International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Transforming Sanskrit: Natural Text-to-Speech with Optimized Encoders

Sabnam Kumari[1], Amita Malik[2]

Submitted:03/09/2024 **Revised:**22/10/2024 **Accepted:**02/11/2024

Abstract: Sanskrit is a very ancient and classical language having a significant impact on science, philosophy, and literature. There are less and fewer skilled speakers of this ancient language, which prevents the easy access to its rich cultural heritage although it is the root of many Indian languages. Thus, resulting in declination of the spoken use of Sanskrit these days. To solve the problem, we need innovative technical solutions that will enable and promote the spoken form of Sanskrit. One way to produce speech to enhance accessibility to the language in the modern era is by using text-to-speech (TTS) synthesis. To help enhance the synthesis quality as well as naturalness of speech, the paper discusses on an improved Sanskrit TTS system having optimized transformer encoding. The system employs Grapheme-to-Phoneme (G2P) to convert Sanskrit text into sounds and use a transformer-based mel-style speaker encoder to extract the speaker's vector. The Gated Convolutional Neural Network (GCNN) captures local features, and GRU, Gated Recurrent Unit, is used for analyzing temporal features. The optimized transformer encoder, optimized by the Adaptive Cheetah Optimization (ACO) algorithm, processes the extracted features The processed output acts as an input to a mel-spectrogram. Later, the mel-spectrogram is converted into high-quality audio waveform using HiFi-GAN vocoder. This complete process leads to a highly effective TTS system that greatly improves speech synthesis for Sanskrit allowing natural sounding speech that closely resembles the voice quality of target speaker. To show that the suggested method is effective, we developed the system with Python and take Vāksañcayaḥ - Sanskrit Speech Corpus dataset for demonstrating our results. The results show a significant improvement in creating speech that resembles the voice of the target speaker.

Keywords: *Text-to-Speech (TTS) synthesis, Grapheme-to-Phoneme, Gated Convolutional Neural Network, Gated Recurrent Unit, Adaptive Cheetah Optimization, HiFi-GAN vocoder*

1. Introduction

Being one of the ancient grammars and a key component of Indian social and cultural psychology, Sanskrit is a unique language that shows an extension of tradition from the Vedic era. "Languages are the repository of thousands of years of people's science and art" [1]. Over time, Sanskrit's influence has diminished significantly, and its usage and scope could be somewhat restricted. Nevertheless, there are numerous materials available on a vast array of aspects and historical eras, most of which are text-based. We believe that in order to provide natural

1Department of Computer Science & Engineering, Deenbandhu Chhotu Ram University of Science & Technology (DCRUST), Sonipat, Haryana, India Shabnam022@gmail.com
2Department of Computer Science & Engineering, Deenbandhu Chhotu Ram University of Science & Technology (DCRUST), Sonipat, Haryana, India amitamalik.cse@dcrustm.org

auditory access to these components, a TTS for Sanskrit is essential.

The objective of generating speech that seems natural from text is still a challenging one to solve. At the moment, the most sophisticated TTS systems are those utilizing deep learning to produce speech that sounds natural. But the primary obstacle to Sanskrit TTS progress is an inadequate supply of high-quality data. Internal building blocks can be combined via deep learning [2] into a single model that directly links input and output. This approach is also known as "end-to-end" learning. An end-to-end text-to-speech system called Tacotron [3], which computes a spectrogram directly from an input text and is frequently referred to as "a black box," has recently shown impressive results despite the need for manually-engineered parametric models based on domain-specific data. Tacotron‘s use of numerous repeating units contributes to its high training expenses [4]. Regular labs cannot do additional research or development without expensive machinery. Even while several people have tried building open replicas of Tacotron, they have not been able to produce speech that is as clear and of high quality as the original models.

