Leading University
Department of Computer Science and Engineering

CSE 4801

End-to-End Pitch-controlled TTS

Zaki Rezwana Chowdhury	1812020051
Humayun Kibria Shakib	1812020121

Department of Computer Science and Engineering

Supervisor

Prithwiraj Bhattacharjee

Lecturer

Department of Computer Science and Engineering

23rd March, 2022

End-to-End Pitch-controlled TTS

A Thesis submitted to the

Department of Computer Science and Engineering

Leading University

Sylhet - 3112, Bangladesh

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

By

Zaki Rezwana Chowdhury	1812020051
Humayun Kibria Shakib	1812020121

Supervisor

Prithwiraj Bhattacharjee

Lecturer

Department of Computer Science and Engineering

23rd March, 2022

Recommendation Letter from the Thesis Supervisor

The project entitled *End-to-End Pitch-controlled TTS* submitted by the students

1. Zaki Rezwana Chowdhury, 1812020051
2. Humayun Kibria Shakib, 1812020121

is a record of research work carried out under my supervision and I, hereby, approve that the report be submitted in partial fulfillment of the requirements for the award of their Bachelor Degrees.

Signature of the Supervisor:

Prithwiraj Bhattacharjee

Lecturer

Date: 23rd March, 2022

Certificate of Acceptance of the Thesis

The thesis entitled *End-to-End Pitch-controlled TTS*
submitted by the students

1. Zaki Rezwana Chowdhury, 1812020051
2. Humayun Kibria Shakib, 1812020121

on 23rd March, 2022
is, hereby, accepted as the partial fulfillment of the requirements for the award of their Bachelor Degrees.

Head of the Dept.	**Chairman, Exam. Committee**	**Supervisor**
Shafkat Kibria	Shafkat Kibria	Prithwiraj Bhattacharjee
Assistant Professor & Head	Assistant Professor	Lecturer
Department of Computer Science and Engineering	Department of Computer Science and Engineering	Department of Computer Science and Engineering

Abstract

Over the past few years, End-to-End Text-to-Speech (TTS) has made significant progress. While end-to-end text-to-speech (TTS) models enabling single-stage training and parallel sampling have been proposed, their sample quality lacks that of two-stage TTS systems. More of it, generating speech with fine-grained prosody control still becomes an open challenge. In this work, we present two parallel end-to-end TTS methods with pitch-control that generate more natural sounding audio than current two-stage models. For efficient and high-fidelity speech synthesis we use a vocoder. As speech audio consists of sinusoidal signals with various periods, vocoder enhances the sample quality. Our model has been trained with a phonetically balanced 13 hours of single speaker speech data. It has obtained 4.47 and 4.45 Mean Opinion Score (MOS) on scale 5.0 as a subjective evaluation 1.26 and 1.09 PESQ as an objective evaluation on scale [-0.5, 4.5]. among all existing on-commercial Bangla TTS systems it is outperforming the basis of naturalness.

Index Terms— Text To Speech; End to End; MOS; FastPitch; VITS; Tacotron; HIFI-GAN;

Acknowledgements

We would like to thank the Department of Computer Science and Engineering, Leading University, Sylhet 3112, Bangladesh, for supporting this course work. We are indebted to our supervisors Arif Ahmad and Prithwiraj Bhattacharjee for providing invaluable feedback and guidance on our analysis and framing, at times responding to queries late at night and early in the morning.

Dedication

We would like to dedicate our work to our parents, relatives and friends.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Ever since modern science came to the surface, the scientific revolution is growing so rapidly that we rely on science everyday. Science is deeply interwoven with humankind that we use science in every field. Speech is one of the most vital forms of communication in our everyday life. Since speech is a primary mode of communication among human beings, it is natural for people to expect to be able to carry out spoken dialogue with computers. This involves the integration of speech technology and language technology. Speech synthesis is the automatic generation of artificial speech forms by the computer. Text-to-Speech is a system that generates speech from given text. In the past couple of years some great work has been done on the TTS system. But for a better TTS system there are many things that should be improved. If we look at a language specific TTS system then it certainly has a vast area to work on. Bangla language is considered as the fifth-most spoken language in the world with speakers around 265 million. So, technology like the TTS system in Bangla should be there with no argument. Being approached in the traditional way, very few significant Bangla TTSs are available today which is not enough. For this reason we have come forward to implement the Bangla TTS system by using the current state of the art. There are different types of TTS approaches. TTS systems are mainly classified into two major parts: Concatenative TTS and Statistical Parametric Speech Synthesis (SPSS) TTS. Conventional ways of implementing TTS systems are concatenative based. There are mainly three types of concatenative approach. First one is unit selection based TTS in which a large database of pre-recorded speech

is needed. The second one is Diphone synthesis. Though it requires minimal use of the database, for any specific language it needs all the sound to sound transitions, which we call the diphone. The third one is domain specific synthesis. Domain-specific TTS system is implemented through concatenating words when diversity of texts is limited. SPSS has mainly two sub-classes. The first one is the Hidden Markov Model (HMM) based TTS system and the second one is Deep Neural Network based TTS system. Frontend processing, duration and acoustic modeling, vocoder are the major components in traditional Deep Neural Network based TTS systems. A new technology named end-to-end TTS synthesis has emerged recently. This type of Neural network based TTS system doesnÃ¢t need a huge pre-recorded speech as well as does not need any hand engineered pre-processing and only requires 3 major building blocks.

1.2 Motivation

What motivations thrives us to do research on Bangla TTS systems? In Leading University(LU), Bangla based research is always being inspired. Definitely Bangla TTS system is a part of Bangla research. Some research has already been done on the Bangla TTS system in Bangladesh. So, we continued to develop a better TTS system by doing research based on the latest TTS technology. Secondly, the motivation for choosing TTS is that it actually helps physically impaired people like sightless and mute people. Moreover TTS systems are being used in many airports for airing the schedules of flights. Besides, modern business also relies on TTS nowadays for communication purposes. These things inspired us to contribute in TTS field and specially in our mother language Bangla. Thirdly, what are the reasons for choosing end to end speech synthesis to implement our Bangla TTS system? SPSS methods worked better than concatenative methods because of having a small footprint. But they require a huge amount of human annotation for language-specific text processing. To solve these issues, modern researchers prefer to develop end-to-end TTS systems. Here speech is generated directly from (text, speech) pairs without any preprocessing. Based on this technology giant tech companies are building TTS systems. So, this thinking motivated us to continue with end to end speech synthesis in Bangla.

Chapter 2

Background Study

TTS researches have made a great advancement for the last few decades. Different types of new approaches have been introduced to the research community. TTS systems are now coming from language specific to language independent platforms. Now, we are going to briefly describe TTS research starting from a concatenative approach. Then how SPSS were introduced and implemented for better efficiency. Finally, we will mention the current state-of-art which is end to end speech synthesis system and will notify researches so far done in different languages using this method.

