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

Learning Speaker Embedding from Text-to-Speech

Jaejin Cho[1], *Piotr Żelasko*[1], *Jesús Villalba*[1,2], *Shinji Watanabe*[1,2], *Najim Dehak*[1,2]

[1]Center for Language and Speech Processing, Johns Hopkins University, USA
[2]Human Language Technology Center of Excellence, Johns Hopkins University, USA
{jcho52,pzelask2,jvillal7,shinjiw,ndehak3}@jhu.edu

Abstract

Zero-shot multi-speaker Text-to-Speech (TTS) generates target speaker voices given an input text and the corresponding speaker embedding. In this work, we investigate the effectiveness of the TTS reconstruction objective to improve representation learning for speaker verification. We jointly trained end-to-end Tacotron 2 TTS and speaker embedding networks in a self-supervised fashion. We hypothesize that the embeddings will contain minimal phonetic information since the TTS decoder will obtain that information from the textual input. TTS reconstruction can also be combined with speaker classification to enhance these embeddings further. Once trained, the speaker encoder computes representations for the speaker verification task, while the rest of the TTS blocks are discarded. We investigated training TTS from either manual or ASR-generated transcripts. The latter allows us to train embeddings on datasets without manual transcripts. We compared ASR transcripts and Kaldi phone alignments as TTS inputs, showing that the latter performed better due to their finer resolution. Unsupervised TTS embeddings improved EER by 2.06% absolute with regard to i-vectors for the LibriTTS dataset. TTS with speaker classification loss improved EER by 0.28% and 2.88% absolutely from a model using only speaker classification loss in LibriTTS and Voxceleb1 respectively.

1. Introduction

Neural Text-to-Speech (TTS) [1] is gaining great attention due to its simpler system pipeline and improved performance compared to a more conventional statistical TTS system [2]. Several neural TTS models include speech encoder modules that aim to extract latent representation to control desired characteristics such as speaker voice, accent, speaking style, or noise to the synthesized speech [3–7], in addition to a text encoder.

Among those models, Multi-speaker Text-To-Speech (M-TTS) model can synthesize speech imitating the voices of multiple speakers. A key component in M-TTS systems is the speaker encoder that extracts a speaker embedding from one or several utterances of the speaker of interests. This embedding is used to customize the TTS output and generate new utterances from the target speaker. The speaker encoder can be jointly trained with the rest modules of the M-TTS. The speaker encoder can also be trained ahead and used to produce embeddings to train M-TTS on a multi-speaker dataset [5]. While most of the previous papers focused on improving TTS as their final goal, this work's core focus is analyzing the utility of the speaker encoder for Speaker Verification (SV) tasks.

There are three main reasons we expect the TTS to help learn a better speaker embedding. Firstly, by leveraging the encoded latent representation from the reference speech, TTS controls several aspects of the synthesized speech, such as a speaker's voice, speaking style/prosody, and noise. Thus, training an M-TTS that naturally synthesizes desired speech of multiple voices, implies that the speaker encoder extracts a robust speaker embedding, sufficient to discriminate between the voices of different speakers. Building controllable robust TTS systems has been the aim of several past works, as in [8–10]. In our paper, we focus on improving the speaker encoder module. To that end, we use a system proposed previously in [5], with a major difference – the speaker encoder is trained jointly with the rest of TTS modules [3].

Secondly, it has been previously observed that speaker recognition systems suffer from imbalanced or incomplete coverage of the phonetic variability in the speaker embedding space, especially in short utterances [11]. However, we expect that the speaker encoder in M-TTS can amend this problem. We hypothesize that the speaker encoder learns a speaker representation robust to the phonetic variability, given that the phonetic information is mainly encoded by the text encoder.

Finally, M-TTS training does not explicitly require speaker labels but only the paired speech and transcript data. Also, if we even do not have transcripts, a well pre-trained Automatic Speech Recognizer (ASR) can generate pseudo labels for the speech-only data. Thus, it enables unsupervised learning of a speaker embedding similarly as i-vector, the state-of-the-art unsupervised method for learning a speaker embedding [12]. Note that this setup is similar to recent activities in self-supervised speaker embeddings [13], unsupervised/self-supervised speech processing including ASR/TTS joint modeling [14, 15], voice conversion [16, 17], and zero speech challenge (TTS without T) [18]. However, again, the main difference between them and our work is that the primary focus in this paper is to investigate the effectiveness of the learned speaker embedding for the SV task in this unsupervised setup.

