INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Adapter-Based Extension of Multi-Speaker Text-To-Speech Model for New Speakers

Cheng-Ping Hsieh *Subhankar Ghosh* *Boris Ginsburg*

NVIDIA, USA

{chsieh, subhankarg, bginsburg}@nvidia.com

Abstract

Fine-tuning is a popular method for adapting text-to-speech (TTS) models to new speakers. However, this approach has some challenges. Usually, fine-tuning requires several hours of high quality speech per speaker. Fine-tuning might negatively affect the quality of speech synthesis for previously learned speakers. In this paper, we propose an alternative approach for TTS adaptation based on using parameter-efficient adapter modules. In the proposed approach, a few adapter modules are added between the layers of the pretrained network. The pre-trained model is frozen, and only the adapters are fine-tuned to the speech of a new speaker. Our approach will produce a new model with a high level of parameter sharing with the original model. Our experiments on LibriTTS, HiFi-TTS and VCTK datasets validate our adapter-based method through objective and subjective metrics. The code is open-sourced[1] and the audio samples are available on our demo page[2].

Index Terms: Text-to-speech, speaker adaptation, adapter, few-Shot Learning

1. Introduction

Neural text-to-speech (TTS) models have significantly improved in recent years [1, 2, 3, 4, 5] . These models can synthesize a high-quality natural human voice in single-speaker [6] and multi-speaker [7, 8, 9] settings after being trained on several hours of high-quality recordings. However, to adapt new speaker voices, these TTS models are typically fine-tuned using several hours of studio quality data, which makes scaling TTS models to a large number of speakers very expensive.

Fine-tuning TTS models to new speakers may be challenging for a number of reasons. First, the original TTS model should be pre-trained with a large multi-speaker corpus so that it generalizes well to new voices and different recording conditions. The second challenge is to reduce the amount of speech required to add a new speaker to the existing TTS model. Third, fine-tuning the whole TTS model is very parameter inefficient, since one will need a new set of weights for every newly adapted speaker. Currently, there are two approaches to making the adaptation of TTS more efficient. The first approach is to modify only parameters directly related to speaker identity [10, 11, 12, 13]. The alternative approach is based on adding a light voice conversion post-processing module to the baseline TTS model [14].

In this paper, we propose a new parameter-efficient method for adapting new speakers to the pre-trained multi-speaker TTS model shown in Fig. 1. First, we pre-train a base multi-speaker TTS model on a large and diverse TTS dataset. To adapt the model to new speakers, we add a few lightweight modules in between the layers of the base model. We used vanilla adapters [15], unified adapters [16, 17, 18], or BitFit [19]. Then pre-trained model is frozen, and only adapters are fine-tuned on new speaker data. During inference, we can share most of the model weights and insert small modules to synthesize a new speaker voice. The contributions of this paper are:

- We propose a new adapter-based framework for efficiently adapting new speakers to the TTS model without forgetting previously learned speakers.
- We validate our design through a comprehensive ablation study across different types of adapter modules, amounts of training data, and recording conditions.
- We demonstrate that adapter-based TTS tuning performs similarly to full fine-tuning while requiring significantly less compute and data.

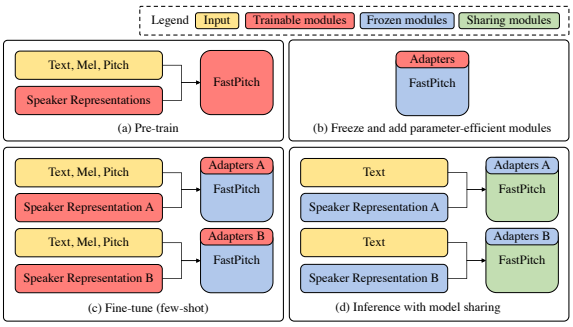

Figure 1: *The proposed pipeline for adaptation of multi-speaker TTS model for new speakers. (a) Pre-train a multi-speaker FastPitch model. (b) Freeze weights of pre-trained FastPitch model and add adapter modules. (c) Only the adapters and speaker representations are fine-tuned for new speaker. (d) Inference by sharing the same model and plugging the lightweight, speaker-specific module.*

