Implementing a Text-to-Speech synthesis model on a Raspberry Pi for Industrial Applications

Ali Raheem Mandeel
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
alimandeel@tmit.bme.hu

Ammar Abdullah Aggar
Department of Comput. Engineering Ministry of Education
Baghdad, Iraq
ammar3ag@gmail.com

Mohammed Salah Al-Radhi
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
malradhi@tmit.bme.hu

Tamás Gábor Csapó
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
csapot@tmit.bme.hu

***Abstract*— Text-to-Speech (TTS) technology produces human-like speech from input text. It has recently acquired prominence by applying deep neural networks. Nowadays, end-to-end TTS models produce highly natural synthesized speech but require extremely high computational resources. Deploying such high-quality TTS models in a real-time environment has been a challenging problem due to the limited resources of embedding systems and cell phones. This paper demonstrated the implementation of an end-to-end TTS model (FastSpeech 2) in an embedded device (Raspberry Pi4 B+). The objective experimental results showed that the TTS model is compatible with the Raspberry Pi (RPi) with high-quality synthesized speech and acceptable performance in terms of processing speed. Our proposed model could be used in many real-life applications if used together with a mechanism for caching, such as railway announcements and industrial purposes.**

Keywords—Real-time system, speech synthesis, FastSpeech

I. INTRODUCTION

Speech is one of several forms of communication that can enhance cognitive processes when used in conjunction with others, such as visual and haptic forms of communication. In many application contexts, such as mobile environments, it is advantageous to augment the graphical user interface with audio output [1]. Moreover, output audio could be used as an alternative solution when the visual output is disabled at the human-computer interaction machines—for example, using voice Interactive voice response systems [2].

TTS aims to produce natural speech from the input text, which has been a challenging problem. Today, advanced end-to-end TTS models can produce high-quality synthesized speech with prosody, expressivity, and emotional content. Besides, it is possible to have speaker adaptation and conversion solutions with state-of-the-art TTS models [3] [4]. Although, it is a very computational cost process and requires high machine resources. It is reasonable to use small, low-resource embedded devices with TTS models for information communication to be invested in several industrial fields.

To the best of our knowledge, not many TTS systems have previously been implemented on extremely low-resource embedded devices for real-time applications. TTS systems have mainly experimented on computers and smartphones. We preferred Raspberry Pi (RPi) 4 over other options like the Nvidia Jetson Nano because it is cheaper and has sufficient computing power for the TTS application.

Our study aims to implement the state-of-the-art end-to-end TTS model (FastSpeech 2) on a low-resource embedded system (RPi). Also, it analyzes the computational cost and playback latency while keeping the quality of synthesized speech high.

The rest of this paper is structured as follows. Section II contains the related work and background. Section III then describes the design architecture of our suggested model for the RPi. Section IV depicts the practical experimental outcomes. Eventually, Section V includes conclusions.

II. RELATED WORK AND BACKGROUND

Many studies have been conducted to implement TTS models for different applications. In addition, RPi has been recently exploited in deep machine learning that operates for several purposes.

One system by Zainkó and his colleagues has been designed employing the TTS for making announcements at railway stations [5]. It uses the cache mechanism, which uses recorded prompts with "slot filling" of variable data. The synthesized speech had high intelligibility, and the model could be employed in real-time [5]. Another model proposed was to convert handwriting to output audio, which could be used to help blinds in reading. The RPi 3 device was used to capture an image of the handwritten text via a camera and then convert it to an audio file via the TTS engine. This approach needs an internet connection to use the TTS engine [6].

On the other hand, RPi 3 has been used to monitor agriculture using computer vision [7]. The images are gathered from multiple sensor nodes and sent to the RPi via wireless communication. After that, these images will be processed and classified by the computer vision model stored in Rpi, before being sent to the IoT cloud. In addition, RPi 4 model B+ 8 GB was proposed to detect traffic signs in real-time via using a deep learning model [8]. The images are gathered from the traffics via a RPi camera, which are then predicted by the model stored in RPi.

Overall, not many TTS models have been implemented on embedded devices such as the RPi for use in real-time applications.

A. FastSpeech 2

FastSpeech 2 is an end-to-end speech synthesis model that develops non-autoregressive mel-spectrograms from text [9]. It is faster than autoregressive models (e.g., Tacotron) in synthesizing speech and introduces more variation in speech characteristics (e.g., pitch, energy, and more accurate duration).

