On the Effect of Purely Synthetic Training Data
for Different Automatic Speech Recognition Architectures

Benedikt Hilmes[1,2,], Nick Rossenbach[1,2,*], Ralf Schlüter[1,2]*

[1]RWTH Aachen University, Germany
[2]AppTek GmbH, Germany

[*]*equal contribution*

`<lastname>@ml.rwth-aachen.de`

Abstract

In this work we evaluate the utility of synthetic data for training automatic speech recognition (ASR). We use the ASR training data to train a text-to-speech (TTS) system similar to FastSpeech-2. With this TTS we reproduce the original training data, training ASR systems solely on synthetic data. For ASR, we use three different architectures, attention-based encoder-decoder, hybrid deep neural network hidden Markov model and a Gaussian mixture hidden Markov model, showing the different sensitivity of the models to synthetic data generation. In order to extend previous work, we present a number of ablation studies on the effectiveness of synthetic vs. real training data for ASR. In particular we focus on how the gap between training on synthetic and real data changes by varying the speaker embedding or by scaling the model size. For the latter we show that the TTS models generalize well, even when training scores indicate overfitting.

Index Terms: synthetic data generation, text-to-speech, speech recognition, semi-supervised training

1. Introduction

Current literature shows the capability of synthetic data to complement real data and thus improve automatic speech recognition (ASR) training through various ways and techniques [1, 2, 3, 4, 5] . Commonly end-to-end architectures are trained with a combination of real and synthetic data, where especially models like the attention-based encoder-decoder (AED) benefit from the additional synthetic data [2, 4, 5]. However, we still lack a good understanding of how well synthetic data is able to replace real data. To this end, we suggest to use synthetic training data *only* to analyze and compare its ASR training utility against real data. Such a study helps to gain further insight on the discrepancy between synthetic and real data. Recent work has presented large TTS systems trained on much more data than typically available for academically defined tasks [6, p. 5]. But even with such an industrial scale system it was not possible to create synthetic data that is equally utilizable to real data. In this work we explore the corresponding performance gap for three ASR architectures: A classic Gaussian mixture based system using hidden Markov model (GMM-HMM), a hybrid deep neural network HMM (Hybrid) [7] and an AED [8] system. A lot of previous work on synthetic data analysis was done on data generated by autoregressive text-to-speech (TTS) models, like Tacotron-2 [9], with the exception of [10]. In this work we focus on a simple non-autoregressive TTS model which is similar to FastSpeech, but with a BLSTM-decoder, and an encoder with mixed convolutions and BLSTM. There are more recent architectures which exhibit strong performance on standard TTS tasks. However, we consider a more simple and established

TTS architecture that we know from previous experiments [11] to work on ASR-specific training data and thus is expected to be better suited for our analysis. The contributions of this work are as follows:

- Investigating how robust traditional ASR approaches such as GMM-HMM react to synthetic data compared to a more modern Hybrid or AED system.
- Quantifying the impact of low-quality Griffin & Lim vocoding for the usability of audio data.
- Showing how simply increasing the number of TTS model parameters already improves the usability of the synthetic data for ASR.
- Showing that in the context of this work, more sophisticated speaker embedding systems greatly influence the performance.

Especially, the effect of hyper-parameter tuning rarely is covered in TTS-literature due to the expensive MOS evaluations this would ensue. A large scale comparison of training various ASR architectures with both real and synthetic data has been done in [2]. We extend [2] and [11] through analyzing the effect of ASR performance by using synthetic data *only* for ASR training. Our work and software is publicly accessible and will remain as such [1].

