INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

LiteTTS: A Lightweight Mel-spectrogram-free Text-to-wave Synthesizer Based on Generative Adversarial Networks

Huu-Kim Nguyen[1], *Kihyuk Jeong*[1], *Seyun Um*[1], *Min-Jae Hwang*[2], *Eunwoo Song*[3], *Hong-Goo Kang*[1]

[1]Yonsei University, Seoul, Korea,
[2] Search Solutions Inc., Seongnam, Korea,
[3] Naver Corp., Seongnam, Korea
huukim136@dsp.yonsei.ac.kr

Abstract

In this paper, we propose a lightweight end-to-end text-to-speech model that can generate high-quality speech at breakneck speed. In our proposed model, a feature prediction module and a waveform generation module are combined within a single framework. The feature prediction module, which consists of two independent sub-modules, estimates latent space embeddings for input text and prosodic information, and the waveform generation module generates speech waveforms by conditioning on the estimated latent space embeddings. Unlike conventional approaches that estimate prosodic information using a pre-trained model, our model jointly trains the prosodic embedding network with the speech waveform generation task using an effective domain transfer technique. Experimental results show that our proposed model can generate samples 7 times faster than real-time, and about 1.6 times faster than FastSpeech 2, as we use only 13.4 million parameters. We confirm that the generated speech quality is still of a high standard as evaluated by mean opinion scores.

Index Terms: text-to-speech, prosody, vocoder, Hifi-GAN

1. Introduction

Thanks to the advancement in deep generative model-based speech synthesis systems, the quality of text-to-speech (TTS) systems has significantly improved in recent years. In general, these deep generative models are categorized into either autoregressive or non-autoregressive approaches. Some well-known autoregressive TTS models include Tacotron [1], Tacotron 2 [2], Deep Voice 3 [3], and Transformer TTS [4]. For instance, Tacotron 2 can generate audio of quality close to that of natural human voices, but it has a large number of parameters, and its training and inference time are slow due to the nature of autoregressive modeling in the waveform generation process.

To address this issue, non-autoregressive models such as FastSpeech [5], FastSpeech 2 [6], and Parallel Tacotron [7] have been proposed, which perform parallel processing during the inference stage. The key point of these models is that they predict the duration of each phonetic frame and expand it to have an identical duration as a Mel-spectrogram. Non-autoregressive TTS models break the barrier to real-time performance because acoustic frames are generated in parallel. Notably, FastSpeech 2 [6] predicts pitch and energy information from expanded linguistic context, then combines pitch, energy, and expanded text embeddings together before being decoded into a Mel-spectrogram.

Most speech synthesis systems are designed in a two-step manner: generation of Mel-spectrograms from input texts (i.e., a feature prediction module), followed by synthesis of waveforms with a pre-trained neural vocoder given the Mel-spectrograms (i.e., a waveform generation module) [8–13]. Although this two-step process is useful for training large and hard-to-train networks in settings where resources are limited, it may lead to an error propagation issue between the two modules. In addition, since the network needs a decoder to generate Mel-spectrograms in the feature prediction module, the network must be sufficiently large. If we can remove the Mel-spectrogram decoder in the network, then the model size can be significantly reduced.

In this paper, we propose a fully text-to-wave model that does not generate intermediate speech signal-related representations, i.e. Mel-spectrograms in the feature prediction module. In other words, our model directly synthesizes speech waveforms using a generative adversarial network (GAN) style waveform generation module. We also utilize a domain transfer technique to extract prosodic information directly from text. Since the two encoders that extract text and prosodic embeddings are constructed by small and computationally efficient architectures, our proposed model is small and its processing speed is considerably fast. Therefore, our model is suitable for on-device practical applications.

Our contributions are summarized as follows: 1) We design a fully text-to-wave model which is appropriate for portable devices with a size of only 13.4 million parameters; 2) We introduce an effective way to extract prosodic information from text during the speech generation process by leveraging the concept of domain transfer; 3) We evaluate the performance of our model and compare it with conventional approaches. Our model achieves a mean opinion score (MOS) of 3.84 while we are able to generate samples 7 times faster than real-time, 5 times faster than Tacotron 2 in a CPU environment.

