A Literature review on Text-to-Speech synthesis on Indic Languages

September 3, 2023

1 Abstract

Text to speech (TTS), also known as speech synthesis, is a popular study area in the speech, language and machine learning communities and has many practical applications in business. The quality of synthetic speech has substantially increased in recent years thanks to neural network-based TTS and the advancements in deep learning and artificial intelligence.In this survey, we evaluate the choice of acoustic models, vocoders, supplementary loss functions, training schedules . We further summarize resources related to TTS (e.g., datasets, opensource implementations).[1]

2 Introduction

Text-to-speech (TTS) systems have made quick progress thanks to deep neural networks. With less human feature engineering required than more conventional techniques like formant, concatenative, and statistical parametric speech synthesis, neural TTS may produce high-fidelity real-time speech synthesis . This has made it possible to create high-quality synthetic speech, which is now being used in an increasing variety of applications. .[**?**]

The three main parts of a TTS system are a text analysis module, an acoustic model, and a vocoder, which transforms linguistic data into acoustic features and voice waveforms, respectively. However, when it comes to synthesizing speech in Indian languages, there's still a lot to explore and understand. This is mainly because Indian languages are incredibly diverse, resources are limited, and we haven't fully utilized the latest advancements in neural TTS technology.

The challenges in this field are significant. Indian languages have complex sounds and rhythms, making it tricky to create accurate TTS systems. There's also the issue of capturing the natural melody and tone of speech, which can vary widely. Additionally, the lack of available data and the many different dialects and regional differences further complicate the task. Some Indian languages are

tonal, where the pitch of a word can change its meaning, adding another layer of complexity. In this review paper, we conducted a comprehensive review of neural network techniques that have demonstrated promising outcomes when applied to Indic languages.

In the upcoming subsections, we will be giving a brief overview of the evolution of TTS technologies. Following that, we will introduce some essential aspects of neural TTS.[2]

2.1 Approaches in Text-to-Speech (TTS) Technology

2.1.1 Concatenative synthesis

Concatenative synthesis, a method used in Text-to-Speech (TTS) technology, involves combining pieces of pre-recorded speech stored in a database. Typically, this database contains various speech units, ranging from full sentences to smaller parts like syllables, recorded by voice actors. During TTS, the system searches this database to find and piece together the right speech units for the given input text, producing a speech waveform.

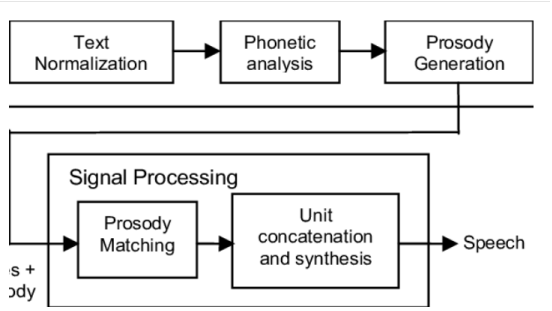


Figure 1: Block diagram of the Concatenative speech synthesis.

Concatenative TTS generally excels in producing speech that is highly intelligible and closely resembles the original voice actor's timbre. However, there are some drawbacks. It requires an extensive database to cover all possible combinations of speech units, making it resource-intensive. Additionally, the generated voice may sound less natural and emotional due to potential discontinuities in stress, emotion, and prosody caused by the concatenation process.

2.1.2 Statistical Parametric Speech Synthesis

Instead of creating speech by stringing together pre-recorded sounds, statistical parametric speech synthesis (SPSS) starts by generating the essential speech parameters and then uses algorithms to turn those parameters into speech. SPSS typically has three main parts: a module to understand the text, a module to

predict speech parameters (like pitch and tone), and a module to turn these parameters into spoken words.

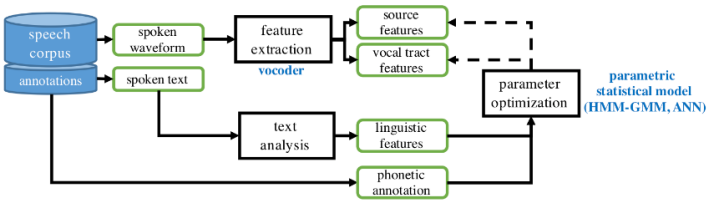


Figure 2: Block diagram of the SPSS system.

The text analysis module processes the text by making it easier to work with, like converting written words into the sounds they represent. Then, it figures out important details like which sounds to make, how long each sound should be, and what parts of speech are involved.