The variance adaptor adds various variance data, such as energy, duration, and pitch, to the hidden sequence after the encoder transforms the phoneme embedding sequence into the phoneme hidden sequence. Ultimately, the mel-spectrogram decoder simultaneously transforms the adapted hidden sequence into the mel-spectrogram sequence. Figure 1 shows the overall structure of FastSpeech 2.

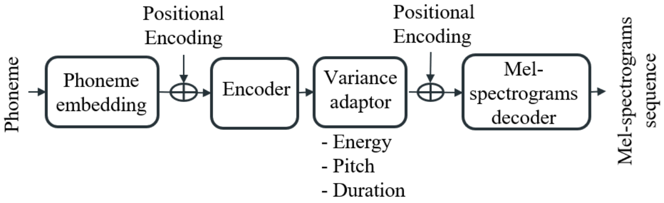


Fig.1. The overall FastSpeech 2 structure

B. NEURAL VOCODER: HIFI-GAN

HiFi-GAN is made up of one generator and two discriminators, namely multi-scale and multi-period [10]. They are trained adversarially, and two losses are employed to enhance training stability. HiFi-GAN produces adequate quality output speech 13.4 times faster than real-time on a CPU comparable autoregressive counterpart (WaveGlow and WaveNet) [10].

C. Raspberry Pi

RPi is a low-cost single-board computer developed by the RPi organization. Since its initial release in 2012, numerous versions of RPi computers have been released, which are classified into three main models: RPi Zero, A, and B, as shown in Figure 2. The compute module, a fourth variant, is primarily used in industrial applications. Moreover, the fundamentals of these three models are highly similar, featuring a system on a chip consisting of an integrated CPU (central processing unit) and on-chip graphics processing unit (GPU), a power input of 5 V DC, and onboard memory.

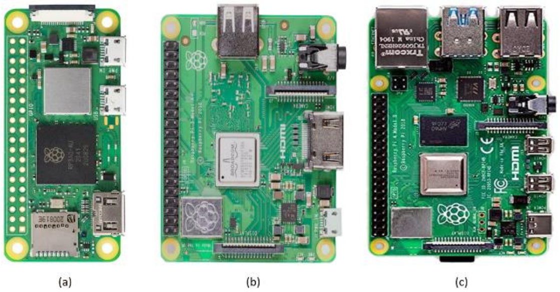


Fig. 2. (a) RPi Zero (a) RPi A (a) RPi Zero B

The RPi has the capability to perform all functions of a standard computer. Users can connect a screen, keyboard, and mouse to a RPi controlled by a Linux Desktop environment or traditional operating system without any additional settings.

Furthermore, the RPi 4 model has several features such as Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC 1.5GHz, 8 GB ram, 2 USB 3.0 ports, 2 USB 2.0 ports, Gigabit Ethernet RJ45 jack, a standard 40-pin GPIO (general-purpose input/output) header (fully backward-compatible with previous boards), a camera interface, two micro-HDMI ports, a micro-SD card slot, and a display interface.

III. Methods

We built the overall system, as illustrated in Figure 3. The process of building our system consists of two main steps: hardware setup and preparing the TTS model to be compatible with the low-resource embedded system. After that, we tested it by calculating the computational cost and objective metrics.

A. Hardware setup

We utilized a 64 GB SD card memory for storage. We installed the RPi OS (64-bit) desktop operating system on the RPi. Then, the MobaXterm toolbox has been used to set up a local connection between our RPi and a personal computer to download the TTS model and create the environment. We powered the RPi with 5v and connected an external speaker device to the RPi through a pole stereo audio.

B. Environment and TTS model setup

We used the open-source pre-trained FastSpeech2 model[1] (PyTorch implementation of Microsoft's text-to-speech system). FastSpeech2 was trained on LJSpeech dataset, which includes one female speaker (24 hours). We prepared the environment by installing all required Python libraries such as numpy, librosa, scipy, and torch. We downloaded the FastSpech2 model to the RPi. The FastSpeech's transformer encoder converts the phoneme sequence into a hidden sequence. After that, the mel-spectrogram decoder transforms the sequence into mel-spectrograms.

We used the pre-trained HiFi-GAN vocoder[2] to convert the predicted mel-spectrogram into speech waveform (of a sample rate of 22050 Hz). HiFi-GAN vocoder has two generator versions (V1 and V3). The first generator (V1) is designed to acquire higher speech naturalness, while the V3 generator is intended to run faster. In our proposed system design, we used the V1 generator to obtain high-quality synthesized speech.

