Interspeech 2022
18-22 September 2022, Incheon, Korea

TriniTTS: Pitch-controllable End-to-end TTS without External Aligner

Yoon-Cheol Ju[1], *Il-Hwan Kim*[1], *Hong-Sun Yang*[1], *Ji-Hoon Kim*[1], *Byeong-Yeol Kim*[1], *Soumi Maiti*[2], *Shinji Watanabe*[2]

[1]AIRS Company, Hyundai Motor Group, Seoul, Republic of Korea
[2]Carnegie Mellon University, Pittsburgh, PA, USA

{windtoker, i.h.kim, hongsun.yang, jihoon_k, byeol}@hyundai.com,
smaiti@andrew.cmu.edu, shinjiw@cmu.edu

Abstract

Three research directions that have recently advanced the text-to-speech (TTS) field are end-to-end architecture, prosody control modeling, and on-the-fly duration alignment of non-auto-regressive models. However, these three agendas have yet to be tackled at once in a single solution. Current studies are limited either by a lack of control over prosody modeling or by the inefficient training inherent in building a two-stage TTS pipeline. We propose TriniTTS, a pitch-controllable end-to-end TTS without an external aligner that generates natural speech by addressing the issues mentioned above at once. It eliminates the training inefficiency in the two-stage TTS pipeline by the end-to-end architecture. Moreover, it manages to learn the latent vector representing the data distribution of the speeches through performing tasks (alignment search, pitch estimation, waveform generation) simultaneously. Experimental results demonstrate that TriniTTS enables prosody modeling with user input parameters to generate deterministic speech, while synthesizing comparable speech to the state-of-the-art VITS. Furthermore, eliminating normalizing flow modules used in VITS increases the inference speed by 28.84% in CPU environment and by 29.16% in GPU environment.

Index Terms: TriniTTS, speech synthesis, text-to-speech, end-to-end architecture, pitch control

1. Introduction

Text-to-speech (TTS) synthesis has seen a significant amount of advances in the quality of synthesized speech since the application of deep learning in the field. Typically neural TTS systems follow a two-stage pipeline, acoustic model to generate mel-spectrogram and vocoder model to synthesize speech samples from mel-spectrogram. With auto-regressive models like Tacotron [1] and WaveNet [2], neural TTS system can produce very high-quality speech. Later, to solve the slow sample by sample synthesis of auto-regressive models, parallel generation architectures were proposed in both acoustic [3, 4, 5] and vocoder [6, 7, 8]. There are also significant advances towards diversifying the generated speech by imposing certain constraints on the models, such as duration length scale and pitch [5, 9, 10].

Two-stage TTS pipeline for speech generation has an inefficient training procedure due to the fact that two models have to be trained separately with ground-truth mel-spectrograms. As a result, the input data distribution of the training and the inference steps are mismatched in the second-stage model. The discrepancy in the input data distribution degrades the speech quality. To solve this discrepancy, one of the proposed solutions [1] is to utilize the mel-spectrogram generated by the first-stage model to train the second-stage model. However, such a cascade procedure increases training inefficiency in that the second-stage model depends on the feature prediction of the first-stage model.

Alternatively, the end-to-end TTS model [11, 12, 13] removes these training complexities by training a single model. Additionally, using a single model has the potential of learning more powerful latent features as it is not limited to learning the intermediate acoustic features (e.g., mel-spectrogram).

The current state-of-the-art model VITS (Variational Inference with adversarial learning for end-to-end Text-to-Speech) [11, 14] applies an end-to-end architecture based on conditional variational auto-encoder (VAE) [15] and dynamic programming for monotonic alignment search on-the-fly. By sampling the latent vector in VAE structure, it shows outstanding performance in generating the same contextual speeches with varied rhythms and pitches. However, the rhythm and pitch are not controllable due to the fact that the latent vector is sampled stochastically.