2.1 Review Previous Work

There are very few works available based on Bangla TTS. The first attempt of implementing Bangla TTS system was concatenative based. It was a unit selection based TTS system named Katha [1] developed by Firoz Alam et. al. from BRAC University in 2007. Festival [2] toolkit was used for frontend preprocessing in this TTS system. In 2009, a diphone concatenation based TTS system Subachan [3] was developed by Abu Naser et. al. from Shahjalal University of Science and Technology (SUST).

After the introduction of SPSS based TTS systems, a significant improvement had been done on TTS research. Hidden Markov Model based [4] systems became common then. One of the popular implementations of HMM based TTS was HTS [5] developed in the early 2000s. In 2014,

for Bangla an HMM-based SPSS system [6] was also developed. But over smoothness of acoustic modeling and limitations of vocoders resulted in producing less natural output in HMM-based system [7].

When Deep neural network (DNN) [8] is introduced for acoustic modeling in SPSS [7] [9], a great enhancement occurred in TTS researches. Neural networks had been used for acoustic modeling since 1990 [10]. But the availability of dataset and higher computational power have helped the system giving impressive results [11].

Networks like recurrent neural networks (RNNs) [7], long short-term memory RNNs (LSTM-RNNs) [12], and deep bidirectional LSTM-RNNs (BLSTM-RNNs) [13] significantly improved the performance of SPSS systems. Although till now no significant works have been done to implement DNN-based Bangla SPSS. In 2016, Google [14] released a commercial TTS but it is not open source. But Google has released some language based resources[15] for Bangla TTS so that Bangla research community can start a research based on these resources.

Many eminent researchers and institutions shared many open source tools for TTS research so that TTS researchers can be benefited. Such a TTS system is Idlak Tangle [16]. It was developed through an open source speech recognition system named Kaldi [17]. Another DNN based open source tool is Merlin [18]. Edinburgh University developed this tool with frontend processing toolkit Ossian [19] and Festival. For synthesizing speech from acoustic parameters WORLD [20] vocoder is used. Under open source licenses these tools are freely available.

SPSS methods are better and preferred over concatenative approaches. But they need a huge amount of human annotation. That means they need language specific text processing. So, for resolving this problem researchers are now interested to develop end to end TTS systems. Hence speech is directly generated only from (text, speech) pairs. No preprocessing is needed for this system. Currently tech companies which are now in top position in the world are working with end to end speech synthesis.

For the last couple of years, there are some great significant works have already been done. Some of them are Deep Voice 1 [21], 2 [22], 3 [23] from Baidu, Char2Wav [24] from MILA, FastSpeech [25] from Microsoft etc. Google has also released some TTS research based papers named WaveNet [26], Tacotron [27], and Tacotron 2 [28].

It is obvious that Commercial companies do not make their project open source for all. But many individual researchers have implemented these systems and those systems are available in online. So, those can be a good starting point for new TTS researchers for starting their research.

By observing all of the cases we adopted some good implementations of open source tools released by [29] based on Google's paper Tacotron [27] for our end to end Bangla TTS system.

Chapter 3

Data Collection and Pre-processing

The first step of building a good TTS system is to collect a large amount of speech data along with corresponding transcriptions. But there is not so much of speech data available for low resourced language like Bangla. Google has released 3 hours of speech data [15] for Bangla TTS research. Brac university has released 13 hours of speech data [30] for TTS system. And this dataset is standard for any TTS research for Bangla. We then made it compatible with our model by doing some processing.

3.1 Data Source

In this section we will focus firstly on the checking of the text corpus then their corresponding speech.

3.1.1 Text Corpus

For developing phonetically balanced Bangla text corpus we consulted the literature from [31–33]. Then from various domain we collected the text data. We made it sure that our corpus have all possible pronunciations for Bangla. Our final dataset contains more than 10700 utterences. A preview of our dataset is presented in table 3.1

Total sentences	10701
Total words	14, 13, 369
Total unique words	16, 498
Minimum words in a sentence	4
Maximum words in a sentence	19
Average words in a sentence	8.96
Total duration of speech (hours)	13 : 09 : 19
Average duration of each sentence (seconds)	5.29

Table 3.1: Summary of Dataset

3.1.2 Speech Data

All the recordings were stored in the .wav file format. The sample rate of wave files are 48KHz. The duration of collected speech is around 13 hours.

3.2 Data Pre-processing

3.2.1 Data Cleaning

As we implemented VITS and FastPitch [34] for our Bangla TTS, we had to do some pre-processing to make the dataset compatible with our Model. So, at first we removed all the sentences containing less than or equal to 3 words and greater than or equal to 12 words and also removed their corresponding speech. Then we removed the silence from the starting and ending part from all the .wav files for better learning in training. Thus we made the dataset compatible with our system. We convert stereo audio files to mono audio files. We also splitted big sentences and made it less than 15s long. To meet our training criteria we normalize the amplitude/ volume of the files.

3.2.2 Text Normalization

Text normalization is the process of converting raw text to normalized text[1]. Before a text is ready to undergo the process of TTS, it must first be pre-processed to remove the ambiguities

[1]Normalized text means pronounceable form of text

and convert some Non Standard Words (NSWs) into their standard word pronunciations. TTS systems work with text in everyday use (real text). A text must be pre-processed to remove the ambiguities. Even Non standard words have to convert into their standard pronunciation through text normalization. We had to normalize our data before passing to the training part. So, our training data were fully text normalized. Numerical words ambiguity, abbreviations, etc. issues of text normalization were applied to the dataset part for more accurate pronunciation.

Chapter 4

Methodology

In this section, we explain our proposed methods and the architecture of them. We have worked on both VITS-TTS with FastPitch TTS.[35] VITS-TTS is a fully end-to-end text-to-speech system that needs a conditional VAE formulation; alignment estimation derived from variational inference; adversarial training for improving synthesis quality. On the other hand, FastPitch requires a vocoder. FastPitch is a feed-forward model based on FastSpeech[36] that improves the quality of synthesized speech. Combined with HiFi-GAN [37] the model learns to predict and use pitch in a low resolution of one value for every input symbol, it makes it easy to adjust pitch interactively, enabling practical applications in pitch editing. We will describe the methodology weâve used for our TTS system and will also mention why we chose the current one in that part.

4.1 Architecture of VITS

4.1.1 Variational Inference

VITS can be expressed as a conditional VAE with the objective of maximizing the variational lower bound, also called the evidence lower bound (ELBO)[1][2], of the intractable marginal log-likelihood of data. The training loss then becomes the negative ELBO, which can be viewed as the sum of reconstruction loss and KL divergence.