2. Related works

Several studies [6, 14, 15] have investigated methods to directly generate speech waveforms from input text without using an intermediate feature decoding process. For instance, FastSpeech 2s uses Parallel WaveGAN [13] to synthesize speech waveforms directly from context information. However, it still uses an additional decoder that performs an auxiliary task to improve the training performance. On the other hand, SpeedySpeech [16] and LightSpeech [17] are models that aim to run on on-device real-time applications. As such, these two TTS models must use only a small number of parameters while still generating reasonable speech quality. However, SpeedySpeech still operates in a two-step processing style, and LightSpeech utilizes a neural architecture search technique to design a low-cost model with a baseline model, unlike our work.

3. Proposed model: LiteTTS

Fig. 1(a) illustrates the overall architecture of our proposed model. The model is composed of a prosody encoder ($\mathbf{E}^p$), a text encoder ($\mathbf{E}^t$), a domain transfer encoder ($\mathbf{E}^f$), an alignment

http://dx.doi.org/10.21437/Interspeech.2021-188

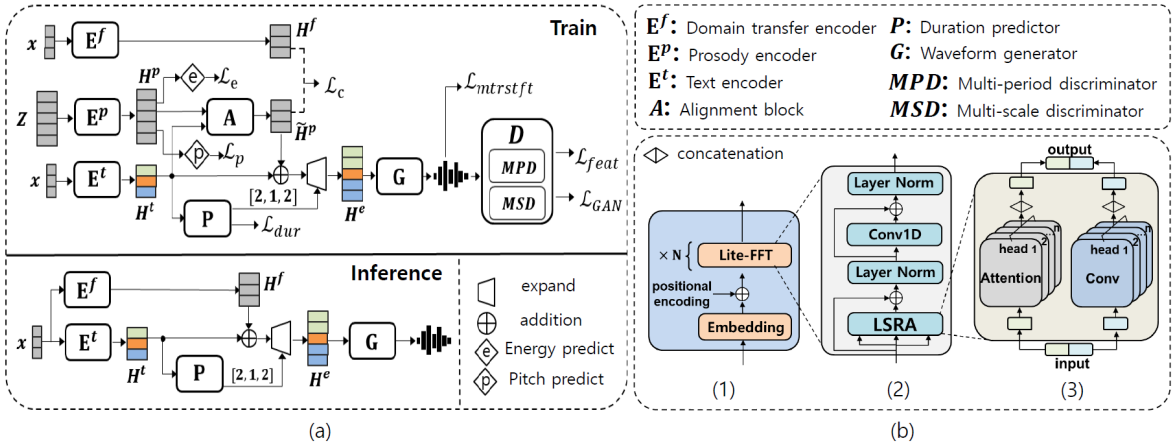

Figure 1: *(a) Overall training and inference architectures. During training, phonetic feature* $\boldsymbol{x}$ *and acoustic feature* $\mathbf{Z}$ *extracted from the same sentence are used as the model's inputs. During inference, the prosody encoder* $\mathbf{E}^p$ *and alignment blocks* $\mathbf{A}$ *are removed, but the model still uses identical phonetic features as input for both text and prosody encoders (*$\mathbf{E}^t$ *and* $\mathbf{E}^p$*). (b) Architecture of the text and domain transfer encoders: (1) overall architecture of encoder; (2) lite-FFT block; and (3) long-short range attention (LSRA) block.*

module ($\mathbf{A}$), a duration predictor ($\mathbf{P}$), a waveform generator ($\mathbf{G}$), and a discriminator block ($\mathbf{D}$). A sequence of m phonemes and an n-length reference Mel-spectrogram are respectively denoted as $\boldsymbol{x} = [x_1, x_2, ..., x_m]$ and $\mathbf{Z} = [z_1, z_2, ..., z_n]$, which are given as the inputs to the model during training time. Basically, phonetic information is obtained by a text encoder, and prosody-related information is provided by either the prosody encoder (during training) or the domain transfer encoder (during inference time). The combination of these two sources of information is expanded to have the same length as a Mel-spectrogram so that the waveform generator can directly synthesize speech from it. Lastly, the discriminator block distinguishes whether the speech is generated or recorded reference.

