INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

# ADAPTERMIX: Exploring the Efficacy of *Mixture of Adapters* for Low-Resource TTS Adaptation

*Ambuj Mehrish*[1], *Abhinav Ramesh Kashyap*[2], *Li Yingting*[3], *Navonil Majumder*[1], *Soujanya Poria*[1]

[1]Singapore University of Technology and Design, Singapore
[2]ASUS Intelligent Cloud Services (AICS) Singapore,
[3]Beijing University of Posts and Telecommunications, China

ambuj_mehrish@sutd.edu.sg, abhinav_kashyap@asus.com, cindyyting@bupt.edu.cn,
navonil_majumder@sutd.edu.sg, sporia@sutd.edu.sg

## Abstract

There are significant challenges for speaker adaptation in text-to-speech for languages that are not widely spoken or for speakers with accents or dialects that are not well-represented in the training data. To address this issue, we propose the use of the "mixture of adapters" method. This approach involves adding multiple adapters within a backbone-model layer to learn the unique characteristics of different speakers. Our approach outperforms the baseline, with a noticeable improvement of 5% observed in speaker preference tests when using only one minute of data for each new speaker. Moreover, following the adapter paradigm, we fine-tune only the adapter parameters (11% of the total model parameters). This is a significant achievement in parameter-efficient speaker adaptation, and one of the first models of its kind. Overall, our proposed approach offers a promising solution to the speech synthesis techniques, particularly for adapting to speakers from diverse backgrounds.
**Index Terms**: Text to Speech, Adapters, Mixture of Adapters

## 1. Introduction

One of the key aspects of text-to-speech (TTS) technology is the ability to capture the unique acoustic mannerisms of a given speaker, which can include characteristics such as accent, intonation, rhythm, and other vocal traits that are associated with a speaker's identity [1]. This can be especially important in applications where the speaker's identity is a key factor, such as in voice assistants or interactive voice response systems. Thus, capturing these vocal idiosyncrasies in the generated speech is challenging and often requires many hours of reference speech samples from the speaker. Performing TTS tasks using large reference samples could be infeasible due to various reasons, such as limited memory budget, privacy concerns, or logistical issues. To address this challenge, we suggest a low-resource TTS approach that utilizes reference samples no longer than 1 minute. This strategy will help overcome the aforementioned limitations and enable efficient TTS performance with minimal resources.

TTS is being integrated into a wide range of applications, from virtual assistants to audiobooks, making it more accessible and useful than ever before. As text-to-speech technology continues to evolve, it has the potential to revolutionize communication and accessibility for people with disabilities or language barriers. One of the biggest challenges in TTS research is improving the naturalness and expressiveness of speech [2, 3, 4]. This involves using algorithms to convert written text into spoken words, and there are many techniques available to make the resulting speech sound more natural and expressive. For example, prosody modelling [5, 6] can help to convey intonation, stress, and rhythm, while neural network-based models such as [7, 8] can produce speech with more natural sounding timbre and articulation.

Few-shot *speaker adaptation* is a challenging task in TTS, which aims to personalize synthesized speech to match the characteristics of a specific speaker. This can involve modifying the acoustic model of the TTS to adjust for factors such as pitch, tone, and pronunciation [9, 10, 11]. Adaptive TTS models can be used to achieve few-shot speaker adaptation through methods such as pre-trained speaker embedding models with a small amount of reference speech [10] or fine-tuning multi-speaker TTS. However, fine-tuning requires a large amount of training data to avoid over-fitting and catastrophic forgetting [12].

In this paper, we propose a low-resource speaker adaptation approach based on the mixture of adapters (MoA) [13, 14, 15]. MoA has been widely used in NLP tasks such as text [13], question answering [14], and machine translation [16], by fine-tuning pre-trained models with minimal adapter parameters. In TTS, MoA can train a parameter-efficient module on a small amount of data, enabling fine-tuning of a pre-trained TTS model for a new speaker without forgetting previous knowledge. MoA's multiple adapter modules capture fine-grained information about the speaker's speech such as prosody, speaking rate or accent and adapt the acoustic model more effectively than traditional fine-tuning, saving time and computational resources while maintaining or improving performance.

