MODULE COMPARISON OF TRANSFORMER-TTS FOR SPEAKER ADAPTATION BASED ON FINE-TUNING

Katsuki Inoue*, Sunao Hara*, Masanobu Abe*
* The Graduate School of Interdisciplinary Science and Engineering in Health Systems, Okayama University, Okayama, Japan
E-mail: k_inoue@a.cs.okayama-u.ac.jp, hara@okayama-u.ac.jp, abe-m@okayama-u.ac.jp

***Abstract*—End-to-end text-to-speech (TTS) models have achieved remarkable results in recent times. However, the model requires a large amount of text and audio data for training. A speaker adaptation method based on fine-tuning has been proposed for constructing a TTS model using small scale data. Although these methods can replicate the target speaker's voice quality, synthesized speech includes the deletion and/or repetition of speech. The goal of speaker adaptation is to change the voice quality to match the target speaker's on the premise that adjusting the necessary modules will reduce the amount of data to be fine-tuned. In this paper, we clarify the role of each module in the Transformer-TTS process by not updating it. Specifically, we froze character embedding, encoder, layer predicting stop token, and loss function for estimating sentence ending. The experimental results showed the following: (1) fine-tuning the character embedding did not result in an improvement in the deletion and/or repetition of speech, (2) speech deletion increases if the encoder is not fine-tuned, (3) speech deletion was suppressed when the layer predicting stop token is not fine-tuned, and (4) there are frequent speech repetitions at sentence end when the loss function estimating sentence ending is omitted.**

Index Terms: end-to-end speech synthesis, speaker adaptation, fine-tuning, transformer, module analysis

I. Introduction

Text-to-speech (TTS) is a technology that generates human-like speech from inputted texts. End-to-end TTS [1], [2], [3], [4], [5] is now being investigated actively. It is made up of the encoder-decoder architecture with an attention mechanism that generates an acoustic feature sequence from a character/phoneme sequence. Unlike conventional statistical parametric speech synthesis (SPSS) systems based on hidden Markov models (HMMs) [6] and deep neural networks (DNNs) [7], end-to-end TTS systems can be trained with raw text data directly without requiring linguistic features like phonemes, syllables, accents, intonation, and mora. As an end-to-end TTS, Tacotron [1] that addresses the mapping between character and Mel-spectrogram was proposed. Many types of end-to-end neural architectures have been proposed, e.g., Tacotron 2 [2], Deep voice 3 [3], Transformer-TTS [4], [8], and FastSpeech [5].

To build an end-to-end TTS system, a large amount of paired speech and texts is needed for training, e.g., a few hours to 20 hours of recordings for a single speaker [9]. This makes constructing end-to-end TTS systems with various speaker's voice quality expensive. Knowledge transfer approaches such as transfer learning [10], fine-tuning [11] and multi-task learning [12] are promising options for solving the problem of data quantity. Fine-tuning, which needs only a dozen minutes of single-speaker recordings, has been extensively studied as a basis for adaptation techniques. Arik et al. proposed a speaker adaptation that is based on fine-tuning a pretrained multi-speaker model using a few audio-text pairs of an unseen speaker [13]. Chen et al. proposed cross-lingual transfer learning in end-to-end TTS for low-resource languages [14]. Tits et al. proposed an adaptation method that is based on fine-tuning target neutral speech and emotional speech sequentially [15]. In these methods, the entire model is updated.

However, if the model is sufficiently trained, it should be sufficient to update only those parts of the model that are relevant to the change in the task. For example, if the task that needs to be fine-tuned is developed using the same language, say English, then the character/word embedding does not need to be updated. Like the analysis of neurons [16] or the analysis of attention [17], knowing what the modules of a neural network represent allows us to know the appropriate modules to be updated. We have previously investigated semi-supervised speaker adaptation using unpaired speech data [18]. The results show that the fine-tuned model tends to relatively increase speech deletion when compared to the pretrained model. Similarly, in the study of [15], it can be seen that the intelligibility of the speech generated by the fine-tuned model decreases. To solve this problem, we compared the modules to be updated when fine-tuning and checked their effectiveness.

