INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Self-supervised speaker embeddings

Themos Stafylakis[1*], *Johan Rohdin*[2*], *Oldřich Plchot*[2], *Petr Mizera*[1], *Lukáš Burget*[2]

[1]Omilia - Conversational Intelligence, Athens, Greece
[2]Brno University of Technology, Faculty of Information Technology, IT4I Centre of Excellence, Czechia

{tstafylakis,petr.mizera}@omilia.com, {rohdin,iplchot,burget}@fit.vutbr.cz

Abstract

Contrary to i-vectors, speaker embeddings such as x-vectors are incapable of leveraging unlabelled utterances, due to the classification loss over training speakers. In this paper, we explore an alternative training strategy to enable the use of unlabelled utterances in training. We propose to train speaker embedding extractors via reconstructing the frames of a target speech segment, given the inferred embedding of another speech segment of the same utterance. We do this by attaching to the standard speaker embedding extractor a decoder network, which we feed not merely with the speaker embedding, but also with the estimated phone sequence of the target frame sequence.

The reconstruction loss can be used either as a single objective, or be combined with the standard speaker classification loss. In the latter case, it acts as a regularizer, encouraging generalizability to speakers unseen during training. In all cases, the proposed architectures are trained from scratch and in an end-to-end fashion. We demonstrate the benefits from the proposed approach on the VoxCeleb and Speakers in the Wild Databases, and we report notable improvements over the baseline.

Index Terms: speaker recognition, self-supervised learning, deep learning

1. Introduction

In recent years, deep learning classifiers and representations have surpassed the performance of shallow and fully probabilistic counterparts in several tasks of speech recognition and computer vision, often by a large margin. A key ingredient towards this success has been the availability of large annotated datasets, which enabled very deep architectures to be trained using supervised learning approaches. The availability of large in-domain corpora played a major role in building robust speaker recognition models, too. The success of Joint Factor Analysis and i-vectors can largely be attributed to such corpora, which enabled modeling correlations between acoustic units [1, 2]. More recently, deep learning architectures outperformed such methods in most of the speaker recognition benchmarks [3].

On the other hand, these architectures require the datasets to be labelled with respect to speaker, which was not the case with i-vectors. As an unsupervised model, an i-vector extractor does not require utterances with associated speaker labels for training [2]. Labelled utterances are needed merely for training the backend classifier (typically a PLDA) which requires much less data, thanks to its relatively small number of trainable parameters.

In this paper, we introduce a training architecture capable of learning speaker embeddings with only few or no speaker labels. The structure we add to the standard speaker embedding network is a decoder network, which learns how to reconstruct speech segments in the frame-level, using a mean-squared error loss. A key idea of the method is the conditioning of the decoder not merely on the embedding extracted by the encoder (i.e., the embedding extractor), but also on the phonetic sequence of the decoding speech segment, as estimated by an independently trained Automatic Speech Recognition (ASR) model. Such a conditioning allows for a decoding loss over speech frames to apply for learning in an end-to-end fashion using standard back-propagation. It moreover enables learning speaker embeddings that capture only the idiosyncratic characteristics of a speaker, rather than irrelevant information about the phonetic sequence. The latter property is further improved by extracting two different segments from an utterance: the first for feeding the encoder and extracting the embedding, and the second one for using it as target for the decoder, together with its associated phone sequence.

We show that the proposed decoder loss can be combined with the standard x-vector architecture and loss (i.e., cross-entropy over training speakers) yielding significant improvement. Finally, we consider a semi-supervised learning scenario, where only a small fraction of the training utterances contain speaker labels and we show how the proposed architecture can leverage both labelled and unlabelled utterances. All our experiments are conducted on VoxCeleb[4] and Speakers In The Wild[5] benchmarks.

*Equal Contribution.

2. Related work

2.1. Speaker recognition using autoencoders

There have been several attempts in speaker recognition to make use of reconstruction losses. Most of them are based on (plain or variational) autoencoders, either in an unsupervised way or using speaker labels [6, 7, 8]. Other such approaches aim at reducing the phonetic variability of short segments by learning a mapping from short segments to the whole utterance. The main weakness of these methods is the fact that they operate over fixed, utterance-level representations, typically i-vectors [9]. Our approach of conditioning the reconstruction on the estimated phone sequence of each segment can be employed, enabling such approaches to be revisited in an end-to-end fashion. Other recent approaches aiming at enhancing the x-vector architecture with adversarial loss are also relevant, since they propose joint training of the network with auxiliary losses and structures which are removed in runtime [10, 11, 12].