In our experiments, we first compared different types of input texts in TTS training, including a human transcript, an ASR-generated transcript, and phone alignment. We found that replacing the human transcript with ASR-generated outputs does not degrade the speaker encoder training in TTS. Also, using the phone alignment input works better than using the ASR transcript in training the TTS for the speaker encoder. We compared the proposed method with other state-of-the-art methods and found that it consistently outperforms them in most experimental setups, both in supervised and unsupervised settings. We conclude that learning the speaker embedding with the TTS criterion indeed helps with the SV task.

2. Learning Speaker Embedding with Text-to-Speech

The main goal of this paper is to investigate whether an M-TTS model can help the speaker encoder to learn better embeddings for the SV task. To that end, we first trained an M-TTS based on the Tacotron 2 [1] architectural design. The original Tacotron 2

http://dx.doi.org/10.21437/Interspeech.2020-2970

is composed of a text encoder, a decoder, and a vocoder. A text encoder $\mathrm{Enc}^{\mathsf{txt}}(\cdot)$ encodes a J-length text input sequence $W = \{w_i \in \mathcal{V} | i = 1, ..., J\}$ with a vocabulary $\mathcal{V}$ into a sequence of D-dimensional hidden vectors $\mathbf{H} = \{\mathbf{h}_i \in \mathbb{R}^D | i = 1, ..., J\}$ as follows:

$$\mathbf{H} = \mathrm{Enc}^{\mathsf{txt}}(W). \quad (1)$$

Then, the decoder $\mathrm{Dec}(\cdot)$ predicts a T-length sequence of target acoustic features $\mathbf{O} = \{\mathbf{o}_t \in \mathbb{R}^{D_{\mathsf{a}}} | t = 1, ..., T\}$ with D_{a}-dimensional features, e.g., Mel-filter-banks, based on a context vector generated by an attention mechanism over hidden vectors $\mathbf{H}$ as follows.

$$\mathbf{o}_t = \mathrm{Dec}(\mathbf{o}_{t-1}, \mathbf{H}) \quad (2)$$

The prediction happens in an auto-regressive fashion, i.e., the prediction of $\mathbf{o}_t$ is performed with the previous acoustic feature $\mathbf{o}_{t-1}$ as a condition. The condition $\mathbf{o}_{t-1}$ would be a ground truth defined as $\mathbf{o}^*_{t-1}$ during training (so called teacher-forcing) or a predicted one during inference.

To enable this model to generate voices of multiple speakers, a speaker encoder $\mathrm{Enc}^{\mathsf{spk}}(\cdot)$ is added to encode a D_{e}-dimensional global speaker embedding vector $\mathbf{e} \in \mathbb{R}^{D_{\mathsf{e}}}$ as follows:

$$\mathbf{e} = \mathrm{Enc}^{\mathsf{spk}}(\mathbf{O}). \quad (3)$$

The speaker embedding vector $\mathbf{e}$ is concatenated to every encoded text vector $\mathbf{H}$ over the sequence [5]. That is, the multi-speaker decoder function is extended from Eq. (2) as follows:

$$\mathbf{o}_t = \mathrm{Dec}^{\mathsf{mlt}}(\mathbf{o}_{t-1}, \mathrm{Cat}(\mathbf{H}, \mathbf{e})), \quad (4)$$

where $\mathrm{Cat}(\cdot)$ is a concatenation function between $\mathbf{H}$ and $\mathbf{e}$. We exclude the explanation of the vocoder, which is usually trained separately to generate a waveform from the Mel-spectrogram. While in [5], they use a pretrained speaker encoder, we propose to jointly train the speaker encoder $\mathrm{Enc}^{\mathsf{spk}}(\cdot)$ in Eq. (3) with the rest of the TTS blocks. Thus, the TTS reconstruction loss $\mathcal{L}^{\mathsf{tts}}$ is defined as follows:

$$\mathcal{L}^{\mathsf{tts}} = \sum_t \left| \mathbf{o}^*_t - \mathrm{Dec}^{\mathsf{mlt}}(\mathbf{o}^*_{t-1}, \mathrm{Cat}(\mathbf{H}, \mathrm{Enc}^{\mathsf{spk}}(\mathbf{O}^*)) \right|_p, \quad (5)$$

where $|\cdot|_p$ denotes an Lp-norm. The actual TTS loss function is a combination of the L1 and L2 losses. M-TTS training does not require speaker labels to learn a embedding extractor, which can be used to reconstruct the speech from different speakers.