MoA has several advantages over traditional fine-tuning approaches, such as faster training time, better generalization to new tasks, and improved robustness to domain shifts. However, MoA also has some limitations, such as the need for many adapters to achieve good performance, the risk of overfitting on small datasets, and the difficulty of interpreting the adapter weights. While MoA is a promising technique for improving the performance of TTS systems with minimal resources, its use is likely to increase in the future.

To this end, we take a two-phase training approach: i) train a transformer-based encoder-decoder TTS model on a large TTS dataset, LibriTTS [17] with 100h of clean speech samples from 251 speakers; ii) adapt this backbone model to work with short duration speech samples. The first phase is meant to learn the TTS task, traditionally by optimizing all the parameters in the network. Whereas, the second phase adds only a small fraction of the parameters, in the form of Adapters [18], only to optimize them to work with shorter reference speech samples. As compared to the recently proposed by Hsieh [19], which trains an adapter for each speaker, our approach is much more scalable to a large number of speakers as the adapters are shared among the speakers. To learn the key lower-dimensional vocal qualities, from the speech samples in the context of the target text, we add multiple parallel adapters to the transformer decoder and aggregate their outputs, akin to *mixture of experts* [20]. We call our setup ADAPTERMIX.


10.21437/Interspeech.2023-1568

## 2. Related Work

There are various approaches to few-shot speaker adaptation for TTS systems [21, 22, 23]. The main idea behind these methods is to first train a TTS model [24, 7] on a large dataset of speech from multiple speakers [17, 25]. This enables the model to learn general speech and language patterns. Subsequently, the model is fine-tuned on a smaller dataset of speech from a new speaker [10]. One significant advantage of using a pretrained model for few-shot adaptation is that it can quickly adapt to a new speaker with minimal data by leveraging the knowledge learned from the large dataset. This is particularly useful when collecting significant amounts of data for each new speaker is impractical or infeasible [9]. Numerous techniques have been proposed for few-shot speaker adaptation using pretrained TTS models [10, 11], including meta-learning [26], transfer learning [27, 28], and domain adaptation [29]. These approaches aim to enhance adaptation performance and reduce the data requirements for successful adaptation. A notable example of such an approach is Adaspeech [30], which achieves efficient adaptation and edge-affordable memory storage by incorporating conditional layer normalization into the decoder.

Adapters-based approaches have shown remarkable performance in low-resource TTS by efficiently using only a fraction of parameters compared to fully fine-tuned models. In [19], the residual adapter was recently introduced in TTS, but it trains a specific adapter for each target speaker, limiting its scalability as the number of speakers increases. In our approach, we also use residual adapters in the decoder but with $N$ adapters in each layer, which are shared by all target speakers, and a routing algorithm selects tokens during training. This design significantly reduces the number of parameters and allows scalability for larger datasets compared to [19].

## 3. Proposed Method

We first pretrain a multi-speaker Transformer TTS model [31] on LibraTTS corpus [17] and then adapt the model to new speakers by only training the *mixture of adapters* module inserted into the decoder layer of the pretrain model. In this section, we will briefly discuss the architecture of the Transformer TTS backbone model, followed by ADAPTERMIX.

### 3.1. Transformer TTS backbone architecture

Transformer [32] architecture has been successfully applied to TTS systems [31, 24], consisting of a text encoder, duration and pitch predictors, decoder, and vocoder. The text encoder uses self-attention and feedforward neural networks to generate a new representation of the input text. The duration predictor and pitch predictor estimate phoneme lengths and fundamental frequency, respectively. The decoder takes the encoded text, predicted phoneme durations, and predicted F0 values to generate spectrograms, which are converted into a time-domain waveform by a vocoder. Additional Postnet and linear projection layers are used to enhance speech quality.

### 3.2. AdapterMix

We adapt the pretrained transformer TTS backbone to a new target speaker using mixture of adapter modules inserted into every layer of the decoder after the feed-forward sub-layer, as shown in Figure 1a. *Mixture of adapters* module consists of $N$ lightweight neural modules and a routing algorithm for independently selecting top-$k$ tokens for individual adapters. By selecting tokens that are highly relevant to speaker identity, we can effectively adapt the model to new speakers while minimizing the need for extensive fine-tuning. Whereas the individual adapters are meant to capture complementary characteristics of the tokens, in context of the speaker, that are relevant to TTS.

