STREAMSPEECH: LOW-LATENCY NEURAL ARCHITECTURE FOR HIGH-QUALITY ON-DEVICE SPEECH SYNTHESIS

Georgi Shopov[1], *Stefan Gerdjikov*[1,2], *Stoyan Mihov*[1]

[1]Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
[2]Faculty of Mathematics and Informatics, Sofia University, Bulgaria
gshopov@lml.bas.bg, stefangerdzhikov@fmi.uni-sofia.bg, stoyan@lml.bas.bg

ABSTRACT

Neural text-to-speech (TTS) systems have recently demonstrated the ability to synthesize high-quality natural speech. However, the inference latency and real-time factor (RTF) of such systems are still too high for deployment on devices without specialized hardware. In this paper, we describe StreamSpeech – an optimized architecture of a complete TTS system that produces high-quality speech and runs faster than real time with imperceptible latency on resource-constrained devices by utilizing a single CPU core. We divide the standard TTS processing pipeline into three phases with respect to their operating resolution and optimize them separately. Our main novel contribution is the introduction of a lightweight convolutional acoustic model decoder, which enables streaming and low-latency speech generation. Experiments show that the resulting complete TTS system achieves 79 ms latency, 0.155 RTF on a low-power notebook x86 CPU and 276 ms latency, 0.289 RTF on a mid-range mobile ARM CPU with no noticeable difference in the quality of the generated speech.

Index Terms— Text-to-speech optimizations, low latency, on-device synthesis

1. INTRODUCTION

Modern TTS systems employ deep neural networks to synthesize high-quality natural-sounding speech. However, such systems are usually very computationally demanding and require specialized hardware (e.g. GPUs and TPUs) to run in real time. Recently, non-autoregressive acoustic models such as FastPitch [1] and FastSpeech 2 [2] have been proposed to significantly increase the inference speed over previous autoregressive models [3]. Furthermore, linear prediction has been utilized in the LPCNet [4] vocoder to greatly improve the efficiency of neural speech synthesis. However, the inference latency and RTF of modern TTS systems based on the standard FastSpeech 2 and LPCNet models are still too high for deployment in resource-constrained scenarios and on devices without specialized hardware.

To address this problem, recent research has been focused on the optimization of both the acoustic model and the vocoder networks. LightSpeech [5] leverages neural architecture search to find lightweight architectures to further speed up the inference of FastSpeech 2, while maintaining the voice quality. It also utilizes depth-wise separable convolutions (SepConv) [6], which are considerably more memory and computationally efficient compared to vanilla convolutions. Many LPCNet variants have also been studied with the aim to reduce the complexity of the model without sacrificing speech quality. DurIAN [7], FeatherWave [8] and Subband LPCNet [9] adopt multi-band signal processing to generate several speech samples in parallel in a single step. In [10], the authors propose using tensor decomposition to reduce the complexity of the dual fully-connected layer of the vocoder. More recently, Valin et al. [11] have combined weight quantization and hierarchical softmax to further improve the efficiency of LPCNet.

In contrast, research on the optimization of a complete TTS system aiming to achieve low-latency on-device synthesis has been more limited [12, 13, 14, 15]. He et al. [14] and Wu et al. [15] propose to use a recurrent acoustic decoder with a multi-rate attention mechanism that attends to lower rate features on the character, syllable, and word levels instead of higher rate frame-level features. In order to maintain inference speed independent of the input utterance length, they enforce a hard limit on the length of the attention context via dynamic max pooling. Others [16] fix the computational bottlenecks of the original Tacotron 2 [3] decoder by decreasing the LSTM width and progressively decreasing the receptive fields of the post-net convolutions. They also demonstrate that many instances of the LPCNet vocoder can be used to process independent parts of the input Mel spectrogram in parallel on separate threads.

In this paper, we describe StreamSpeech – an optimized TTS architecture based on FastSpeech 2 and LPCNet that produces 24 kHz high-quality speech and runs faster than real time with imperceptible latency on resource-constrained devices. Contrary to the work mentioned above, our approach does not employ recurrent or attention-based acoustic decoders. Moreover, we focus on reducing the total computational load of the vocoder, instead of using multiple threads to run it in parallel.