3.1. Model architecture

Text encoder ($\mathbf{E}^t$). The text encoder takes $\boldsymbol{x}$ as input and generates phonetic embeddings $\boldsymbol{H}^t = \mathbf{E}^t(\boldsymbol{x})$, where $\boldsymbol{H}^t = [\boldsymbol{h}_1^t, \boldsymbol{h}_2^t, ..., \boldsymbol{h}_m^t]$. Fig. 1(b) shows the detailed architecture of the text encoder. Input $\boldsymbol{x}$ is first embedded using a learnable phonetic embedding lookup table before adding positional encodings to it. Later, the combined embeddings are passed through multiple lite Feed-Forward Transformer (lite-FFT) blocks to obtain high-level phonetic embeddings $\boldsymbol{H}^t$.

The architecture of lite-FFT blocks (Fig. 1(b)) is a modified version of the Transformer to preserve the efficiency when running on low-resource devices. Specifically, we adopt the architecture of long-short range attention (LSRA) [18] to replace the attention module. Unlike in the conventional Transformer, where a single attention module is utilized to aggregate the information of the input, the LSRA module uses two branches that process the input information in parallel. As illustrated in Fig. 1(b3), the attention branch aims to gather the global knowledge of the input when the convolution branch is expected to obtain the local information. LSRA allows for the use of fewer parameters while still providing effective performance.

Prosody encoder ($\mathbf{E}^p$). Prosody[1] is one of the fundamental components of speech; thus, it needs to be included in the TTS training process in order to effectively model the data distribution [6, 19, 20]. FastSpeech 2 provides various prosodic knowledge such as energy and pitch to the model by using predictor networks. However, using a separate predictor network for each prosodic factor is neither resource-friendly nor optimal in terms of quality. Moreover, around three-fourths of connections in neural networks can be removed without damaging the overall performance because neural networks are likely to be overparameterized [21, 22]. Therefore, a single network may be able to carry out more than one specific task.

Motivated by this, we design a single network which extracts multiple prosodic factors from the input acoustic features. Our prosody encoder takes acoustic features $\mathbf{Z}$ as input, and outputs the prosody embeddings $\boldsymbol{H}^p = [\boldsymbol{h}_1^p, \boldsymbol{h}_2^p, ..., \boldsymbol{h}_n^p]$, where $\boldsymbol{H}^p = \mathbf{E}^p(\mathbf{Z})$. The single prosody network $\mathbf{E}^p$ is built such that it is able to productively provide multiple prosodic factors together. Specifically, embeddings $\boldsymbol{H}^p$ are followed by pitch prediction and energy prediction tasks to ensure that the prosody embeddings $\boldsymbol{H}^p$ contain both kinds of information. The pitch and energy prediction loss functions are defined as:

$$\mathcal{L}_p = \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{p}_i - \bar{\boldsymbol{p}}_i\|_1 \quad \text{and} \quad \mathcal{L}_e = \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{e}_i - \bar{\boldsymbol{e}}_i\|_1, \quad (1)$$

where $\boldsymbol{p}$, $\boldsymbol{e}$ denote the ground-truth pitch and energy while $\bar{\boldsymbol{p}}$, $\bar{\boldsymbol{e}}$ represent the predicted pitch and energy, respectively.

Alignment block (A). The role of alignment block $\mathbf{A}$ is to transform n-length frame-level prosody embeddings $\boldsymbol{H}^p$ into m-length phoneme-level prosody representations $\tilde{\boldsymbol{H}}^p$, where $\tilde{\boldsymbol{H}}^p = \mathbf{A}(\boldsymbol{H}^p)$. In the training stage, the prosody embeddings $\tilde{\boldsymbol{H}}^p$ are then combined with phonetic embeddings to form $\boldsymbol{H}^c = \boldsymbol{H}^t + \tilde{\boldsymbol{H}}^p$. Here, $\boldsymbol{H}^c$ carries both the phoneme and prosody information of the input utterance.

Duration predictor (P). The duration predictor is employed to predict the duration of each phoneme, which literally means how many frames in the Mel-spectrogram that a phoneme corresponds to. Given the text embeddings $\boldsymbol{H}^t$, the vector $\bar{\boldsymbol{d}} = [\bar{d}_1, \bar{d}_2, ..., \bar{d}_m]$ represents the predicted duration of m input phonemes as $\bar{\boldsymbol{d}} = \mathbf{P}(\boldsymbol{H}^t)$. It is also worth mentioning that $\sum_{i=1}^{m}\bar{d}_i = n$. The loss function for training the duration predictor is defined as:

$$\mathcal{L}_{dur} = \frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{d}_i - \bar{\boldsymbol{d}}_i\|_1, \quad (2)$$

where $\boldsymbol{d}$ represents the ground-truth duration.