The conventional methods of synthesis were concatenative [5-7] and parametric, resulting in speech that was muted due to much more complex pipelines. Moreover, irregularities and abnormalities could appear in the speech output. The field has experienced enormous growth due to the quick evolution of deep learning-based techniques. It has been suggested that the current pipelines be substituted with end-to-end generative models like Tacotron2 [8, 9], and Deep Voice [10]. By integrating spectrogram forecasting and synthesis of speech into a single pipeline, these models have demonstrated state-of-the-art performance. But these end-to-end systems need a large amount of processing power and tens of hours of audio data. These issues are addressed by the proposed Sanskrit TTS system, which uses an optimized transformer encoder to improve voice synthesis while decreasing the requirement for vast datasets and processing resources. The main contributions of this work are listed below:

- To address phonetic complexity and data limitations, an advanced Sanskrit TTS system is developed using an optimized transformer encoder.
- Consider the two input sets such as text data that needs to be converted into speech and reference audio signal which is the small sample of the target speaker's voice as a reference to guide the synthesis process.
- Then, we Convert the input text into phonemes grapheme-to-phoneme conversion, which are the basic units of sound and we extract a speaker vector from the reference audio, which captures the unique characteristics of the target speaker's voice.
- G2P conversion is integrated, along with GCNN and GRU for local and temporal feature extraction.
- The Adaptive Cheetah Optimization algorithm is applied to optimize the transformer's performance.
- HiFi-GAN vocoder is utilized to generate high-quality audio waveforms from mel-spectrograms.

- The system is evaluated using metrics like WER, MCD, SECS, MOS, and SMOS, demonstrating significant improvements in naturalness and speaker resemblance.
- Python is used to implement the system for better accessibility and scalability.

The following sections of the paper will be organized in this manner. Section 2 presents an overview of the related work. Section 3 describes the proposed work. Section 4 discusses the study's results and assessments. Section 5 includes the final conclusion and plans for further work.

2. Related works

Yinghao Aaron Li and Cong Han et al., [11] presented the Style-Based Generative Model for Natural and Diverse Text-to-Speech Synthesis. Through self-supervised learning, Style TTS was generating speech with the same emotional and prosodic tone as the reference speech without needing explicit labels for these categories. A limitation of Style TTS systems was that mapping different prosodic patterns from reference recordings can be challenging. This could result in difficulty maintaining smooth control over subtle, highly specialized stylistic variations, which was affect the naturalness or consistency of the synthesized speech. These systems often struggle with fine-tuning prosody while preserving the intended style, especially for nuanced or complex speech patterns.

Ye Jia et al. [12] presented Transfer Learning from speaker verification to multi-speaker text-to-speech synthesis. This method synthesized natural speech from speakers not encountered during training, using the understanding of speaker variation gained by a discriminatively learnt speaker encoder in the multi-speaker TTS challenge. It has been proved that the model learns a high-quality speaker description when it can synthesize speech in the voice of speakers other than those utilized during training using randomly selected speaker embeddings. Finally, a similar pattern emerged: the model struggled to distinguish between the voice of the speaker and the prosody of the reference speech.

Chengyi Wang et al. [13] introduced neural codec language models, which are zero-shot text to speech synthesizers. They presented VALL-E, a language model technique for TTS that used intermediate models in the form of audio codec code. That demonstrated the ability to learn in context in zero-shot circumstances. On VCTK and Libri Speech, they attain new state-of-the-art zero-shot TTS outcomes. Moreover, VALL-E could maintain the synthesis of the speaker's emotions and the surrounding acoustics while producing a variety of outputs for various sampling-based decoding techniques.

Ruchika Kumari et al. [14] demonstrated a robust adaptive neural network-based text-to-speech synthesizers for the Hindi language. This paper laid out a text-to-speech synthesizer for the Hindi language that was based on the language-based restrictions recommended for constructing parameters such as intonation, duration, and syllable phases, as well as the parameters of Mel-frequency cepstral (MFCC) features that were gathered for processing. The suggested ALO-ANN method outperforms all other methods in terms of accurate prediction. To investigate the individual feature results, the

durations, intensities, and basic frequencies of the syllables were evaluated in conjunction with these aspects. Objective data such as the correlation coefficient, standard deviation, and average prediction error are used to determine intonation representation.