The main contributions of our work can be summarized as follows. (1) We divide the TTS processing pipeline into three phases in correspondence with their operating resolutions: character-level (Text Analysis and FastSpeech 2 encoder), frame-level (FastSpeech 2 decoder), and sample-level (LPCNet), which we optimize separately. (2) In order to optimize the computational complexity of the FastSpeech 2 encoder, we utilize depth-wise separable convolutions. We demonstrate that the low character-level resolution of the encoder makes streamability and further optimizations of this module unnecessary. (3) We replace the non-autoregressive transformer-based FastSpeech 2 decoder with a lightweight streamable convolutional decoder to achieve constant low latency without

 DOI: 10.1109/ICASSP49357.2023.10096566

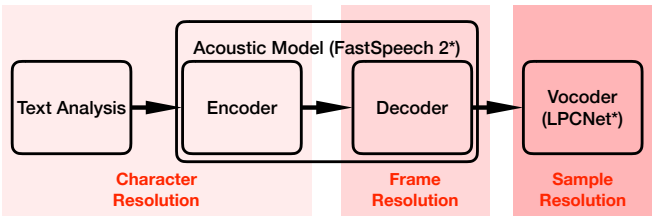

Fig. 1. Overall architecture of the TTS system. (*) The baseline system utilizes the original FastSpeech 2 and LPCNet models, whereas StreamSpeech implements the optimizations described in Section 3.

affecting the synthesized speech quality. (4) In terms of inference speed, we focus on optimizing the vocoder, which operates at the highest sample-level resolution in the system. We employ multi-band signal processing and hierarchical softmax to significantly improve the efficiency of LPCNet, while maintaining the voice quality.

2. BASELINE ARCHITECTURE

Following the recent trend, our TTS processing pipeline consists of three main modules: text analysis, acoustic model and vocoder (see Figure 1).

Text analysis. The text analysis module performs text normalization and verbalization (expansion of numbers, abbreviations, etc.) using contextual rules and dictionaries. It also performs grapheme-to-phoneme conversion using a combination of rules and a transformer-based neural network. [1] The rules and dictionaries are implemented as a cascade of 26 finite-state transducers (FSTs) [17, 18].

Acoustic model. The acoustic model is based on the FastSpeech 2 architecture [2]. For the baseline we use the original FastSpeech 2 model, which has 27M parameters. It is composed of an encoder – which consists of 4 Feed-Forward Transformer (FFT) blocks, duration, pitch, and energy predictors, and a decoder – which consists of 4 FFT blocks and a final spectrogram projection.

Vocoder. The vocoder is based on the LPCNet architecture [4]. However, the 16 kHz sampling rate of the speech generated by the original implementation does not provide satisfactory sound quality. Therefore, for our baseline implementation we use a modified LPCNet model, which synthesizes 24 kHz speech. Following FeatherWave [8], we increase the receptive field of the frame rate network by using a stack of five 1×3 one-dimensional convolutions with 256 channels, each followed by a tanh activation. Additionally, we modify the vocoder to accept the output features of FastSpeech 2 – 80-dimensional Mel spectrograms representing 10.7-ms frames (256 samples), as input. To account for the increased sampling rate, we extract 24 (instead of 16) linear prediction coefficients from the input Mel spectrograms.

3. STREAMSPEECH ARCHITECTURE

The processing pipeline of the presented baseline architecture consists of three phases that require operation in increasingly finer resolutions (see Figure 1). In this section, we first present the framework used for our performance evaluations. Then, we profile the baseline system and propose several optimizations to its components. Each optimization is motivated by the resolution at which the corresponding component operates.

[1] Our grapheme-to-phoneme approach is the subject of a forthcoming paper.

Module	#Params	x86		A76	
		LAT	RTF	LAT	RTF
Text Anal.	*6.5M	46	0.005	162	0.019
Encoder	15.2M	88	0.010	328	0.039
Decoder	11.6M	409	0.049	1676	0.199
Vocoder	1.7M	4	0.383	8	0.725
Total	35M	547	0.447	2174	0.982

Table 1. Profiling of the components of the baseline system. Latency is given in milliseconds. The RTF is equal to the processing time divided by the duration of the generated speech. (*) The text analysis module also contains 26 FSTs.

3.1. Performance analysis

The dataset used in all of our performance evaluations consists of 600 utterances with lengths uniformly distributed between 10 and 250 characters. The durations of the corresponding synthesized speech vary between 0.85 and 16.23 seconds. The average utterance length and duration are 127 characters and 8.34 seconds respectively. We conduct all of our experiments on a low-power notebook CPU (Intel Core i5-5257U @ 2.7 GHz, referred to as x86) and a mid-range mobile CPU (Cortex-A76 @ 2.25 GHz, referred to as A76) using a single core.