We propose TriniTTS, a neural TTS approach which performs 1) deterministic and controllable prosody modeling, 2) in an end-to-end manner, 3) without using an external aligner. To the best of our knowledge, TriniTTS is the first work attempting to address pitch controllability, two-stage pipeline training inefficiency, and the dependency on an external aligner at once. We aim for higher training efficiency and controllable prosody modeling with stable speech synthesis quality. TriniTTS achieves mean opinion score (MOS) close to VITS while showing a faster inference speed than VITS in real-time factor (RTF) analysis. We will publicly release the demo page with the synthesized audios. [1]

2. Model description

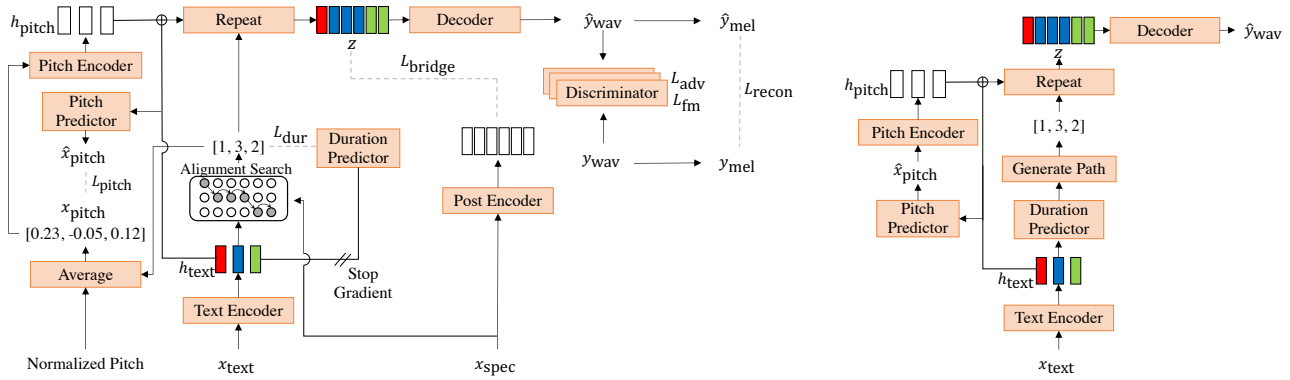
This section explains the overall architecture and implementation in the five sub-sections: text encoder, post encoder, decoder, pitch control, and alignment search. The diagrams of the overall architectures in the training and inference stage are shown in Figure 1.

TriniTTS and VITS both take text as input and generate waveform output in an end-to-end manner. However, there are some contrasting points between those two models. First, each uses a different alignment search algorithm [16, 17]. Second, pitch-related modules are attached in TriniTTS while pitch information is not explicitly processed in VITS. Last, VITS follows a conditional variational auto-encoder structure that involves a sampling process while TriniTTS applies straight forward deterministic process.

In Figure 1, it is shown that there are modules only dedicated to the training stage that are not in the inference stage.

[1]https://hkmc-airlab.github.io/trinitts/

10.21437/Interspeech.2022-925



(a) *Training diagram* (b) *Inference diagram*

Figure 1: *Training and inference diagrams of TriniTTS*

The post encoder, the discriminator in the decoder, and alignment search modules are used only for training, not in the inference stage.

In the multi-speaker model, speaker embedding is given additionally as input to pitch predictor, duration predictor, query and key encoders in the alignment search module, post encoder, and decoder to condition the speaker information with each speaker id.

2.1. Text encoder

The pre-processed sequence of text tokens is individually mapped to embeddings from the embedding lookup table. The sequence of embeddings is inserted into the transformer [18] based text encoder to learn the text hidden state h_{text}.

2.2. Post encoder

The post encoder corresponds to the posterior encoder in VITS [11]. Same as in VITS, we use non-causal WaveNet [2] residual blocks for the wide receptive field to capture a latent representation of a given spectrogram. When it comes to VITS, the posterior encoder extracts the parameters representing the posterior distribution. Then, a latent vector is sampled from normal Gaussian distribution parameterized by the one above. After all, the latent vector is fed into the decoder to generate raw waveforms. In TriniTTS, it is similar to VITS in that it takes spectrograms x_{spec} as input for the post encoder. However, unlike VITS, where the sampled latent vector from the posterior encoder works as a direct input to the decoder, the post encoder in TriniTTS only guides the intermediate representations from the prior encoder part by imposing bridge loss. Bridge loss is defined as $L1$ loss between the latent vector of the posterior data distribution and the intermediate representations generated by the prior encoder part given x_{text} as input:

$$\mathcal{L}_{\text{bridge}} = \|PostEnc(x_{\text{spec}}) - PriorEnc(x_{\text{text}})\|_1 \quad (1)$$

where $PostEnc(.)$ denotes for the post encoder and $PriorEnc(.)$ encompasses the entire modules used to generate intermediate representations z respectively.