The acoustic models, often based on things like hidden Markov models, learn from examples of both the linguistic features (like which sounds to make) and the acoustic features (like how to make those sounds). These features are extracted from recorded speech using a vocoder.

Finally, **The vocoder** takes the predicted acoustic features and turns them into spoken words. SPSS has some benefits over older speech synthesis methods: it sounds more natural, it's easier to adjust how it sounds, and it doesn't require as much recorded data. However, there are downsides too: sometimes the speech doesn't sound very clear, and it can still sound a bit robotic.

Hidden Markov Models (HMM)

HMM-based synthesis is a statistical parametric speech synthesis method based on Hidden Markov Models (HMM). In this technique, the frequency spectrum (vocal tract), fundamental frequency (vocal source), and duration (prosody) of speech are modelled simultaneously by HMMs. Speech waveforms are generated from HMMs themselves based on the maximum likelihood criterion . It consists of two phases, the training phase and the synthesis phase. The spectrum and excitation parameters are extracted from the speech database and modelled by context-dependent HMMs in the training phase. A model usually consists of three states that represent the beginning, the middle and the end of the phone. The synthesis phase deals with the generation of speech signals by concatenating the context-dependent HMMs according to the text to be synthesized.

The major advantages of the HMM-based speech synthesis approach are

its voice characteristics can be easily modified and can be applied to various languages with little modification. A variety of speaking styles or emotional speech can be synthesized using the small amount of speech data. Also the techniques developed in Automatic Speech Recognition (ASR) can also be easily applied with less effort and its footprint is relatively small.[3]

Neural Speech Synthesis

Neural Speech Synthesis, also known as neural TTS, is a technology that uses deep neural networks to create synthetic speech. In the past, we used methods like Hidden Markov Models (HMM) for speech synthesis, but neural TTS replaced them. One of the early neural models was WaveNet, which was a significant advancement. It could directly produce speech waveforms from text. Other models like DeepVoice 1/2 improved upon traditional speech synthesis components with neural networks.

Later, some models simplified the process even further. They took character or phoneme sequences as input and used mel-spectrograms to represent speech features. These models are called end-to-end models because they generate speech directly from text. Examples include Tacotron 1/2, Deep Voice 3, and FastSpeech 1/2.

Then, there are fully end-to-end TTS systems like ClariNet, FastSpeech 2, and EATS. These systems generate speech waveforms straight from text without relying on previous synthesis methods. The key advantages of neural network-based speech synthesis are that it produces high-quality and natural-sounding voices and requires less manual preprocessing and feature development compared to older synthesis techniques.

3 DNN approaches of TTS(Acoustic Models)

3.1 DeepVoice 3

Deep Voice 3, a fully-convolutional sequence-to-sequence architecture tailored for Text-to-Speech (TTS) applications, as depicted in Fig. The architecture boasts the capability to seamlessly convert a diverse array of textual features, such as characters, phonemes, and stresses, into a wide spectrum of vocoder parameters. These encompass mel-band spectrograms, linear-scale log magnitude spectrograms, fundamental frequency, spectral envelope, and aperiodicity parameters. These vocoder parameters, in turn, serve as invaluable inputs for audio waveform synthesis models.

The Deep Voice 3 architecture comprises three essential components:

Encoder: At its core, we employ a fully-convolutional encoder. Its primary role is to transform textual features into a finely-honed internal representation.

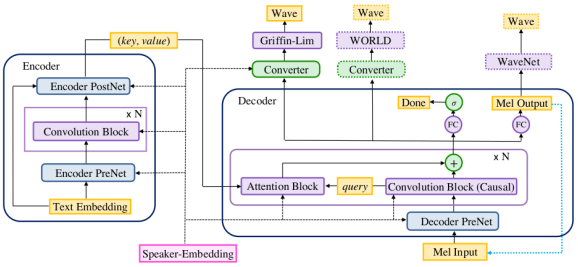


Figure 3: Block diagram of DeepVoice 3

Decoder: Complementing the encoder, we incorporate a fully-convolutional causal decoder. This decoder, equipped with a multi-hop convolutional attention mechanism, orchestrates the conversion of the acquired learned representation into a compact, low-dimensional audio representation—specifically, mel-scale spectrograms. This transformation is executed in an autoregressive fashion.

Converter: Our architecture features a fully-convolutional post-processing network known as the converter. Unlike the decoder, the converter operates in a non-causal manner, affording it the ability to leverage future context information. Its primary task is to predict the final vocoder parameters, a choice dependent on the specific vocoder in use.