[1]Prosody can be attributed to various aspects. Among them, pitch, energy and duration are crucial components. As duration information is given by the duration predictor $\mathbf{P}$, we hereinafter denote prosody as pitch and energy information.

After that, embeddings $\boldsymbol{H}^c$ are expanded based on the predicted duration $\bar{\boldsymbol{d}}$. For instance, the i^{th} frame of $\boldsymbol{H}^c$ will be repeated d_i times and they are stacked all together. As a result, we obtain $\boldsymbol{H}^e = [\boldsymbol{h}_1^e, \boldsymbol{h}_2^e, ..., \boldsymbol{h}_n^e]$ as an expanded version of $\boldsymbol{H}^c$. Embeddings $\boldsymbol{H}^e$ are then passed through a projection layer before going to the waveform generator module **G**. This projection layer is not illustrated in Fig. 1(a) to keep it simple.

Domain transfer encoder ($\mathbf{E}^f$). The phoneme sequence input $\boldsymbol{x}$ is fed into this encoder to generate the m-length embeddings as $\boldsymbol{H}^f = \mathbf{E}^f(\boldsymbol{x})$. The embeddings $\boldsymbol{H}^f$ are expected to share the same prosody domain knowledge with embeddings $\tilde{\boldsymbol{H}}^p$ by constraining them to be close to each other using a loss function $\mathcal{L}_c$ that encourages their similarity. We experimented with various types of losses such as pair-wise ranking loss [23,24] and cosine similarity, but found that simply using L1 loss for $\mathcal{L}_c$ performs better than the others.

The underlying intuition behind this is that the prosody information in $\tilde{\boldsymbol{H}}^p$ will be transferred to $\boldsymbol{H}^f$ as training goes on thanks to the loss function $\mathcal{L}_c$. Since $\tilde{\boldsymbol{H}}^p$ and $\boldsymbol{H}^f$ are coupled during training, at inference time, the domain transfer block will try to extract $\boldsymbol{H}^f$ with meaningful pitch and energy information. Later, $\boldsymbol{H}^f$ will be provided directly to phonetic embeddings $\boldsymbol{H}^t$ as a substitute for $\tilde{\boldsymbol{H}}^p$.

Waveform generator and discriminator (G & D). In conventional approaches, it is common to design a decoder following the encoders to generate intermediate speech representations (e.g. Mel-spectrograms). However, despite our decision to design a Mel-spectrogram-free structure to keep the model lightweight and inexpensive, our approach still works well. In this module, a fixed-length segment of hidden embeddings $\boldsymbol{H}^e$ is used as the input of generator **G** during training. Generator **G** then upsamples its input to produce raw waveforms, finalizing a fully end-to-end process. Discriminator **D** attempts to determine whether its input is synthesized or recorded reference.

3.2. Training losses

We adopt the architecture of the generator and discriminator in Hifi-GAN [12]. The discriminator block **D** contains two submodules: multi-period discriminator (**MPD**) and multi-scale discriminator (**MSD**). Each submodule includes multiple sub-discriminators which handle audio inputs in different periodicities in the case of **MPD** and in different scales in the case of **MSD**. Similar to [12], we train the waveform generation and the discriminator networks with the objective functions used for LSGAN [25]:

$$\begin{aligned}\mathcal{L}_{GAN}(D;G) &= \mathbb{E}_{v,s}\left[\sum_{k=1}^{K}(D_k(v)-1)^2 + (D_k(G(s)))^2\right], \\ \mathcal{L}_{GAN}(G;D) &= \mathbb{E}_{s}\left[\sum_{k=1}^{K}(D_k(G(s))-1)^2\right], \end{aligned} \tag{3}$$

where v denotes the ground-truth waveforms in different scales or periodicities, s represents $\boldsymbol{H}^e$, K is the total number of sub-discriminators and D_k indicates the sub-discriminator in either **MPD** or **MSD**. Feature matching loss [26] is additionally applied to the generator:

$$\mathcal{L}_{feat}(G;D) = \mathbb{E}_{x,s}\left[\sum_{k=1}^{K}\sum_{i=1}^{T}\frac{1}{N_i}\|D_k^i(x) - D_k^i(G(s))\|_1\right], \tag{4}$$