Our hypotheses are given below:

1) If the fine-tuning task is in the same language, the character embedding does not need to be updated.
2) If sufficient pretraining has been done, only the decoder, which maps the acoustic features from the encoded features, should be updated. This means the encoder does not need to be updated.
3) Because the loss that estimates the end of a sentence tends to cause over-training, updating the layer predicting stop token or using the loss is not necessary for fine-tuning.

We adopted the Transformer architecture [19] for the TTS because of its high performance and fast training properties [4], [8]. The results of our experiment showed that (1) if the task before and after fine-tuning is in English, updating the character embedding does not contribute to performance

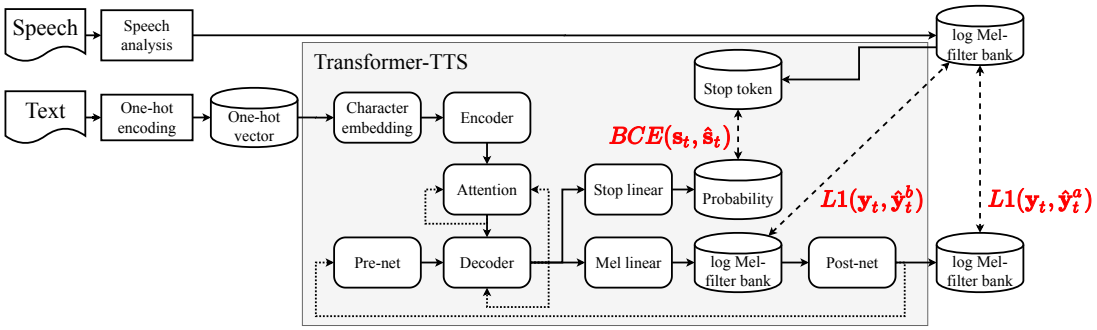

Fig. 1. The training process for Transformer-TTS. Plain, dotted, and dashed lines indicate forward flow of current step, feedback flow from past step, and loss calculation, respectively.

improvement; (2) freezing encoders tends to cause speech deletion; (3) freezing the layer that predicts stop token can reduce speech deletion; and (4) not including binary cross entropy (BCE) in the loss function tends to cause speech repetition.

II. Transformer-TTS

Figure 1 shows the training process that is common to both pretraining and fine-tuning.

A. Training process

Transformer-TTS (shown in Figure 1) are directly trained to map C-length character id sequence $\mathbf{X} = \{\mathbf{x}_c\}_{c=1}^{C}$ to T-length speech feature sequence $\mathbf{Y} = \{\mathbf{y}_t\}_{t=1}^{T}$. One-hot vectors are used as the character id. Also, log Mel-filter bank features are typically used as the speech features. The mapping from $\mathbf{X}$ to $\mathbf{Y}$ can be represented in the following manner:

$$\mathbf{e}_c = \text{Character-Embedding}(\mathbf{x}_c) \tag{1}$$
$$\mathbf{h}_c = \text{Encoder}(\mathbf{e}_c) \tag{2}$$
$$a_{t,c} = \text{Attention}(\mathbf{h}_c, \mathbf{q}_{t-1}, \mathbf{a}_{t-1}) \tag{3}$$
$$r_t = \sum_{c=1}^{C} a_{t,c}\mathbf{h}_c \tag{4}$$
$$\mathbf{v}_{t-1} = \text{Pre-net}(\mathbf{y}_{t-1}) \tag{5}$$
$$\mathbf{q}_t = \text{Decoder}(\mathbf{q}_{t-1}, \mathbf{r}_t, \mathbf{v}_{t-1}) \tag{6}$$
$$\hat{s}_t = \text{Linear}(\mathbf{q}_t) \tag{7}$$
$$\hat{\mathbf{y}}_t^b = \text{Linear}(\mathbf{q}_t) \tag{8}$$
$$\hat{\mathbf{y}}_t^a = \hat{\mathbf{y}}_t^b + \text{Post-net}(\hat{\mathbf{y}}_t^b) \tag{9}$$