[1] https://github.com/ming024/FastSpeech2
[2] https://github.com/jik876/hifi-gan

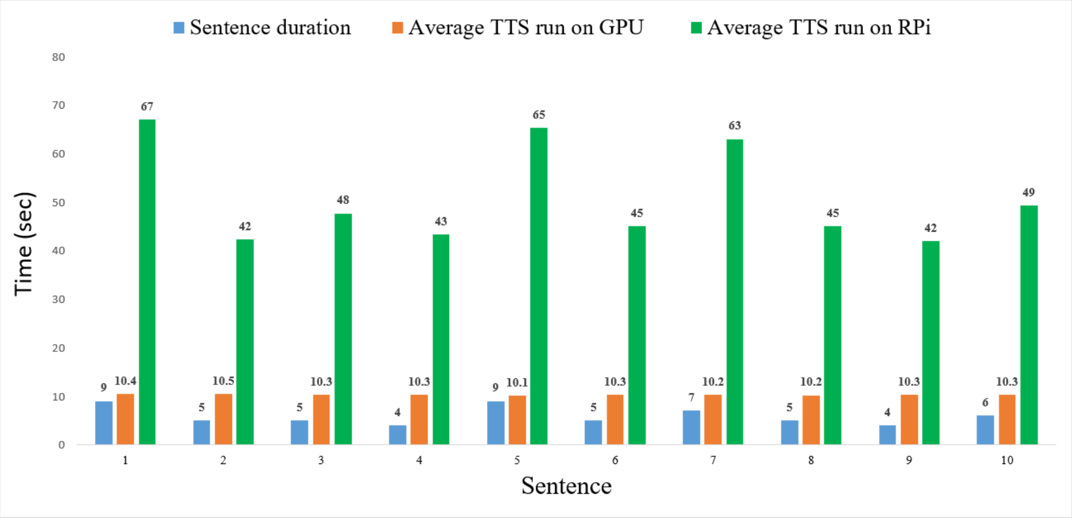


Fig. 4. AVERAGE RUNTIME.

.

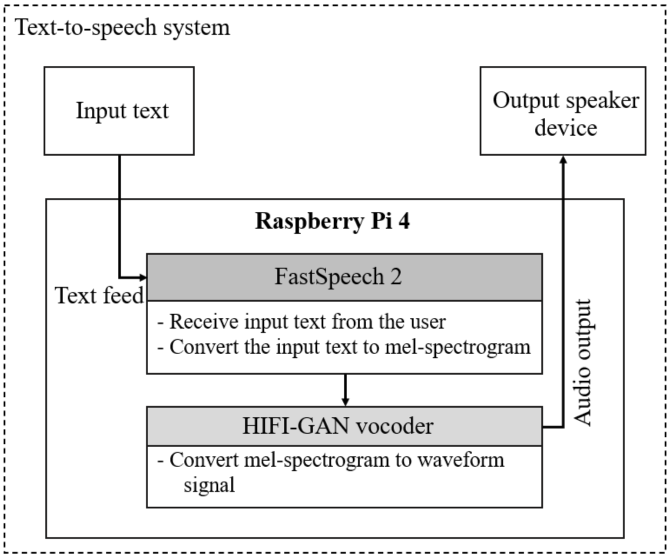

Text-to-speech system
Input text
Output speaker device
Raspberry Pi 4
Text feed
FastSpeech 2
- Receive input text from the user
- Convert the input text to mel-spectrogram
HIFI-GAN vocoder
- Convert mel-spectrogram to waveform signal
Audio output

Fig. 3. System Overview

IV. RESULTS

We conducted experiments to assess the effectiveness of our suggested solution in a close-to-real-time setting.

A. *Computational Cost*

The computational performance of the TTS model on RPi was measured with several waveform lengths. The TTS model was timed while creating individual utterances, without batching, on the RPi. The Real-Time Factor (RTF) was obtained by dividing the entire time for synthesizing a sentence by the total duration required to produce it. The test set has ten sentences (samples) and waveforms durations from 4.0 to 9.0 seconds.