[1]Source-code: https://github.com/jaywalnut310/vits
[2]Demo: https://jaywalnut310.github.io/vits-demo/index.html

4.1.1.1 Reconstruction Loss

As a target data point in the reconstruction loss, we use a mel-spectrogram instead of a raw waveform. We upsample the latent variables to the waveform domain through a decoder and transform to the mel-spectrogram domain. Then the loss between the predicted and target mel-spectrogram is used as the reconstruction loss. This can be viewed as maximum likelihood estimation assuming a Laplace distribution for the data distribution and ignoring constant terms. We define the reconstruction loss in the mel-spectrogram domain to improve the perceptual quality by using a mel-scale that approximates the response of the human auditory system. Note that the mel-spectrogram estimation from a raw waveform does not require trainable parameters as it only uses STFT and linear projection onto the mel-scale. Furthermore, the estimation is only employed during training, not inference. In practice, we do not upsample the whole latent variables but use partial sequences as an input for the decoder, which is the windowed generator training used for efficient end-to-end training.

4.1.1.2 KL-Divergence

The input condition of the prior encoder is composed of phonemes text extracted from text and an alignment A between phonemes and latent variables. The alignment is a hard monotonic attention matrix with dimensions representing how long each input phoneme expands to be time-aligned with the target speech. Because there are no ground truth labels for the alignment, we must estimate the alignment at each training iteration. In our problem setting, we aim to provide more high-resolution information for the posterior encoder. We, therefore, use the linear-scale spectrogram of target speech as input rather than the mel-spectrogram[38]. The factorized normal distribution is used to parameterize our prior and posterior encoders. We found that increasing the expressiveness of the prior distribution is important for generating realistic samples.

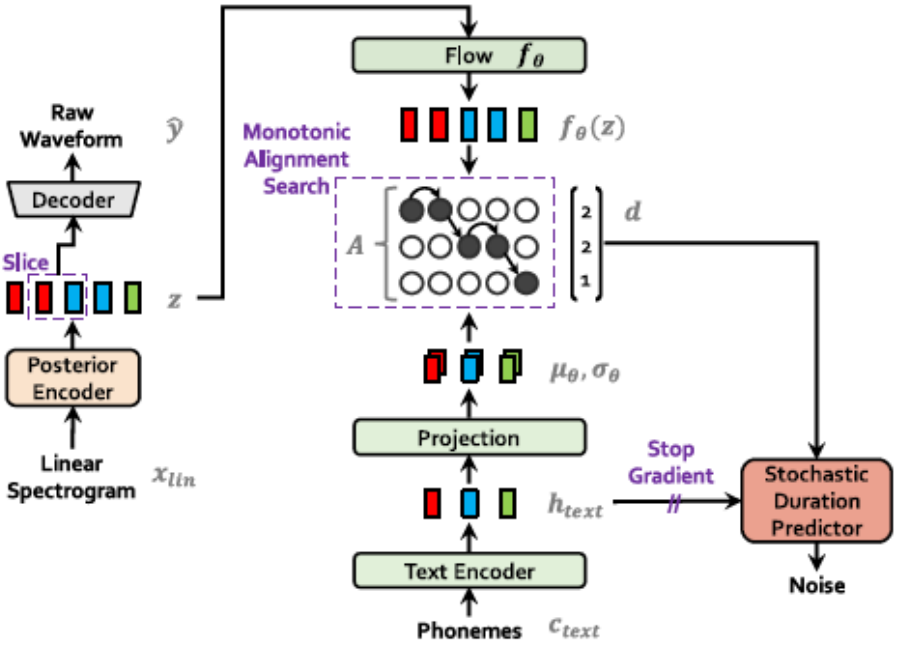

Figure 4.1: The architecture of VITS TTS.

4.1.2 Alignment Estimation

4.1.2.1 Monotonic Alignment Search

To estimate an alignment A between input text and target speech, we adopt Monotonic Alignment Search, a method to search an alignment that maximizes the likelihood of data parameterized by a normalizing flow where the candidate alignments are restricted to be monotonic and non-skipping following the fact that humans read text in order without skipping any words. To find the optimum alignment,[39] use dynamic programming. Applying MAS directly in our setting is difficult because our objective is the ELBO, not the exact log-likelihood. We, therefore, redefine MAS to find an alignment that maximizes the ELBO, which reduces to finding an alignment that maximizes the log-likelihood of the latent variables where the candidate alignments are restricted to be monotonic and non-skipping following the fact that humans read

4.1.2.2 Duration Prediction from Text

We can calculate the duration of each input token by summing all the columns in each row of the estimated alignment. The duration could be used to train a deterministic duration predictor, as proposed in previous work [28], but it cannot express the way a person utters at different speaking rates each time. To generate human-like rhythms of speech, we design a stochastic duration predictor so that its samples follow the duration distribution of given phonemes. The stochastic duration predictor is a flow-based generative model that is typically trained via maximum likelihood estimation. The direct application of maximum likelihood estimation, however, is difficult because the duration of each input phoneme is 1) a discrete integer, which needs to be equantized for using continuous normalizing flows, and 2) a scalar, which prevents high-dimensional transformation due to invertibility. We apply variational equantization and variational data augmentation to solve these problems. To be specific, we introduce two random variables, which have the same time resolution and dimension as that of the duration sequence, for variational dequatization and variational data augmentation, respectively. We sample the two variables through an approximate posterior distribution. The resulting objective is a variational lower bound of the log-likelihood of the phoneme duration. The training loss is then the negative variational lower bound. We apply the stop gradient operator, which prevents back-propagating the gradient of inputs, to the input conditions so that the training of the duration predictor does not affect that of other modules. The sampling procedure is relatively simple; the phoneme duration is sampled from random noise through the inverse transformation of the stochastic duration predictor, and then it is converted to integers.

4.1.3 Adversarial Training

To adopt adversarial training in our learning system, we add a discriminator that distinguishes between the output generated by the decoder and the ground truth waveform. In this work, we use two types of loss successfully applied in speech synthesis; the least-squares loss function for adversarial training, and the additional feature matching loss for training the generator.

4.1.4 Model Architecture

The overall architecture of the proposed model consists of a posterior encoder, prior encoder, decoder, discriminator, and stochastic duration predictor. The posterior encoder and discriminator are only used for training, not for inference.

4.1.4.1 Prior Encoder

The prior encoder consists of a text encoder that processes the input phonemes text and normalizing that improves the flexibility of the prior distribution. The text encoder is a transformer encoder that uses relative positional representation instead of absolute positional encoding. We can obtain the hidden representation text from text through the text encoder and a linear projection layer above the text encoder that produces the mean and variance used for constructing the prior distribution. The normalizing flow is a stack of affine coupling layers consisting of a stack of WaveNet residual blocks. For simplicity, we design the normalizing flow to be a volume-preserving transformation with the Jacobian determinant of one.