2.2. Speech synthesis, recognition, and factorization

Recently, speaker embeddings have been deployed in text-to-speech (TTS) and voice conversion [13, 14, 15]. The embeddings are typically extracted using a pretrained network (e.g., a d- or x-vector extractor), which may be fine-tuned to the task [14]. Conditioning the decoder on speaker embeddings (together with the text of the target utterance) is crucial for training multispeaker TTS systems and producing synthetic speech for target speakers unseen during training. Although our method shares certain similarities with this family of TTS methods (especially in the decoder), our goals are different. Rather than employing a pretrained speaker recognition model to extract embeddings, we demonstrate that speaker-discriminative training is feasible using merely a reconstruction loss over speech segments and training the overall network jointly. Finally, a recently introduced ASR approach for integrating ASR and TTS into a single cycle during training has also certain similarities with our method ([16]), and the same holds for the deep factorization method for speech proposed in [17].

2.3. Self-supervised learning

The approach of extracting speaker embeddings via reconstructing different parts of a sequence can be considered as an application of self-supervised learning, where a network is trained with a loss on a pretext task, without the need for human annotation. Models using self-supervised learning for initialization are now state-of-the-art in several domains and tasks, such as action recognition, reinforcement learning, and natural language understanding [18, 19, 20, 21, 22].

3. The proposed architecture

In this section we describe the network used in training and we provide rationale for certain algorithmic choices. The architecture is depicted in Figure 1. Architectural details are given in Section 4.2.

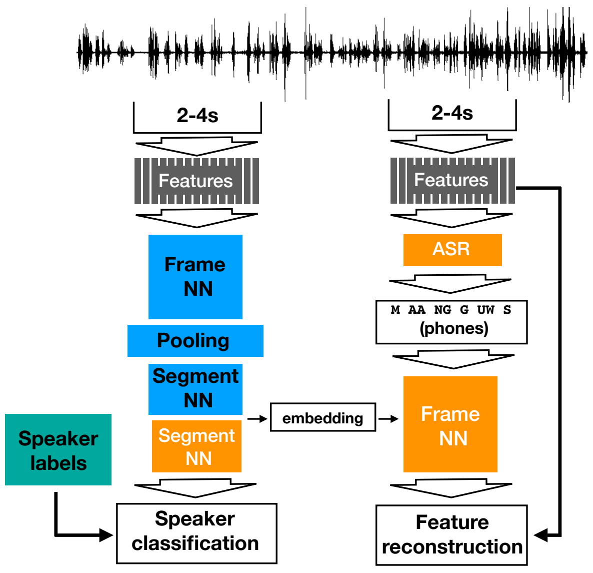

Figure 1: *Block-diagram of the architecture. Note that only the blue part of the network is required in runtime.*

3.1. Notation

We denote by $\mathbf{Y} = \{\mathbf{y}_t\}_{t=1}^{T}$ the frame sequence of an utterance, and the corresponding estimated phones sequence by $\hat{\mathbf{p}} = \{\hat{p}_t\}_{t=1}^{T}$. We also denote an associated version of the same utterance by $\tilde{\mathbf{Y}} = \{\tilde{\mathbf{y}}_t\}_{t=1}^{T}$, where tilde indicates corruption by noise, reverberation, or another data augmentation scheme. The encoder consumes randomly extracted segments from $\tilde{\mathbf{Y}}$, denoted by $\tilde{\mathbf{Y}}_e = \{\tilde{\mathbf{y}}_t\}_{t=t_e}^{t_e+\tau_e-1}$, where τ_e is randomly sampled in $[200, 400]$. Let also $\mathbf{Y}_d = \{\mathbf{y}_t\}_{t=t_d}^{t_d+\tau_d-1}$ be another segment of the same utterance (without data augmentation), and let $\hat{\mathbf{p}}_d$ denote its corresponding estimated phone sequence. The encoder part of the network (i.e., the embedding extractor) is denoted by

$$\mathbf{x}_e = f_e(\tilde{\mathbf{Y}}_e; \mathbf{W}_e), \tag{1}$$

and it is a function parametrized by $\mathbf{W}_e$.