If there is an available speaker ID label l^*_s for a speaker s, an additional projection layer $\mathrm{Proj}(\cdot)$ can be added to the speaker encoder to calculate the speaker classification loss.

$$\mathcal{L}^{\mathsf{spk}} = \mathrm{AngSoftMax}(l^*_s, \mathrm{Proj}(\mathbf{e}_s)), \quad (6)$$

where $\mathrm{AngSoftMax}(\cdot)$ denotes an angular softmax loss used for speaker classification [19]. $\mathbf{e}_s$ is an embedding vector obtained by Eq. (3) with the acoustic features of the speaker s. This can be considered as multi-task learning for both TTS and speaker classification. The diagram for the aforementioned description is in Figure 1. In the figure, the TTS loss $\mathcal{L}^{\mathsf{tts}}$ is a sum of L1, L2 reconstruction losses for filter bank prediction, as introduced in Eq. (5) and the additional binary cross entropy loss for the stop token prediction. Optionally, we can also include the speaker classification loss $\mathcal{L}^{\mathsf{spk}}$, as introduced in Eq. (6). Once the M-TTS training finishes either with or without speaker classification loss, only the speaker encoder is used to extract embeddings for SV.

Notably, most of the speech recordings do not include human transcripts W used in Eq. (1), except for ASR-oriented

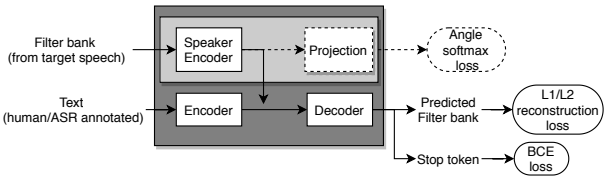

Figure 1: *Multi-speaker TTS module used to learn speaker embedding. The outer most black box represents the TTS module while the grey box inside represents the speaker-related module. Square boxes are modules of the system, and rounds are losses. Dotted parts are added when a speaker classification loss (angle softmax loss here) is added.*

corpora, since obtaining human transcripts is a lengthy and expensive process. Therefore, it is important to check whether automatically generated text inputs from ASRs are suitable for training M-TTS for the speaker encoder. For that purpose, we explored using transcripts generated from ASR systems, that is $\hat{W} = \mathrm{ASR}(\mathbf{O})$ instead of W as an input of Eq. (1). We can have different ASR systems with different Word-Error-Rate (WER)s, as well as using a phone alignment from a hybrid ASR system.

3. Experimental Setup

3.1. Datasets

We use two datasets in our experiments. The first dataset was the *train-clean-100* and *train-clean-360* subsets of the LibriTTS [20], which is read speech corpus designed for TTS research. These subsets are composed of speech considered clean in terms of signal-to-noise ratio (SNR) since their SNRs are higher than 20dB. The subsets were divided into *dev* and *test*, having 1000 and 150 speakers respectively, without speaker overlap. The *dev* was used for training M-TTS and SV backend; the *test* was used for the SV evaluation using the extracted embeddings. We created SV trials from the *test* set, where each trial is a pair of enrollment and test utterances. We made all possible utterance pair combinations removing cross-gender pairs. For each trial, the system determines whether both utterances in the trial belong to the same speaker or not. This dataset has both human transcript and speaker ID labels. With this dataset, we would like to see how appropriately the M-TTS system works depending on the level of supervision, i.e., the amount of the labels used in training, by comparing the systems to an i-vector [12] unsupervised system and a Residual Network (ResNet)-based supervised system [21].