The input $\mathbf{X}$ is embedded into a continuous feature sequence $\mathbf{E} = \{\mathbf{e}_c\}_{c=1}^{C}$. Then, $\mathbf{E}$ is encoded into a hidden feature sequence $\mathbf{H} = \{\mathbf{h}_c\}_{c=1}^{C}$ by the encoder network like a self-attention mechanism [19]. Next, a hidden feature sequence $\mathbf{H}$ is used as the input in the decoder network with a source-target attention mechanism [20]. Next, the temporary output $\hat{\mathbf{Y}}^b = \{\hat{\mathbf{y}}_t^b\}_{t=1}^{T}$ is generated by using the decoder network's output. The end-of-sequence (EOS) probability $\hat{s}_t$ at each frame t is predicted at the same time. Finally, the output $\hat{\mathbf{Y}}^a = \{\hat{\mathbf{y}}_t^a\}_{t=1}^{T}$ is generated from $\hat{\mathbf{Y}}^b$ and residual factor through the Post-net.

Given the speech feature sequence $\mathbf{Y}$ and the corresponding character sequence $\mathbf{X}$ from pretraining dataset $\mathcal{D}^P$, the whole of the TTS model is optimized by minimizing the following TTS loss:

$$\begin{aligned}\mathcal{L}(\mathbf{X}, \mathbf{Y}) =& \frac{1}{T}\sum_{t=1}^{T} \text{L1}(\mathbf{y}_t, \hat{\mathbf{y}}_t^b) + \frac{1}{T}\sum_{t=1}^{T} \text{L1}(\mathbf{y}_t, \hat{\mathbf{y}}_t^a) \\ &+ \frac{1}{T}\sum_{t=1}^{T} (s_t \ln \hat{s}_t + (1 - s_t)\ln(1 - \hat{s}_t)),\end{aligned} \tag{10}$$

where $\text{L1}(\cdot)$ represents an L1 norm, last term represents a BCE, and s_t represents a label that indicates if the input at time t is the EOS ($s_t = 1$) or not ($s_t = 0$). Note that the $\hat{\mathbf{y}}_t$ is conditioned on the ground-truth of the previous feature sequence $\{\mathbf{y}_{t'}\}_{t'=1}^{t-1}$, i.e., teacher-forcing is used in training.

B. Modules and their contributions

1) Character embedding: The character embedding (shown in eq. 1) takes on the role of mapping from a one-hot vector to a D-dimensional continuous value vector [2], [4]. If the tasks before and after fine-tuning are in the same language, the occurrence frequency and role of the alphabet are kept. Therefore, it is assumed that freezing the character embedding does not affect performance improvement after fine-tuning.

2) Encoder and decoder: The encoder (shown in eq. 2) takes on the role of a mapping from continuous value vectors to hidden feature vectors, while the decoder (shown in eq. 6) takes on the role of a mapping from hidden feature vectors and outputs of the previous step to outputs of the current step. Hidden features can be regarded as compressed features that adequately represent the acoustic features. If the encoder is sufficiently pretrained and the hidden feature does not depend on acoustic feature prediction, it might be sufficient that the decoder is only adapted to the target speaker. In other words, freezing the encoder allows us to check whether the hidden features are independent of the target speaker.