TABLE I. TIME CONSUMED TO RUN EACH MODEL ON TEN SENTENCES

Sentence	Sentence duration (sec)	TTS run on GPU (sec)			TTS run on RPi (sec)		
		1st run	2nd run	3rd run	1st run	2nd run	3rd run
1	9	10.6	10.5	10.24	77	62	62
2	5	10.2	10.3	10.9	42	42	43
3	5	10.2	10.4	10.24	48	48	47
4	4	10.3	10.2	10.28	43	44	43
5	9	10.1	10.2	10.09	65	66	65
6	5	10.0	10.9	10.14	45	45	45
7	7	10.1	10.3	10.29	63	63	63
8	5	10.3	10.2	10.12	45	45	45
9	4	10.3	10.4	10.27	44	41	41
10	6	10.4	10.2	10.44	50	49	49

We ran the measurement three times for each sample and calculated the average of all sentence values (see Table 1 and Figure 4). It was determined that the time spent loading models from the file system was not crucial for the system under evaluation. The RTF of the Raspberry TTS model was compared to the RTF of a TTS model on a high-performance NVidia Titan X graphics processing unit (GPU). We found out that the RTF of the GPU-based TTS model is 1.9, while the RTF of the RPi-based TTS model is 8.93. We can observe that the TTS model on the RPi generates waveforms slower than the GPU-based TTS model, but it is still runs at an acceptable speed.

B. *Objective Evaluation*

We conducted objective measurements of synthesized speech quality and intelligibility to assess the quality of the proposed system. We utilized normalized covariance metric (NCM) and frequency-weighted segmental SNR (fwSNRseg). The measurements were done frame-by-frame, and the outcomes were averaged over the synthesized utterances for the female speaker. We compared the results of the TTS model on the RPi to those of the TTS model on the GPU.

- Normalized Covariance Metric (NCM):

It accurately predicts the intelligibility of noise-corrupted speech with non-linear distortions [11]. It is based on the Speech Transmission Index (STI) [12], which considers the Hilbert envelope's covariance coefficient r between the original and synthetic speech signals (see Equation 1). W is the weight vector given to the STI of K bands, which may be calculated using the articulation index.

$$NCM = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\sum_{i=1}^{K} W_{i,j}\cdot\log\frac{r_{i,j}^2}{1-r_{i.j}^2}}{\sum_{i=1}^{K} W_{i,j}}\right) \qquad (1)$$

- frequency-weighted segmental SNR (fwSNRseg):

It calculates the segmental SNR per spectral band and then collects the weighed SNRs from the whole bands (see Equation 2) [13]. $X^2_{i,j}$ and $Y^2_{i,j}$ are the critical-band magnitude spectra of the original and synthesized speech signals in the j^th frequency band, W is the weight vector, and K is the bands' number.

$$fwSNR_{\text{seg}} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\sum_{i=1}^{K} W_{i,j}\cdot\log\frac{X_{i,j}^2}{X_{i,j}^2-Y_{i,j}^2}}{\sum_{i=1}^{K} W_{i,j}}\right) \qquad (2)$$

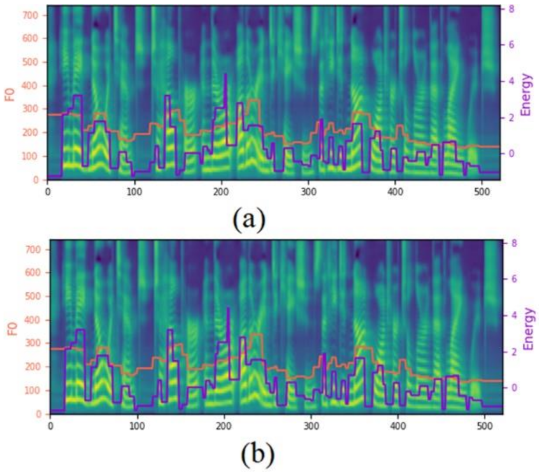
We measured two metrics for ten sentences, and averaged them as shown in Table 1. As a result, temporal envelope-based techniques were advantageous for modeling the noise component, and the proposed system tends to show significantly high-quality speech synthesis among all two metrics – i.e., the synthesized speech quality is not different when inference is run on RPi vs on GPU.

TABLE II. THE OBJECTIVE EVALUATION

System type	**The objective metrics**	
	NCM	***fwSNRseg***
RPi TTS model	0.03	0.857
GPU TTS model	0.03	0.857

C. *Demonstration sample (spectrogram)*

Figure 5 depicts the spectrograms of synthesized speech by the FastSpeech2 model on two machines. The spectrograms are for the sentence “He made no attempt to help her and there are other indications that he did not want her to learn that language.”. We can notice that FastSpeech2 works similarly on both the high-resources GPU machine and the other machine – similarly to Sec IV.B, there is no significant difference.