The overarching objective function that we seek to optimize is a linear combination of the losses emanating from both the decoder and the converter.To facilitate effective learning, we adopt a multi-task training approach. This separation of the decoder and converter streamlines attention learning, with the loss for mel-spectrogram prediction playing a pivotal role. The attention mechanism is honed through the gradients stemming from mel-spectrogram prediction, alongside vocoder parameter prediction.[4]

3.2 Tacotron2

Tacotron 2 is a neural network architecture for text to speech that uses a recurrent sequence-to-sequence feature prediction that maps the text character embeddings to the mel-scale spectrograms. To sum that down Tacotron takes text as an input and uses Natural Language Processing tools to convert it to character embedded sequences and then thrown into a recurrent sequence-to-sequence feature that predicts a sequence of mel spectrum frames. The mel spectrum is a visual way of looking at audio data in a time-frequency domain with the bins representing pitch classes. The model also generates a time-domain waveform that is taken from the mel spectrum using a modified version of an architecture known as WaveNet. This is achieved by doing a inverse Fourier Transform which is a function that transforms data from the time-frequence domain to the

time-power domain. Having the 2 different acoustic representations allows us to train the two separately.

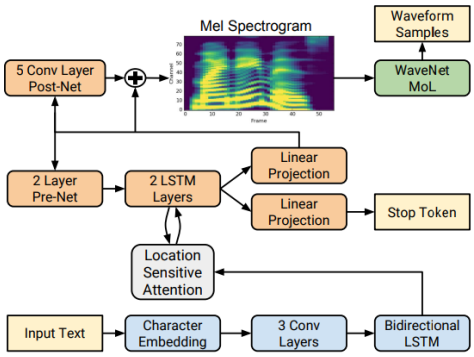


Figure 4: Block diagram of the Tacotron 2 system architecture.

In Tacotron the mel spectrograms are computed through a short-time Fourier transform using a 50ms frame size and a Hann window function. The Hann window function is a smoothing technique that averages the frequencies to produce and visualize a smoother and more bearable waveform. The audio data is scaled to a range between 125 Hz to 7.6 kHz which is the range of the human voice and a Log compression is also put in place to limit the dynamic range or how loud and quiet the audio data can be. This filtering will make it easier for the network to analyze and predict the audio as well as save processing power. The 2 acoustic representations are then fed into a neural network that is composed of an encoder and a decoder.

The encoder converts a character sequence into a hidden feature representation and the decoder takes that to predict a spectrogram while the decoder in an autoregressive recurrent neural network which predicts a mel spectrogram from the encoded input sequence one frame at a time.[5]

Wavenet Vocoder

Tacotron2 uses a modified version of the WaveNet architecture from [8] to invert the mel spectrogram feature representation into time-domain waveform samples. As in the original architecture, there are 30 dilated convolution layers, grouped into 3 dilation cycles, i.e., the dilation rate of layer k (k = 0 . . . 29) is 2k (mod 10). To work with the 12.5 ms frame hop of the spectrogram frames, only 2 upsampling layers are used in the conditioning stack instead of 3 layers. Instead of predicting discretized buckets with a softmax layer, we follow PixelCNN++

and Parallel WaveNet and use a 10-component mixture of logistic distributions (MoL) to generate 16-bit samples at 24 kHz. To compute the logistic mixture distribution, the WaveNet stack output is passed through a ReLU activation followed by a linear projection to predict parameters (mean, log scale, mixture weight) for each mixture component. The loss is computed as the negative log-likelihood of the ground truth sample.

Evaluation

Below shows a comparison of our method against various prior systems. In order to better isolate the effect of using mel spectrograms as features, we compare to a WaveNet conditioned on linguistic features[8] with similar modifications to the WaveNet architecture as introduced above. We also compare to the original Tacotron that predicts linear spectrograms and uses Griffin-Lim to synthesize audio, as well as concatenative and parametric baseline systems, both of which have been used in production at Google. We find that a system was proposed significantly outperforms all other TTS systems, and results in an MOS comparable to that of the ground truth audio.

System	MOS
Parametric	3.492 ± 0.096
Tacotron (Griffin-Lim)	4.001 ± 0.087
Concatenative	4.166 ± 0.091
WaveNet (Linguistic)	4.341 ± 0.051
Ground truth	4.582 ± 0.053
Tacotron 2	4.526 ± 0.066