The measurements of the inference latency and speed of the baseline system are presented in Table 1. The total latency of the system is about 0.55 seconds on the x86 CPU and 2.2 seconds on the A76 CPU. This heavily hampers the use of the system for interactive speech synthesis applications such as speech interfaces. The high latency of the baseline system is caused by the Text Analysis, FastSpeech 2 encoder and FastSpeech 2 decoder modules, because they employ a global attention mechanism, which necessitates the processing of the whole input and prohibits streaming. However, three quarters of the latency is attributed only to the execution of the FastSpeech 2 decoder, since it operates at the higher frame-level resolution. Thus, we focus mainly on architectural changes to the FastSpeech 2 decoder that make it streamable and substantially reduce its latency. Additionally, we optimize the computational complexity of the encoder to further improve the overall latency of the system. In terms of inference speed, three quarters of the computational load is due to the vocoder, because it operates at a substantially higher resolution than the rest of the components. Therefore, we also concentrate on significantly improving the efficiency of LPCNet.

3.2. Acoustic model optimizations

To optimize the acoustic model, we first experimented with the automatically discovered architecture of LightSpeech [5], which has been shown to achieve 6.5 times inference speedup. Even though this modification decreased the latency of the complete TTS system to around 0.5 seconds on the A76 CPU, this still limits its use on mobile devices. Moreover, in our setup this resulted in perceptible degradation in the quality of the generated speech. Therefore, we

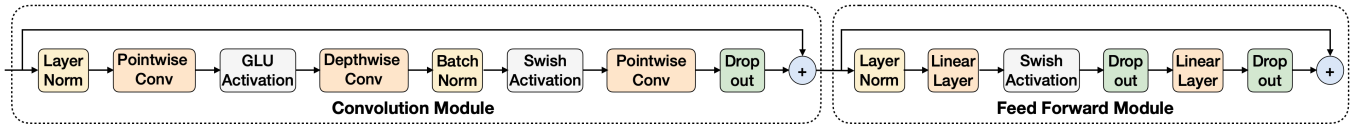

Fig. 2. The architecture of a StreamSpeech decoder block. The StreamSpeech decoder consists of a stack of 5 blocks.

Architecture	Variant	x86		A76	
		LAT	RTF	LAT	RTF
FFT	Baseline	409	0.049	1676	0.199
	+SepConv	197	0.024	749	0.090
StreamSpeech	$Rate = 1$	8	0.088	19	0.115
	$Rate = 6$	9	0.024	22	0.057
	$Rate = \infty$	118	0.014	447	0.054

Table 2. Latency and RTF of the examined decoder architectures.

adopt a different approach.

Encoder. We speculate that in the FastSpeech 2 architecture speech prosody is modeled mainly by the encoder, which utilizes multi-head self-attention to capture long-term dependencies. Similar to the work in [15], we decide to retain the non-streaming architecture of the encoder so that it can better model the prosody of longer utterances. In order to improve its efficiency, we substitute the vanilla convolutions in the FFT blocks and the predictors with depthwise separable convolutions, as suggested in LightSpeech [5]. Table 4 shows that this reduces the encoder's parameters by a factor of 3.8 and its RTF by a factor greater than 3.3. It also demonstrates that the low character-level operating resolution of the encoder makes further optimizations and streaming of this module unnecessary.

Decoder. Table 2 demonstrates that by applying the same optimizations as those for the encoder (+SepConv) improved the latency and RTF of the baseline decoder by a factor greater than 2. However, the resulting total system latency remains greater than 1 second on A76 which is still too high for interactive applications required to run on devices with limited computational capacity. Thus, we propose major architectural modifications to the decoder.

We speculate that the role of the decoder in the FastSpeech 2 architecture is mainly to model the coarticulation phenomena of speech production. Since coarticulation is a local occurrence, there is no need of global information to resolve it. Therefore, we replace the FFT blocks, which gather global information via an attention mechanism, with new convolutional blocks that capture local context progressively layer by layer. As shown in Figure 2, the StreamSpeech decoder blocks consist of a convolutional module followed by a feed-forward module. We use the convolutional and feed-forward modules introduced in the Conformer [19] speech recognition model, because they have proven to be very effective in capturing the relative offset-based local correlations in natural speech. In our setup, we set the hidden dimension of the modules to 256 and the kernel size of the convolutions to 31. We observe that a stack of 5 of the proposed blocks delivers speech quality which is hardly distinguishable from that of the baseline model.