4.1.4.2 Decoder

The decoder is essentially the HiFi-GAN generator[37]. It is composed of a stack of transposed convolutions, each of which is followed by a multireceptive field fusion module (MRF). The output of the MRF is the sum of the output of residual blocks that have different receptive field sizes.

4.1.5 Discriminator

We follow the discriminator architecture of the multi-period discriminator proposed in HiFi-GAN.[37] The multi-period discriminator is a mixture of Markovian window-based sub-discriminators, each of which operates on different periodic patterns of input waveforms.

4.1.6 Model Configurations

In this section, we mainly describe the parts of VITS[34] as we followed configurations of HiFi-GAN[37] for several parts of our model: we use transformer encoder and residual blocks as those of HiFi-GAN.

4.1.7 Prior Encoder and Posterior Encoder

The normalizing flow in the prior encoder is a stack of four affine coupling layers, each coupling layer consisting of four HiFi-GAN residual blocks. As we restrict the affine coupling layers to be volume-preserving transformations, the coupling layers do not produce scale parameters. The posterior encoder, consisting of 16 HiFi residual blocks, takes linear-scale log magnitude spectrograms and produces latent variables with 192 channels[37].

4.1.8 Decoder and Discriminator

The input of our decoder is latent variables generated from the prior or posterior encoders, so the input channel size of the decoder is 192. For the last convolutional layer of the decoder, we remove a bias parameter, as it causes unstable gradient scales during mixed precision training. For the discriminator, HiFi-GAN uses the multi-period discriminator containing five sub-discriminators with periods [2; 3; 5; 7; 11] and the multi-scale discriminator containing three sub-discriminators. To improve training efficiency, we leave only the first sub-discriminator of the multi-scale discriminator that operates on raw waveforms and discard two sub-discriminators operating on average-pooled waveforms. The resultant discriminator can be seen as the multi-period discriminator with periods [1; 2; 3; 5; 7; 11].

4.1.8.1 Stochastic Duration Predictor

The stochastic duration predictor estimates the distribution of phoneme duration from a conditional input text. For the efficient parameterization of the stochastic duration predictor, we stack residual blocks with dilated and depth-separable convolutional layers. We also apply neural spline flows which take the form of invertible nonlinear transformations by using monotonic rational-quadratic splines, to coupling layers. Neural spline flows improve transformation expressiveness with a similar number of parameters compared to commonly used affine coupling layers. For the multi-speaker setting, we add a linear layer that transforms speaker embedding and add it to the input text.

4.2 Architecture of FastPitch

4.2.1 Model Description

The architecture of FastPitch is shown in Figure 4.3. It is based on FastSpeech and composed mainly of two feed-forward Transformer (FFTr) stacks . The first one operates in the resolution of input tokens, the second one in the resolution of the output frames. Let x = (x1; : : : ; xn) be the sequence of input lexical units, and y = (y1; : : : ; yt) be the sequence of target mel-scale spectrogram frames. The first FFTr stack produces the hidden representation h = FFTr(x). The hidden representation h is used to make predictions about the duration and average pitch of every character with a 1-D CNN.

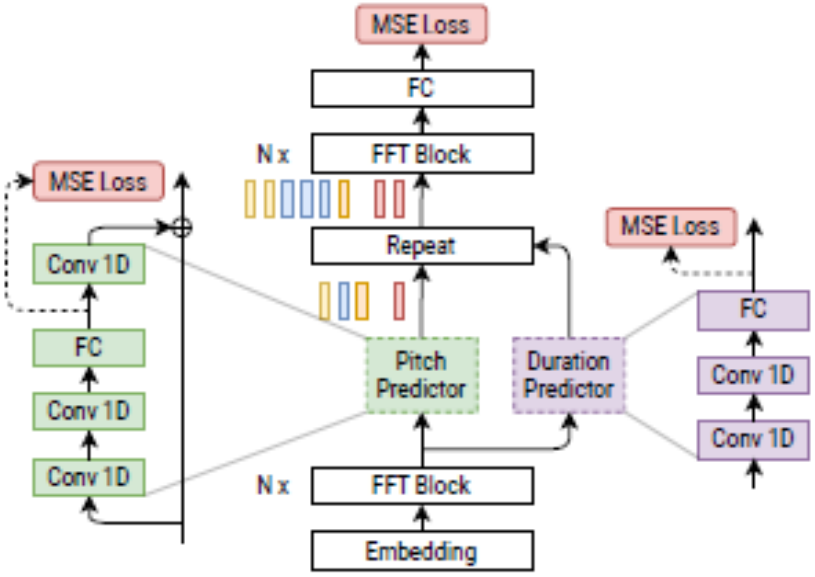

Figure 4.2: Architecture of FastPitch follows FastSpeech.

4.2.2 Duration of Input Symbols

Durations of input symbols are estimated with a Tacotron-2 model trained on LJSpeech. Tacotron-2 has a single attention matrix, we do not need to choose between attention heads, as it would be necessary with a multi-head Transformer model. FastPitch is robust to the quality

of alignments. We observe that durations extracted with distinct Tacotron-2 models tend to differ, where the longest durations have approximately the same locations, but may be assigned to different characters. Surprisingly, those different alignment models produce FastPitch models, which synthesize speech of similar quality.

4.2.3 Pitch of Input Symbols

We obtain ground truth pitch values through acoustic periodicity detection using the accurate autocorrelation method. The windowed signal is calculated using Hann windows. The algorithm finds an array of maxima of the normalized autocorrelation function, which become the candidate frequencies. The lowest-cost path through the array of candidates is calculated with the Viterbi algorithm. The path minimizes the transitions between the candidate frequencies. We set windows size to match the resolution of training mel-spectrograms, to get one value for every frame. we tried averaging to three pitch values per every symbol, in the hope to capture the beginning, middle and ending pitch for every symbol. However, the model was judged inferior.

4.3 Architecture of Tacotron-2

4.3.1 Model Description

Tacotron-2 is a neural network architecture for speech synthesis directly from text. It consists of two components - a recurrent sequence-to-sequence feature prediction network with attention which predicts a sequence of mel spectrogram frames from an input character sequence, a modified version of WaveNet which generates time-domain waveform samples conditioned on the predicted mel spectrogram frames. In contrast to the original Tacotron, Tacotron-2 uses simpler building blocks in the encoder and decoder.