2.3. Decoder

Same as VITS, we use HiFi-GAN [19] architecture as a decoder to convert intermediate representations z to waveforms $\hat{y}_{\text{wav}}$. Using the adversarial training mechanism [20], the generator G aims to generate waveforms that are similar to ground truth waveforms. The strength of HiFi-GAN [19] module lies in the discriminator D. Multi-scale discriminator and multi-period discriminator modules increase the capability to distinguish whether the given speech is real or fake with more diverse and wide receptive fields. Adversarial loss of the generator $\mathcal{L}_{\text{adv}}(G)$ and the discriminator $\mathcal{L}_{\text{adv}}(D)$ are defined as:

$$\mathcal{L}_{\text{adv}}(G) = \mathbb{E}_z \left[(1 - D(G(z)))^2\right] \quad (2)$$

$$\mathcal{L}_{\text{adv}}(D) = \mathbb{E}_{(y_{\text{wav}}, z)} \left[(1 - D(y_{\text{wav}}))^2 + (D(G(z))^2\right] \quad (3)$$

Not only adversarial loss $\mathcal{L}_{\text{adv}}$ but also feature matching loss $\mathcal{L}_{\text{fm}}$ [21] and reconstruction loss $\mathcal{L}_{\text{recon}}$ are used to learn to generate a waveform similar to the target waveform. Feature matching loss $\mathcal{L}_{\text{fm}}$ [21] is defined as:

$$\mathcal{L}_{\text{fm}}(G) = \mathbb{E}_{(y_{\text{wav}}, z)} \left[\sum_{l=1}^{T} \frac{1}{N_l} \|D^l(y_{\text{wav}}) - D^l(G(z))\|_1\right] \quad (4)$$

where T denotes the number of layers used in the discriminator D and D^l outputs the feature map of each layer in the discriminator D. We calculate $\mathcal{L}_{\text{fm}}$ as we compare the feature maps of y_{wav} and G(z) and normalize it by the number of features N_l.

Generated waveform $\hat{y}_{\text{wav}}$ and target waveform y_{wav} are both converted to mel-spectrogram $\hat{y}_{\text{mel}}$ and y_{mel} each for reconstruction loss $\mathcal{L}_{\text{recon}}$ as defined in:

$$\mathcal{L}_{\text{recon}} = \|\hat{y}_{\text{mel}} - y_{\text{mel}}\|_1 \quad (5)$$

2.4. Pitch control

We define pitch control task as adjusting the pitch of the generated speech under the unit of each token. The amount of the pitch value changed should be proportional to the deviation of the tuned pitch control parameter. We can dedicate the task to serializing pitch predictor and pitch encoder modules. Pitch predictor predicts the normalized pitch value $\hat{x}_{\text{pitch}}$ assigned to individual tokens. The pitch encoder module generates the pitch hidden state h_{pitch} over each token based on a sequence of float pitch values. Then the text hidden state h_{text} and the pitch hidden state h_{pitch} are summed together to form the intermediate representations. It manages to learn the joint distribution of the text and the pitch data in the intermediate representations. In the inference stage, we tune the pitch control parameter to adjust the

predicted pitch values of each token so that we manipulate the pitch hidden state h_{pitch} added to the text hidden state h_{text}.

We implemented pitch predictor and pitch encoder with the same configuration as those of Fastpitch[5] [5]. The ground truth of pitch value over spectrogram frames is needed for the pitch estimation task. We use the pitch extraction algorithm previously used in Fastpitch [5]. It uses pyin algorithm [22] to extract the pitch value of each waveform which selects the most probabilistic result by Viterbi decoding out of the candidates pre-calculated by yin [23] algorithm. We use the optimal alignment found in the alignment search to calculate the average of ground-truth pitch x_{pitch} over a duration of each token which we give as a target to the pitch predictor. We define pitch loss $\mathcal{L}_{\text{pitch}}$ as:

$$\mathcal{L}_{\text{pitch}} = \|\hat{x}_{\text{pitch}} - x_{\text{pitch}}\|_2^2 \tag{6}$$

2.5. Alignment search

The alignment search task is to find the alignment mapping between text tokens and spectrogram frames that maximizes the likelihood of text given spectrograms. Recent studies [11, 17, 24] show that under monotonic alignment assumption between text tokens and spectrogram frames, we can find the optimum alignment that maximizes the likelihood of text given spectrograms as a joint likelihood of each text token and spectrogram frames.