Table 1: Mean Opinion Score (MOS) evaluations with 95% confidence intervals

3.3 FastSpeech2

The overall model architecture of FastSpeech 2 is shown in Figure. The encoder converts the phoneme embedding sequence into the phoneme hidden sequence, and then the variance adaptor adds different variance information such as duration, pitch and energy into the hidden sequence, finally the mel-spectrogram decoder converts the adapted hidden sequence into mel-spectrogram sequence in parallel. We use the feed-forward Transformer block, which is a stack of selfattention layer and 1D-convolution as in FastSpeech, as the basic structure for the encoder and mel-spectrogram decoder. Different from FastSpeech that relies on a teacher-student distillation pipeline and the phoneme duration from a teacher

model, FastSpeech 2 makes several improvements. First, we remove the teacher-student distillation pipeline, and directly use ground-truth mel-spectrograms as target for model training, which can avoid the information loss in distilled mel-spectrograms and increase the upper bound of the voice quality. Second, our variance adaptor consists of not only duration predictor but also pitch and energy predictors, where 1) the duration predictor uses the phoneme duration obtained by forced alignment as training target, which is more accurate than that extracted from the attention map of autoregressive teacher model, the additional pitch and energy predictors can provide more variance information, which is important to ease the one-to-many mapping problem in TTS. Third, to further simplify the training pipeline and push it towards a fully end-to-end system, we propose FastSpeech 2s, which directly generates waveform from text, without cascaded mel-spectrogram generation (acoustic model) and waveform generation (vocoder). In the following subsections, we describe detailed designs of the variance adaptor and direct waveform generation in our method.[6]

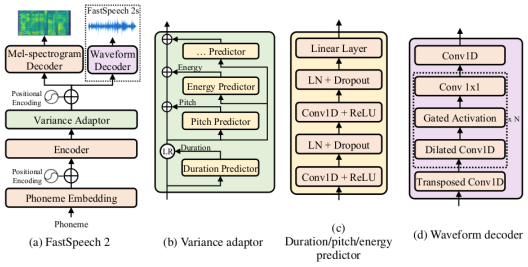


Figure 5: Block diagram of the fastspeech2.

VARIANCE ADAPTOR

The variance adaptor aims to add variance information (e.g., duration, pitch, energy, etc.) to the phoneme hidden sequence, which can provide enough information to predict variant speech for the one-to-many mapping problem in TTS. We briefly introduce the variance information as follows: 1) phoneme duration, which represents how long the speech voice sounds; 2) pitch, which is a key feature to convey emotions and greatly affects the speech prosody; 3) energy, which indicates framelevel magnitude of mel-spectrograms and directly affects the volume and prosody of speech. More variance information can be added in the variance adaptor, such as emotion, style and speaker, and we leave it for future work. Correspondingly, the variance adaptor consists of 1) a duration predictor (i.e., the length regulator, as used in FastSpeech), 2) a pitch predictor, and 3) an energy predictor. In training, we take the ground-truth value of duration, pitch and energy extracted from the recordings as input into the hidden sequence to predict the target speech. At the same time, we use the ground-truth duration, pitch and energy as targets to train the duration, pitch

and energy predictors, which are used in inference to synthesize target speech. The duration, pitch and energy predictors share similar model structure (but different model parameters), which consists of a 2-layer 1D convolutional network with ReLU activation, each followed by the layer normalization and the dropout layer, and an extra linear layer to project the hidden states into the output sequence. In the following paragraphs, we describe the details of the three predictors respectively.

Duration Predictor

Duration Predictor The duration predictor takes the phoneme hidden sequence as input and predicts the duration of each phoneme, which represents how many mel frames correspond to this phoneme, and is converted into logarithmic domain for ease of prediction. The duration predictor is optimized with mean square error (MSE) loss, taking the extracted duration as training target. Instead of extracting the phoneme duration using a pre-trained autoregressive TTS model in FastSpeech, we use Montreal forced alignment (MFA) tool3 to extract the phoneme duration, in order to improve the alignment accuracy and thus reduce the information gap between the model input and output.

Pitch Predictor

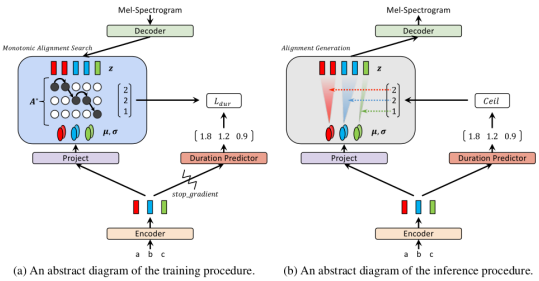
Pitch Predictor Previous neural network based TTS systems with pitch prediction often predict pitch contour directly. However, due to high variations of ground-truth pitch, the distribution of predicted pitch values is very different from ground-truth distribution, To better predict the variations in pitch contour, we use continuous wavelet transform (CWT) to decompose the continuous pitch series into pitch spectrogram and take the pitch spectrogram as the training target for the pitch predictor which is optimized with MSE loss. In inference, the pitch predictor predicts the pitch spectrogram, which is further converted back into pitch contour using inverse continuous wavelet transform (iCWT). We describe the details of pitch extraction, CWT, iCWT and pitch predictor architecture in Appendix D. To take the pitch contour as input in both training and inference, we quantize pitch F0 (ground-truth/predicted value for train/inference respectively) of each frame to 256 possible values in log-scale and further convert it into pitch embedding vector p and add it to the expanded hidden sequence.