2. Speech Synthesis

As TTS system we use the non-autoregressive (NAR) model from [11], which is closely related to [12], extended with Gaussian Upsampling [13]. It consists of a phoneme encoder, duration predictor and a feature decoder, which follow exactly the design as in [11]. For both duration prediction and decoding we pass a speaker embedding to enable multi speaker capabilities. In the simplest case this is generated by look-up table on the speaker ID which is learned during training, but we also investigate different more elaborate approaches, by generating fixed speaker embeddings through stronger pre-trained models. Training is done on the reference durations, extracted from a given alignment. As spectrogram targets we use globally normalized 80-dimensional log-mel features with frame shift 12.5 ms and window size of 50 ms. The spectrogram predictions are transformed into 512-dimensional linear features for Griffin & Lim (G&L) vocoding [14] via a pre-trained bi-directional LSTM (BiLSTM) network. As validated in the baseline experiments of this paper, simple G&L vocoding does not reduce the quality of the generated audio for ASR training. Counting both neural models, our architecture consists of 63M parameters. The phoneme set consists of ARPA-BET phoneme sym-

---

[1]https://github.com/rwth-i6/returnn-experiments/tree/master/2024-pure-synthetic-data

Table 1: *Evaluation on LibriSpeech dev-clean and dev-other corpora. Only data from LibriSpeech train-clean-100 is used for TTS training. Vocoding only used features extracted from the real data, vocoded by Griffin-Lim. TTS-Durations indicates whether the model predicts phoneme durations via the duration predictor or is fed the ground truth durations from the alignment.*

Data	Silence Removal	TTS-Durations	Data Length	WER [%]				
				GMM-HMM		AED		Hybrid
				clean	other	clean	other	other
Real	No	-	100.6h	**8.1**	**25.9**	**7.5**	**18.9**	**15.0**
	Yes	-	88.7h	8.7	28.0	7.8	20.3	15.5
Vocoding Only	No	-	100.6h	8.7	27.7	7.6	19.4	15.2
	Yes	-	88.7h	9.5	28.8	8.0	20.1	16.1
Synthetic	Yes	pred.	81.1h	10.0	32.2	14.1	37.6	26.2
		real	88.3h	9.7	32.2	10.9	31.8	26.8

bols without stress marker. We mark word boundaries and possible silence with a `[space]` token between the last phoneme of a word and the first of the next respectively.

3. Speech Recognition

Our **GMM-HMM** model is implemented in RASR [16], which is functionally close to the Montreal forced aligner (MFA) commonly used for TTS works such as FastSpeech-2 [17]. Training parameters are optimized on LibriSpeech-100h with focus on ASR performance. The overall training consists of the several steps, which in more detail are explained in [11]. In this work we use the GMM-HMM twofold. First we use the alignments produced by the system as ground truth alignments for duration prediction in our TTS. For this we calculate the Viterbi path for our best alignment, setting a duration of zero for `[space]` tokens where no silence was aligned. For recognition, we use the model together with a pre-trained 4-gram count-based language model (LM) from the LibriSpeech [18] dataset.

On top of the GMM-HMM we use a **Hybrid** model which predicts the frame-label posterior probability by a neural network. We use the final alignment output from the ASR GMM-HMM as training targets. The neural network (NN) consists of a stack of 8 1024-dimensional BiLSTM layers followed by a linear layer with softmax activation and output size 12001 to match the corresponding CART [19] labels. In total the model consists of 210M parameters. For recognition we again use the LibriSpeech 4-gram LM.

Our last model is an **AED** model as used in [11] with 12 conformer blocks as encoder and single layer LSTM for the decoder, resulting in 98M parameters. We use BPE labels [21] with 2k merge operations as the output units. Different from the TTS model we use a frame shift to 10 ms and the window size to 25 ms for the feature extraction. The models uses a downsampling factor of 6. To improve training stability we increase the number of encoder layers over time, starting with 2 layers which are increased by 2 every 5 sub-epochs beginning the first increase after 10 sub-epochs reaching full model size at 10 full epochs (30 sub-epochs). For data augmentation we use SpecAugment [20] and apply speed-pertubation via librosa.resample()[2] uniformly distributing scales 0.9/1.0/1.1 among the input. Recognition results are without the use of an external language model.
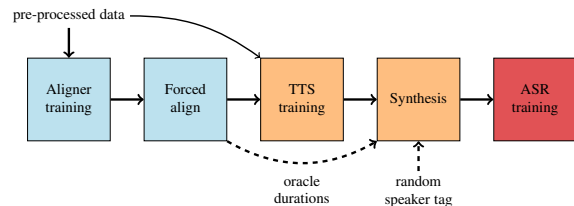