Unlike in VITS where the optimum alignment is found solely over dynamic programming [16], we apply the duration search algorithm[17, 24] which involves not only dynamic programming but also neural networks for query and key encodings for alignment mapping.

In the alignment search module, we have query encoder Φ_{query} with text hidden state h_{text} as input and key encoder Φ_{key} with spectrograms x_{spec} as input. Query and key encoders generate hidden vectors ϕ_{query} and ϕ_{key} respectively for alignment map searching. Soft alignment map A_{soft} [17] is built based on the learned pairwise affinity between query vector ϕ_{query} and key vector ϕ_{key}. With the constraint of monotonic alignment, all possible candidates of alignments are extracted out of the soft alignment map to find the most likely path. Forward-sum algorithm [17] is used to calculate the loss using CTC loss for the possible candidates. Duration predictor learns to estimate the duration of each token given text hidden state h_{text} as input and the extracted optimal alignment as a target.

3. Experiments

3.1. Training

3.1.1. Dataset

The model is trained both on single speaker and multi-speaker datasets. We use the LJSpeech dataset [25] for the single speaker model. We set aside 100 and 500 samples each for the validation and test sets. For the multi-speaker model, the VCTK dataset [26] is used for training. We downsampled the audios of the VCTK dataset to 22,050 Hz same as the LJSpeech dataset. We also trimmed the audio clips using librosa [2] trim function with the threshold of 20 and back margin of 0.5 seconds to remove reasonable silence intervals while preserving the speech part safely. To extract pitch value for the pitch control task, we use librosa [2] pyin function to generate the pitch contour of waveforms. Spectrograms and pitch contour information are calculated in advance with filter length to 1024 and hop size to 256 at the dataset preparation steps. Parameters for short-time Fourier transform are equal in both single and multi-speaker models. We set filter length to 1024, hop length to 256, and window length to 1024 to extract spectrograms. We use a phoneme-based phonemizer [3] for text processing.

3.1.2. Training configuration

For both single speaker and multi-speaker models, we trained the model on 4 Nvidia V100 GPUs. We set the batch size as 16 examples per GPU. We enabled automatic mixed-precision mode. We applied the same optimizer configuration from VITS. We used AdamW optimizer [27] with initial learning rate of 2 $\times$ 10^{-4}, β_1 = 0.8, β_2 = 0.99, weight decay λ = 0.01. We set learning rate decay as 0.999875 after every epoch.

3.2. Speech synthesis quality test

To evaluate synthesis quality with human ratings, we conducted MOS test using Amazon MTurk system. First, we set the score scale to 5 in terms of speech quality. Then, we generated 60 samples from the test dataset sentences using each model to compare the ground truth and the synthesized audios. At least 20 raters were asked to rate the naturalness of each sample during the tests. We designed the MOS test for both single and multi-speaker models. Additionally, we did the MOS test on the generated speeches with the pitch parameter set to pitch shift + 40Hz to check the naturalness of the speeches with ascended pitch level.

3.2.1. Single speaker

We compare our model with well-known and publicly available baseline models for the single speaker model. We use VITS as a baseline. Additionally, we compare two other two-stage models. For acoustic, we use flow-based Glow-TTS and transformer-based Fastpitch. For vocoder, HiFi-GAN is used. For Glow-TTS [4], Fastpitch [5] and HiFi-GAN [6], we use pre-trained weights from publicly released implementations. We trained TriniTTS up to 1000 epochs which are approximately 200K steps. We trained VITS under the same configuration as TriniTTS.

To demonstrate that the pitch controlling function transforms the pitch level of generated speech proportional to the input parameter, we generate audios with and without pitch control and calculate the mean and the standard deviation of pitch over 100 generated speeches.

In Table 1, the standard deviation indicates how widespread fundamental frequency (F0) ranges amongst the generated speeches are, which is related to the dynamicity of the speech. TriniTTS records the higher standard deviation value than Fastpitch + HiFi-GAN. Comparing mean values among the columns in Table 1, we verify that the amount of changes in the mean value of pitch is close to the parameter given as input for pitch shifting.

The evaluation results for the naturalness of the generated speeches are shown in Table 2. Without pitch shifting, the differences between the models are within the margin of confidence intervals. However, testing over generated speeches with

[2]https://github.com/librosa/librosa

[3]https://github.com/bootphon/phonemizer

[4]https://github.com/jaywalnut310/glow-tts

[5]https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/FastPitch

[6]https://github.com/jik876/hifi-gan

pitch control, we observe that TriniTTS outperforms Fastpitch + HiFi-GAN in terms of speech naturalness.