As illustrated in Figure 1b, ADAPTERMIX consists of $N$ residual adapters and a routing algorithm. The architecture of the residual adapter is also shown in Figure 1b. Each residual adapter first applies layer normalization [33] to a $d_{model}$ dimensional input vector $h_l \in \mathbf{R}^{d_{model}}$ (subscript $l$ represents $l^{th}$ decoder layer). Following normalization, the output is projected down to a bottleneck dimension $r$, followed by a nonlinearity (ReLU [34]), up projection to the model dimension $d_{model}$ and finally a residual connection. The residual adapter can be described in the following way:

$$\hat{h}_l = h_l + ReLU(LayerNorm(h_l)W_{down})W_{up} \quad (1)$$

where $W_{down} \in \mathbf{R}^{d_{model} \times r}$ and $W_{up} \in \mathbf{R}^{r \times d_{model}}$ are the down and up projection weights.

ADAPTERMIX use an expert choice routing strategy similar to [20], where top-$k$ tokens are selected independently for each adapter. Depending on the number of adapters $N$, capacity $c$ (hyperparameter), and length of the sequence $n$, $k$ is determined dynamically as $k = \frac{n \times c}{N}$. As a result, tokens can be allocated to a variable number of adapters, enabling flexible allocation. The token-to-adapter affinity score $S \in \mathbf{R}^{n \times N}$ is computed using standard softmax operation between input $h_l$ and $W_g \in \mathbf{R}^{d_{model} \times N}$, where $W_g$ denotes adapters embeddings. Following equations can be used to explain routing

$$S = Softmax(h_l, W_g) \quad (2)$$

Top $k$ largest entries of each row in $S^T$ are selected as an input to the $N$ adapters:

$$G, I = TopK(S^T, k). \quad (3)$$

The input to each adapter $h_l^{in}$ is obtained by permutation matrix $P = Onehot(I)$ is the one hot version of $I$. This helps us to pick up the appropriate token representation for a given adapter as:

$$h_l^{in} = Ph_l. \quad (4)$$

The output of $i^{th}$ the adapter $h_l^{out}[i]$ is computed using Equation 1, where $h_l^{in}[i]$ is input to each adapter. Finally, all the adapter outputs are combined with $G$ and $P$ from Equation 3 using Einstein summation (einsum) operations:

$$\hat{h}_l = h_l + \sum_i PGh_l^{out}[i]. \quad (5)$$

## 4. Experiments

### 4.1. Baselines

As a common baseline, we evaluated ADAPTERMIX against two other methods: full fine-tuning and residual adapter [19]. The residual adapter-based baseline is hereafter referred to as `Adapter`. During full fine-tuning, all parameters of the backbone model are updated, while for `Adapter`, only adapter parameters are updated, keeping the backbone frozen.

### 4.2. Training setup

The multi-speaker backbone model utilized a Transformer architecture, specifically a Transformer-TTS model [1], which consisted

[1]The source codes of AdapterMix along with the checkpoints are publicly available at `https://github.com/declare-lab/adapter-mix`.