Energy Predictor

We compute L2-norm of the amplitude of each short-time Fourier transform (STFT) frame as the energy. Then we quantize energy of each frame to 256 possible values uniformly, encoded it into energy embedding e and add it to the expanded hidden sequence similarly to pitch. We use an energy predictor to predict the original values of energy instead of the quantized values and optimize the energy predictor with MSE loss.

3.4 Glow-TTS

Each component of Glow-TTS is briefly explained in this section, and the overall model architecture and model configurations are shown in Figure.[7]



(a) An abstract diagram of the training procedure. (b) An abstract diagram of the inference procedure.

Figure 6: Block diagram of the Glow-TTS.

Decoder

The core part of Glow-TTS is the flow-based decoder. During training, we need to efficiently transform a mel-spectrogram into the latent representation for maximum likelihood estimation and our internal alignment search. During inference, it is necessary to transform the prior distribution into the mel-spectrogram distribution efficiently for parallel decoding. Therefore, our decoder is composed of a family of flows that can perform forward and inverse transformation in parallel. Specifically, our decoder is a stack of multiple blocks, each of which consists of an activation normalization layer, invertible 1x1 convolution layer, and affine coupling layer. We follow the affine coupling layer architecture of WaveGlow, except we do not use the local conditioning .
For computational efficiency, we split 80-channel mel-spectrogram frames into two halves along the time dimension and group them into one 160-channel feature map before the flow operations. We also modify 1x1 convolution to reduce the time-consuming calculation of the Jacobian determinant. Before every 1x1 convolution, we split the feature map into 40 groups along the channel dimension and perform 1x1 convolution on them separately. To allow channel mixing in each group, the same number of channels are extracted from one half of the feature map separated by coupling layers and the other half, respectively.

Encoder and Duration Predictor

The encoder structure of Transformer TTS with two slight modifications. We remove the positional encoding and add relative position representations into the self-attention modules instead. We also add a residual connection to the

encoder pre-net. To estimate the statistics of the prior distribution, we append a linear projection layer at the end of the encoder. The duration predictor is composed of two convolutional layers with ReLU activation, layer normalization, and dropout followed by a projection layer. The architecture and configuration of the duration predictor are the same as those of FastSpeech.

4 DNN approaches of TTS(Fully E2E Model)

4.1 FastSpeech2s

To enable fully end-to-end text-to-waveform generation, in this subsection, we extend FastSpeech 2 to FastSpeech 2s, which directly generates waveform from text, without cascaded mel-spectrogram generation (acoustic model) and waveform generation (vocoder). As shown in Figure o fastspeech2, FastSpeech2s generates waveform conditioning on intermediate hidden, which makes it more compact in inference by discarding mel-spectrogram decoder and achieve comparable performance with a cascaded system. We first discuss the challenges in non-autoregressive text-to-waveform generation, then describe details of FastSpeech 2s, including model structure and training and inference processes.

Challenges in Text-to-Waveform Generation

When pushing TTS pipeline towards fully endto-end framework, there are several challenges: 1) Since the waveform contains more variance information (e.g., phase) than mel-spectrograms, the information gap between the input and output is larger than that in text-to-spectrogram generation. 2) It is difficult to train on the audio clip that corresponds to the full text sequence due to the extremely long waveform samples and limited GPU memory. As a result, we can only train on a short audio clip that corresponds to a partial text sequence which makes it hard for the model to capture the relationship among phonemes in different partial text sequences and thus harms the text feature extraction.

Methodology

To tackle the challenges above, we make several designs in the waveform decoder: 1) Considering that the phase information is difficult to predict using a variance predictor, we introduce adversarial training in the waveform decoder to force it to implicitly recover the phase information by itself. 2) We leverage the mel-spectrogram decoder of FastSpeech 2, which is trained on the full text sequence to help on the text feature extraction. As shown in Figure 1d, the waveform decoder is based on the structure of WaveNet including non-causal convolutions and gated activation . The waveform decoder takes a sliced hidden sequence corresponding to a short audio clip as input and upsamples it with transposed

1D-convolution to match the length of audio clip. The discriminator in the adversarial training adopts the same structure in Parallel WaveGAN which consists of ten layers of non-causal dilated 1-D convolutions with leaky ReLU activation function. The waveform decoder is optimized by the multi-resolution STFT loss and the LSGAN discriminator loss following Parallel WaveGAN. In inference, we discard the mel-spectrogram decoder and only use the waveform decoder to synthesize speech audio.