2. Method

In this section, we first describe the architecture of our pre-trained multi-speaker FastPitch – a non-autoregressive TTS model conditioned on speaker representations, as shown in Fig. 2. Next, we introduce parameter-efficient adapter modules including vanilla adapter, unified adapters, and BitFit (see Fig. 3). Finally, we explain how we insert or unfreeze the lightweight learnable modules to fine-tune our pre-trained model for speaker adaptation.

[1]https://github.com/NVIDIA/NeMo

[2]https://hsiehjackson.github.io/adapter-tts-demo/

10.21437/Interspeech.2023-2313

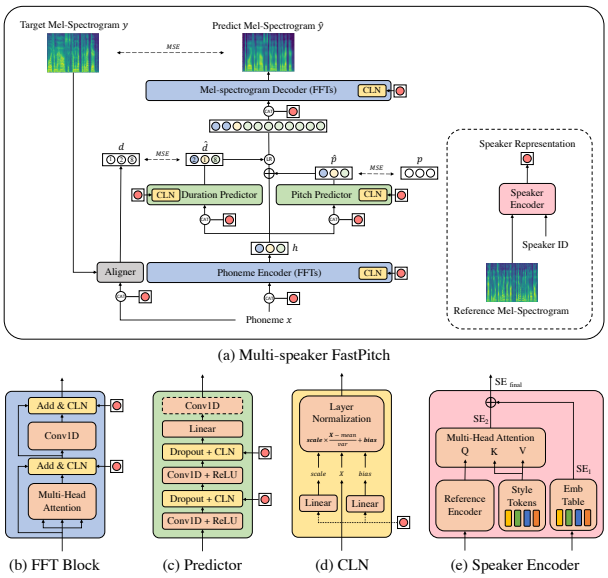

Figure 2: *Architecture of proposed multi-speaker FastPitch. It is composed of phoneme encoder, mel-spectrogram decoder, duration and pitch predictor, aligner, and speaker encoder. We control speaker identity by using conditional layer normalization (CLN) and concatenating inputs with speaker representation.*

2.1. Base multi-speaker TTS model

FastPitch We use FastPitch [5] as base TTS model. FastPitch is composed of four components including two feed-forward transformer (FFT) stacks as phoneme encoder and mel-spectrogram decoder, and two convolutional modules as pitch and duration predictor. The encoder operates on the input phoneme tokens x and produces a hidden state h which is used to predict the average pitch of each token $\hat{p}$ and duration $\hat{d}$ by the pitch and duration predictor respectively. The decoder takes the length-regulated hidden representations from the sum of encoder outputs h and pitch $\hat{p}$ to produce the mel-spectrogram sequence $\hat{y}$. To train the pitch predictor, we use the ground-truth pitch p, derived using PYIN [20] and averaged over the input tokens. For the duration predictor, we use a learnable aligner from [21]. The training loss is composed of MSE between predicted and ground-truth modalities plus the alignment loss L_{align}:

$$L = ||\hat{y} - y||_2^2 + \alpha||\hat{p} - p||_2^2 + \beta||\hat{d} - d||_2^2 + \gamma L_{align}.$$

Speaker Representation Typically, speakers are represented by adding a speaker embedding table to the TTS model. The limitation of this approach is that it cannot generalize to out-of-sample speakers. Therefore, we combine speaker embeddings (SE_1) from a look-up table with speaker embeddings (SE_2) obtained from global style tokens (GST) [22] for a particular speaker. GST embeddings capture prosodic styles for a particular speaker. From a reference spectrogram, the convolutional recurrent neural network-based encoder learns the style tokens, and the contribution weights of the tokens are learned by a multi-head attention layer. By adding (SE_1) and (SE_2), the final speaker embedding SE_{final} is obtained. The benefit of this method is that we can learn a wide range of acoustic expressiveness without any explicit style or prosody labels for unseen speakers.