where T indicates the number of layers in the sub-discriminator D_k, N_i is the total number of features in the i^{th} layer, and D^i represents the features of that layer. Additionally, to further enhance the stability, we apply an auxiliary loss called multi-resolution short-time Fourier transform (STFT) loss [13], denoted as $\mathcal{L}_{mrstft}$. It captures the dissimilarity between the STFTs of the generated and ground-truth waveforms in multiple configurations (e.g. frame size, hop size, FFT size). To this end, the final loss is calculated as follows:

$$\begin{aligned}\mathcal{L}_G = \mathcal{L}_{GAN}(D;G) + \lambda_f\mathcal{L}_{feat}(G;D) + \mathcal{L}_{dur} \\ +\lambda_m\mathcal{L}_{mrstft} + \mathcal{L}_p + \mathcal{L}_e + \lambda_c\mathcal{L}_c,\end{aligned} \tag{5}$$

$$\mathcal{L}_D = \mathcal{L}_{GAN}(G;D), \tag{6}$$

where λ_f, λ_m and λ_c are set to be 2, 30, and 5, respectively.

Table 1: *The detailed architecture of the proposed model*

LSRA	**Attention branch**: 2-head attention **Convolution branch**: 2-head convolution (Conv-3-64-RB → Conv-3-32-RB → Conv-3-16-RB → Conv-3-32-RB → Conv-3-64-RB → Conv-3-128-RB for each head)
Conv1D	Conv-9-1024-R → Conv-1-256-Dropout(0.5)
$\mathbf{E}^t$, $\mathbf{E}^f$	256-dim embedding → 2×lite-FFT
$\mathbf{E}^p$	2×lite-FFT
A	Scaled dot-product attention
P	Conv-3-256-RB-Dropout(0.5)×2 → Linear-1
p, e	Conv-3-256-RB-Dropout(0.5) → Linear-1

B stands for batch normalization, and R for ReLU activation. Conv-i-j denotes a convolution with kernel size i and channel j, Linear-1 denotes a linear layer with output feature dimension 1, and p, e indicate pitch and energy predictors. Architectures of **G** and **D** are the same as in [12].

4. Experiments and results

4.1. Dataset and model configurations

We train our model using the LJSpeech corpus [27]. For Mel-spectrogram configurations, we set the window size, hop size, and fast Fourier transform (FFT) size to be 1024, 256, and 1024, respectively, with a sampling rate of 22,050 Hz. We use the AdamW [28] optimizer with $\beta_1 = 0.8$ and $\beta_2 = 0.99$. The learning rate is set to 0.0002, then it is decayed with a factor of 0.999 during training. The batch size is set to 16. We train the model for one million steps, but we disable the effect of $\mathcal{L}_c$ loss up to two hundred thousand steps. Table 1 shows detailed components of the architecture of our model.

4.2. Results

We show the effectiveness of our model by making comparisons in various aspects with the following baseline TTS frameworks:

- Tacotron 2 feature predictor [2] + Hifi-GAN vocoder
- FastSpeech 2 feature predictor [6] + Hifi-GAN vocoder

Since our proposed model utilizes the generator adopted from Hifi-GAN, we use it as the vocoder for Tacotron 2 and FastSpeech 2 models for fair comparisons.

Inference speed, complexity and model size. As shown in Table 2, our model requires only 13.4 million parameters, which is less than half that of Tacotron 2 and FastSpeech 2. On the other hand, in order to demonstrate that our model can generate speech audio speedily in CPU environment, we measure the real-time factor (RTF), which basically denotes how much time is needed to generate one second of audio.

We set up a consistent environment to generate speech using every model for fair comparisons. We use an Intel(R)

Table 2: *Comparisons between our proposed model and other TTS frameworks*

Model	# params (M)	RTF	GMACs	MOS	PER (%)
Tacotron 2 + Hifi-GAN	29.4	0.71 ± 0.034	100.5	3.88 ± 0.12	10.19
FastSpeech 2 + Hifi-GAN	28.4	0.22 ± 0.015	39.3	3.76 ± 0.11	9.49
Proposed model	**13.4**	$\mathbf{0.14 \pm 0.013}$	**27.0**	3.84 ± 0.11	**5.75**

RTFs and MACs are calculated after synthesizing speech from text input which consists of 101 phonemes. For RTFs, the process is iterated over 100 times and the averaged value is taken.