4.2 ClariNet

ClariNet fully convolutional text-to-wave architecture for end-to-end TTS. Our architecture is based on Deep Voice 3 (DV3), a convolutional attention-based TTS system . DV3 is capable of converting textual features (e.g., characters, phonemes and stresses) into spectral features (e.g., log-mel spectrograms and log-linear spectrograms). These spectral features can be used as inputs for a separately trained waveform synthesis model, such as WaveNet. In contrast, we directly feed the hidden representation learned from the attention mechanism to the WaveNet through some intermediate processing, and train the whole model from scratch in an end-to-end manner.
Conditioning the WaveNet on hidden representation is crucial to the success of training from scratch. Indeed, we tried to condition WaveNet on predicted mel-spectrogram from DV3, thus the gradients of WaveNet loss can backpropagate through DV3 to improve the text-to-spectrogram model. When the whole model is trained from scratch, we found it performs slightly worse than the separate training pipeline. The major reason is that the predicted mel-spectrogram from DV3 can be inaccurate at early training, and may spoil the training of WaveNet. In order to get satisfactory results, one need pretrain DV3 and WaveNet, then fine-tune the whole system.
The proposed architecture consists of four components:

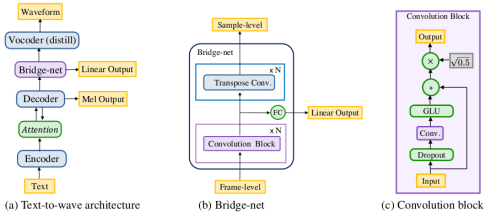


Figure 7: (a) Text-to-wave model converts textual features into waveform. All components feed their hidden representation to others directly. (b) Bridge-net maps frame-level hidden representation to sample-level through several convolution blocks and transposed convolution layers interleaved with softsign nonlinearities. (c) Convolution block is based on gated linear unit.

- Encoder: A convolutional encoder as in DV3, which encodes textual features into an internal hidden representation.

- Decoder: A causal convolutional decoder as in DV3, which decodes the encoder representa- tion with attention into the log-mel spectrogram in an autoregressive manner.
- Bridge-net: A convolutional intermediate processing block, which processes the hidden representation from the decoder and predicts the log-linear spectrogram. Unlike the decoder, it is non-causal and can thus utilize future context information. In addition, it upsamples the hidden representation from frame-level to sample-level.
- Vocoder: A Gaussian autoregressive WaveNet to synthesize the waveform, which is condi- tioned on the upsampled hidden representation from the bridge-net. This component can be replaced by a student IAF distilled from the autoregressive vocoder.

The overall objective function is a linear combination of the losses from decoder, bridge-net and vocoder; we simply set all coefficients to one in experiments. We introduce bridge-net to utilize future temporal information as it can apply non-causal convolution. All modules in our architecture are convolutional, which enables fast training 7 and alleviates the common difficulties in RNN-based models (e.g., vanishing and exploding gradient problems). Throughout the whole model, we use the convolution block from DV3 as the basic building block. It consists of a 1-D convolution with a gated linear unit (GLU) and a residual connection.[8]

5 Neural Network based Vocoders

5.1 Wavenet

WaveNet has achieved significant success in various fields, including image, handwriting, video, text, music, and human speech processing. This model excels in handling raw audio signals, demonstrating extreme autoregression capabilities at speeds of up to 24,000 samples per second. In the WaveNet framework, input X represents text, and it produces audio waveforms as output Y. Its primary function is to convert audio waveforms into speech signals. To achieve this, WaveNet relies on the time series of audio samples to generate speech samples directly. The original work on WaveNet was conducted by DeepMind. During audio sample generation, each sample is influenced by the previous ones.

WaveNet employs convolutional layers, to process digitized raw audio waveforms. These convolution layers take in the input and produce a waveform sample.