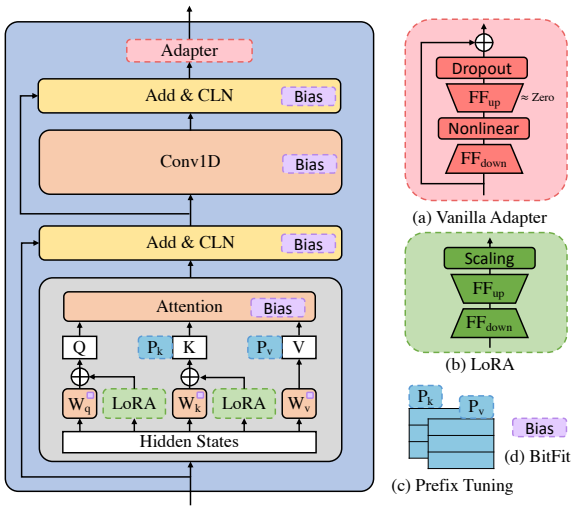

Figure 3: *Illustration of parameter-efficient tuning modules in transformer architecture. LoRA and Prefix Tuning are only used in FFTs while Adapter and BitFit can be applied to any components in FastPitch.*

Multi-Speaker FastPitch Each component of FastPitch: encoder, decoder, pitch predictor, duration predictor, and aligner, was conditioned using a speaker representation vector. The inputs to each component are concatenated with the speaker representation vector. Following [13], we leverage conditional layer normalization (CLN) to better condition our model with the speaker representation. The CLN network is made up of two linear layers that project the speaker representation vector into scale and bias vectors. We substitute all normalization layers in the encoder, pitch and duration predictor, and decoder with the CLN layer.

2.2. Adapter Modules

Vanilla Adapter Vanilla Adapters [15] are small modules inserted between layers of a frozen pre-trained network. During training, the gradient only updates the adapters while other parameters are fixed. The adapter layer generally uses a down-projection feed-forward network $(\boldsymbol{FF_{down}})$ to project the input to a lower dimensional bottleneck, followed by a non-linear activation function and an up-projection feed-forward network $(\boldsymbol{FF_{up}})$. To stabilize training, a near-identity initialization is required, so the adapter has a skip-connection internally. With the skip connection, the original network can stay unaffected when training starts. In our design, we optionally add dropout and layer normalization as well as the zero initialization of the final layer to serve this module as an identity operation. Moreover, instead of placing adapters inside transformer layers as in [15], we propose to insert them after the outputs of each transformer layer. Specifically, we generalize adapters to be inserted after any module.

Unified Adapters A unified framework has been developed in [18] to integrate multiple parameter-efficient modules, including vanilla adapters and their variants like **LoRA** [16] and **Prefix Tuning** [17]. LoRA injects trainable low-rank matrices into the self-attention network in each transformer layer to update the query and key. The architecture is similar to that of an

Table 1: *Comparison of parameter-efficient methods and ablation study of speaker-related modules on objective metrics. We weighted mean looked-up speaker embeddings as new speaker embedding and unfreeze CLN as trainable modules.*