3.2. Purely self-supervised training

We train the network using a decoder, which is a network implementing the following function

$$\hat{\mathbf{Y}}_d = f_d(\mathbf{x}_e, \hat{\mathbf{p}}_d; \mathbf{W}_d). \tag{2}$$

The decoder is parametrized by $\mathbf{W}_d$ and receives as input the embedding $\mathbf{x}_e$ and the estimates phone sequence $\hat{\mathbf{p}}_d$. Note that the embedding $\mathbf{x}_e$ and target frames $\mathbf{Y}_d$ correspond in general to different segments of the same utterance. The architecture is trained using mean-squared error (MSE) loss, i.e.

$$\mathcal{L}_{MSE}(\mathbf{Y}_d, \hat{\mathbf{Y}}_d) = \frac{1}{\tau_d} \sum_{t=t_d}^{t_d+\tau_d-1} \left\|\mathbf{y}_t - \hat{\mathbf{y}}_t\right\|^2. \tag{3}$$

The above equations show that the encoder and decoder can be trained jointly using standard backpropagation.

3.3. Combining self-supervision with cross-entropy over speakers

In case the utterance $\mathbf{Y}$ has a speaker label s we may combine the decoding loss with the standard cross-entropy loss over speakers. The speaker classifier estimates the posterior distribution over the set of training speakers, i.e.,

$$P(s|\mathbf{x}_e, \mathbf{W}_c) = f_c(\mathbf{x}_e; \mathbf{W}_c), \tag{4}$$

it is parametrized by $\mathbf{W}_c$ and has a softmax as final layer. The cross-entropy (CE) loss $\mathcal{L}_{CE}(s, P(s|\mathbf{x}_e, \mathbf{W}_c))$ can be added to the decoding loss and the overall network can be trained jointly, i.e.

$$\mathcal{L}_{joint}(\cdot) = \delta\mathcal{L}_{CE}(\cdot) + \alpha\mathcal{L}_{MSE}(\cdot), \tag{5}$$

where $\delta = \{0, 1\}$ indicates whether the particular utterance has a speaker label, and $\alpha \geq 0$ is a scalar for balancing the two losses.

3.4. Discussion

3.4.1. Encoding and decoding segments

The rationale for defining the encoding and decoding sequences as different segments of the same utterance is to encourage the encoder to learn embeddings that encode information about the way a speaker pronounces acoustic events unseen in the encoding sequence. Note also that we keep the decoding target sequence clean (i.e., without augmentation). We do so for encouraging the encoder to learn how to denoise speech sequences, an approach similar to denoising autoencoders [23].

3.4.2. *Representation of the phonetic sequence*

For passing the phonetic sequence to the decoder, we choose to define estimated phones as phonetic units, in the form of one-hot vectors. Clearly, there are several other options, such as bottleneck features, senones or characters.

One of the reasons why we did not use bottleneck features is that they inevitably carry speaker-discriminative information, as experience shows (recall that speaker recognition is feasible with plain bottleneck features [24]). Therefore, passing bottleneck features to the decoder could create information leakage, preventing the embedding from capturing useful speaker-discriminative information. Another drawback of bottlenecks is that they would tie the remaining network to a specific bottleneck extractor. Contrarily, symbolic entities such as phones allow for different ASR models to be used for estimating $\hat{\mathbf{p}}_d$ or even for using the ground truth phones, when available.

On the other hand, senones would result in a much larger and harder to train decoding network, while the senone posteriors would be much less spiky compared to phones, and hence far from resembling one-hot vectors. Furthermore, passing senones to the decoder seems unnecessary; the decoder can recover the context-dependence of each phone since it is conditioned on the overall phone sequence $\hat{\mathbf{p}}_d$.

Finally, using characters would require additional complexity to align the two sequences, such as an attention mechanism employed in TTS approaches [14]. For these reasons, we consider phones in the form of one-hot vectors as the appropriate representation and level of granularity for this task and setup.