Since the StreamSpeech decoder architecture is feed-forward and convolutional, it can be applied in a streaming way, which implies constant latency and inference speed independent of the input length. The latency of the decoder module represents the time delay between receiving the result from the encoder and producing the first Mel spectrogram. In the proposed architecture, the latency is determined entirely by the kernel size and padding of the convolutions, and the number of stacked blocks. We use 5 blocks and convolutions with kernel size 31 and padding 15. Therefore, to generate its first Mel spectrogram the decoder has to process an input of length 91. After this initial sequence has been processed, the decoder can continue to operate sequentially by consuming a single input and producing a single output.

Processing inputs one by one results in the lowest latency. However, it causes inferior utilization of low-level SIMD instructions [20, 21], which substantially increases the RTF. To strike balance between latency and RTF, we parameterize the streaming mode of the decoder so that it processes simultaneously $Rate$ number of inputs in a single step. To measure its optimal RTF, we also profile the decoder in non-streaming mode ($Rate = \infty$), in which the whole input is processed in parallel.

Table 2 presents the latency and RTF of the StreamSpeech decoder. Operating in streaming mode with $Rate = 1$, the proposed approach drastically reduces the latency compared to the FFT-based architectures, rendering it practically insignificant. On the x86 CPU, however, this is at the expense of a 3.66 times increase in the RTF with respect to the SepConv FFT variant and 6.28 times increase with respect to $Rate = \infty$. Table 2 also shows that streaming with $Rate = 6$ causes insignificant increase in latency with respect to $Rate = 1$, while achieving RTF which is close to the optimal ($Rate = \infty$). Thus, in StreamSpeech, we utilize $Rate = 6$.

Vocoder Architecture	x86	A76
LPCNet	0.383	0.725
+ Multi-Band	0.192	0.360
+ Hierarchical Softmax	0.123	0.202

Table 3. RTF of the examined vocoder architectures.

3.3. Vocoder optimizations

The original LPCNet vocoder is designed to predict audio signals sample by sample, which significantly slows down the audio generation process. We adopt the multi-band parallel generation approach [7, 8, 9] and modify the original LPCNet model to take inputs from multiple subbands and predict excitations for all subbands simultanously through multiple dual fully-connected and softmax layers. Using this approach, the audio in each subband is downsampled by a factor of N (the number of frequency bands). In our setup, we employ 4 frequency bands and analysis/synthesis FIR filters of order 64. Table 3 shows that by using multi-band processing, we achieve twofold inference speedup.

Detailed analysis of the performance of the multi-band LPCNet revealed that at least 45% of the computational load is focused on the evaluation of the 4 dual fully-connected layers and the softmax acti-

Module	#Params	x86		A76	
		LAT	RTF	LAT	RTF
Text Anal.	6.5M	46	0.005	162	0.019
Encoder	4M (-11.2M)	23 (-65)	0.003 (-0.007)	89 (-239)	0.011 (-0.028)
Decoder	3.7M (-7.9M)	9 (-400)	0.024 (-0.025)	22 (-1654)	0.057 (-0.142)
Vocoder	1.7M	1 (-3)	0.123 (-0.260)	3 (-5)	0.202 (-0.523)
Total	15.9M (-19.1M)	79 (-468)	0.155 (-0.292)	276 (-1898)	0.289 (-0.693)

Table 4. Profiling of the components of StreamSpeech. The numbers in green show the reductions with respect to the baseline system.

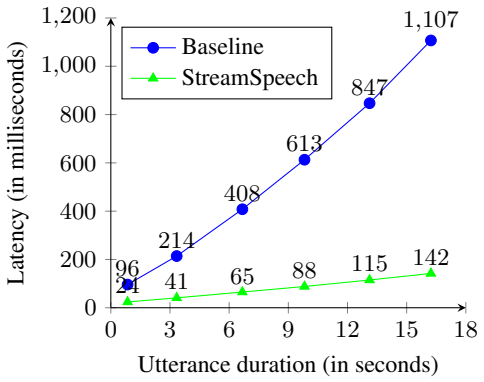

Fig. 3. Latency of the baseline and the StreamSpeech TTS systems with respect to the utterance duration measured on the x86 CPU.

vations. Thus, we adopt the hierarchical softmax approach of Valin et al. [11] to represent the output distribution of each frequency band as an 8-level binary tree, with each branch probability being computed as a sigmoid output. Thereby, we reduce the number of dual fully-connected outputs that have to be evaluated during sampling from 256 to 8. The results (see Table 3) demonstrate that by using hierarchical softmax we achieve 36% improvement in the RTF on the x86 CPU and 44% on the A76 CPU.