Method	SECS ↑	CFSD ↓	MSE_P ↓	MSE_D ↓	Params
Different parameter-efficient methods					
BitFit	0.452	56.4	71.0	**19.1**	2.2M
PrefixTuning (FFTs)	0.067	83.2	75.6	22.7	0.6M
LoRA (FFTs)	0.141	62.7	77.7	22.3	2.8M
Vanilla adapter (FFTs)	**0.568**	**30.0**	**65.9**	21.3	2.4M
Different speaker-related modules					
Vanilla adapter (FFTs/Predictors/Aligner)	0.575	28.3	**62.7**	16.9	3.5M
+ speaker embedding	**0.586**	**27.2**	63.6	**16.2**	3.5M
+ speaker embedding + CLN	0.540	46.9	66.8	17.2	7.8M
speaker embedding + CLN [13]	0.513	53.5	68.0	21.7	4.3M
Full fine-tuning	0.604	31.0	73.8	19.7	53.4M

adapter but without an activation function and with a fixed scaling scalar. Prefix Tuning prepends trainable vectors to the keys and values of the self-attention network in each transformer layer. In other words, we concatenate the original key and value matrices with additional prefix vectors and perform multi-head attention as usual. Compared to Adapter, LoRA and Prefix Tuning are only applied to self-attention network in transformers. We also use another simple tuning approach **BitFit** [19]. This approach only updates bias vectors while fixing other parameters in the pre-trained model.

2.3. Parameter-efficient fine-tuning

To adapt to the new speaker adaptation data, we only update the parameter-efficient modules and speaker representation modules, keeping the rest of the weights in the pre-trained FastPitch frozen. First, we insert parameter-efficient modules in our pre-trained model. We add a vanilla adapter to the phoneme encoder, mel-spectrogram decoder, pitch and duration predictor, as well as the aligner. We experimented by adding LoRA and Prefix Tuning to the self-attention network in the encoder and decoder. BitFit can be used in any layer that contains bias terms. Second, as described in the Speaker Representation section of 2.1, we obtain speaker embedding (SE_2) from GST using a reference spectrogram. We add this speaker embedding with speaker embedding (SE_1) obtained by weighted mean of all pre-trained speaker embeddings from lookup table to form the final speaker embedding SE_{final} as shown in Figure 2. The weights were learned from gradients during fine-tuning. Third, we also unfreeze the linear layers of scale and bias in each CLN because this module's effectiveness in controlling speaker identity has been verified in [13]. Thus, with this small number of trainable parameters, we can optimize our TTS model for new speaker adaptation in a parameter-efficient way.

3. Experiments and Results

3.1. Dataset

We used LibriTTS [7] for pre-training. From the original train-clean-360 set, we choose the top 100 speakers with the highest amount of data, totaling 42.5 hours. The top five longest-duration speakers from the original test-clean set are used to create our test set of 10 unseen speakers (5 men and 5 women) for the evaluation of speaker adaptation. To validate the generalization abilities to multiple acoustic conditions, we experiment on VCTK [8] and HiFi-TTS [9] datasets. To simulate few-shot scenario, the test sets are composed of 15 minutes data for each speaker. For each test speaker, we randomly choose 20 unseen utterances to evaluate the adaptation voice quality.

After the data collection, we normalize and tokenize the raw text sequence into phoneme tokens. Also, we pre-process the speech waveform into mel-spectogram under the sampling rate 22kHz and pre-compute the pitch [20] and alignment prior [21] before training.

3.2. Experiment Setup

We pre-train multi-speaker FastPitch for 500 epochs on 8 V100 GPUs with batch size 16 and learning rate 1×10^{-3}. In the fine-tuning stage, we freeze all model parameters and only update the proposed speaker representation and parameter-efficient modules. We train the model as well as our baselines on a single NVIDIA A5000 GPU for $\sim$1500 steps using Adam optimizer, batch size 8 and 2×10^{-4} learning rate. The adaptation process may take 10 to 15 minutes depending on the data size. We use HiFi-GAN [23], trained on mel-spectrograms from pre-trained FastPitch, as the vocoder to convert mel-spectrograms to waveforms. The vocoder was not fine-tuned on the new adapted speakers.