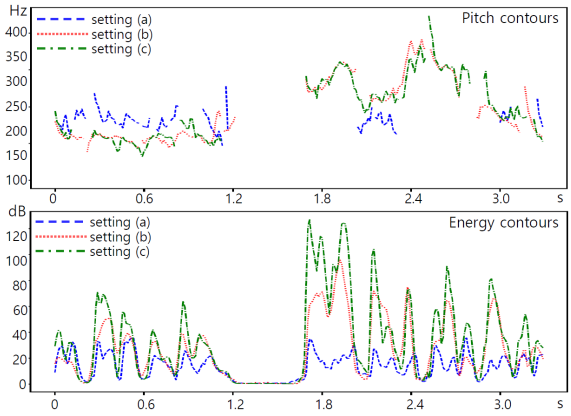

Figure 2: *Pitch and energy contours of synthesized speech using three settings: (a) Only $\boldsymbol{H}^t$, (b) Only $\boldsymbol{H}^f$, (c) Both $\boldsymbol{H}^t$ and $\boldsymbol{H}^f$.*

Core(TM) i7-7700K CPU @ 4.20 GHz with a single thread for performing inference. As seen in Table 2, our model can generate audio samples about 7 times faster than real-time and about 5 and 1.6 times faster than Tacotron 2 and FastSpeech 2, respectively. One of the reasons for this is that during the inference stage, phonetic embeddings $\boldsymbol{H}^t$ and transferred prosody $\boldsymbol{H}^f$ are calculated in parallel fashion so that we can save valuable generation time. This additionally contributes to having lower multiply-accumulate operations (MACs). As shown in Table 2, our model uses only 27.0 GMACs, which is significantly less costly than both Tacotron 2 and FastSpeech 2. In our experiments, all RTFs seem to meet the real-time requirement, as we use a fairly powerful CPU. However, in practical on-device applications, less processing power will be available; we expect that this is when our model will be most effective.

Evaluation. To subjectively measure the speech quality, we conduct a test in which fourteen candidates were asked to independently rate the quality of fifteen generated utterances[2] with the scores ranging from 1 to 5 under the following criteria, ordered from highest to lowest importance: *stability to noise and artifacts, good pronunciation, naturalness*. After that, we calculate the mean opinion score (MOS) for each model. As shown in Table 2, our model obtains a MOS of 3.84, compared to 3.76 of FastSpeech 2. Although the MOS result for Tacotron 2 is slightly higher than ours due to the adoption of autoregressive modeling, two scores are statistically insignificant.

In addition, we conduct an automatic speech recognition (ASR) task to investigate the intelligibility of our model and the two baselines. For each model, we create a separate corpus from the available LJSpeech transcripts. These three new LJSpeech datasets are used to train three different ASR models using the Listen, Attend and Spell (LAS) model [29]. The results are shown in the **PER** column in Table 2. We see that the ASR task achieves the best performance when using speech generated from our model, as it results in the lowest phoneme error rate (PER). We observe that speech generated from FastSpeech 2 is occasionally unstable due to noise and artifacts, while speech synthesized from Tacotron 2 is prone to mispronunciation or repeated words in spite of high naturalness. The separate training the TTS model and vocoder may be another reason for the higher PERs of FastSpeech 2 and Tacotron 2.

[2]Audio samples are available at: https://dsp136.github.io/2021-04-01-interspeech-samples/

Table 3: *ASR results corresponding to ablation inference*

PER(a) (%)	PER(b) (%)	PER(c) (%)
5.07	34.23	5.75

4.3. Ablation inference

At inference time, our model normally uses the combination of phonetic embeddings $\boldsymbol{H}^t$ and prosody embeddings $\boldsymbol{H}^f$ to generate speech. We run ablation studies to investigate the impact of this step by deliberately synthesizing speech using three settings: (a) use only $\boldsymbol{H}^t$, (b) use only $\boldsymbol{H}^f$, (c) use both $\boldsymbol{H}^t$ and $\boldsymbol{H}^f$.