To test the proposed systems on a more challenging and closer to the real-world scenario, we also used Voxceleb1 dataset [22]. This corpus is composed of conversational speech utterances with moderate noise, which are processed from interview videos of 1,251 celebrities uploaded on Youtube. The corpus does not have human transcripts but has speaker labels. The Voxceleb1 *dev* and *test* subsets were used for training and evaluating the model, respectively. No data augmentation was done in training.

3.2. M-TTS system configuration

We used ESPnet-TTS [23] as our M-TTS system. We are planning to make this proposed SV system publically available as an open-source[1].

[1] https://github.com/JaejinCho/espnet_spkidtts.git

Table 1: *EER(%) in SV according to the change of speaker classification loss weight*

Spkloss_W		0	0.001	0.01	0.03	0.3	3
EER (%)	LibriTTS	1.36	1.31	1.16	**1.08**	1.13	1.20
	Voxceleb1	12.62	11.68	6.10	**6.03**	6.82	7.19

For the speaker encoder within the M-TTS system, we used the same network design, ResNet-LDE, as in [21]. The ResNet-LDE network is different from the original x-vector [24] system in that it replaces Time Delay Neural Network (TDNN) layers with a residual network with 2D convolution layers and replaces the pooling layer with a Learnable Dictionary Encoding (LDE) layer [25].

When an additional speaker classification (angular softmax) loss (dotted parts in Figure 1) was added to the M-TTS loss, the speaker loss was weighted before added. Table 1 shows how the weight value affects performance. Note that regardless of the dataset, the weight value 0.03 showed the lowest Equal-Error-Rate (EER). Thus, results for the M-TTS plus speaker classification loss systems are reported using this value throughout the paper.

For the SV back-end system training, the LDA dimension reduction to 150, followed by PLDA [26], was used throughout all the experiments.

3.3. ASR systems description

To examine how ASR-generated text inputs compared to the human transcript affect the TTS training for speaker embeddings, three ASR models were used. The first ASR model was a joint CTC-attention based end-to-end model [27] with convolution and Long Short-Term Memory (LSTM) layers. The training corpus was the Wall Street Journal (WSJ) corpus [28]. The second one was also the same end-to-end model but with transformer architecture [29] and trained on the LibriSpeech corpus [30]. The third model was a hybrid ASR [31] model using factorized TDNN [32] trained with Lattice-Free Maximum-Mutual-Information (LF-MMI) criterion [33], also using Librispeech. With this model, we generated both the transcripts and the phonetic alignments.

The WERs calculated on the *train-clean-100* and *train-clean-360* subsets of the LibriTTS were 44.0, 2.7, and 5.15(%) for the first, second, and third ASR models respectively. The WERs could not be calculated on the Voxceleb1 corpus since there was no human transcript available. However, we expected them to be worse compared to the WERs on the LibriTTS subsets, considering the Voxceleb1 corpus is more challenging due to mismatched acoustic conditions and spontaneous speaking style. Pre-trained models available online were used for the second end-to-end model and the hybrid model.

4. Results and Analysis

4.1. LibriTTS results

4.1.1. Transcript source analysis

First, different TTS systems, either with or without a speaker classification loss, were trained using different text inputs, either transcribed by human annotators or generated from ASR systems. Once the TTS systems had been trained, speaker encoders were used to extract speaker embeddings to evaluate an SV task. Table 2 shows the comparative results with EER and MinDCF at p=0.01. Comparing from the first to fourth rows in Table 2, we observe that regardless of using a manual or ASR transcript, the SV performance was not affected.

Table 2: *Comparison between different text inputs for M-TTS training: SV evaluation results on LibriTTS with EER(%) and MinDCF with p=0.01. Human Trans. means human transcript.*

TTS text input (ASR training)	WER(%)	M-TTS		M-TTS + SpkID loss	
		EER(%)	MinDCF	EER(%)	MinDCF
Manual Trans.	N/A	1.34	0.510	1.13	0.502
E2E ASR Trans. (WSJ)	44.00	1.36	0.510	1.08	0.497
E2E ASR Trans. (LibriSpeech)	2.70	1.32	0.511	1.06	0.496
Hybrid ASR Trans. (LibriSpeech)	5.15	1.49	0.516	1.06	**0.493**
Hybrid Phn. Align. SR1 (LibriSpeech)	N/A	**1.12**	**0.501**	**1.04**	0.502
Hybrid Phn. Align. SR3 (LibriSpeech)	N/A	1.31	0.524	1.11	0.509

