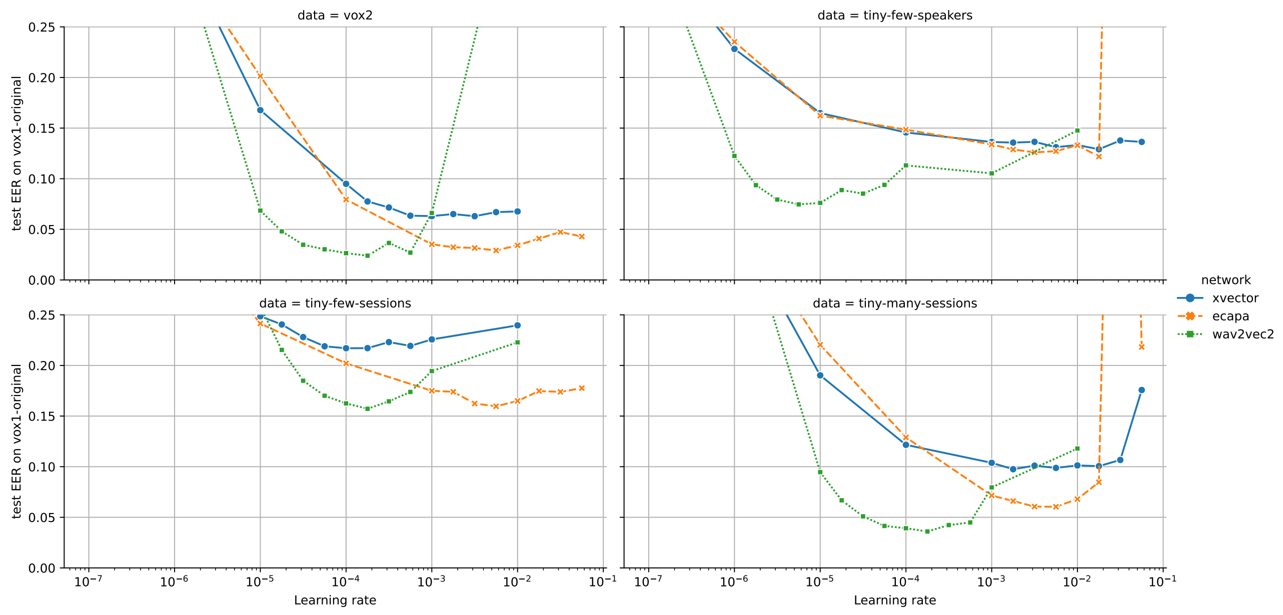

Figure 1: *The results of the learning rate search. We plot the learning rates of phase 1 and phase 2 of the grid search on the x-axis, and the EER on the vox1-o development set on the y-axis. Each dot represents a single experiment, no variation is measured.*

all regularisation parameters mentioned in §4.1, or 2) only enable dropout, 3) only enable LayerDrop, or 4) only enable masking.

The results are shown in Table 4. When we ablate on the learning rate schedule, we observe that for *tiny-few-speakers* all three schedules perform worse than the baseline. For *tiny-many-sessions*, an exponentially decaying schedule seems to perform slightly better than our baseline, while a constant schedule, as well as a cyclic schedule with 1 cycle, perform worse. Next, we looked at the network weights and the freezing schedule. We observe that using randomly initialized weights prevents convergence to a reasonable performance. When using the self-supervised pre-trained weights without any freezing, the performance is slightly worse than the baseline. Freezing the whole network for the first learning rate cycle seems beneficial, as it improves on the baseline, while freezing the feature extractor CNN for n_{steps} results in degraded performance. Finally, we observe that disabling all regularisation degrades performance. With only enabling LayerDrop regularization we achieve similar performance to the baseline, while only enabling masking, and only enabling dropout, perform similar to disabling regularisation.

Table 4: *Ablation on the wav2vec2 network trained on the* tiny-few-speakers *and* tiny-many-sessions *datasets. Evaluation is done on the* vox1-h *evaluation set. Each experiment is run 3 times.*

	EER (mean, std in %) on *vox1-h*			
ablation	few-speakers		many-sessions	
baseline (Table 3)	15.19	0.24	6.72	0.04
LR schedule				
constant	16.77	0.26	8.80	0.44
exp. decay	16.68	0.20	6.67	0.04
1 cycle	15.79	0.23	8.59	0.23
weights				
random init	33.35	0.16	46.24	0.06
pretraining BASE	15.16	0.17	7.18	0.19
and freeze CNN	15.48	0.25	7.83	0.45
or freeze 1st cycle	14.52	0.11	6.38	0.17
regularisation				
none	16.67	0.27	7.73	0.07
dropout	16.67	0.11	8.01	0.12
layerdrop	15.03	0.14	6.72	0.21
masking	16.21	0.18	7.87	0.24

5. Conclusion

Similar to ASR [1], we have shown that the wav2vec2 network, when initialised with self-supervised weights, has better performance, and needs fewer training iterations, than the X-vector and ECAPA-TDNN network on two out of the three tiny datasets. Although ECAPA-TDNN performed slightly better on the *tiny-few-sessions* dataset, the performance of all three networks was poor. As indicated by the results on *tiny-few-sessions*, a dataset with many speakers but no session variability leads to poor performance. As all networks had better performance on *tiny-few-speakers* compared to *tiny-few-sessions*, it seems that having more sessions in a limited dataset should be a priority above having many speakers. However, *tiny-few-speakers* has a mean of 51 sessions per speaker, compared to a mean of 8 for *tiny-many-sessions*, while achieving worse EERs. It would be interesting if future work can find an optimal amount of sessions per speaker.