Junrui Ni et al., [15] laid out the Unsupervised TTS synthesis via Unsupervised Automatic Speech Recognition. The two components of their suggested unsupervised text-to-speech system are an orientation module that produces pseudo-text and a synthesis module that employs real text for training and pseudo-text for inference. With roughly 10–20 hours of speech in seven languages, the suggested unsupervised system could perform on par with the supervised system. This work's modest decreased intelligibility for non-English languages as compared to supervised TTS models was one of its limitations.

A cross-lingual, multi-speaker neural end-to-end text-to-speech system that can mimic speaker characteristics and synthesize speech in several languages was presented by Mengnan Chen et al. [16]. A neural speaker embedding network that has been trained separately was shown by them; it was capable of describing the latent structure of various speakers and language pronunciations. Our findings demonstrated that the multi-speaker TTS model was capable of extracting from the latent space both speaker features and language pronunciations with speaker embedding. They also confirmed that the suggested approach can effectively handle cross-lingual jobs with a minimal amount of audio data. The use of a Griffin-Lim vocoder, which may provide less-than-ideal audio quality when compared to more sophisticated neural vocoders, was a limitation of this work.

A Robust Transformer-Based text-to-speech Model was presented by Naihan Li et al. [17]. A robust neural TTS model based on Transformer, called Robu Trans (Robust Transformer), was proposed by them. In contrast to transformer TTS, our approach feeds input texts to the encoder after first converting them to linguistic features, such as prosodic and phonemic features. They produced exceptional results, resulting in very high-quality and realistic synthesized sounds. In addition to solving the robustness issue, the suggested model achieves parity MOS with 4.36 on the general set, with transformer TTS with 4.37 and Tacotron2 with 4.37. Nevertheless, the robustness problem with existing neural TTS models leads to anomalous audios (poor instances), particularly for unusual text.

The pipeline for creating artificial child speech in a situation with minimal training data was presented by Rishabh Jain et al. [18]. Additionally, a subjective evaluation technique appropriate for child speech generated was presented and illustrated. These phrases could use some refinement. Using a trained adult speech wav2vec2 ASR model, the WER for generated child voices was 25.63, while the WER for real child voices was 15.27. They succeed in significantly raising the vocoder's quality. The audio patterns that had produced did not significantly improve in quality, and some of the synthesis showed the presence of extra noise.

3. Proposed Optimized TTS System for Sanskrit

The objective of this work is to develop an advanced TTS synthesis system for Sanskrit that enhances speech synthesis accuracy and naturalness. To achieve this, the methodology begins with converting text into phonetic representation using a Grapheme-to-Phoneme tool. A Transformer-based mel-style speaker encoder extracts speaker vectors from reference audio, capturing distinct vocal attributes. Local and temporal speaker characteristics are further refined through the use of Gated Convolutional Neural Networks for local features and Gated Recurrent Units for temporal features. These features are then processed by an optimized transformer encoder, with the optimization performed using the Adaptive Cheetah Optimization algorithm. The final mel-spectrogram output is converted into high-quality audio waveforms using the HiFi-GAN vocoder, resulting in a robust TTS system that significantly improves speech synthesis for Sanskrit by generating natural-sounding speech that closely mirrors the target speaker's vocal attributes. Figure 1 depicts the proposed framework's structure.