4. PERFORMANCE EVALUATION

In Table 4, we present the performance of the StreamSpeech TTS system along with the achieved improvements with respect to the baseline. The experiments are conducted within the framework described in Subsection 3.1. The StreamSpeech architecture achieves 79 ms latency, 0.155 RTF on the x86 CPU and 276 ms latency, 0.289 RTF on the A76 CPU. Next, we analyze the effect of the input utterance length on the performance of the system.

The decoder and the vocoder of the StreamSpeech system have constant latency and RTF that are not affected by the length of the input. However, the text analysis and the encoder modules operate on the whole input sequence in a non-streaming manner. Table 4 demonstrates that the vocoder is still responsible for at least 70% of the computational load of the system. This, combined with the constant RTF of the vocoder, means that the RTF of the complete system will stay practically constant for utterances of reasonable length. The latency, on the other hand, behaves differently. Figure 3 shows the dependency of the latency of the baseline and of the StreamSpeech systems on the utterance duration. The graph demonstrates the accomplished drastic improvements in the relation between latency and utterance duration. It also highlights the fact that the StreamSpeech system manages to maintain low latencies (under 150 ms) even for long utterances.

5. QUALITY EVALUATION

We conduct experiments to assess the quality of the speech synthesized with StreamSpeech and compare it with that of the baseline. We train all acoustic models with the scheme from [2] and all vocoders with the scheme from [4]. In the experiments, we compute three objective measures: Mel cepstral distortion (MCD), log-F_0 root mean square error (F_0 RMSE) and character error rate (CER) of a neural speech recognition system, on 130 held-out utterances.

Architecture	MCD	F_0 RMSE	CER %
Baseline	5.59 ± 0.54	0.344 ± 0.104	1.39
+ SepConv	5.59 ± 0.53	0.319 ± 0.092	**1.34**
+ ConvDec	$\mathbf{5.53 \pm 0.54}$	0.321 ± 0.101	1.49
+ MB	5.67 ± 0.54	$\mathbf{0.313 \pm 0.092}$	1.49
+ HSM	5.60 ± 0.54	0.318 ± 0.103	1.44
+ NR	6.45 ± 0.92	0.320 ± 0.096	1.52

Table 5. Effect of the proposed optimizations on several objective measures. SepConv stands for "separable convolutions", ConvDec for "convolutional decoder", MB for "multi-band", HSM for "hierarchical softmax" and NR for "noise reduction".

Table 5 demonstrates how the progressive application of the proposed optimizations affects the objective measures. In terms of MCD, StreamSpeech (+HSM in Table 5) achieves the same result as the baseline. On the other hand, it improves the result of the baseline on the F_0 measure. The improvement occurs when we employ multi-band processing in the vocoder, since then the lowest band, which contains the base frequency, obtains a separate term in the loss function. The difference between the baseline and StreamSpeech in terms of CER is insignificant. [2]

In our last experiment (+NR in Table 5), we apply a threshold when sampling from the distributions generated by the vocoder to reduce the excessive noise [4]. The results show deterioration in the objective measures, which can be explained with the fact that the vocoder model is trained without applying the sampling threshold. However, subjective evaluations demonstrate the benefits of using the threshold in terms of noise reduction and reveal no perceptible quality degradation. Audio samples are available online. [3]

6. CONCLUSION

In this work, we propose several architectural modifications and optimizations of a complete TTS system. We demonstrate that

[2]Throughout the development of StreamSpeech several internal subjective evaluations have also demonstrated no degradation in speech quality.

[3]https://lml.bas.bg/~gshopov/tts-arch.html

the transformer-based FastSpeech 2 decoder can be replaced with a lightweight streamable convolutional decoder to achieve a dramatic latency reduction. Additionally, we significantly speedup the FastSpeech 2 encoder and the LPCNet vocoder. The resulting StreamSpeech architecture achieves low latency and faster than real time speech synthesis on resource-constrained devices without degradation in the quality of the produced speech. The architecture was implemented in the Bulgarian TTS engine NeuralSpeechLab which is widely used by the visually impaired people in Bulgaria. [4]

7. REFERENCES

[1] Adrian Łańcucki, "FastPitch: Parallel text-to-speech with pitch prediction," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6588–6592.