In the table, the M-TTS systems trained with phone alignment inputs with the frame Sub-sampling Rate (SR) 1 performed the best. Here, frame SR means how many acoustic frames were used to predict one phone label. For example, the hybrid ASR we used generated one phone label every three acoustic frames, i.e., frame SR is 3. Thus, to make SR 1, we up-sampled each predicted phone label by 3. One possible reason for the best systems could be the aligned phoneme inputs to TTS reduce the burden for the speaker encoder to include phoneme or pronunciation information since that information can be more easily learned by the text encoder module. This might also happen with the transcript inputs, but the degree might be less. Another possible reason is that the silence and short pause duration information obtained by the aligned phoneme input could make TTS training more stable.

Note that there was no significant difference in SV performance between using an ASR with high WER or one with low WER, as it is shown between E2E ASR Transcript (WSJ) and E2E ASR Transcript (LibriSpeech) in Table 2. Considering that the difference in the WERs is quite large, this is an interesting observation. For the 44.0% WER ASR-generated transcript, we investigated the quality of the trained TTS outputs, and most of them were nonsensical sounds due to the attention not trained well. Nevertheless, it did not seem to affect the quality of speaker representations negatively. One explanation for this result could be that although the TTS failed to learn the attention, it still tries to include in the synthesized speech other speech characteristics such as a speaker's voice to reduce the loss. The attention problem was solved with transcripts from ASRs having lower WERs, synthesizing reasonable speech.

4.1.2. Model comparison by level of supervision

Table 3 compares several models with different levels of supervision. We have i-vector and ResNet-LDE as previously proposed models. As unsupervised learning for speaker embedding, i-vector and M-TTS without speaker ID loss trained with the ASR phone alignment (M-TTS w/ ASR Phn. Ali. SR1, the fifth row in Table 3) were compared, showing the latter system outperformed i-vector largely in both EER and MinDCF.

Next, as a supervised training setup with either a human transcript or speaker ID labels, two M-TTS systems (the 3rd and the last rows in Table 3) were compared to ResNet-LDE. Here, ResNet-LDE showed a similar result to the M-TTS system trained with only the human transcript (M-TTS), which suggests that a speaker embedding can be learned implicitly by

Table 3: *Comparison between proposed M-TTS models to existing models: SV results on LibriTTS with EER(%) and MinDCF with p=0.01. Human Trans. means human transcript.*

System	Metrics		Label usage	
	EER(%)	MinDCF	Human Trans.	SpkID
i-vector	3.18	0.639		
ResNet-LDE	1.32	**0.500**		◯
M-TTS	1.34	0.510	◯	
M-TTS + SpkID loss	1.13	0.502	◯	◯
M-TTS w/ ASR Phn. Ali. SR1	1.12	0.501		
M-TTS + SpkID loss w/ ASR Phn. Ali. SR1	**1.04**	0.502		◯

training an M-TTS with a transcript. Meanwhile, an M-TTS trained on ASR phone alignments with only speaker ID labels (M-TTS + SpkID loss w/ ASR Phn. Align. SR1) outperformed both ResNet-LDE and M-TTS.

Adding speaker classification loss to M-TTS in training (4nd row in Table 3) outperformed the pure discriminative system (ResNet-LDE). This implies that multi-task learning for both TTS and the speaker classification enables better speaker embedding learning for the SV task.

To sum up, the unsupervised training setup, M-TTS w/ ASR Phn. Ali. SR1, outperformed the i-vector and the state-of-the-art ResNet-LDE that learns speaker embedding in a supervised way. Then, adding a speaker ID loss to the M-TTS w/ ASR Phn. Ali. SR1 outperformed the ResNet-LDE system further.

4.2. Voxceleb1 results

4.2.1. ASR transcript analysis

All the ASRs used on this dataset were the same as used in LibriTTS. Although the performances of ASRs cannot be calculated due to unavailable human transcripts, it is expected that the WERs are worse compared to ones calculated on LibriTTS due to domain mismatch between the ASR training data (read and clean speech) and Voxceleb1 (conversational and moderately noisy speech).