3.3. Evaluation Metrics

To measure the voice quality, we conduct both objective and subjective evaluations of the synthesized and ground-truth speech. For objective evaluation, we first calculate the average Speaker Embedding Cosine Similarity (SECS) between the reference and measured audios by a speaker verification model [24] to estimate speaker similarity. Further, we compute Conditional Fréchet Speech Distance (CFSD) [14] between the generated speech and actual recording to measure signal quality. Besides, we also evaluate mean square error for pitch (MSE_P) and duration (MSE_D) to access prosody similarity. The error is computed against ground-truth speech.

For subjective evaluation, we conduct human evaluations with a 5-scale MOS (mean opinion score) for naturalness and SMOS (similarity MOS) for speaker similarity on Amazon Mechanical Turk. Each audio sample is rated by 20 workers. We average the scores of all speakers as the final scores.

3.4. Results

We use four voices (two males and two females) to study how different types of adapters and speaker-related modules de-

Table 2: *Comparison of different amount of training data on both subjective and objective metrics. We fine-tune adapters in all FastPitch components and the weights to average looked-up speaker embeddings as the adapter results shown here. Note that we omit* $\times 10^{-3}$ *in reported MSE scores for simplicity.*

Method	**Trainset, min**	**MOS** ↑	**SMOS** ↑	**SECS** ↑	**CFSD** ↓	**MSE**$_P$ ↓	**MSE**$_D$ ↓
Vanilla adapter	1	3.81 ± 0.04	3.38 ± 0.05	0.421	31.4	129.5	25.2
	5	3.84 ± 0.04	**3.50 ± 0.05**	0.466	26.9	108.3	21.5
	15	3.83 ± 0.04	3.46 ± 0.05	0.492	24.2	**90.2**	19.1
	60	**3.86 ± 0.04**	3.47 ± 0.05	**0.520**	**23.2**	119.6	**18.3**
Full fine-tuning	1	3.56 ± 0.04	3.34 ± 0.05	0.461	35.4	135.6	30.4
	5	3.72 ± 0.04	3.44 ± 0.05	0.522	26.2	102.6	25.8
	15	**3.77 ± 0.04**	**3.46 ± 0.05**	0.537	25.4	**90.5**	24.6
	60	3.75 ± 0.04	3.43 ± 0.05	**0.542**	**22.1**	106.4	**22.9**

Table 3: *Comparison of datasets with different acoustic conditions on subjective metrics.*

	MOS ↑			**SMOS** ↑		
Method	**LibriTTS**	**VCTK**	**HiFi-TTS**	**LibriTTS**	**VCTK**	**HiFi-TTS**
Recording	4.11 ± 0.03	3.99 ± 0.03	4.02 ± 0.03	3.82 ± 0.04	3.69 ± 0.04	3.62 ± 0.04
Vanilla adapter	**4.02 ± 0.03**	3.82 ± 0.04	**3.84 ± 0.04**	**3.45 ± 0.04**	3.22 ± 0.05	**3.36 ± 0.04**
Full fine-tuning	3.92 ± 0.04	**3.83 ± 0.04**	3.79 ± 0.04	3.37 ± 0.04	**3.26 ± 0.04**	3.31 ± 0.04

scribed in section 2.3 affects the quality of TTS adaptation. The results are shown in Table 1.

When inserting parameter-efficient modules in FFTs in the encoder and decoder blocks, vanilla adapters significantly outperform other approaches on all metrics except duration error. These results may be attributed to the locations we inserted modules. For language, the self-attention layer plays a crucial role, so a small update on this module can obtain good performance, such as LoRA and Prefix Tuning. For speech, however, an update of the convolution layer is essential, which is why the vanilla adapter had better performance.

Next, we insert adapters into predictors and aligner. Adding weighted mean speaker embedding improves the performance while unfreezing CLN may degrade the speech metrics. Moreover, vanilla adapters get better scores than just using speaker embedding and CLN [13], and they obtain comparable quality to full fine-tuning when using only 7% parameters.