It's important to note that WaveNet is an unconditioned model, meaning it lacks information about the structure of speech. Without conditioning, it cannot generate meaningful audio. When trained on human speech audio, the generated

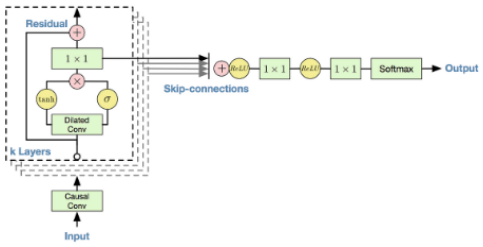


Figure 8: Block diagram of the Wavenet architecture.

sound may resemble human speech but is often fragmented or interrupted. To address this issue, the WaveNet team incorporated vocoder constraints from an existing text-to-speech (TTS) system as an additional input alongside the raw audio input. This approach resulted in higher-quality voice synthesis for users. However, it's worth mentioning that this model is computationally expensive. To achieve high-quality results, it typically requires a minimum of 40 convolution layers. Generating 1 second of 16 kHz audio involves processing 16,000 samples, which can take up to 4 minutes—a duration that may not be suitable for modern technology standards. To address this issue, Baidu Deep Voice introduced a faster version of WaveNet, which is 400 times quicker than the original.

5.2 WaveRNN

The architecture WaveRNN vocoder is a hybrid model that combines residual blocks and an upsampling network, followed by GRU (Gated Recurrent Unit) and fully connected (FC) layers, as illustrated in Figure

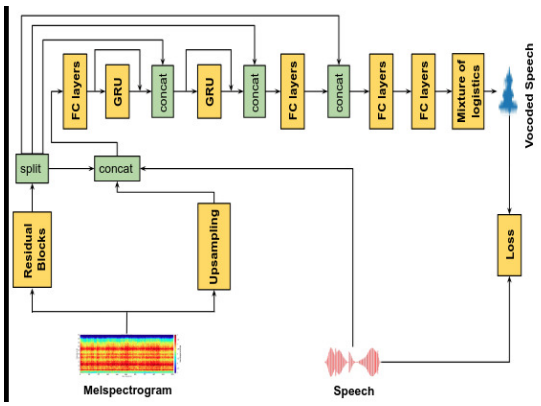


Figure 9: Block diagram of WaveRNN.

The architecture can be conceptually divided into two primary networks: the conditional network and the recurrent network. The conditional network comprises a combination of a residual network and an upsampling network, which incorporates three scaling factors. Initially, we map the input acoustic features, specifically the mel-spectrograms, into a latent representation using multiple residual blocks. This latent representation is subsequently split into four segments, which serve as inputs for the subsequent recurrent network. The purpose of the upsampling network is to adjust the temporal dimension of the input signal to match the desired size.

The outputs of the two convolutional networks, namely the residual and upsampling networks, along with the input speech data, are fed into the recurrent network. Within the recurrent network, two unidirectional GRUs are employed, alongside several FC layers. This network design not only reduces computational complexity by minimizing the number of parameters but also leverages temporal context to enhance prediction accuracy.

Furthermore, the model incorporates a continuous univariate distribution, specifically a mixture of logistic distributions as described in reference [23]. This distribution enables straightforward calculation of probabilities for the observed discretized values in the speech data. Finally, the model applies a loss function known as the discretized mixture logistic loss to train and optimize itself on the discretized speech samples.

5.3 HIfi-GAN

HiFi-GAN stands out as a prominent GAN-based neural vocoder known for its efficient parallel synthesis capabilities. In the GAN training paradigm, the model undergoes adversarial training, where a generator aims to deceive a discriminator, and the discriminator's role is to distinguish between genuine and generated samples iteratively. HiFi-GAN incorporates specialized discriminators, namely the multi-period discriminator (MPD) and the multi-scale discriminator (MSD), designed to enhance fidelity by addressing distinct aspects of speech waveform properties.

In Figure, we adopt the HiFi-GAN generator to synthesize raw waveform data from the decoder's output. The HiFi-GAN generator achieves this by employing transposed convolution to upsample the decoder's output, aligning it with the length of the raw waveform. This ensures that the output of the decoder matches the length of the ground-truth waveform's mel-spectrogram. The training process involves not only adversarial loss but also auxiliary losses, namely the feature matching loss and the mel-spectrogram loss. These auxiliary losses are introduced to improve speech quality and training stability.