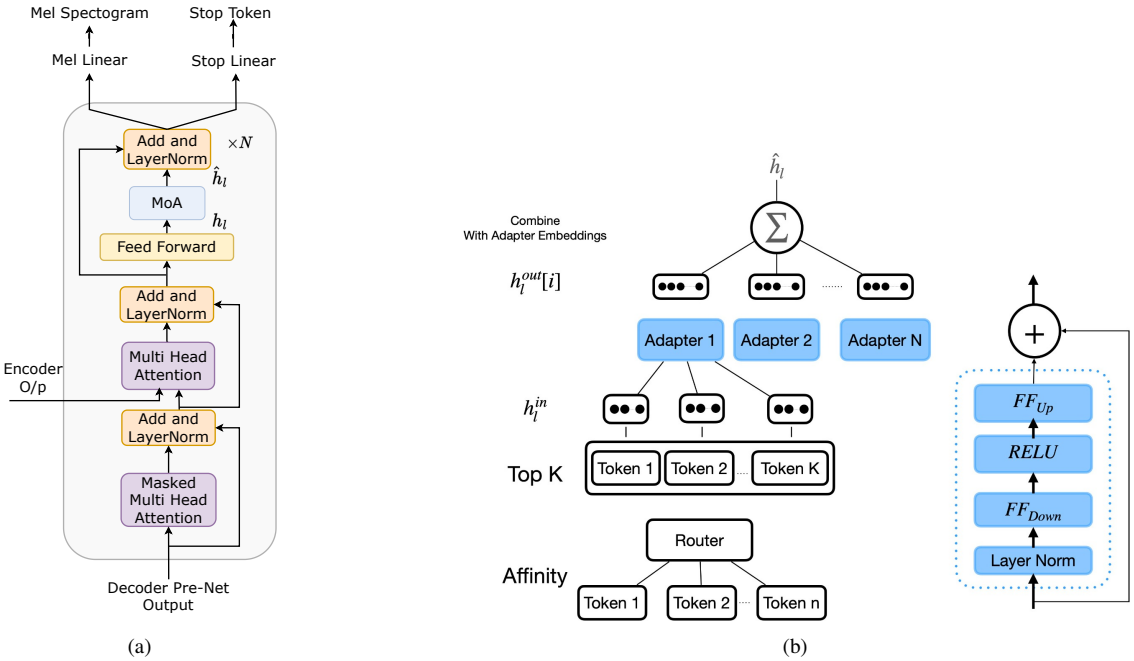


Figure 1: *(a) Transformer TTS [31] decoder architecture with one MoA module. (b) The MoA module comprises $N$ residual adapters (left) Every adapter chooses $k$ closest tokens and processes it. The same token can be processed by multiple adapters. The outputs of the adapters are combined. Additionally, the architecture of the standard residual adapter is illustrated on the right in the same diagram.*

of 4 encoders and 6 decoder layers, with a hidden state dimension of $d_{model} = 256$. This model also incorporated other speech synthesis modules, including post-net, pre-net, and variance adapter, as described in Section 3.

To pretrain the backbone model, we used the train-clean-100 split of the multi-speaker English LibriTTS corpus [17], which contains 100 hours of 24 kHz English speech from 251 speakers. We downsampled the speech to 22.05 kHz and trained the model for 900k steps using the Adam optimizer. Overall, the Transformer-TTS model had 3.6M parameters

We randomly selected ten speakers (five male and five female) from the CSTR VCTK corpus [25] for our speaker adaptation experiments. We then divided the selected speakers' utterances into three groups based on training duration: 1 min, 10 min, and 15 min. We downsampled the speech to 22.05 kHz, used the Adam optimizer for training, and trained all models on a single NVIDIA Tesla A6000 GPU. To ensure a fair comparison between the baselines (Finetune, `Adapter`) and MoA (ADAPTERMIX), we trained each one of them for 10k steps. We applied a warm-up period of 4000 steps and performed learning rate annealing at 6000, 7000, and 8000 steps, with an annealing rate of 0.3. All models were trained with a batch size of 64, except in the case of the 1-min training duration, where we used a batch size of 16.

During the adaptation process to a new speaker, we insert adapter modules (ADAPTERMIX and `Adapter`) into the decoder layer of the backbone model, as shown in Figure 1a. Unless stated otherwise, the bottleneck dimension $r$ is set to 128. Target speakers may have different variance information, such as duration, pitch, energy, etc., than the speakers on which the backbone model is pretrained. To capture the variance of target speakers, we add a single residual adapter to the output of the variance adapter with $r = 64$. We optimize only the adapter modules (ADAPTERMIX and `Adapter`) for the target speaker, while keeping the entire backbone frozen, including batch normalization. We also fine-tune the target speaker's speaker embedding to ensure that the variance adapters are properly conditioned by the learned target speaker embedding, in addition to the adapter modules (ADAPTERMIX and `Adapter`). By doing so, we can better capture the target speaker's unique characteristics and improve the overall quality of the synthesized speech.