Figure 1: *Experiment pipeline for synthetic data training.*

4. Pipeline

In general our pipeline consists of 5 steps analogue to [11], which is visualized Figure 1. First the aligner is trained on the pre-processed data and a forced alignment generated as duration reference for the TTS model. With that the TTS is trained and used for generation of synthetic data. Afterwards the ASR models are trained on the generated data for a final evaluation. We remove unnatural long silence portions by using the silence filter from FFmpeg[3] with a threshold of -50dB. Synthesis is done on two portions of data, the TTS training data and a similar amount of unseen text data. Synthesizing the data seen during TTS allows our analysis to be done with as little TTS errors as possible, while using unseen data helps verifying the validity of our results. In the baseline case we randomize the speaker ID, using speaker IDs of train-clean-100. The synthesized data is then used to train the different ASR systems **without** any additional real data. Similar to [11], we additionally let the TTS model synthesize the data with access to the target durations which were seen during training. We denote this in the tables by marking the durations as *real* durations.

5. Experiments

5.1. Data and Training

For all our experiments we use the the train-clean-100 subset of LibriSpeech as supervised training data. While common literature usually synthesizes an unseen part of train-clean-360 [1, 2, 4], we also conduct experiments on the previously seen training data. Comparing results to synthesizing unseen data, we can observe generalization effects of the TTS. As cross validation (CV) set for ASR training we use a combination of dev-clean and dev-other. For TTS training we construct our own CV set where we split 4 sequences per speaker from the training

---

[2]https://librosa.org/doc/latest/generated/librosa.resample.html

[3]https://ffmpeg.org/ffmpeg-filters.html#silenceremove

Table 2: *Generalization Results. Evaluation of LibriSpeech dev-clean and dev-other corpora. Only data from LibriSpeech train-clean-100 is used for TTS training. The synthetic data is created by using either text from train-clean-100h (LS-100) or an equivalent subset of train-clean-360h (100h-LS-360). γ denotes scaling the hidden dimension of the TTS by this factor. ω denotes scaling the number of layers by this factor.*

Data	(synth.) Audio Data	TTS-Dur.	Scale Type	Model Scale	WER [%]				
					GMM-HMM		AED		Hybrid
					clean	other	clean	other	other
Real	LS-100	-	-	-	**8.1**	**25.9**	**7.5**	**18.9**	**15.0**
Synth.	LS-100	pred.	-	-	10.0	32.2	14.1	37.6	26.2
			Dimension	$\gamma = 1.5$	9.8	31.3	11.7	32.4	23.4
				$\gamma = 2.0$	**9.5**	**31.0**	**10.5**	**30.5**	**22.1**
			Layers	$\omega = 2$	9.7	31.8	13.5	36.1	24.4
		real	-	-	9.7	32.2	10.9	31.8	26.8
			Dimension	$\gamma = 1.5$	9.6	30.7	9.7	28.9	24.5
				$\gamma = 2.0$	9.4	**30.5**	**9.3**	**27.6**	**23.1**
			Layers	$\omega = 2$	**9.2**	30.9	10.2	29.9	24.8
	100h-LS-360	pred.	-	-	10.3	32.9	19.1	43.8	26.9
			Dimension	$\gamma = 1.5$	9.9	32.9	16.1	38.0	24.5
				$\gamma = 2.0$	**9.8**	**31.3**	**15.0**	**35.7**	**23.2**
			Layers	$\omega = 2$	10.1	32.2	18.6	41.7	24.7

Table 3: *Train and cross-validation (CV) mean-average error (MAE) Scores of TTS models.*