Fig. 2 shows the pitch and energy contours of speech generated using the three settings. As expected, for case (a), the contours of pitch and energy are clearly flat, which implies that no prosodic information resides in phonetic embeddings $\boldsymbol{H}^t$. In contrast, for case (b), diverse prosodic information is given to the synthesized speech, as observed in the contours. Interestingly, generated speech in case (c) tends to follow a similar trend of prosody compared to case (b).

We also perform ASR tasks for the synthesized speech from cases (a), (b) and (c). The obtained phoneme error rates are shown in Table 3 as PER(a), PER(b) and PER(c), respectively. Unsurprisingly, PER(a) is much smaller than PER(b) (5.07% vs 34.23%), as case (b) omits all phonetic information. Interestingly, PER(a) is lower than PER(c). We believe that in setting (c), the presence of prosody makes the recognition task more difficult. This result also suggests that removing prosodic information before training may be a promising direction to improve the performance of ASR.

5. Conclusion

In this paper, we proposed a TTS model that directly generates speech waveforms from text inputs. Since the model is lightweight and it can synthesize speech rapidly, it is particularly suitable for on-device TTS applications. Additionally, by leveraging the concept of domain transfer, the model extracts prosodic information from text inputs. We confirm that the quality of generated speech is high, as expressed by both subjective and objective tests. Interesting research directions for the future may include investigating a more effective similarity loss function rather than the simple L1 loss, and exploring different discriminator architectures.

6. Acknowledgments

This work was supported by Clova Voice, NAVER Corp., Seongnam, Korea.

7. References

[1] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017, pp. 4006–4010.

[2] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *Proc. 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 *arXiv preprint arXiv:1710.07654*, 2018.

[4] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proc. AAAI*, 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. NIPS*, 2019.

[6] 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 *arXiv preprint arXiv:2006.04558*, 2020.

[7] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. Weiss, and Y. Wu, "Parallel Tacotron: Non-autoregressive and controllable TTS," in *arXiv preprint arXiv:2010.11439*, 2020.

[8] A. van den 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," in *arXiv preprint arXiv:1609.03499*, 2016.

[9] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proc. ICML*, 2018, pp. 2410–2419.

[10] R. Prenger, R. Valle, and B. Catanzaro, "WaveGlow: A flow-based generative network for speech synthesis," in *Proc. ICASSP*, 2019, pp. 3617–3621.

[11] 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. NIPS*, 2019.

[12] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *arXiv preprint arXiv:2010.05646*, 2020.

[13] 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.

[14] W. Ping, K. Peng, and J. Chen, "ClariNet: Parallel wave generation in end-to-end text-to-speech," in *arXiv preprint arXiv:1807.07281*, 2019.

[15] 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 *arXiv preprint arXiv:2011.0356*, 2021.

[16] J. Vainer and O. Dušek, "SpeedySpeech: Efficient neural speech synthesis," in *Proc. Interspeech*, 2020, pp. 3575–3579.

[17] R. Luo, X. Tan, R. Wang, T. Qin, J. Li, S. Zhao, E. Chen, and T.-Y. Liu, "LightSpeech: Lightweight and fast text to speech with neural architecture search," in *arXiv preprint arXiv:2102.04040*, 2021.

[18] Z. Wu*, Z. Liu*, J. Lin, Y. Lin, and S. Han, "Lite transformer with long-short range attention," in *Proc. ICLR*, 2020.

[19] Y. Wang, D. Stanton, Y. Zhang, R.-S. Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *Proc. ICML*, 2018, pp. 5180–5189.

[20] R. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron," in *Proc. ICML*, 2018, pp. 4693–4702.

[21] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *arXiv preprint arXiv:1611.06440*, 2016.

[22] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017.

[23] H. Zhou, Y. Liu, Z. Liu, P. Luo, and X. Wang, "Talking face generation by adversarially disentangled audio-visual representation," in *Proc. AAAI*, 2019, pp. 9299–9306.

[24] J.-X. Zhang, Z.-H. Ling, and L.-R. Dai, "Non-parallel sequence-to-sequence voice conversion with disentangled linguistic and speaker representations," in *IEEE/ACM Trans. Audio, Speech, and Lang. Process.*, 2019, pp. 540–552.

[25] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. ICCV*, 2017, pp. 2813–2821.

[26] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *arXiv preprint arXiv:1512.09300*, 2016.

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

[28] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *arXiv preprint arXiv:1711.05101*, 2017.

[29] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.