3.4.3. *Semi and weakly supervised learning*

The proposed architecture defines a principled way of utilizing unlabelled data in x-vector training. There are other losses for supervised training with which one may combine it, such as the triplet loss [25, 26, 27]. Note that both cross-entropy and triplet losses cannot leverage unlabelled utterances (e.g., by splitting the same utterances into multiple segments), unless one assumes that each utterance in the training set is coming from a different speaker (which is typically not the case). On the other hand, our self-supervised method requires only the knowledge that two segments belong to the same speakers, while it can be extended to encode and decode on segments coming from different utterances of the same speaker. This would make it suitable also for certain weakly supervised learning settings, where labels indicate merely that two or more utterances are coming from the same speaker, without excluding the possibility that other utterances may belong to the same speaker as well. In such cases, a training criterion that does not require exclusive labels (cross-entropy loss) or negative pairwise labels (triplet loss) seems to be the only principled method for learning speaker representations.

4. Experiments

4.1. VoxCeleb and SITW datasets

We evaluate the systems on the Speakers in the wild (SITW) [5] *core-core eval* set and the VoxCeleb 1 *test* set [4]. We use the SITW core-core development set for tuning various hyperparamters of the systems. For preparing the data, we use the Kaldi [28] SITW recipe (`sitw/v2`). This recipe uses VoxCeleb 1 and 2 [29] for training data. We use the recipe as is, except that we do not include VoxCeleb 1 test set in the training set. The number of speakers in the training set is 7146 and the number of utterances is 2081192 including augmentations. For *semisupervised* experiments, we randomly selected 1000 speakers, having in total 227998 utterances.

4.2. Implementation and training details

4.2.1. *Implementation and decoder*

We use the TensorFlow toolkit [30] for implementing our systems. As baseline, we use the standard Kaldi x-vector architecture [31], i.e., five TDNN layers with ReLU activation functions followed by batch normalization, followed by a pooling layer that accumulates mean and standard deviation, followed by two feed-forward layers with ReLU and batch normalization, and finally a softmax layer for classifying speakers. Different from Kaldi, we apply a global normalization on the input features and batch normalization also after the pooling layer. As discussed above, the loss is CE over training speakers.

The reconstruction network (i.e., the decoder) consists of five layers that operates framewise. Its input are the phone labels represented as one-hot vector and its output predicts the 30-dimensional feature vectors. The input layer is either (a) a feed-forward layer or (b) a TDNN layer with a context of three frames on each side (denoted by ctx). The other layers are feed-forward layers with an output dimension 166 (i.e., same as the number of phone labels). All layers except the last one are followed by ReLU and batch normalization. The embedding is appended to the input of each layer. The loss for the reconstruction is the MSE between the real and predicted features.

In the experiments, we use minibatches containing 150 segments. The lengths of the segments are 2-4s. We use the ADAM optimizer [32], starting with a learning rate of 1e-2 which we then halve whenever the loss on a validation set does not improve for 32 epochs, where an epoch is defined to be 400 minibatches. In the semisupervised experiment, each batch contains 150 labelled segments and 150 unlabelled segments and only the labelled segments will be used to calculate the speaker classification loss.

4.2.2. *The ASR model*

The frame-level phone labels are generated using the official Kaldi [28] Tedlium speech recognition recipe (`s5_r3`). This recipe uses a TDNN based acoustic model with i-vector adaptation and an RNN based language model. Phone posteriors are obtained from the lattices using the forward-backward algorithm and then converted to hard labels. There are 39 phones, each coming in four different versions depending on their position in the word, plus a silence (SIL) and noise class (NSN) that has 5 versions each, resulting in 166 *phone classes*.

4.2.3. *PLDA Backend*

We used an identical backend to the one in the Kaldi x-vector recipe. This backend involves a preprocessing step which first reduces the x-vector dimension by LDA from 512 to 128, and then applies a nonstandard variant of length-norm[1]. The backend was implemented in python based on our in-house toolkit *Pytel*. For the fully supervised experiments we, as the Kaldi recipe, use the 200k longest utterances, resulting in 6298 speakers. For the semi-supervised experiments, we use all of these utterances where the speaker is among the 10000 randomly selected, resulting in 899 speakers and 31785 utterances.

[1] https://github.com/kaldi-asr/kaldi/blob/master/src/ivector/plda.cc

4.3. Experimental Results

4.3.1. Fully labelled training set

The results using the standard training set of VoxCeleb are given in Table 1.