Scale Type	Model Scale	Spectrogram MAE		Duration MAE	
		Train	CV	Train	CV
-	-	0.305	0.376	1.373	1.461
Dimension	$\gamma = 1.5$	0.278	0.375	1.201	1.448
	$\gamma = 2.0$	0.264	0.373	1.090	1.453
Layers	$\omega = 2$	0.296	0.369	1.333	1.451

data resulting in 1004 sequences. To generate phoneme representations for words not contained in the LibriSpeech lexicon we use Sequitur [15]. For the AED model we use byte-pair encoding [21] with 2000 merge operations. We evaluate all of our models on *dev-clean* and *dev-other* and do not apply silence removal. The NAR-TTS is trained for 200 steps, the GMM-HMM for 100 EM-steps, the Hybrid for ∼13 full epochs and the AED model for ∼165 full epochs. All experiments were done on a single consumer 11Gb GPU for easy reproducibility. As optimizer we use Adam [22] with a learning rate decay factor of 0.9 based on the CV score.

5.2. Effect of Vocoding and Data Preprocessing

Table 1 shows a comparison of our three baseline models. For this we first train the models on train-clean-100, with and without the silence removal. All three models degrade by a similar amount, which is to be expected, since silence portions of both training and test data differ. In lines three and four of Table 1, the effect of vocoding is shown. For this we extract mel features from the real data and convert them back to audio with our vocoder. Here a first difference of the ASR models becomes visible. The degradation of GMM-HMM ranges from 0.8% to 1.8% absolute for dev-other, depending on the inclusion of silence removal. For the two neural models this degradation is much less, ranging from an improvement of 0.2% to a degradation of 0.6 % word error rate (WER). When replacing the real data with TTS generated audio again the models behave differently. For dev-clean GMM-HMM is able to keep the best

performance of ∼25% relative increase, while for AED with predicted phoneme durations the WER doubles. When using the aligned durations the WER of the AED improves by 4% absolute while improvements for GMM-HMM are only marginal. For dev-other this effect is similar, but in this case dominated by the fact that the GMM-HMM is already showing weak performance on the more noisy data. For the Hybrid model the performance degrades by around two-thirds, with the special exception that TTS with oracle durations do not help the model. While the TTS converges without silence removal, contrary to previous experiments for autoregressive models in [2], results are significantly worse and thus omitted from the table.

5.3. Model sizes and Generalization

Next-up we investigate the influence of different model sizes on both the generalization capabilities of the TTS model and the influence on ASR training. For this we chose two different model scaling approaches. TTS literature usually reports on a single set of hyperparameters, but due to the possibility of automatic evaluation through ASR training and recognition, we can conduct a study on different choices. In the first approach we scale TTS model dimensions by a factor γ, meaning that e.g. for γ = 1.5 the 512-dimensional layers are increased to 768. Analogue we indicate scaling the layer amount with ω, meaning that e.g. for ω = 2 there are twice as many BiLSTM layers in the TTS models. As seen in the upper part of Table 2 increasing the model size or the layer count for audio generation in both cases helps the ASR models. Here the AED model benefits the most from the larger models, increasing relative performance on dev-clean by ∼30% and on dev-other by ∼20%. While the Hybrid model shows improvements of ∼15% on dev-other, the GMM-HMM model only improves marginally with data generated by larger TTS models. This also contradicts to the common idea that deeper models are able to hold even with larger models, while having a friction of the parameters. In the case of generating synthetic data for ASR training this paradigm seems to be not trivially realizable. We hypothesize that larger TTS systems are able to better replicate the acoustic structure of the train-

Table 4: *Speaker Embedding results. WER [%] evaluation of LibriSpeech dev-clean and dev-other corpora. Gaussian upsampling with SAT alignment used for TTS. No shuffling means TTS sees the same speaker embedding as during training.*