(a)

(b)

Fig. 3. The spectrogram plot of two synthesized sounds (a) the synthesized speech of the RPi TTS model (b) the synthesized speech of the GPU TTS model

V. CONCLUSIONS AND FUTURE WORK

Small low-resource devices with limited capabilities, particularly portable devices, are likely to be utilized in infocommunications systems. Speech synthesis can be invested in small appliances as the preliminary communication pipeline. In this study, the FastSpeech 2 end-to-end TTS model was implemented on an embedded system (Raspberry Pi4 B+). Objective experiments have been done to assess the performance of the proposed system. The outcomes were encouraging, indicating that the proposed system's performance is adequate for standard audio-enabled industrial applications, when extended with a caching mechanism. However, the TTS model computation needs to be improved further. The footprint size (910 Megabytes) is also adequate, although it should be reduced to preserve memory space. To talk about such implementation details, we are happy to invite colleagues experienced in speech technology and/or embedded systems to the WINS 2023 workshop for a fruitful discussion.

We will investigate HifiGAN's other versions, like V2 and V3. Moreover, we will implement several TTS models and compare their performance on RPi. A caching mechanism will also be considered.

ACKNOWLEDGMENT

The research was partially funded by the National Research, Development and Innovation Office of Hungary (FK 142163 grant). T.G.Cs. was supported by the Bolyai János Research Fellowship of the Hungarian Academy of Sciences and by the ÚNKP-22-5-BME-316 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund. The Titan X GPU used was donated by NVIDIA.

REFERENCES

[1] M. Pleva et al., “Speech and Mobile Technologies for Cognitive Communication and Information Systems”, In: CogInfoCom 2011 : Proceedings of the 2nd International Conference on Cognitive Infocommunications: 7. - 9.7.2011, Budapest. - Budapest : University of Technology and Economics, 2011 pp. 1-5.

[2] C. B. Dickson, “Text to Speech Interactive Voice Response System,” The Journal of the Acoustical Society of America, vol. 130, no. 2, pp. 1087, 2011, doi: 10.1121/1.3625678.

[3] A. R. Mandeel, M. S. Al-Radhi, and T. G. Csapó, “Investigations on speaker adaptation using a continuous vocoder within recurrent neural network based text-to-speech synthesis,” Multimedia Tools and Applications, Oct. 2022, doi: 10.1007/s11042-022-14005-5.

[4] M. S. Al-Radhi, T. G. Csapó, and G. Németh, “Continuous vocoder applied in deep neural network based voice conversion,” Multimedia Tools and Applications, vol. 78, no. 23, pp. 33549–33572, Sep. 2019, doi: 10.1007/s11042-019-08198-5.

[5] C. Zainkó, M. Bartalis, G. Németh, and G. Olaszy, “A polyglot domain optimised text-to-speech system for railway station announcements,” Interspeech 2015, Sep. 2015, doi: 10.21437/interspeech.2015-311.

[6] P. V. N. Reddy, “Text to Speech Conversion Using Raspberry-Pi for Embedded System,” International Journal of Innovative Research in Science, Engineering and Technology, vol. 1, no. 1, pp. 144–148, Jan. 2012, doi: 10.15680/ijirset.2012.0101019.

[7] R. Kamath, M. Balachandra, and S. Prabhu, “Raspberry Pi as Visual Sensor Nodes in Precision Agriculture: A Study,” IEEE Access, vol. 7, pp. 45110–45122, 2019, doi: 10.1109/access.2019.2908846.

[8] Aggar, Ammar & Rahem, Abd Alrazak. (2023). An Automatic Traffic Signs Detection and Recognition System Based on Raspberry Pi 4. 10.13140/RG.2.2.24267.87848.

[9] Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “Fastspeech 2: Fast and high-quality end-to-end text to speech,” arXiv:2006.0455, 2020

[10] Kong, J. Kim, J. Bae, ”HiFi-GAN: generative adversarial networks for efficient and high fidelity speech synthesis,” International Conference on Neural Information Processing Systems (NeurIPS), pp. 17022–17033, 2020.

[11] J. Ma, Y. Hu, and P. C. Loizou, “Objective measures for predicting speech intelligibility in noisy conditions based on new band-importance functions,” The Journal of the Acoustical Society of America, vol. 125, no. 5, pp. 3387, 2009, doi: 10.1121/1.3097493.

[12] H. J. M. Steeneken and T. Houtgast, “A physical method for measuring speech‐transmission quality,” The Journal of the Acoustical Society of America, vol. 67, no. 1, pp. 318‐326, Jan. 1980, doi: 10.1121/1.384464.

[13] Y. Hu and P. C. Loizou, “Evaluation of objective measures for speech enhancement,” Interspeech 2006, Sep. 2006, doi: 10.21437/interspeech.2006-84.