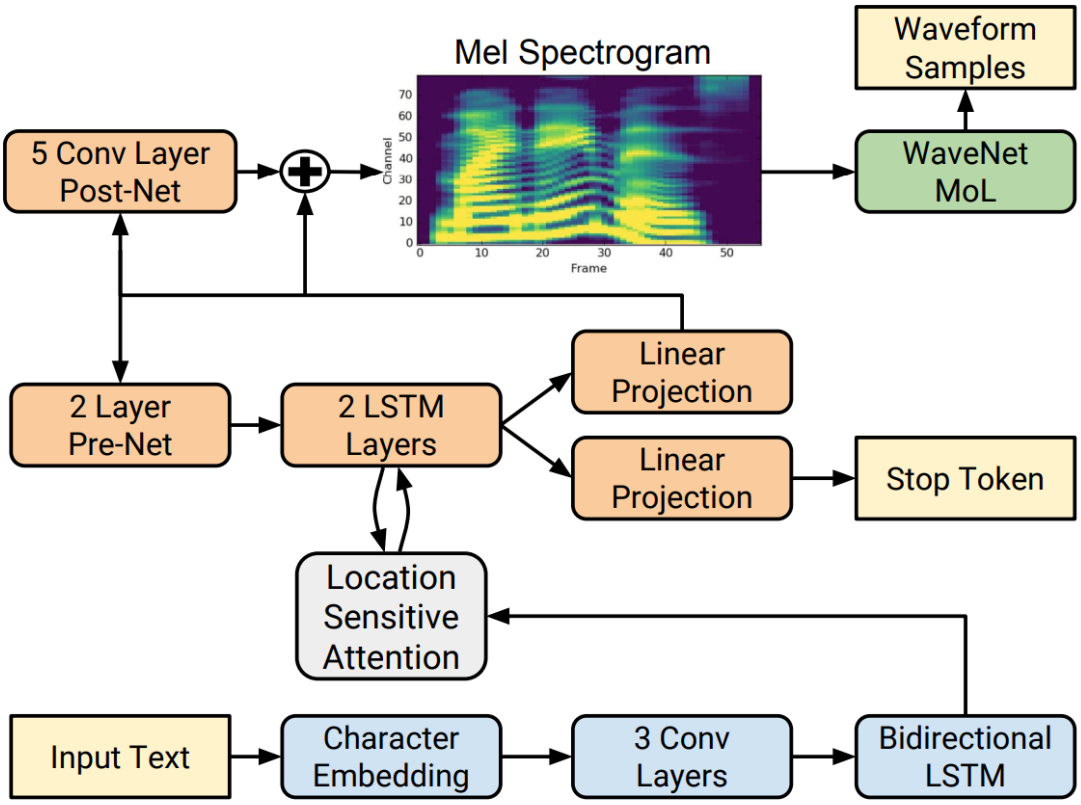

Figure 4.3: Architecture of Tacotron-2

Tacotron combines a sequence-to-sequence recurrent network with attention to predicted mel spectrograms with a HiFi-GAN vocoder. The resulting system synthesizes speech with Tacotron-level prosody and HiFi-GAN level audio quality.

4.4 HiFi-GAN

HifiGAN is a neural vocoder model for text-to-speech applications. It is intended as the second part of a two-stage speech synthesis pipeline, with a mel-spectrogram generator such as FastPitch as the first stage.

4.4.1 Model Architecture

HifiGAN is a neural vocoder based on a generative adversarial network framework, During training, the model uses a powerful discriminator consisting of small sub-discriminators, each one focusing on specific periodic parts of a raw waveform. The generator is very fast and has a small footprint, while producing high quality speech.

4.4.2 How we used it?

We used it as the second part of Tacotron-2 and FastPitch Text-to-Speech synthesis pipeline. Where Hifi-GAN takes a mel spectrogram and returns audio.

Chapter 5

Result and Analysis

There are mainly two types of evaluation for measuring the performance of a TTS system. One is objective evaluation and the other is subjective evaluation. Now, we are going to describe each of them briefly. Finally, we will present our numerical results from these evaluations and graphical comparisons among our TTS system and some currently existing Bangla TTS systems.

5.1 Objective Evaluation

What is objective evaluation? Objective evaluation is nothing but a mathematical comparison of the generated signal and original signal. For objective evaluation, we have accepted the Perceptual Evaluation of Speech Quality (PESQ) [40] measurement.

5.1.1 PESQ

There are different types of PESQ measurement. Among them raw-PESQ and MOS-LQO are popular measurement. The ranges of raw-PESQ and MOS-LQO are [-0.5,4.5] and [1.0, 5.0] respectively. We have specifically choosen raw-PESQ score for our model performance evaluation. Two waveforms are required for measuring the PESQ score. One is original waveform and another is the generated one as mentioned earlier in the definition. Below we will mention how the experiment has been done.

5.1.2 PESQ Calculation

For calculating the PESQ score, we took 100 random sentences and their original waveforms from the test dataset. Then we synthesized waveforms from that 100 sentences using our TTS system. Then the original and generated waveforms of corresponding sentences were sent to the PESQ system pair wisely. The figure 5.3 shows the PESQ score of every 100 generated waveforms from the system.