### 4.3. Objective Evaluation

In this study, we evaluated the mel-spectrogram reconstruction capability of the synthesized speech using the mel-cepstrum distortion (MCD) [35]. To assess the intelligibility of the synthesized speech, we used the word error rate (WER). Additionally, we calculated an objective speaker similarity measure by computing the mean cosine similarity between the embeddings of the ground truth and synthetic speech samples, using a neural speaker embedding system called deep speaker [36]. The embeddings used for computing cosine similarity had a dimension of 512. Each speaker similarity test had a total of approximately 200 utterances. Although all the models produced similar MCD scores, Table 1 highlights performance gaps between ADAPTERMIX and the other two baselines. Notably, ADAPTERMIX outperforms `Adapter` in terms of WER, but performs comparably in terms of MCD for all durations. Thus, although ADAPTERMIX scores slightly lower in mel-spectrogram reconstruction capability, it generates more intelligible speech than `Adapter`.

The cosine similarity scores of ADAPTERMIX are comparable to those of the finetuned model and better than those of `Adapter` for all time intervals. For a model trained on 10 minutes of data per speaker, ADAPTERMIX achieves a cosine similarity score of 0.7324, compared to 0.7362 and 0.7091 for the finetuned model and `Adapter`, respectively. This result indicates that ADAPTERMIX has better speaker adaptation capabilities than the other baselines. In the objective evaluation,

Table 1: *Subjective (MOS) and objective comparison among full Finetune,* `Adapter`*, and* ADAPTERMIX.

| Method (#param %) | 1min | | | | 10min | | | | 15min | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MOS↑ | MCD↓ | WER↓ | Cosine Sim↑ | MOS↑ | MCD↓ | WER↓ | Cosine Sim↑ | MOS↑ | MCD↓ | WER↓ | Cosine Sim↑ |
| Ground Truth | 4.01± 0.33 | - | 0.19036 | - | 4.10 ± 0.41 | - | 0.1853 | - | 4.17 ± 0.41 | - | 0.1861 | - |
| Finetune | 3.45± 0.48 | 5.7450 | **0.2489** | **0.747 ± 0.0065** | 3.18 ± 0.29 | 5.6453 | **0.2228** | **0.7362 ± 0.0069** | 3.54 ± 0.49 | 5.7058 | **0.2056** | **0.7374 ± 0.0080** |
| `Adapter` (1.57%) | 2.82 ± 0.52 | **5.2482** | 0.3911 | 0.6733± 0.0072 | 3.13 ± 0.42 | **5.2763** | 0.2445 | 0.7091 ± 0.0053 | 3.19 ± 0.42 | **4.9430** | 0.2731 | 0.6957 ± 0.0062 |
| ADAPTERMIX (11.62%) | 3.33 ± 0.31 | 5.5943 | 0.2987 | 0.7037 ± 0.0119 | 3.66 ± 0.31 | 5.4216 | 0.2270 | 0.7324 ± 0.0060 | 3.53 ± 0.37 | 5.3641 | 0.2260 | 0.7170 ± 0.0081 |

Fine-tuning has an edge over ADAPTERMIX, which can be credited to the number of trainable parameters in the fine-tuned model. After conducting a subjective evaluation (refer to Section 4.4), it was found that ADAPTERMIX performs as well as the fine-tuned model and, in some experiments, even outperforms it. These results demonstrate the effectiveness of ADAPTERMIX, which utilizes only 11% of the trainable parameters present in the full fine-tuned model.

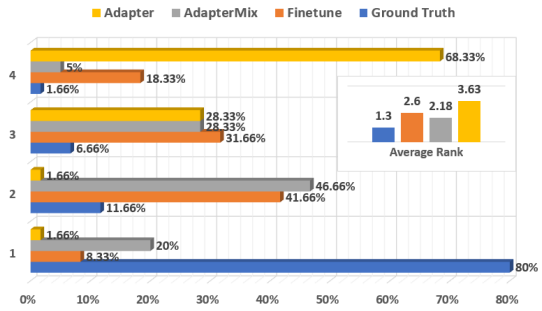


Figure 2: *Ranking of the % of speech samples synthesized by each model based on the MOS score.*

### 4.4. Subjective Evaluation

We conducted subjective evaluations of synthesized samples using crowdsourced Mean Opinion Score (MOS), which provides a quantitative measure of the fidelity of the synthesized audio relative to the original audio[2]. In this evaluation, listeners were asked which sample sounded more natural. To determine speaker individuality, we conducted an XAB test [37] to assess speaker similarity. The target speaker's reference speech is presented as X, and the speech synthesized by ADAPTERMIX and `Adapter` are presented to listeners in a random order as A and B. Listeners were asked to choose which of A or B sounded more similar to X. A total of 20 listeners with backgrounds in NLP and speech participated in the experiment and were presented with 60 (+20 reference samples) synthesized speech samples. The evaluation samples consisted of speech samples synthesized using models trained on 1 min, 10 min, and 15 min durations. Table 1 reports the MOS scores for different models, while the preference test between ADAPTERMIX and `Adapter` is presented in Figure 3.