Table 1: *The statistics of pitch value with pitch shifting (Hz)*

Model	pitch+0Hz mean	pitch+0Hz stddev	pitch-40Hz mean	pitch+40Hz mean
Fastpitch +HiFiGAN	208.62	43.27	171.82 (-36.8)	250.14 (+41.52)
TriniTTS	194.66	46.47	159.59 (-35.07)	243.11 (+48.45)

Table 2: *MOS with 95% confidence intervals on the LJSpeech dataset*

Model	MOS	MOS (pitch+40Hz)
Ground truth	4.26 $\pm$0.03	4.29 $\pm$0.03
Fastpitch + HiFi-GAN	4.01 $\pm$0.05	4.04 $\pm$0.04
GlowTTS + HiFi-GAN	4.08 $\pm$0.05	-
VITS	4.03 $\pm$0.05	-
TriniTTS	4.05 $\pm$0.05	4.09 $\pm$0.05

3.2.2. Multi-speaker

We use VITS and Fastpitch + HiFi-GAN for comparison without pitch control as baseline models. Then, we use Fastpitch + HiFi-GAN as a baseline for comparison with pitch shift + 40Hz. We trained Fastpitch up to 500 epochs of 300K steps under the same configuration as TriniTTS and used pre-trained HiFi-GAN [6]. For VITS, $\text{VITS}_{\text{pre-trained}}$ is loaded from the pre-trained weights [7] trained up to 800K steps in Nvidia V100 4 GPUs [11]. For further training convergence analysis, we prepared two versions of the model weights for TriniTTS. The one TriniTTS_{300K} is trained up to 500 epochs of 300K steps, and another one TriniTTS_{30K} is the checkpoint at 30K steps for later training convergence analysis.

The evaluation results for the naturalness of the generated speeches are shown in Table 3. Without pitch control, the differences between the models are within the margin of confidence intervals. However, testing over generated speeches with pitch shift + 40Hz, we observe that TriniTTS_{300K} outperforms Fastpitch + HiFi-GAN in speech naturalness.

Moreover, TriniTTS_{30K} in Table 3 shows similar or slightly worse MOS compared to not only TriniTTS_{300K} but also other baseline models as well. We demonstrate that the training convergence reaches fast in TriniTTS multi-speaker model due to the additional pitch information given as input and speaker embedding conditioned to the alignment search module, which leads to faster alignment searching.

3.3. Inference speed

In real-life applications, inference speed is an important factor in speech synthesis. We generated 100 speeches randomly extracted from the LJSpeech test dataset using VITS and TriniTTS single speaker model to compare the inference speed. We use RTF as a comparison indicator: the elapsed time to generate speech divided by the length of the generated speech. We used Nvidia T4 for GPU and Intel Xeon 2.10 GHz for the CPU environment. We set the environment variable OMP_NUM_THREADS to 1 for CPU inference speed test.

[7] https://github.com/jaywalnut310/vits

Table 3: *MOS with 95% confidence intervals on the VCTK dataset*

Model	MOS	MOS (pitch+40Hz)
Ground truth	4.27 $\pm$0.03	4.30 $\pm$0.03
Fastpitch + HiFi-GAN	4.10 $\pm$0.05	4.01 $\pm$0.05
$\text{VITS}_{\text{pre-trained}}$	4.01 $\pm$0.05	-
TriniTTS_{300K}	4.07 $\pm$0.04	4.08 $\pm$0.04
TriniTTS_{30K}	4.05 $\pm$0.04	4.04 $\pm$0.05

In Table 4, TriniTTS shows faster inference than VITS in both CPU and GPU environments. We presume that faster inference comes from the elimination of normalizing flow module and sampling in VITS architecture even though pitch prediction-related modules are added. The decrease in the number of parameters in Table 5 demonstrates that the size of the model has become smaller in TriniTTS than VITS.