It's worth noting that the auxiliary mel-spectrogram loss, in this context, represents the L1 loss between the mel-spectrogram of the synthesized wave-

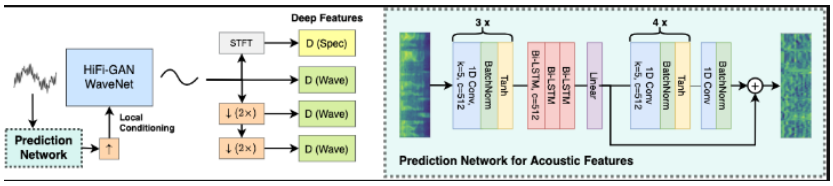
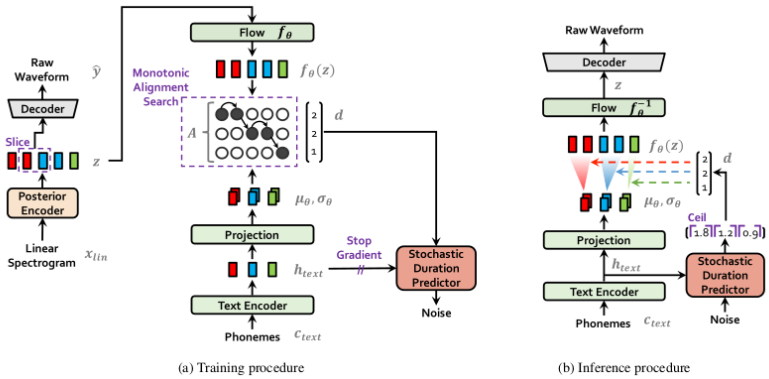


Figure 10: Block diagram of Hifi-GAN.

form and that of the ground-truth waveform. This loss is a distinct concept from the mel-spectrogram loss utilized in FastSpeech2. The training objective of HiFi-GAN follows the principles of LSGAN, where the generator's loss comprises adversarial loss and auxiliary losses.

5.4 VITS

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.[9]



(a) Training procedure (b) Inference procedure

Figure 11: Block diagram of VITS.

POSTERIOR ENCODER

For the posterior encoder, we use the non-causal WaveNet residual blocks used in WaveGlow and Glow-TTS . A WaveNet residual block consists of layers of dilated convolutions with a gated activation unit and skip connection. The linear projection layer above the blocks produces the mean and variance of the normal posterior distribution. For the multi-speaker case, we use global conditioning in

residual blocks to add speaker embedding.

PRIOR ENCODER

The prior encoder consists of a text encoder that processes the input phonemes text and a normalizing flow 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.

DISCRIMINATOR

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

Stochastic duration predictor

The stochastic duration predictor estimates the distribution of phoneme duration from a conditional input htext . 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

6 Datasets for Various Languages

6.1 Mozilla Common Voice Dataset

Common Voice is an audio dataset that consists of a unique MP3 and corresponding text file. There are 9,283 recorded hours in the dataset. The dataset also includes demographic metadata like age, sex, and accent. The dataset consists of 7,335 validated hours in 60 languages. **Mozilla dataset**.

6.2 Shrutilipi

Shrutilipi is a labelled corpus obtained by mining parallel audio and text pairs at the document scale from All India Radio news bulletins for 12 Indian languages: Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Odia, Punjabi, Sanskrit, Tamil, Telugu, Urdu. The corpus has over 6400 hours of data across all languages. **Shrutilipi Dataset**

6.3 Indic TTS

A special corpus of Indian languages covering 13 major languages of India. It comprises of 10000+ spoken sentences/utterances each of mono and english recorded by both Male and Female native speakers. Speech waveform files are available in .wav format along with the corresponding text. **Indic TTS**

7 Conclusion

Unfortunately, even with rapid growth in the TTS technology in the last few years, TTS systems for all official languages of India are not available till date. The technological advancement reflects for around 13 Indian languages till date which is less than 60% of the total number of official languages and the development of a generic working model for all official languages is far from the horizon. Majority of the available research focuses on the Hindi, Bengali, Marathi, Tamil, Telugu, Malayalam, Gujarati, Odia, Assamese, Manipuri and Kannada, languages. A fewer research are available for the Punjabi, Nepali and Urdu language. However, the technological advancement is yet to reach in -Maithili, Santali, Kashmiri, Sindhi, Konkani, Dogri and Manipuri language. Also, research in the unofficial Indian languages spoken by the 3.38% of the population is not yet started till date showing widespread scope for future developments. . .

References

[1] G. K. Kumar, P. S. V. au2, P. Kumar, M. M. Khapra, and K. Nandakumar, "Towards building text-to-speech systems for the next billion users," 2023.

[2] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, "A survey on neural speech synthesis," 2021.

[3] S. Panda, A. Nayak, and S. Champati Rai, "A survey on speech synthesis techniques in indian languages," *Multimedia Systems*, vol. 26, 08 2020.

[4] 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," 2018.

[5] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," 2018.

[6] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," 2022.

[7] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," 2020.

[8] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," 2019.

[9] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," 2021.