Table 1 illustrates that ADAPTERMIX outperforms `Adapter`, particularly in low-resource scenarios (1 min and 10 min). ADAPTERMIX performs comparably or better than full fine-tuning in all three training scenarios while being significantly more parameter-efficient (approximately 10 times lower than full fine-tuning). Also, the XAB test results (Figure 3) show that samples generated by ADAPTERMIX are preferred over ADAPTERMIX by 43.63% and 31% for training durations of 10 min and 1 min, respectively. Furthermore, ADAPTERMIX achieved competitive or better performance in both naturalness and speaker similarity compared to the full fine-tuning under low-resource conditions. The results presented in Figure 2 show the subjective ranking of speech samples synthesized by each model. Interestingly, the ADAPTERMIX model achieved an average rank of 2.18, outperforming full fine-tuning which only received a rank of 2.6. This suggests that speech samples generated using ADAPTERMIX are preferred over those generated using full fine-tuning. Notably, the ADAPTERMIX only optimized 11.62% of the backbone model parameters to achieve these results, showing its efficiency against full fine-tuning.

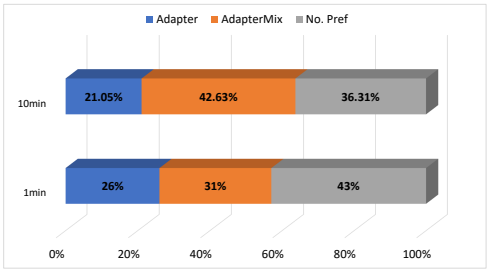


Figure 3: *XAB speaker similarity test results for* 1 *min and* 10 *min of train data.*

## 5. Conclusions and Future Works

The present paper proposes a novel approach, called ADAPTERMIX, for efficient TTS speaker adaptation under low-resource settings, which leverages a mixture of adapters. ADAPTERMIX achieves competitive or better performance than both fine-tuning and single adapter baselines in terms of naturalness and speaker similarity. Remarkably, ADAPTERMIX only optimizes 11.62% of the total backbone model parameters and outperforms full model fine-tuning for low-resource speaker adaptation. Future extensions of this work could significantly improve the quality of synthesized speech and reduce the mismatch between TTS synthesized speech and target speaker speech. The proposed adapters and mixture of adapters provide a parameter-efficient method for adapting TTS. In future work, we could explore more challenging scenarios of extremely low-resource settings, where adaptation data is limited to less than one minute. It would also be interesting to investigate other parameter-efficient adapters, such as tiny adapters, instead of residual adapters, and explore different stochastic routing algorithms.

## 6. Acknowledgement

This project is supported by the AcRF MoE Tier-2 grant (Project no. T2MOE2008, and Grantor reference no. MOE-T2EP20220-0017) titled: "CSK-NLP: Leveraging Commonsense Knowledge for NLP", and the SRG grant id: T1SRIS19149 titled "An Affective Multimodal Dialogue System".

[2]Audio samples are available at `https://adaptermix.github.io`

# 7. References


[1] A. Mehrish, N. Majumder, R. Bhardwaj, R. Mihalcea, and S. Poria, "A review of deep learning techniques for speech processing," 2023.

[2] M. S. Al-Radhi, T. G. Csapó, and G. Németh, "Improving the expressiveness of tts synthesis with non-autoregressive neural vocoding," in *Speech Research conference*, p. 94.

[3] M. Kim, S. J. Cheon, B. J. Choi, J. J. Kim, and N. S. Kim, "Expressive Text-to-Speech Using Style Tag," in *Proc. Interspeech 2021*, 2021, pp. 4663–4667.

[4] L. B. de MM Marques, L. H. Ueda, F. O. Simões, M. Uliani Neto, F. O. Runstein, E. J. Nagle, B. D. Bó, and P. D. Costa, "Diffusion-based approach to style modeling in expressive tts," in *Intelligent Systems: 11th Brazilian Conference, BRACIS 2022, Campinas, Brazil, November 28–December 1, 2022, Proceedings, Part I*. Springer, 2022, pp. 253–267.