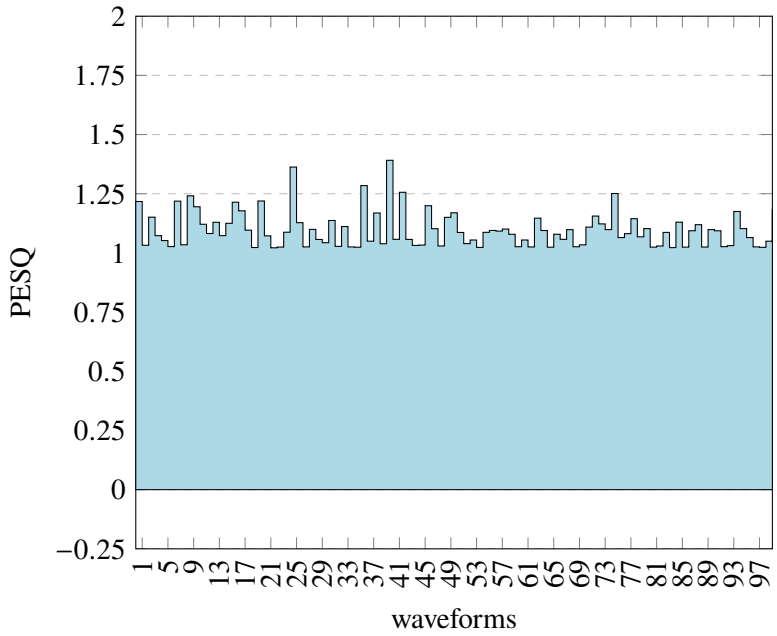

Figure 5.1: PESQ score of 100 synthesized wave in VITS-TTS

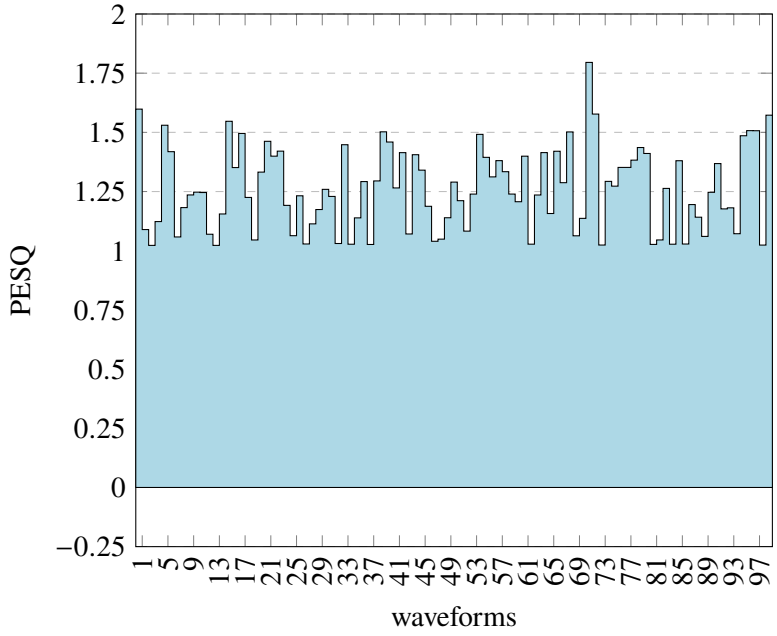

Figure 5.2: PESQ score of 100 synthesized wave in FastPitch-TTS

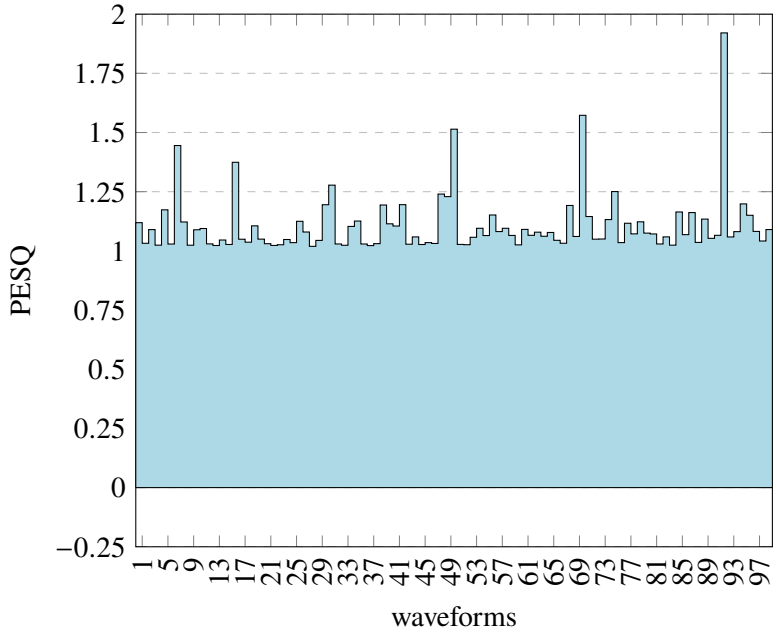

Figure 5.3: PESQ score of 100 synthesized wave in Tacotron-2 TTS

Here, X-axis denotes waveforms and Y-axis denotes the score for each waveform. We calculated the raw PESQ for every stimuli. The calculation is done by a standard program for raw PESQ. Finally, we have taken the average value of the PESQ scores of 100 waveforms and received **1.09** on Vits-TTS, **1.26** on FastPitch-TTS and **1.11** on Tacotron 2 PESQ.

5.2 Subjective Evaluation

When human judgement is involved, it is called subjective evaluation. Several techniques were introduced for subjective evaluation but most popular one for TTS system is the Mean Opinion Score (MOS) [41] and we have also adopted MOS as subjective evaluation.

5.2.1 MOS

Actually, the naturalness of TTS system is calculated through MOS. For calculating the MOS, five labels are given from which individual listener has to choose one for each synthetic waveform after listening. The five labels are Excellent, Good, Fair, Poor and Bad. These labels are mapped into pre-defined numbers. The mapping between label and number is given in table 5.1.

Label	**Rating**
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

Table 5.1: Label vs Rating

After that the arithmetic mean of those numbers is taken which we declare as MOS. Individual rating is integer type but the final result can be a real value also. The equation of calculating MOS is given below:

$$MOS = \frac{1}{N}\sum_{n=1}^{N} R_n \tag{5.1}$$

5.2.2 MOS Calculation

For finding MOS, we conducted a crowdsourcing. 41 volunteers, native speakers of Bangla from different backgrounds participated to rate the waveforms. Then, we provided 2 waveforms generated by our TTS system. The 2 waveforms are available at https://hkshakib.github.io/tts-mos.github.io/. They put a rating (rating details described in the table 5.1) to each of the 2 waveforms on the basis of naturalness. So, we received 41 ratings for each of the 2 waveforms. We calculated average rating or formally MOS for each of the 2 waveforms using the formula 5.1. The final MOS is equal to the arithmetic mean of MOS of 2 waveforms. The figure 5.6 denotes the MOS of 2 waveforms.