Currently, the self-supervised learning optimization of wav2vec2 uses a contrastive loss to distinguish a masked segment from other segments in the same utterance. For speaker recognition, it might be beneficial to include segments from other utterances, which could model inter- and intra-speaker variance. Such a change could then perhaps result in even better performance on the tiny datasets, and specifically on *tiny-few-sessions*. We are also interested in future work pre-training on a dataset other than LibriSpeech, which has limited variability per speaker. It might also be relevant to pre-train on VoxCeleb2, so that the model has a prior on speech patterns, requiring less fine-tuning.

6. References

[1] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations,"

in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. ACL 2019*. Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[3] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on Speaker Verification and Language Identification," in *Proc. Interspeech 2021*, 2021, pp. 1509–1513.

[4] N. Vaessen and D. A. Van Leeuwen, "Fine-tuning wav2vec2 for speaker recognition," in *ICASSP 2022*, 2022, pp. 7967–7971.

[5] S. Evain *et al.*, " LeBenchmark: A Reproducible Framework for Assessing Self-Supervised Representation Learning from Speech," in *Proc. Interspeech 2021*, 2021, pp. 1439–1443.

[6] L. Pepino, P. Riera, and L. Ferrer, "Emotion Recognition from Speech Using wav2vec 2.0 Embeddings," in *Proc. Interspeech 2021*, 2021, pp. 3400–3404.

[7] J. Yuan, X. Cai, R. Zheng, L. Huang, and K. Church, "The role of phonetic units in speech emotion recognition," *arXiv preprint arXiv:2108.01132*, 2021.

[8] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[9] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 173–182.

[10] 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.

[11] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in *Proc. Interspeech 2020*, 2020, pp. 3830–3834.

[12] D. Cai, W. Wang, and M. Li, "An iterative framework for self-supervised deep speaker representation learning," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6728–6732.

[13] J. Thienpondt, B. Desplanques, and K. Demuynck, "The idlab voxceleb speaker recognition challenge 2020 system description," 2020. [Online]. Available: https://arxiv.org/abs/2010.12468

[14] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, J. Wu, L. Zhou, S. Ren, Y. Qian, Y. Qian, J. Wu, M. Zeng, X. Yu, and F. Wei, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," 2021. [Online]. Available: https://arxiv.org/abs/2110.13900

[15] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[16] A. Poddar, M. Sahidullah, and G. Saha, "Speaker verification with short utterances: a review of challenges, trends and opportunities," *IET Biometrics*, vol. 7, no. 2, pp. 91–101, 2018.

[17] R. K. Das, S. Abhiram, S. R. M. Prasanna, and A. G. Ramakrishnan, "Combining source and system information for limited data speaker verification," in *Proc. Interspeech 2014*, 2014, pp. 1836–1840.

[18] H. Jayanna and S. Mahadeva Prasanna, "An experimental comparison of modelling techniques for speaker recognition under limited data condition," *Sadhana*, vol. 34, no. 5, pp. 717–728, 2009.

[19] W.-W. Lin and M.-W. Mak, "Wav2spk: A simple DNN architecture for learning speaker embeddings from waveforms." in *Proc. Interspeech*, 2020, pp. 3211–3215.

[20] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, jun 2020. [Online]. Available: https://doi.org/10.1145/3386252

[21] R. Li, J.-Y. Jiang, J. L. Li, C.-C. Hsieh, and W. Wang, *Automatic Speaker Recognition with Limited Data*. New York, NY, USA: Association for Computing Machinery, 2020, p. 340–348. [Online]. Available: https://doi.org/10.1145/3336191.3371802

[22] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[23] J. Wang, K.-C. Wang, M. T. Law, F. Rudzicz, and M. Brudno, "Centroid-based deep metric learning for speaker recognition," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3652–3656.

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

[25] C. Zhang and K. Koishida, "End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances," in *Proc. Interspeech 2017*, 2017, pp. 1487–1491.

[26] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library." Curran Associates, Inc., 2019, pp. 8024–8035.

[27] 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," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[28] M. Ravanelli *et al.*, "Speechbrain: A general-purpose speech toolkit," 2021.

[29] A. Nagrani, J. S. Chung, J. Huh, A. Brown, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, "Voxsrc 2020: The second voxceleb speaker recognition challenge," 2020. [Online]. Available: https://arxiv.org/abs/2012.06867

[30] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 2, pp. 652–662, 2019.

[31] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.

[34] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[36] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 464–472.

[37] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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

[39] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 2613–2617.

[40] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*. Springer, 2016, pp. 646–661.

[41] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," *arXiv preprint arXiv:1909.11556*, 2019.

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