Data	TTS-Durations	Embedding Type	Shuffle Embedding	WER [%]				
				GMM-HMM		AED		Hybrid
				clean	other	clean	other	other
Real	-	-	-	**8.1**	**25.9**	**7.5**	**18.9**	**15.0**
Synthetic	pred.	Linear	Yes	10.0	**32.2**	**14.1**	37.6	26.2
			No	**9.8**	33.0	14.7	39.3	25.5
		X-Vectors	Yes	10.3	33.3	16.1	40.7	26.4
			No	10.3	32.5	**14.1**	36.5	24.0
		Resemblyzer	Yes	10.1	32.4	15.5	38.4	25.2
			No	10.1	32.6	14.6	**36.2**	**23.8**
	real	Linear	Yes	9.7	32.2	10.9	31.8	26.8
			No	9.6	33.1	10.9	31.6	26.2
		X-Vectors	Yes	9.8	32.4	10.9	32.6	26.6
			No	**9.5**	**31.2**	10.1	28.3	24.7
		Resemblyzer	Yes	9.8	32.0	11.0	31.2	25.7
			No	9.8	**31.0**	**9.9**	**27.8**	**24.5**

ing data, which then reflects in better synthesis of seen data. In the lower section of Table 2 we show results on an unseen portion of train-clean-360. Namely, we chose transcriptions that in the original corpus relates to 100h of data, hence the name 100h-LS-360. As to be expected the general performance of the TTS on unseen data is worse than on the training data, still the larger TTS models are able to improve the performance of the ASR models, even though training scores indicated overfitting, as visible in Table 3. This confirms our perception that for synthetic data generation TTS training loss scores are not meaningful as an indicator for generalisation and performance on a held-out dataset. A notable difference is visible in the performance of the different ASR models on the unseen data. The GMM-HMM is almost able to replicate the performance compared to the seen training data, with an absolute difference of 0.3% WER on dev-clean and dev-other for the best results. A similar result is visible for the Hybrid model, where the degradation is only 1.1% WER absolute. In the case of AED the performance is a lot worse, where the best performance with unseen data degrades by almost 50% relative on dev-clean and ∼15% relative on dev-clean. This indicates that modelling errors done by the TTS model during synthesis of unseen data hurt the performance of the AED a lot more than for the other two models.

5.4. Speaker Representations

As a last study we investigate the influence of the speaker embedding on the performance of the synthetic data in ASR training. In order to generate more expressive speaker embeddings we train an X-Vector model [23] on train-clean-100, as well as taking embeddings directly from the pre-trained *Resemblyzer*[4] model [24]. We train our TTS model by replacing the lookup table by the generated embeddings. The results of this can be found in Table 4. For our baseline keeping the speaker tags random does not make the synthesized data worse for ASR training. Adding the embeddings generated by both X-Vectors and Resemblyzer does not help improve over the initial baseline and rather the performance degrades, especially in the case of AED. Only when not shuffling the embeddings during synthesis the pre-trained embeddings are able to keep up with the baseline,

surpassing it together with real durations. From this we conclude that overall the speaker embeddings generated by the TTS model generalize well. Feeding embeddings from more elaborate models without changes to the model makes the TTS overfit instead of benefiting from richer embeddings. Nevertheless, in the correct setting, they can provide meaningful information to the model, as seen in the results with real durations.

6. Conclusions

In this work we used a text-to-speech (TTS) model for the generation of synthetic data for automatic speech recognition (ASR) training. We modified the TTS system in different aspects and investigated how this impacts the ASR training on the synthetically generated data. Increasing the size of the TTS led to more overfitting according to the training and validation scores. Still, when using the enlarged TTS for synthetic data generation, the ASR performance would improve. This means hyperparameter tuning for TTS and proper evaluation is required before drawing conclusions from ASR training procedures involving synthetic data. Basing model selection solely on loss scores does not suffice. In a second set of experiments, we increased the TTS complexity by adding pre-trained networks for speaker modeling. In contrast to enlarging the model, the results were less conclusive. Only in some of the experimental settings the ASR performance would improve, and the improvements were not consistent among the different ASR architectures used. It seems that the TTS tends to overfit on the given embeddings, which is reflected in the performance increase when using real durations and using the same speaker embedding as seen during training for synthesis. We made the additional observation, that the vocoding of log-mel-features using a low-quality method such as Griffin & Lim does not strongly degrade the utilization of the audio data for ASR training, while reducing the overhead for data generation significantly. Overall we have seen that gap between real and synthetic data is smaller for traditional ASR systems. Real phoneme variations and stronger speaker embeddings affected these systems much less than an attention-encoder-decoder ASR systems. Future work should aim to find suitable aspects in synthetic data which correlate with the ASR performance across different model conditions.