The results are shown in Table 4. TTS training with Hybrid ASR Transcript (LibriSpeech) was skipped here since it did not improve with regard to E2E in the LibriTTS experiments. The performance gaps between using transcripts and phone alignments became larger on Voxceleb1, compared to LibriTTS. Different from what is observed on LibriTTS, Hybrid ASR Phn. Align. SR3 (LibriSpeech) worked better than Hybrid ASR Phn. Align. SR1 (LibriSpeech). This inconsistent result is possibly due to less accurate phone alignment prediction on Voxceleb1.

4.2.2. Model comparison by level of supervision

The systems in the Hybrid ASR Phn. Align. SR3 (LibriSpeech) row in Table 4 were compared to previously published systems. The results are shown in Table 5. In an unsupervised scenario of learning speaker embedding, two i-vector systems and M-TTS w/ ASR Phn. Align. SR3. were compared. Although the 2048-GMM i-vector system outperformed the proposed system, the comparison result could be different with more data since the Neural Network (NN) based systems are known to require more data in training to perform better than classical methods, e.g. i-vector. Compared to ResNet-LDE, the proposed M-TTS w/ ASR Phn. Align. SR3. works better even though the proposed system did not use any speaker labels in training.

As supervised ways of learning speaker embedding, we

Table 4: *Comparison between different text inputs for M-TTS training on Voxceleb1: SV evaluation results on Voxceleb1 with EER(%) and MinDCF with p=0.01.*

TTS text input (ASR training corpus)	M-TTS		M-TTS + SpkID loss	
	EER(%)	MinDCF	EER(%)	MinDCF
E2E ASR Transcript (WSJ)	12.62	0.792	6.03	0.478
E2E ASR Transcript (LibriSpeech)	13.49	0.797	6.50	0.554
Hybrid ASR Phn. Align. SR1 (LibriSpeech)	7.21	0.617	4.53	0.443
Hybrid ASR Phn. Align. SR3 (LibriSpeech)	**6.37**	**0.556**	**4.11**	**0.416**

Table 5: *Comparison of models by supervision level on Voxceleb1: SV results on Voxceleb1 with EER(%) and MinDCF with p=0.01.*

System	EER(%)	MinDCF	SpkID usage
1024-GMM i-vector [22]	8.8	n/a	
2048-GMM i-vector [35]	5.51	0.462	
ResNet-LDE	6.99	0.563	◯
SincNet+LIM [34]	5.80	n/a	◯
M-TTS w/ ASR Phn. Align. SR3	6.37	0.556	
M-TTS + SpkID loss w/ ASR Phn. Align. SR3	**4.11**	**0.416**	◯

compared ResNet-LDE and M-TTS + SpkID loss w/ ASR Phn. Align. SR3. The proposed model outperformed ResNet-LDE significantly. It also outperformed SincNet+LIM [34], which learns semi-supervised embedding using mutual information.

5. Conclusion

In this work, M-TTS systems including a speaker encoder were used to learn speaker embeddings for the SV task. To train speaker embeddings with this method on a dataset without transcripts, we compared using manual transcripts, ASR transcripts from E2E and hybrid systems, and phone alignments predicted from the hybrid ASR, as TTS text inputs. We observed that phone alignments performed better than ASR transcripts. Compared to generative i-vectors and discriminative ResNet-LDE, the proposed supervised TTS model using only speaker labels achieved better performance. Regarding unsupervised systems, our unsupervised TTS outperformed the i-vector model with 1024 Gaussians but not with regard to the larger version with 2048 Gaussians. How to improve the unsupervised version is the object of further investigation.

A handicap of the proposed model is the high computing cost of training TTS models, roughly 10× higher than a pure discriminative model. One way to accelerate computation is to increase the reduction factor in TTS training. Another possible future direction is to study an in-depth relationship between the quality of the synthesized speech generated from M-TTS and the SV performance using the speaker encoder of the M-TTS. Finally, using one part of an utterance to extract the speaker embedding while using another part for reconstruction in M-TTS training can further improve the embedding by disentangling speaker and phone information [13]. We intend to explore these directions in future work.

6. References

[1] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *ICASSP 2018*. IEEE, 2018, pp. 4779–4783.