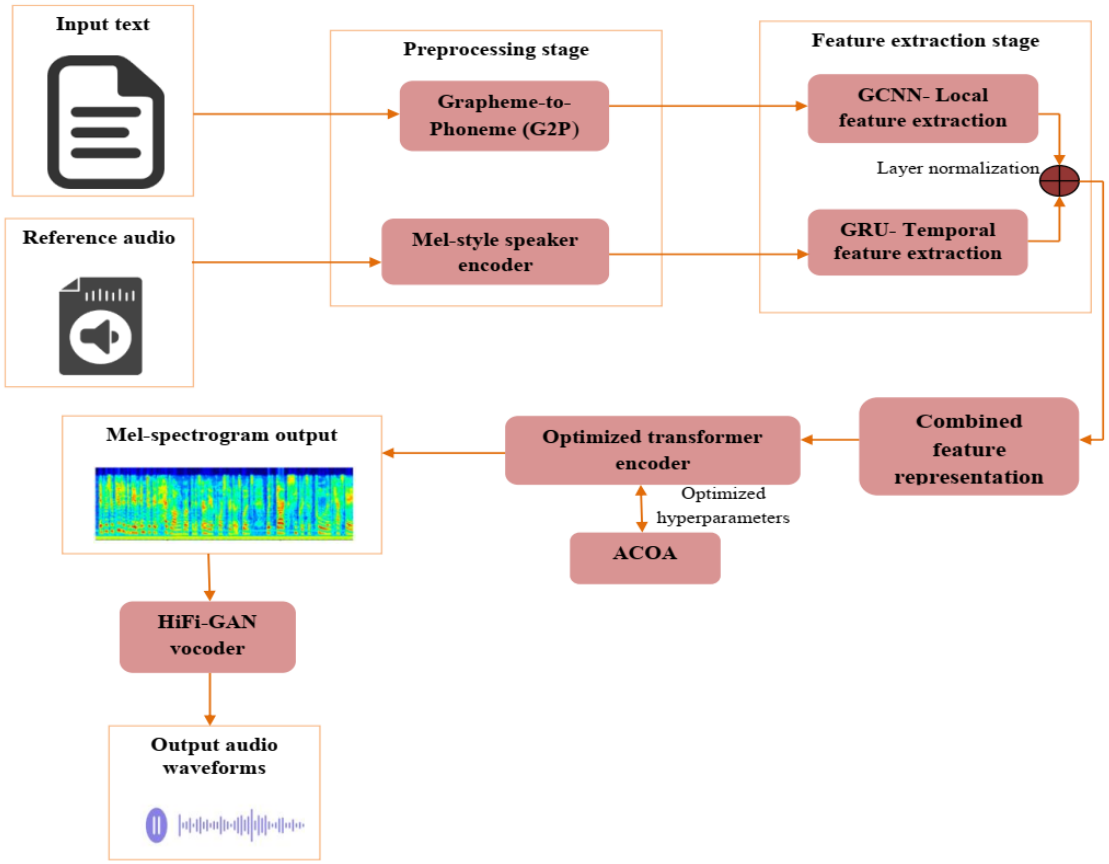


Figure 1: Structure of proposed framework

3.1 Input preparation

The proposed system takes two main inputs: a text sequence that needs to be converted into speech and a reference audio signal, a small sample of the target speaker's voice. The text is converted into its phonetic representation through a Grapheme-to-Phoneme conversion, ensuring accurate

pronunciation. The reference audio is processed to extract a speaker vector, capturing the unique vocal characteristics of the target speaker.

3.1.1 Grapheme-to-Phoneme Conversion

For the Grapheme-to-Phoneme conversion in our Text-to-Speech system, we utilized the G2P tool available at Kyubyong's GitHub repository (https://github.com/Kyubyong/g2p). This tool is a neural network-based model designed to map text (graphemes) to their corresponding phonetic representations (phonemes) accurately. The G2P conversion process is a critical component in our pipeline, ensuring that the input text is translated into phonemes, which are then used to generate natural-sounding speech. By leveraging Kyubyong's G2P tool, we were able to achieve reliable and efficient phoneme generation, contributing to the overall quality and intelligibility of the synthesized speech.