Table 1: *Results with fully supervised training. EER refers to* Equal Error Rate *in % and mDCF refers to* Minimum Detection Cost *with* $\mathrm{P(tar)} = 0.01$. *All* N_f *≈7k speakers are used in PLDA training. Baseline refers to the standard x-vector recipe.*

	SITW		VoxCeleb	
	EER	mDCF	EER	mDCF
Spk, baseline	3.879	0.382	4.088	0.431
Self	5.823	0.540	6.707	0.628
Self, cln	5.685	0.542	6.538	0.612
Self, cln, ctx	4.844	0.485	**5.758**	**0.536**
Self, cln, ctx, same	**4.760**	**0.478**	6.347	0.580
Spk+Self	**3.311**	0.362	3.961	0.386
Spk+Self, cln	3.362	0.356	3.881	0.380
Spk+Self, cln, ctx	3.362	**0.353**	**3.759**	**0.344**

The results show that the self-supervised models are capable of extracting speaker-discriminant embeddings. The use of clean (cln) decoding utterances yields slightly better results, as it enforces the encoder to act as a denoiser. Moreover, conditioning the decoding TDNN on a 7-frame phonetic context (ctx) is clearly beneficial compared to conditioning it merely on the phone of the target frame.

We also observe that using same segments for encoding and decoding (same) yields inferior performance on VoxCeleb, while on SITW their performance is equivalent. A plausible explanation is that VoxCeleb contains shorter segments compared to SITW. Hence, as encoding and decoding on different segments encourages the network to learn how to reconstruct phonetic subsequences unseen in the encoding segments, it is expected to be more beneficial for short durations.

When the two losses are combined (Spk+Self), the model clearly outperforms plain x-vector (Spk). In this case, the self-supervised loss has a regularization effect, constraining the network to learn representations that generalize well to unseen speakers. Again, the use of context yields superior performance, although in this case the differences are less significant.

4.3.2. Partly labelled training set

In this set of experiments, we assume that only a fraction of the training utterances is labelled. Hence, in the results we provide in Table 2 the PLDA is trained with N_r = 899 VoxCeleb speakers out of $N_f \approx 7K$.

In the first two experiments in Table 2 we use standard CE over speaker loss. We observe the severe degradation when the number of speakers used to train the x-vector baseline is reduced to $N_r = 899$ (Spk, baseline). For comparison, we report the experimental results where the full set of speakers is used for training the x-vector model (Spk, full set).

The results using only self-supervision with context are clearly superior to those of pure x-vectors, due to the capacity of self-supervision in leveraging all available utterances during training. Moreover, when the two losses are combined, the results become even better, especially in terms of minDCF. Finally, we observe again the gains in performance by using different encoding and decoding segments.

Table 2: *Results with semi-supervised training. EER refers to* Equal Error Rate *in % and mDCF refers to* Minimum Detection Cost *with* $\mathrm{P(tar)} = 0.01$. *In all experiments, the same set of* N_r *= 899 randomly selected speakers is used for PLDA training. Baseline refers to the standard x-vector recipe.*

	SITW		VoxCeleb	
	EER	mDCF	EER	mDCF
Spk, baseline	6.616	0.593	7.709	0.658
Spk, full set	4.347	0.424	4.804	0.472
Self, cln	6.748	0.615	7.619	0.668
Self, cln, ctx	5.793	**0.548**	**6.644**	**0.589**
Self, cln, ctx, same	**5.768**	**0.548**	7.503	0.612
Spk+Self, cln	5.715	0.497	6.972	0.543
Spk+Self, cln, ctx	**5.416**	**0.488**	**6.315**	**0.534**

5. Conclusions and future work

In this paper, we introduced a new way of training speaker embedding extractors using self-supervision. We showed that a typical TDNN-based extractor can be trained without speaker labels, using a decoder network to approximate in the MSE sense a speech segment of the same utterance. A key idea for enabling decoding is the conditioning of the decoder on both the embedding and the phonetic sequence of the decoding segment, as estimated by an ASR model. Furthermore, we showed that the proposed loss can be combined with the standard cross-entropy, yielding notable improvements. Finally, we demonstrated its effectiveness on semi-supervised learning, i.e., when only a small fraction of the training set is labelled. Both additional networks we introduced (decoder and ASR model) are only needed during training, leaving the standard x-vector architecture unchanged in runtime.

The proposed approach can be extended in several ways. The method of conditioning the decoder on the phonetic sequence of the speech segment paves the way for revisiting methods such as variational autoencoders in an end-to-end fashion. Speech synthesis approaches may also benefit from the proposed method, e.g., by training embedding extractors jointly with TTS from scratch. Finally, there is large room for improvement in the architecture (e.g., by using a recurrent or attentive decoder or deeper and wider encoder [33]), in the training scheme (e.g., by varying the duration of encoding and decoding segments), and in the way the existing speaker labels are used in training (e.g., by extracting the two segments from different utterances of the same speaker).