3) Output layer and loss function: The acoustic feature output layer (shown in eq. 8) and the post-net (shown in eq. 9) take on the role of predicting log Mel-filterbank features. This prediction error is measured quantitatively using the sum of L1 loss (shown in the first and second terms of eq. 10). Figure 2 shows the transitions of the sum of L1 loss in the pretraining

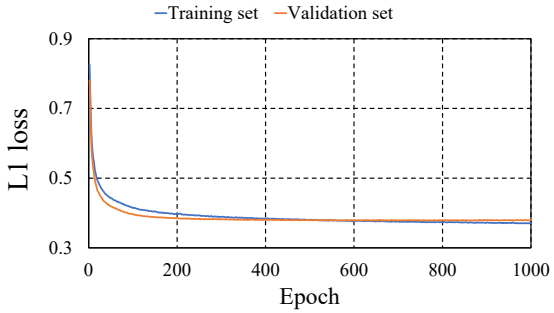

Fig. 2. L1 loss transition on pretraining

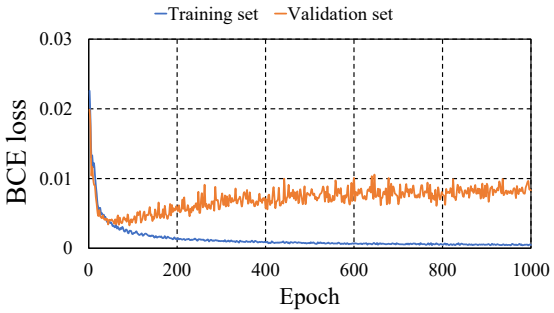

Fig. 3. BCE loss transition on pretraining

described in section III. As the purpose of speaker adaptation is to synthesize the speech of a new speaker, the acoustic feature output layer and the post-net need to be updated.

The probability output layer (shown in eq. 7) takes on the role of predicting the EOS. This prediction error is measured quantitatively using BCE loss (shown in the last term of eq. 10). Figure 3 shows the transitions of BCE loss in the pretraining described in section III. It can be seen that the BCE loss of the training set steadily decreases, while the BCE loss of the validation set increases from an early step. Thus, over-training may negatively affect the training results when fine-tuning.

III. Experimental setup

We froze three modules to check the effect of fine-tuning, on character embedding (char. embed), encoder, and probability output layer (prob. output). Also, BCE loss was excluded from the loss function (shown in eq. 10) to check if the presence or absence of this loss has an effect on performance.

We used four types of end-to-end TTS models that consist of an all fine-tuned model and fine-tuned models with frozen modules as shown in Table I. The first model was the pretrained TTS model, which is used as the baseline of the speaker adaptation. The second model is a speaker adaptation model based on fine-tuning [15], [18]. The third model is a multi-speaker end-to-end TTS model based on zero-shot feature embedding [21], [22], which is also trained with x-vector [23]. The feature embedding is another type of speaker adaptation method. The model controls the speaker's identities by using an auxiliary input vector, like x-vector. The zero-shot means the target speaker is not included in the training set. The fourth model is a multi-speaker end-to-end TTS model based on N-shot feature embedding. The N-shot means N utterances of the target speaker are included in the training set.

A. Dataset

For the pretraining of the TTS model, we used the LJ speech dataset [9], which consists of an approximately 24-hour speech of a female English speaker. The speech was sampled at 22.05 kHz and quantized by 16 bits. We used $12,100$ utterances as a training set, and 500 utterances as a development set.

For the adaptation of the pretrained TTS model based on fine-tuning, we used the subset of LibriTTS [24], which consists of female speakers (speaker 237) from "test-clean". The original LibriTTS consisted of approximately 585 hours of speech by $2,456$ English speakers. The speech was sampled at 24 kHz, quantized by 16 bits, and downsampled to 22.05 kHz. We divided the data into a training set, a development set and a test set in the ratio 90%, 5% and 5%, respectively.

For the training of the feature embedding-based adapted model, LibriTTS [24] was used. The training set consisted of two sets of clean speech data ("train-clean-100" and "train-clean-360"). The development set consisted of clean speech data ("dev-clean"). For the zero-shot, the target speakers were not included in both the training and development sets. For the N-shot, the target speaker was added to the training set.