Table 4: *RTF on the inference of 100 wavs*

Model	RTF on CPU (diff)	RTF on GPU (diff)
VITS	0.3467	0.0048
TriniTTS	0.2454 (-28.84%)	0.0034 (-29.16%)

Table 5: *Number of parameters in the inference stage*

Model	Num of params (diff)
VITS	35.35M
TriniTTS	31.27M (-11.5%)

4. Conclusion

This paper proposes TriniTTS, an end-to-end TTS model with an on-the-fly alignment estimator and pitch controlling function. It managed to integrate an end-to-end architecture and on-the-fly alignment estimation with pitch controlling process in order to generate diverse speeches deterministically in a simpler training manner. In terms of speech naturalness, TriniTTS achieved speech quality close to the other state-of-the-art models while outperforming Fastpitch + HiFi-GAN in pitch-controlled speeches. It also generates speech faster than VITS in the inference stage both in CPU and GPU environments. We verified that the training steps got much more straightforward by applying end-to-end architecture and on-the-fly alignment search.

We will extend our approach towards changing the decoder module to diverse generators and discriminators [28, 29] and conditioning more information for the dataset, such as energy factor to control volume over each token.

5. Acknowledgements

We appreciate Hyung Yong Kim, Jihwan Park, Yushin Lim, Yunkyu Lim and Shukjae Choi for helpful discussions and advice in preparation for this paper.

6. References

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

[2] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[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," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 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 *Proc. International Conference on Learning Representation (ICLR)*, 2020.

[5] A. Łańcucki, "Fastpitch: Parallel text-to-speech with pitch prediction," in *Proc. ICASSP*, 2021, pp. 6588–6592.

[6] 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. ICASSP*, 2020, pp. 6199–6203.

[7] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[8] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, "UnivNet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation," in *Proc. Interspeech*, 2021, pp. 2207–2211.

[9] W.-N. Hsu, Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen *et al.*, "Hierarchical generative modeling for controllable speech synthesis," in *Proc. International Conference on Learning Representation (ICLR)*, 2018.

[10] K. Lee, K. Park, and D. Kim, "STYLER: Style factor modeling with rapidity and robustness via speech decomposition for expressive and controllable neural text to speech," in *Proc. Interspeech*, 2021.

[11] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *International Conference on Machine Learning (ICML)*, 2021, pp. 5530–5540.

[12] J. Donahue, S. Dieleman, M. Binkowski, E. Elsen, and K. Simonyan, "End-to-end adversarial text-to-speech," in *Proc. International Conference on Learning Representation (ICLR)*, 2020.

[13] R. J. Weiss, R. Skerry-Ryan, E. Battenberg, S. Mariooryad, and D. P. Kingma, "Wave-Tacotron: Spectrogram-free end-to-end text-to-speech synthesis," in *Proc. ICASSP*, 2021, pp. 5679–5683.

[14] T. Hayashi, R. Yamamoto, T. Yoshimura, P. Wu, J. Shi, T. Saeki, Y. Ju, Y. Yasuda, S. Takamichi, and S. Watanabe, "ESPnet2-TTS: Extending the edge of TTS research," *arXiv preprint arXiv:2110.07840*, 2021.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[16] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-TTS: A generative flow for text-to-speech via monotonic alignment search," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 8067–8077.

[17] R. Badlani, A. Łancucki, K. J. Shih, R. Valle, W. Ping, and B. Catanzaro, "One TTS alignment to rule them all," *arXiv preprint arXiv:2108.10447*, 2021.

[18] 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. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[19] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 17 022–17 033.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014.

[21] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proc. International Conference on Machine Learning (ICML)*, 2016, pp. 1558–1566.

[22] M. Mauch and S. Dixon, "PYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *Proc. ICASSP*, 2014, pp. 659–663.

[23] A. Cheveigné and H. Kawahara, "YIN, A fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, pp. 1917–1930, 2002.

[24] K. J. Shih, R. Valle, R. Badlani, A. Lancucki, W. Ping, and B. Catanzaro, "RAD-TTS: Parallel flow-based TTS with robust alignment learning and diverse synthesis," in *Proc. ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

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

[26] J. Yamagishi, C. Veaux, and K. MacDonald, "CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92)," 2019.

[27] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. International Conference on Learning Representation (ICLR)*, 2018.

[28] J. H. Kim, S. H. Lee, J. H. Lee, and S. W. Lee, "Fre-GAN: Adversarial frequency-consistent audio synthesis," in *Proc. Interspeech*, 2021, pp. 3246–3250.

[29] T. Kaneko, K. Tanaka, H. Kameoka, and S. Seki, "iSTFTNet: Fast and lightweight mel-spectrogram vocoder incorporating inverse short-time Fourier transform," *arXiv preprint arXiv:2203.02395*, 2022.