[2] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *speech communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[3] E. Nachmani, A. Polyak, Y. Taigman, and L. Wolf, "Fitting new speakers based on a short untranscribed sample," in *ICML*, 2018, pp. 3680–3688.

[4] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: 2000-speaker neural text-to-speech," in *International Conference on Learning Representations*, 2018.

[5] 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. 4480–4490.

[6] Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *ICML*, 2018, pp. 5167–5176.

[7] W.-N. Hsu, Y. Zhang, R. J. Weiss, Y.-A. Chung, Y. Wang, Y. Wu, and J. R. Glass, "Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization," in *ICASSP 2019*. IEEE, 2019, pp. 5901–5905.

[8] W.-N. Hsu, Y. Zhang, R. Weiss, H. Zen, Y. Wu, Y. Cao, and Y. Wang, "Hierarchical generative modeling for controllable speech synthesis," in *International Conference on Learning Representations*, 2019.

[9] G. E. Henter, J. Lorenzo-Trueba, X. Wang, and J. Yamagishi, "Deep encoder-decoder models for unsupervised learning of controllable speech synthesis," *arXiv preprint arXiv:1807.11470*, 2018.

[10] K. Akuzawa, Y. Iwasawa, and Y. Matsuo, "Expressive speech synthesis via modeling expressions with variational autoencoder," in *Proc. Interspeech 2018*, 2018, pp. 3067–3071.

[11] R. Vogt, J. Pelecanos, N. Scheffer, S. Kajarekar, and S. Sridharan, "Within-session variability modelling for factor analysis speaker verification," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[12] 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, 2010.

[13] T. Stafylakis, J. Rohdin, O. Plchot, P. Mizera, and L. Burget, "Self-Supervised Speaker Embeddings," in *Proc. Interspeech 2019*, 2019, pp. 2863–2867.

[14] A. Tjandra, S. Sakti, and S. Nakamura, "Listening while speaking: Speech chain by deep learning," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 301–308.

[15] T. Hori, R. Astudillo, T. Hayashi, Y. Zhang, S. Watanabe, and J. Le Roux, "Cycle-consistency training for end-to-end speech recognition," in *ICASSP 2019*. IEEE, 2019, pp. 6271–6275.

[16] A. van den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6306–6315.

[17] J. Lorenzo-Trueba, J. Yamagishi, T. Toda, D. Saito, F. Villavicencio, T. Kinnunen, and Z. Ling, "The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 195–202.

[18] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black, L. Besacier, S. Sakti, and E. Dupoux, "The Zero Resource Speech Challenge 2019: TTS Without T," in *Proc. Interspeech 2019*, 2019, pp. 1088–1092.

[19] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[20] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech," in *Proc. Interspeech 2019*, 2019, pp. 1526–1530.

[21] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, L. P. García-Perera, F. Richardson, R. Dehak, P. A. Torres-Carrasquillo, and N. Dehak, "State-of-the-art speaker recognition with neural network embeddings in nist sre18 and speakers in the wild evaluations," *Comput. Speech Lang.*, vol. 60, 2020.

[22] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Comput. Speech Lang.*, vol. 60, 2020.

[23] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan, "ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit," in *ICASSP 2020*. IEEE, 2020, pp. 7654–7658.

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

[25] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

[26] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." in *Odyssey*, vol. 14, 2010.

[27] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *ICASSP 2017*. IEEE, 2017, pp. 4835–4839.

[28] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[29] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, "A comparative study on transformer vs rnn in speech applications," *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 449–456, 2019.

[30] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *ICASSP 2015*. IEEE, 2015, pp. 5206–5210.

[31] 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. CONF. IEEE Signal Processing Society, 2011.

[32] 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.

[33] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," *Interspeech 2016*, pp. 2751–2755, 2016.

[34] M. Ravanelli and Y. Bengio, "Learning Speaker Representations with Mutual Information," in *Proc. Interspeech 2019*, 2019, pp. 1153–1157.

[35] Z. Peng, S. Feng, and T. Lee, "Mixture factorized auto-encoder for unsupervised hierarchical deep factorization of speech signal," in *ICASSP 2020*. IEEE, 2020, pp. 6774–6778.