B. Feature and text representation

For the single-speaker TTS model (pretraining, fine-tuning), the waveform that was sampled at more than 22.05 kHz was downsampled at 22.05 kHz. The 80-dimensional log Mel-filter bank features were extracted, and the text was tokenized as characters and mapped into a 76-character set: 52 alphabetic letters (A-Z, a-z), 5 punctuation marks ('‚.!?), 17 special marks ("()[]{}-/:; æœêé¯–), and 2 special tags ⟨unk⟩, ⟨space⟩ as unknown, space tokens, respectively.

For the multi-speaker TTS model (feature-embedding), the waveform was downsampled at 22.05 kHz. The 80-dimensional log Mel-filter bank features were extracted, and the text was tokenized as characters and mapped into the same character set as single-speaker TTS models. To control the speaker identity, x-vector [23] was extracted as a speaker embedding feature.

C. Model configuration

For the TTS model, we used the Transformer architecture [19]. The TTS encoder and decoder both had 6 layers with 1536 units. The pretraining model was trained for 1000 epochs using 2.01×10^{-7} as the initial learning rate. The fine-tuning models were trained for 100 epochs using 2.01×10^{-8} as the initial learning rate. The feature-embedding models were also trained for 100 epochs using 5.10×10^{-8} as the initial learning rate. The characters were embedded into a 384-dimensional vector.

During the waveform synthesis, ParallelWaveGAN [25] trained the mapping from 80-band log Mel-filter bank features to waveform was used as a vocoder. The ParallelWaveGAN was trained with the LJ speech dataset [9].

TABLE I
TRAINING CONFIGURATION.

Training method	Training data	Duration	Target speaker
Pretraining	the LJ speech dataset	23.0 hours	unseen
Fine-tuning	Target speaker ("test-clean")	20.0 minutes	seen
Feature-embedding zero-shot	LibriTTS ("train-clean-100", "train-clean-360")	245.1 hours	unseen
Feature-embedding N-shot	LibriTTS including target speaker	247.0 hours	seen

TABLE II
OBJECTIVE EVALUATION RESULTS OF THE CEPSTRUM RMSE [DB] AND THE CHARACTER ERROR RATE (CER) OF SYNTHESIZED SPEECH [%]. SUB. AND DEL., INS. INDICATE SUBSTITUTION ERROR [%], DELETION ERROR [%] AND INSERTION ERROR [%], RESPECTIVELY. TAIL INS. INDICATES THE INSERTION ERROR ON LAST 5 WORDS [%].

ID	Training method	Update part	Loss	RMSE	Sub.	Del.	Ins.	CER	(Tail Ins.)
1	Ground-truth	—	—		0.4	1.0	0.4	1.8	(0.4)
2	Pretraining	All	L1 + BCE	30.1	2.9	7.4	1.1	11.4	(0.6)
3	Fine-tuning	All	L1 + BCE	20.4	1.7	13.8	0.6	16.1	(0.4)
4		w/o char. embed	L1 + BCE	20.2	2.4	14.6	0.7	17.7	(0.3)
5		w/o char. embed, encoder	L1 + BCE	21.3	3.4	23.8	2.5	29.7	(0.4)
6		w/o prob. output	L1 + BCE	**19.4**	2.1	**8.8**	2.2	**13.1**	(1.9)
7		All	L1	20.5	2.2	4.5	23.8	30.5	(23.5)
8		w/o char. embed	L1	20.5	3.1	4.6	20.0	27.7	(19.4)
9		w/o char. embed, encoder	L1	21.2	4.9	12.8	8.9	26.6	(8.1)
10		w/o prob. output	L1	20.4	3.5	6.1	31.7	41.3	(31.0)
11	Feature-embedding zero-shot	All	L1 + BCE	19.5	2.4	5.6	0.6	8.6	(0.5)
12	Feature-embedding N-shot	All	L1 + BCE	19.1	3.0	14.1	2.0	19.0	(1.9)

IV. EXPERIMENTS

A. Objective evaluation

Unseen speech data for training was used as a test set, which included 26 sentences. All the TTS models generated speech by using the ground-truth text. The x-vector was extracted from target utterance to indicate the upper bound of feature-embedded models.