[5] T. Raitio, J. Li, and S. Seshadri, "Hierarchical prosody modeling and control in non-autoregressive parallel neural tts," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7587–7591.

[6] C.-M. Chien and H.-y. Lee, "Hierarchical prosody modeling for non-autoregressive speech synthesis," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 446–453.

[7] 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 *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[8] 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*.

[9] E. Cooper, C.-I. Lai, Y. Yasuda, F. Fang, X. Wang, N. Chen, and J. Yamagishi, "Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6184–6188.

[10] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, "Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone," in *International Conference on Machine Learning*. PMLR, 2022, pp. 2709–2720.

[11] Y. Wu, X. Tan, B. Li, L. He, S. Zhao, R. Song, T. Qin, and T.-Y. Liu, "Adaspeech 4: Adaptive text to speech in zero-shot scenarios," *arXiv preprint arXiv:2204.00436*, 2022.

[12] H. Hemati and D. Borth, "Continual speaker adaptation for text-to-speech synthesis," *arXiv preprint arXiv:2103.14512*, 2021.

[13] Y. Wang, S. Mukherjee, X. Liu, J. Gao, A. H. Awadallah, and J. Gao, "Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models," *arXiv preprint arXiv:2205.12410*, 2022.

[14] J. Jiang and N. Zheng, "Mixphm: Redundancy-aware parameter-efficient tuning for low-resource visual question answering," *arXiv preprint arXiv:2303.01239*, 2023.

[15] A. Chronopoulou, M. E. Peters, A. Fraser, and J. Dodge, "Adaptersoup: Weight averaging to improve generalization of pretrained language models," *arXiv preprint arXiv:2302.07027*, 2023.

[16] C. Baziotis, M. Artetxe, J. Cross, and S. Bhosale, "Multilingual machine translation with hyper-adapters," *arXiv preprint arXiv:2205.10835*, 2022.

[17] 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," *arXiv preprint arXiv:1904.02882*, 2019.

[18] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[19] N. Morioka, H. Zen, N. Chen, Y. Zhang, and Y. Ding, "Residual adapters for few-shot text-to-speech speaker adaptation," *arXiv preprint arXiv:2210.15868*, 2022.

[20] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. Dai, Z. Chen, Q. Le, and J. Laudon, "Mixture-of-experts with expert choice routing," *arXiv preprint arXiv:2202.09368*, 2022.

[21] 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.

[22] Z. Kons, S. Shechtman, A. Sorin, C. Rabinovitz, and R. Hoory, "High quality, lightweight and adaptable tts using lpcnet," *arXiv preprint arXiv:1905.00590*, 2019.

[23] S. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, "Neural voice cloning with a few samples," *Advances in neural information processing systems*, vol. 31, 2018.

[24] 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," *arXiv preprint arXiv:2006.04558*, 2020.

[25] J. Yamagishi, C. Veaux, and K. MacDonald, "Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92)," 2019.

[26] S.-F. Huang, C.-J. Lin, D.-R. Liu, Y.-C. Chen, and H.-y. Lee, "Meta-tts: Meta-learning for few-shot speaker adaptive text-to-speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1558–1571, 2022.

[27] 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.

[28] P. Neekhara, J. Li, and B. Ginsburg, "Adapting tts models for new speakers using transfer learning," *arXiv preprint arXiv:2110.05798*, 2021.

[29] X. Zhang, J. Wang, N. Cheng, and J. Xiao, "Tdass: Target domain adaptation speech synthesis framework for multi-speaker low-resource tts," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[30] M. Chen, X. Tan, B. Li, Y. Liu, T. Qin, S. Zhao, and T.-Y. Liu, "Adaspeech: Adaptive text to speech for custom voice," *arXiv preprint arXiv:2103.00993*, 2021.

[31] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[33] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[35] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222–2235, 2007.

[36] 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.

[37] H. Mizuno and M. Abe, "Voice conversion algorithm based on piecewise linear conversion rules of formant frequency and spectrum tilt," *Speech communication*, vol. 16, no. 2, pp. 153–164, 1995.