After validating the best design for FastPitch adaptation, we study how much training data is required for this setting, as shown in Table 2. The proposed method outperforms full-model fine-tuning under different data settings. We find that listeners can hardly recognize quality differences even if the model was fine-tuned with vanilla adapters under only 5 minutes of the speech data for new speakers, although objective metrics still demonstrate the improvements for larger sets.

Finally, we adapted the model on speakers from the VCTK and HiFi-TTS datasets to check how the proposed method performs when a new speaker's speech is recorded under different conditions compared to the LibriTTS dataset used for pre-training. In Table 3, vanilla adapters outperform full fine-tuning on naturalness (MOS) and speaker similarity (SMOS) for LibriTTS and HiFi-TTS datasets while obtain similar performance for VCTK dataset. These results show our framework can be generalized to diverse recording conditions.

4. Conclusion

In this work, we propose parameter-efficient method for the adaptation of multi-speaker TTS models to new speakers. The new speaker adaptation is based on adding small adapter modules to the base model. We keep the weights of the base model frozen, and only the parameters of adapters and speaker embeddings are fine-tuned on new speaker data. The experiments show proposed method achieves high speech naturalness, speaker and prosody similarity while requiring significantly less compute. It also performs well even in a low-data setting.

5. References

[1] Y. Wang, R. Skerry-Ryan, D. Stanton *et al.*, "Tacotron: Towards end-to-end speech synthesis," *Proc. Interspeech 2017*, pp. 4006–4010, 2017.

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

[3] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "FastSpeech: Fast, robust and controllable text to speech," *NeurIPS*, 2019.

[4] 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 *ICLR*, 2020.

[5] A. Łańcucki, "FastPitch: Parallel text-to-speech with pitch prediction," in *ICASSP*, 2021.

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

[7] 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," *Interspeech*, 2019.

[8] J. Yamagishi, C. Veaux, K. MacDonald *et al.*, "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit (ver. 0.92)," 2019.

[9] E. Bakhturina, V. Lavrukhin, B. Ginsburg, and Y. Zhang, "Hi-Fi multi-speaker English TTS dataset," in *Interspeech*, 2021.

[10] H. B. Moss, V. Aggarwal, N. Prateek *et al.*, "Boffin TTS: Few-shot speaker adaptation by bayesian optimization," in *ICASSP*, 2020.

[11] Z. Zhang, Q. Tian, H. Lu *et al.*, "AdaDurian: Few-shot adaptation for neural text-to-speech with durian," *arXiv:2005.05642*, 2020.

[12] S. Arik, J. Chen, K. Peng *et al.*, "Neural voice cloning with a few samples," *NeurIPS*, 2018.

[13] M. Chen, X. Tan, B. Li, Y. Liu, T. Qin, sheng zhao, and T.-Y. Liu, "Adaspeech: Adaptive text to speech for custom voice," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=Drynvt7gg4L

[14] A. Gabryś, G. Huybrechts, M. S. Ribeiro *et al.*, "Voice filter: Few-shot text-to-speech speaker adaptation using voice conversion as a post-processing module," in *ICASSP*, 2022.

[15] 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 *ICML*, 2019.

[16] E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models," in *ICLR*, 2021.

[17] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *ACL and IJCNLP*, 2021.

[18] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," in *ICLR*, 2021.

[19] E. B. Zaken, Y. Goldberg, and S. Ravfogel, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *ACL*, 2022.

[20] M. Mauch and S. Dixon, "pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *ICASSP*, 2014.

[21] R. Badlani, A. Łańcucki, K. J. Shih *et al.*, "One TTS alignment to rule them all," in *ICASSP*, 2022.

[22] Y. Wang, D. Stanton, Y. Zhang *et al.*, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *ICML*, 2018.

[23] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," *NeuRIPS*, 2020.

[24] N. R. Koluguri, T. Park, and B. Ginsburg, "TitaNet: Neural model for speaker representation with 1D depth-wise separable convolutions and global context," in *ICASSP*, 2022.