To evaluate the proposed method, we compared the performance of the acoustic feature prediction with two objective measures, namely, root mean squared error (RMSE) of the cepstrum and character error rate (CER) of the synthesized speech. The generated acoustic features were warped to fit the time of the ground-truth acoustic feature. Therefore, speeches repeated at the end of sentences were removed from the RMSE calculation. The cepstrum error is the average of all frames of cepstral distortion (CD) [18], where the cepstral coefficients are 79 dimensions. The CER was computed based on the edit distance between the ground-truth text and text recognized from a generated speech using end-to-end automatic speech recognition (ASR). It was evaluated using an ASR model [18] trained with Librispeech [26]. The ASR model is the Transformer architecture [19] and provided at ESPnet [27]. In end-to-end TTS, the generated speech sometimes includes the deletion and/or repetition of words in the input text due to alignment errors [28]. We used the CER to find and classify deleted speech and repeated speech as deletion error and insertion error, respectively [15], [29]. Comparing with word error rate, CER is suitable to evaluate the correctness of phoneme by ignoring a homonym, for example, "ad" and "add", "brothers" and "brother's". The evaluation by the ASR model has an advantage in reproduction and consistency than human assessment. The insertion error on the last five words was computed after observing the repetition of speech at the tail of the utterance.

B. Evaluation results

Table II shows the cepstrum RMSE and the CER of synthesized speech. In terms of the CER, by observing the results, we found out that the TTS model missed generating speech, which results in the deletion error in most cases. Also, we found that the TTS model repeatedly generated speech, which results in insertion error in most cases.

A comparison of the update of character embedding (Models 3 and 4 in Table II) shows that all updated model have similar RMSE to the model with frozen character embedding. It seems that the deletion of speech is more frequent than in the pretrained model, regardless of updating character embedding. Thus, the character embedding does not need to be updated during fine-tuning.

A comparison of the updated encoder (Models 3 and 5 in Table II) with the model with frozen encoders shows an increase in RMSE in the former. It can also be seen that this model has a significant increase in deletion errors when compared to the pretrained model. We found that the encoder output is not referenced in the deleted part of the speech after checking the source-target attention in generating the acoustic features. This shows that the hidden features are dependent on the generation of acoustic features, and the encoder needs to be updated during fine-tuning.

A comparison of the updated probability output layer (Models 3 and 6 in Table II) with the frozen probability output layer shows the latter has a lower RMSE and a lower deletion error than the model with all updates. It can be seen that updating the probability output layer has a negative effect on the generation of acoustic features. In addition to this, there was a slight increase in insertion errors at the end of the utterance. This indicates that the prediction accuracy was reduced due to the freezing of the probability output layer. As a result, repetition of the utterance occurred. Although contrary to the End-to-End philosophy, speech repetition can

be handled by simple post-processing such as trimming. In contrast, speech deletion is a serious and primary problem because it cannot be handled by post-processing. It can be concluded that the probability output layer does not need to be updated during fine-tuning.

A comparison of the use of BCE (Models 3-6 and 7-10 in table II) to the model with BCE loss shows that the former recorded a significant increase in insertion errors, especially at the end of the utterance. This indicates that a reduced accuracy in estimating the end of the utterance. Although BCE loss is prone to over-training, it is useful in suppressing speech repetition. The BCE needs to used during fine-tuning.

Comparing the different speaker adaptation methods (Models 11 and 12 in Table II), the N-shot model shows a higher increase in CER than the zero-shot model. It can be seen that the target speaker's training data are conducive to the deletion of utterances. However, the best-performing fine-tuned model (Model 6 in Table II) suppressed the deletion of utterances while keeping the RMSE at the same level.