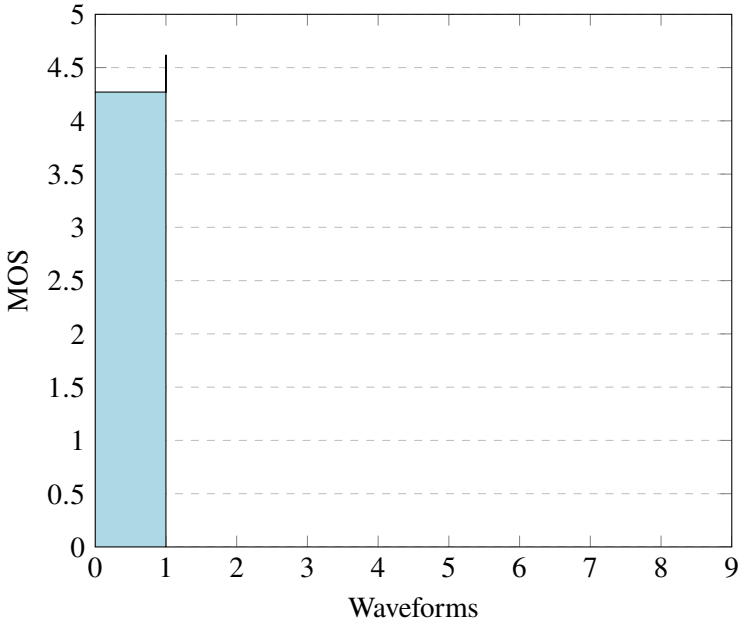

Figure 5.4: MOS scores of 2 waveforms, VITS-TTS

$$MOS = \frac{4.27 + 4.62}{2} \tag{5.2}$$

$$= 4.45 \tag{5.3}$$

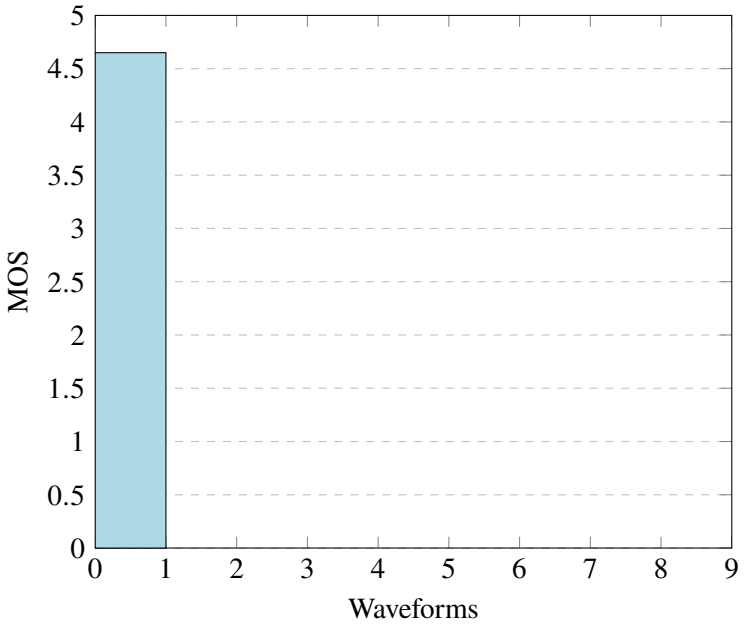

Figure 5.5: MOS scores of 2 waveforms, FastPitch-TTS

$$MOS = \frac{4.65 + 4.29}{2} \tag{5.4}$$

$$= 4.47 \tag{5.5}$$

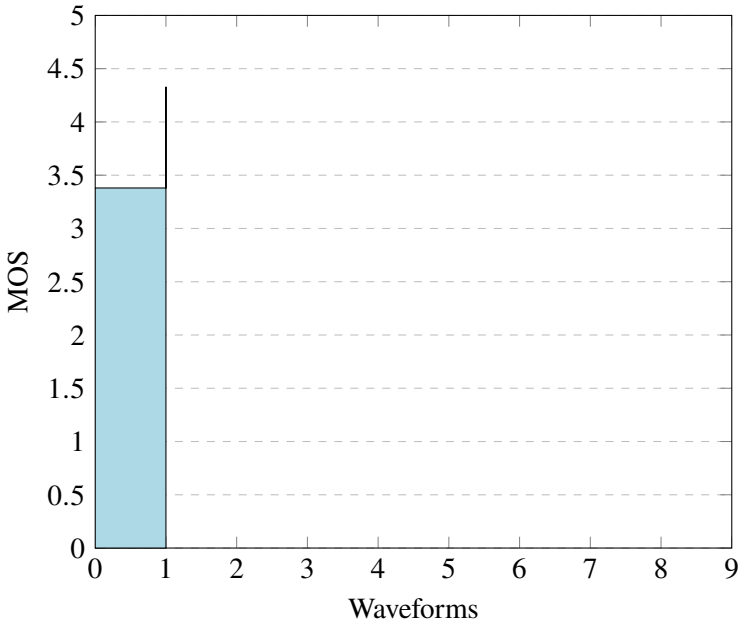

Figure 5.6: MOS scores of 2 waveforms, Tacotron-2 TTS

$$MOS = \frac{3.38 + 4.33}{2} \tag{5.6}$$

$$= 3.86 \tag{5.7}$$

5.3 Discussion and Analysis

In this section, we are going to make a comparison among various existing Bangla TTS systems. We have already calculated the PESQ Score and MOS for our system. We have also calculated both PESQ scores and Mean Opinion Scores for Tacotron-2 TTS, FastPitch TTS, VITS TTS and Tacotron TTS.

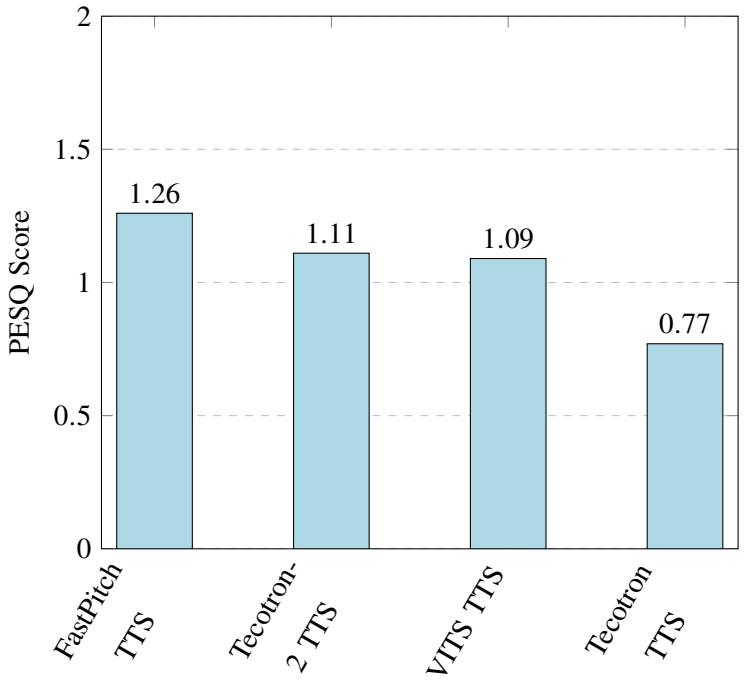

Figure 5.7: Comparison of PESQ Scores of Different TTS Systems

A bar chart in the figure 5.7 is showing a comparison of PESQ scores for different systems mentioned above. We find that the Tecotron[42] has obtained **0.77** PESQ and the FastPitch, Techotron-2 and VITS TTS which we developed have achieved **1.26**, **1.11** and **1.09** PESQ score which is obviously better than Tecotron TTS.

Another comparison of various TTS system is shown in figure 5.8 with respect to Mean Opinion Score (MOS).

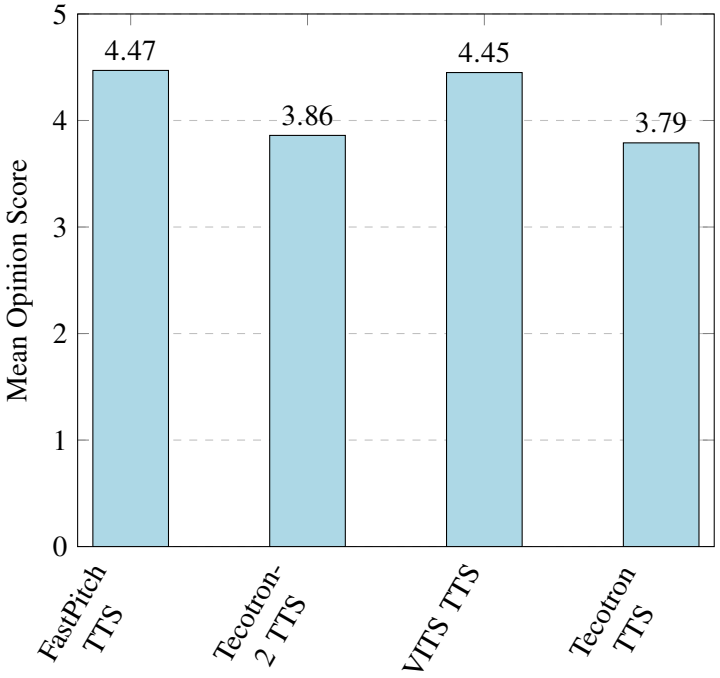

Figure 5.8: Comparison of MOS Scores of Different TTS Systems

We have achieved **4.47** MOS for FastPitch, **3.86** for Tecotron-2 and **4.45** MOS for VITS TTS which is better than previously implemented systems like Tacotron TTS. Generally, PESQ score and MOS of a good TTS system follow same alignment. Interestingly, our PESQ score also correlates with MOS.