6. Acknowledgements

The work was supported by the European Union's Horizon 2020 research and innovation programme Marie Sklodowska-Curie, co-financed by the South Moravian Region under grant agreement No. 665860, Czech Ministry of Interior project No. VI20152020025 "DRAPAK, Google Faculty Research Award program, Czech National Science Foundation (GACR) project "NEUREM3 No. 19-26934X, and by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602.

7. References

[1] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition,"

IEEE Transactions on Audio, Speech, and Language Processing, vol. 15, no. 4, pp. 1435–1447, 2007.

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

[3] 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.

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

[5] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The 2016 speakers in the wild speaker recognition evaluation," in *INTERSPEECH*, 2016, pp. 818–822.

[6] J.-T. Chien and C.-W. Hsu, "Variational manifold learning for speaker recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4935–4939.

[7] J. Villalba, N. Brümmer, and N. Dehak, "Tied variational autoencoder backends for i-vector speaker recognition." in *INTERSPEECH*, 2017, pp. 1004–1008.

[8] A. Silnova, N. Brummer, D. Garcia-Romero, D. Snyder, and L. Burget, "Fast variational bayes for heavy-tailed plda applied to i-vectors and x-vectors," in *INTERSPEECH*, 2018, pp. 72–76.

[9] J. Guo, N. Xu, K. Qian, Y. Shi, K. Xu, Y. Wu, and A. Alwan, "Deep neural network based i-vector mapping for speaker verification using short utterances," *Speech Communication*, vol. 105, pp. 92–102, 2018.

[10] W. Ding and L. He, "MTGAN: Speaker Verification through Multitasking Triplet Generative Adversarial Networks," in *INTERSPEECH*, 2018, pp. 3633–3637.

[11] J. Rohdin, T. Stafylakis, A. Silnova, H. Zeinali, L. Burget, and O. Plchot, "Speaker verification using end-to-end adversarial language adaptation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6006–6010.

[12] G. Bhattacharya, J. Monteiro, J. Alam, and P. Kenny, "Generative adversarial speaker embedding networks for domain robust end-to-end speaker verification," *arXiv preprint arXiv:1811.03063*, 2018.

[13] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Advances in Neural Information Processing Systems*, 2017, pp. 2962–2970.

[14] Y. Chen, Y. Assael, B. Shillingford, D. Budden, S. Reed, H. Zen, Q. Wang, L. C. Cobo, A. Trask, B. Laurie *et al.*, "Sample efficient adaptive text-to-speech," *arXiv preprint arXiv:1809.10460*, 2018.

[15] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, Y. Wu *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Advances in Neural Information Processing Systems*, 2018, pp. 4485–4495.

[16] A. Tjandra, S. Sakti, and S. Nakamura, "End-to-end feedback loss in speech chain framework via straight-through estimator," *arXiv preprint arXiv:1810.13107*, 2018.

[17] L. Li, D. Wang, Y. Chen, Y. Shi, Z. Tang, and T. F. Zheng, "Deep factorization for speech signal," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5094–5098.

[18] O. Wiles, A. Koepke, and A. Zisserman, "Self-supervised learning of a facial attribute embedding from video," in *BMVC*, 2018.

[19] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi, "Self-supervised learning of geometrically stable features through probabilistic introspection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3637–3645.

[20] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1134–1141.

[21] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash, "Boosting self-supervised learning via knowledge transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9359–9367.

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

[23] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[24] A. Lozano-Diez, A. Silnova, P. Matejka, O. Glembek, O. Plchot, J. Pešán, L. Burget, and J. Gonzalez-Rodriguez, "Analysis and optimization of bottleneck features for speaker recognition," in *Proceedings of Odyssey*, vol. 2016. ISCA Bilbao, Spain, 2016, pp. 352–357.

[25] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[26] C. Zhang and K. Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances." in *INTERSPEECH*, 2017, pp. 1487–1491.

[27] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[28] 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*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

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

[30] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th Symposium on Operating Systems Design and Implementation)*, 2016, pp. 265–283.

[31] 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.

[32] 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

[33] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5796–5800.