V. Conclusion

In this paper, we investigated the effects of modules of Transformer-TTS for fine-tuning. We compared all updated model with models that freeze character embedding, encoder, a probability output layer, and binary cross entropy. Objective evaluation results showed that (1) if the task before and after fine-tuning is in English, updating the character embedding does not contribute to performance improvement; (2) freezing of encoders tends to cause speech deletion; (3) freezing the layer that predicts the utterance end can reduce speech deletion; and (4) not including BCE in the loss function tends to cause speech repetition. In conclusion, to realize the TTS model that generates speech of the target speaker with low deletion and repetition, we need to fine-tune the model with freezing character embedding and the layer predicting stop token.

We also found that deletion error after adaptation increased because the encoder output was not referenced by the decoder. In subsequent works, we will investigate the attention mechanism required to avoid increasing speech deletion. We evaluated the fine-tuned models with limited test sentences. As future work, we would like to clarify more the tendency of results and causes by increasing the test sentences.

References

[1] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[2] 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 *Proc. of ICASSP*, 2018, pp. 4779–4783.

[3] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," in *Proc. of ICLR*, 2018.

[4] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proc. of AAAI*, vol. 33, 2019, pp. 6706–6713.

[5] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," in *Proc. of NeurIPS*, 2019, pp. 3165–3174.

[6] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, "Speech synthesis based on hidden Markov models," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1234–1252, 2013.

[7] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. of ICASSP*, 2013, pp. 7962–7966.

[8] S. Karita, X. Wang, S. Watanabe, T. Yoshimura, W. Zhang, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, and R. Yamamoto, "A comparative study on Transformer vs RNN in speech applications," in *Proc. of ASRU*, 2019, pp. 449–456.

[9] K. Ito, "The LJ speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[10] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[12] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[13] S. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, "Neural voice cloning with a few samples," in *Proc. of NIPS*, 2018, pp. 10 019–10 029.

[14] Y.-J. Chen, T. Tu, C.-c. Yeh, and H.-y. Lee, "End-to-end text-to-speech for low-resource languages by cross-lingual transfer learning," in *Proc. of Interspeech*, 2019, pp. 2075–2079.

[15] N. Tits, K. El Haddad, and T. Dutoit, "Exploring transfer learning for low resource emotional tts," in *Proc. of IntelliSys*, 2019, pp. 52–60.

[16] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proc. of ICASSP*, 2013, pp. 8595–8598.

[17] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does bert look at? an analysis of bert ' s attention," in *Proc. of BlackboxNLP*, 2019, pp. 276–286.

[18] K. Inoue, S. Hara, M. Abe, T. Hayashi, R. Yamamoto, and S. Watanabe, "Semi-supervised speaker adaptation for end-to-end speech synthesis with pretrained models," in *Proc. of ICASSP*, 2020, pp. 7634–7638.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of NIPS*, 2017, pp. 5998–6008.

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[21] A. Tjandra, S. Sakti, and S. Nakamura, "Machine speech chain with one-shot speaker adaptation," in *Proc. of Interspeech*, 2018, pp. 887–891.

[22] 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 *Proc. of NIPS*, 2018, pp. 4480–4490.

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

[24] 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. of Interspeech*, 2019, pp. 1526–1530.

[25] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel Wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *Proc. of ICASSP*, 2020, pp. 6199–6203.

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

[27] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: end-to-end speech processing toolkit," in *Proc. of Interspeech*, 2018, pp. 2207–2211.

[28] 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 *Proc. of ICASSP*, 2020, pp. 7654–7658.

[29] F. Biadsy, R. J. Weiss, P. J. Moreno, D. Kanvesky, and Y. Jia, "Parrotron: An end-to-end speech-to-speech conversion model and its applications to hearing-impaired speech and speech separation," *arXiv preprint arXiv:1904.04169*, 2019.