Chapter 6

Conclusion

In this work, we proposed two parallel TTS systems, VITS and FastPitch. VITS, that can learn and generate in an end-to-end manner. We further introduced the resulting system synthesizes natural sounding speech waveforms directly from text, without having to go through predefined intermediate speech representations. Our experimental results show that our method outperforms two-stage TTS systems and achieves close to human quality. We hope the proposed method will be used in many speech synthesis tasks, where two-stage TTS systems have been used, to achieve performance improvement and enjoy the simplified training procedure. FastPitch, a parallel text-to-speech model based on FastSpeech, able to rapidly synthesize high-fidelity mel-scale spectrograms with a high degree of control over the prosody. The model demonstrates how conditioning on prosodic information can significantly improve the convergence and quality of synthesized speech in a feed-forward model, enabling more coherent pronunciation across its independent outputs, and lead to state-of-the-art results. Our pitch conditioning method is simpler than many of the approaches known from the literature. It does not introduce an overhead, and opens up possibilities for practical applications in adjusting the prosody interactively, as the model is fast, highly expressive, and presents potential for multi-speaker scenarios.

References

[1] F. Alam, P. K. Nath, and M. Khan, "Text-to-speech for bangla language using festival," 2007.

[2] "The festival speech synthesis system," accessed: 2019-07-28. [Online]. Available: http://www.cstr.ed.ac.uk/projects/festival/

[3] A. Naser, D. Aich, and M. R. Amin, "Implementation of subachan: Bengali text-to-speech synthesis software," in *International Conference on Electrical & Computer Engineering (ICECE 2010)*. IEEE, 2010, pp. 574–577.

[4] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[5] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The hmm-based speech synthesis system (hts) version 2.0." in *SSW*. Citeseer, 2007, pp. 294–299.

[6] S. Mukherjee and S. K. D. Mandal, "A bengali hmm based speech synthesis system," *arXiv preprint arXiv:1406.3915*, 2014.

[7] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *2013 ieee international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 7962–7966.

[8] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[9] Z.-H. Ling, S.-Y. Kang, H. Zen, A. Senior, M. Schuster, X.-J. Qian, H. M. Meng, and L. Deng, "Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and tuture trends," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 35–52, 2015.

[10] T. Weijters and J. Thole, "Speech synthesis with artificial neural networks," in *IEEE International Conference on Neural Networks*. IEEE, 1993, pp. 1764–1769.

[11] O. Watts, G. E. Henter, T. Merritt, Z. Wu, and S. King, "From hmms to dnns: where do the improvements come from?" in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5505–5509.

[12] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of deep neural network (dnn) for parametric tts synthesis," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3829–3833.

[13] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[14] A. Gutkin, L. Ha, M. Jansche, K. Pipatsrisawat, and R. Sproat, "Tts for low resource languages: A bangla synthesizer," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016, pp. 2005–2010.

[15] "Google international language resources," accessed: 2019-07-28. [Online]. Available: https://github.com/google/language-resources/blob/master/bn/festvox/phonology.json

[16] B. Potard, M. P. Aylett, D. A. Baude, and P. Motlicek, "Idlak tangle: An open source kaldi based parametric speech synthesiser based on dnn." in *INTERSPEECH*, 2016, pp. 2293–2297.

[17] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[18] Z. Wu, O. Watts, and S. King, "Merlin: An open source neural network speech synthesis system." in *SSW*, 2016, pp. 202–207.

[19] "Ossian: A simple language independent text to speech front-end," accessed: 2019-07-28. [Online]. Available: https://github.com/CSTR-Edinburgh/Ossian

[20] M. Morise, F. Yokomori, and K. Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[21] S. Ö. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman *et al.*, "Deep voice: Real-time neural text-to-speech," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 195–204.

[22] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Advances in neural information processing systems*, 2017, pp. 2962–2970.

[23] 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," *arXiv preprint arXiv:1710.07654*, 2017.

[24] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," 2017.

[25] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," *arXiv preprint arXiv:1905.09263*, 2019.

[26] 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.

[27] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[28] 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 *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[29] "A tensorflow implementation of google's tacotron speech synthesis with pre-trained model (unofficial)," accessed: 2020-01-30. [Online]. Available: https://github.com/keithito/tacotron

[30] F. Alam, S. M. Habib, D. A. Sultana, and M. Khan, "Development of annotated bangla speech corpora," in *Spoken Languages Technologies for Under-Resourced Languages*, 2010.

[31] P.-E. Honnet, A. Lazaridis, P. N. Garner, and J. Yamagishi, "The siwis french speech synthesis database? design and recording of a high quality french database for speech synthesis," Idiap, Tech. Rep., 2017.

[32] R. Sonobe, S. Takamichi, and H. Saruwatari, "Jsut corpus: free large-scale japanese speech corpus for end-to-end speech synthesis," *arXiv preprint arXiv:1711.00354*, 2017.

[33] L. Gabdrakhmanov, R. Garaev, and E. Razinkov, "Ruslan: Russian spoken language corpus for speech synthesis," *arXiv preprint arXiv:1906.11645*, 2019.

[34] 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*. PMLR, 2021, pp. 5530–5540.

[35] A. 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.

[36] 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," *arXiv preprint arXiv:2006.04558*, 2020.

[37] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 022–17 033, 2020.

[38] 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," *Advances in neural information processing systems*, vol. 32, 2019.

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

[40] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.

[41] M. Viswanathan and M. Viswanathan, "Measuring speech quality for text-to-speech systems: development and assessment of a modified mean opinion score (mos) scale," *Computer Speech & Language*, vol. 19, no. 1, pp. 55–83, 2005.

[42] P. Bhattacharjee, R. S. Raju, A. Ahmad, and M. S. Rahman, "End-to-end bangla speech synthesis," in *2021 International Conference on Science & Contemporary Technologies (ICSCT)*. IEEE, 2021, pp. 1–6.