7. Acknowledgments

8. References

[1] A. Laptev, R. Korostik, A. Svischev, A. Andrusenko, I. Medennikov, and S. Rybin, "You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation," *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, October 2020.

[2] N. Rossenbach, M. Zeineldeen, B. Hilmes, R. Schlüter, and H. Ney, "Comparing the benefit of synthetic training data for various automatic speech recognition architectures," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 788–795.

[3] A. Tjandra, S. Sakti, and S. Nakamura, "End-to-end feedback loss in speech chain framework via straight-through estimator," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6281–6285.

[4] M. K. Baskar, L. Burget, S. Watanabe, R. F. Astudillo, and J. H. Cernocky, "Eat: Enhanced ASR-TTS for self-supervised speech recognition," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021.

[5] T.-Y. Hu, M. Armandpour, A. Shrivastava, J.-H. R. Chang, H. Koppula, and O. Tuzel, "SYNT++: Utilizing Imperfect Synthetic Data to Improve Speech Recognition," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2022.

[6] Seed-Team and ByteDance, "Seed-tts: A family of high-quality versatile speech generation models," *ArXiv*, vol. abs/2406.02430, p. 5, 2024.

[7] H. A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*. Springer Science & Business Media, 2012, vol. 247.

[8] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[9] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, April 2018.

[10] S. Ueno, M. Mimura, S. Sakai, and T. Kawahara, "Data augmentation for asr using tts via a discrete representation," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, December 2021, pp. 68–75.

[11] N. Rossenbach, B. Hilmes, and R. Schlüter, "On the relevance of phoneme duration variability of synthesized training data for automatic speech recognition," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023, pp. 1–8.

[12] A. Pérez-González-de Martos, A. Sanchis, and A. Juan, "VRAIN-UPV MLLP's system for the Blizzard Challenge 2021," *Festvox Blizzard Challenge 2021*, October 2021. [Online]. Available: http://arxiv.org/abs/2110.15792v1

[13] J. Shen, Y. Jia, M. Chrzanowski, Y. Zhang, I. Elias, H. Zen, and Y. Wu, "Non-attentive tacotron: Robust and controllable neural TTS synthesis including unsupervised duration modeling," *ArXiv*, vol. abs/2010.04301v3, October 2020.

[14] D. W. Griffin, D. S. Deadrick, and J. S. Lim, "Speech synthesis from short-time fourier transform magnitude and its application to speech processing," in *ICASSP '84, San Diego, California, USA, March 19-21, 1984*, pp. 61–64. [Online]. Available: https://doi.org/10.1109/ICASSP.1984.1172423

[15] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, May 2008.

[16] S. Wiesler, A. Richard, P. Golik, R. Schlüter, and H. Ney, "RASR/NN: The RWTH neural network toolkit for speech recognition," in *ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2014.

[17] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "FastSpeech 2: Fast and High-Quality End-to-End Text to Speech," in *International Conference on Learning Representations (ICLR)*, December 2021.

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

[19] K. Beulen and H. Ney, "Automatic question generation for decision tree based state tying," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, vol. 2, May 1998, pp. 805–808.

[20] 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.

[21] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Aug. 2016, pp. 1715–1725.

[22] 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*, Y. Bengio and Y. LeCun, Eds., 2015.

[23] 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)*, April 2018, pp. 5329–5333.

[24] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. Lopez Moreno, Y. Wu *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," *Advances in neural information processing systems*, vol. 31, 2018.