3.1.2 Speaker Vector Extraction

In our study, we use a Transformer-based mel-style speaker encoder [19] to preprocess the reference audio signal. The purpose of the speaker encoder in our method is to train the synthesis network using a reference signal of speech from the intended target speaker. The representation used by the speaker encoder must accurately depict the distinctive qualities of different speakers in order for standardization to be efficient. The encoder extracts a latent speaker vector, represented as SV_{emb}, by processing the reference audio's mel spectrogram. This vector captures the unique characteristics of the target speaker's voice, like pitch, tone, and speaking style.

3.2 Local and Temporal Feature Extraction for Speaker Control Using GCNN and GRU

In this work, we develop a two-branched feature extraction method that employs two separate methods to translate speaker characteristics. To ensure that the generated speech appropriately reflects the speaker's identity, the first channel captures local frame-to-frame information using a GCNN. In order to capture temporal dependencies and overall speaker style, the second channel makes use of a GRU, which improves the synthetic speech's authenticity.

3.2.1 Local feature extraction using GCNN:

In our work, the GCNN is responsible for extracting local features from the input data by capturing frame-to-frame dependencies. It processes the phoneme sequence combined with the speaker vector, ensuring that the generated speech is conditioned on the speaker's identity. The GCNN improves the parallelization and computational efficiency of the system by avoiding sequential procedures that are commonly utilized in recurrent models. To simulate dependencies across extended sequences, the GCNN employs gated temporal convolutions rather than recurrent connections, as in RNNs. With the addition of many convolutional layers to the network, this technique learns both local and global structures from the input data. Figure 1 shows the model architecture.

Formally, the GCNN processes a sequence of input features $w_0, \ldots, w_N$ and produces a representation $H = h_0, \ldots, h_N$, where each h_i is a context representation of the corresponding input w_i. The input sequence is represented as embeddings

$E = [D_{w0}, \ldots, D_{wN}]$, and the hidden layers of the network are computed by convolving these embeddings with the function *f*:

$$H = f * w \tag{1}$$

The features that are output $h_l, l = 1 \ldots L$ are calculated by:

$$h_l = (h_{l-1} * w + b) \otimes \sigma(h_{l-1} * v + c) \tag{2}$$

In which, h_0 is the input feature, $w \epsilon \mathbb{R}^{k \times m \times n}$ and $v \epsilon \mathbb{R}^{k \times m \times n}$ are the convolutional weight matrices, and σ is the sigmoid function. In this case, *n* is the number of filters, *k* is the convolutional kernel's size, and *m* is the input dimension. The biases of filters are denoted by $b \epsilon \mathbb{R}^n$ and $c \epsilon \mathbb{R}^n$.

To compute posterior probabilities, we stack *L*, GCNN layers and add a softmax layer on top of the final one. The softmax layer outputs an estimate of the posterior probabilities *p(s|o)* for states *s*, and provided features *o*. The output of the softmax layer is calculated by,

$$P(s|o) = soft\max(w_s h_L + b_s) \tag{3}$$

Where h_L is the output of the L-th GCNN layers and (w_s; b_s) is the connection weight matrix and bias vector for the softmax layer. As the error signals are backpropagated via the *l*-th GCNN layers, the gradient is computed as follows:

$$\nabla h_l = \nabla(h_{l-1} * w + b) \otimes \sigma(h_{l-1} * v + b) + \sigma'(h_{l-1} * v + c) \nabla(h_{l-1} * v + c) \otimes (h_{l-1} * w + b) \tag{4}$$

GCNN models combine the capabilities of LSTM RNNs and CNNs by using convolutional kernels to collect local features and LSTM-like gating techniques to handle time dependencies. GCNNs allow for more exact learning of local structures than models such as time delay neural networks (TDNN), which rely on linear transformations without gates. Furthermore, unlike highway networks, which employ numerous gates and complex processes, GCNNs simplify the process with a single gate, resulting in more efficient computations and shorter processing times. This unique combination enables GCNNs to efficiently balance local feature learning with temporal dependencies.