[2] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu, "FastSpeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations*, 2021.

[3] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj 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.

[4] Jean-Marc Valin and Jan Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5891–5895.

[5] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Jinzhu Li, Sheng Zhao, Enhong Chen, and Tie-Yan Liu, "LightSpeech: Lightweight and fast text to speech with neural architecture search," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5699–5703.

[6] Lukasz Kaiser, Aidan N. Gomez, and Francois Chollet, "Depthwise separable convolutions for neural machine translation," in *International Conference on Learning Representations*, 2018.

[7] Chengzhu Yu, Heng Lu, Na Hu, Meng Yu, Chao Weng, Kun Xu, Peng Liu, Deyi Tuo, Shiyin Kang, Guangzhi Lei, et al., "DurIAN: Duration informed attention network for multimodal synthesis," *arXiv preprint arXiv:1909.01700*, 2019.

[8] Qiao Tian, Zewang Zhang, Heng Lu, Ling-Hui Chen, and Shan Liu, "FeatherWave: An efficient high-fidelity neural vocoder with multi-band linear prediction," in *INTERSPEECH*, 2020, pp. 195–199.

[9] Yang Cui, Xi Wang, Lei He, and Frank K Soong, "An efficient subband linear prediction for LPCNet-based neural synthesis," in *INTERSPEECH*, 2020, pp. 3555–3559.

[4] The development of NeuralSpeechLab was commissioned by the Union of the Blind in Bulgaria.

[10] Hiroki Kanagawa and Yusuke Ijima, "Lightweight LPCNet-based neural vocoder with tensor decomposition," in *INTERSPEECH*, 2020, pp. 205–209.

[11] Jean-Marc Valin, Umut Isik, Paris Smaragdis, and Arvindh Krishnaswamy, "Neural speech synthesis on a shoestring: Improving the efficiency of LPCNet," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8437–8441.

[12] Zhiying Huang, Hao Li, and Ming Lei, "DeviceTTS: A small-footprint, fast, stable network for on-device text-to-speech," *arXiv preprint arXiv:2010.15311*, 2020.

[13] Sivanand Achanta, Albert Antony, Ladan Golipour, Jiangchuan Li, Tuomo Raitio, Ramya Rasipuram, Francesco Rossi, Jennifer Shi, Jaimin Upadhyay, David Winarsky, et al., "On-device neural speech synthesis," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 1155–1161.

[14] Qing He, Zhiping Xiu, Thilo Koehler, and Jilong Wu, "Multi-rate attention architecture for fast streamable text-to-speech spectrum modeling," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5689–5693.

[15] Chunyang Wu, Zhiping Xiu, Yangyang Shi, Ozlem Kalinli, Christian Fuegen, Thilo Koehler, and Qing He, "Transformer-based acoustic modeling for streaming speech synthesis," in *INTERSPEECH*, 2021, pp. 146–150.

[16] Vadim Popov, Stanislav Kamenev, Mikhail A Kudinov, Sergey Repyevsky, Tasnima Sadekova, Vitalii Bushaev, Vladimir Kryzhanovskiy, and Denis Parkhomenko, "Fast and lightweight on-device TTS with Tacotron 2 and LPCNet," in *INTERSPEECH*, 2020, pp. 220–224.

[17] Stoyan Mihov and Klaus U. Schulz, *Finite-State Techniques: Automata, Transducers and Bimachines*, Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2019.

[18] Haşim Sak, Françoise Beaufays, Kaisuke Nakajima, and Cyril Allauzen, "Language model verbalization for automatic speech recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8262–8266.

[19] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *INTERSPEECH*, 2020, pp. 5036–5040.

[20] Roktaek Lim, Yeongha Lee, Raehyun Kim, and Jaeyoung Choi, "An implementation of matrix–matrix multiplication on the Intel KNL processor with AVX-512," *Cluster Computing*, vol. 21, no. 4, pp. 1785–1795, 2018.

[21] Jonathan Lawrence Peyton, "Programming dense linear algebra kernels on vectorized architectures," M.S. thesis, The University of Tennessee, Knoxville, 2013.