The latent speaker vector is combined with the sequential features of the input to generate the time-expanded speaker vector. Using this combined input as a starting point, a range-limited Soft Gate control signal is generated following sigmoid function and voltage layer processing in the conv3 layer. The CNN branch's output is shown as follows, $h_{|CNN}$:

$$h_{|GCNN} = Sigmoid\,(conv\,(h_i + SV_{emb})) * conv\,(h_i) \tag{5}$$

in which the symbols *conv and sigmoid* denote the convolution process and the sigmoid activation functionality, correspondingly. In order to guarantee which, the speaker vector influences the output features, the CNN branch effectively records the local frame-to-frame data within speech.

3.2.2 Temporal feature extraction using GRU

In our work, the phoneme sequence representing the text data is combined with the implicit speaker vector obtained from the source audio to serve as the GRU model's input. The target speaker's distinct vocal qualities are captured by the speaker vector, while the phoneme sequence provides the basic sound units required for speech

synthesis. This combined input is processed by the GRU, which records speaker style and temporal dependencies during the entire speech. The output of the model combines these temporal and speaker-specific features into a feature representation. This output is crucial for ensuring that the generated speech maintains natural temporal flow and accurately reflects the target speaker's identity.

GRU is an LSTM-based model [20] that maintains LSTM performance while improving the LSTM network layout. The prediction problem of long interval long delay time series can be resolved by the GRU network, which has only two gate structures the update gate and reset gate. The amount of information from the previous moment transferred into the present moment is managed by the update gate. The reset gate regulates how much of the data from the previous moment is discarded. A GRU construction is depicted in Fig. 2. The following formula can be used to determine the output of a GRU unit, assuming that the input sequence is $(x_1, x_{2,} \ldots, x_t)$, followed by a gate reset and an update at t.

$$r_t = \sigma\left(\omega_r * \left[h_{t-1}, x_t\right]\right) \tag{6}$$

$$z_t = \sigma\left(\omega_z * \left[h_{t-1}, x_t\right]\right) \tag{7}$$

$$\hat{h}_t = \tanh\left(\omega_{\hat{h}} \cdot \left[r_t * h_{t-1}, x_t\right]\right) \tag{8}$$

$$h_t = \left(1 - z_t\right) * h_{t-1} + z_t * \hat{h}_t \tag{9}$$

$$\hat{y}_t = \sigma\left(\omega_o \cdot h_t\right) \tag{10}$$

In which, the formula's symbols z_t and r_t denote the update gate, and the result value of the reset gate at time *t*. x_t is the input at time t, ω is the weight of the model and σ is the activation function. h_t and h_{t-1} denotes the output at time t and t-1 correspondingly. In other words, the output of the reset gate is used to adjust the update gate in order to optimise the neuron structure. Using the vector of the speaker SV_{emb} as the beginning state, every frame input h_i is converted by the GRU unit in this way:

$$h_{GRU}(t) = \begin{cases} GRU(SV_{emb}, h_i(t)) & if\ t = 1 \\ GRU(h_{GRU}(t-1), h_i(t)) & else \end{cases} \tag{11}$$

Where the input and output frames of the GRU at every stage are indicated by $h_i(t)$ and $h_{GRU}(t)$, respectively. The very last feature h_{GRU} of the second branch is formed by combining the GRU outputs from all decoding phases. This enables speaker style control over the entire speech. We implement layer normalization (LN) [21] on the outputs from the GCNN and GRU channels, (h_{GCNN} and h_{GRU}), in order to provide constant speaker identity control. In our proposed method, the speaker identification can be precisely controlled at both the local frame level and across the entire utterance by combining the normalised outputs to form the final output of each block.

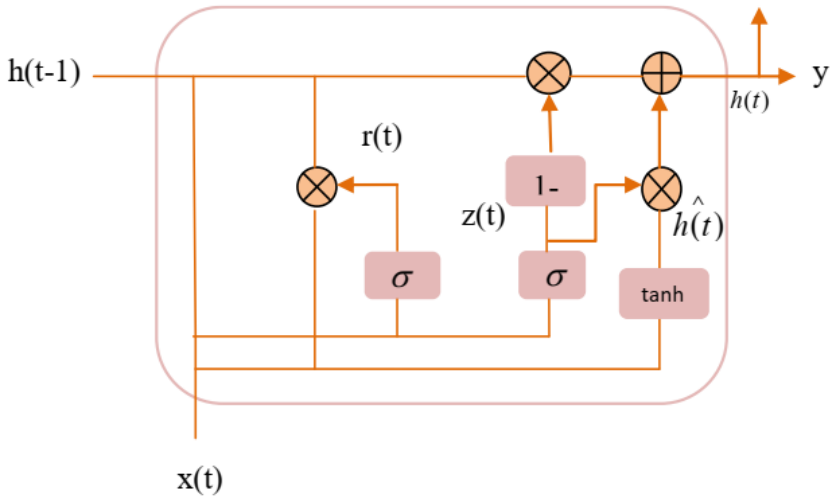


Figure 2: The structure of enhanced GRU's neurons

3.3 Mel-Spectrogram Generation: Optimized Transformer Encoder

The combined outputs from the GCNN and GRU branches are fed into the transformer in our suggested system. While the GRU captures temporal relationships and more detailed background details for the duration of the sequence, the GCNN extracts local features including speaker-specific variations and phonetic specifics. The input for the transformer is a single feature representation that is produced by combining and normalizing these outputs.

The input for the Opt-transformer encoder is the combined feature representation. Positional encodings are added to the combined feature representation because transformers are not automatically aware of the sequence order. Due to the information provided by this encoding, the transformer is able to preserve the input data's sequential structure. Later, this feature embedding, denoted as, $X = (x_1, \ldots\ldots, x_n)$ is transferred to the transformer encoder. Two-dimensional feature embeddings are produced as an output of these features after they go through a learnable linear transformation. Positional data is maintained by adding the positional embedding to the feature embedding, which concatenates the [cls] token. Layer normalisation (LN), an MLP block, and multi-head self-attention (MSA) combine to generate the layers that the embeddings pass throughout. Out of all the feature embeddings obtained from the opt-transformer encoder, just the [cls] token is sent into the MLP head in order to process input data.

Multi-Head Self-Attention Layer

Using k self-attention (SA) or multi-head self-attention operations on two-dimensional embedded features, $E \in R^{(m+1)\times D}$ uses the following equations to learn the relationships between each patch. Query q and key k have a pairwise similarity of A_{ij}, which is the attention weight matrix A.

$$[Q, K, V] = EW_{qkv}$$

$$W_{qkv} \in R^{D\times 3D_h} \quad (12)$$

Every sub-layer adopts residual connections and layer normalisation. Every sublayer outputs data in precisely the same dimension.

In our design, the attended feature vector is translated from x by the encoder block with multi-head attention. Subsequently, feed-forward is applied to every attention vector in order to transform it into a format acceptable by the next level (whether it can be the decoder or as the encoder) $Z = (z_1, \ldots\ldots, z_t)$.

Feed-forward network

Following the self-attention layers of each encoder and decoder is a feed-forward network (FFN). It consists of a nonlinear activation function positioned between two linear transformation layers. The following is the function that represents it:

$$FFN(X) = V_2\sigma(V_1 X) \tag{13}$$

The nonlinear activation function is represented by the symbol σ, while the two linear transformation layers' parameter matrices are V_1 and V_2.

Residual Connection in the Encoder and Decoder

Every sub-layer within the encoder and decoder has a residual link created for it, as seen in Figure 3. Both performance and data transfer are enhanced as a result. The residual connectivity is followed by a layer-normalization [22]. The following are the results from such operations:

$$LayerNorm(X + Attention(X)) \tag{14}$$

The input matrix X is shown to construct the self-attention layer, which then generates the query, key, and value vectors (q, k, and v).

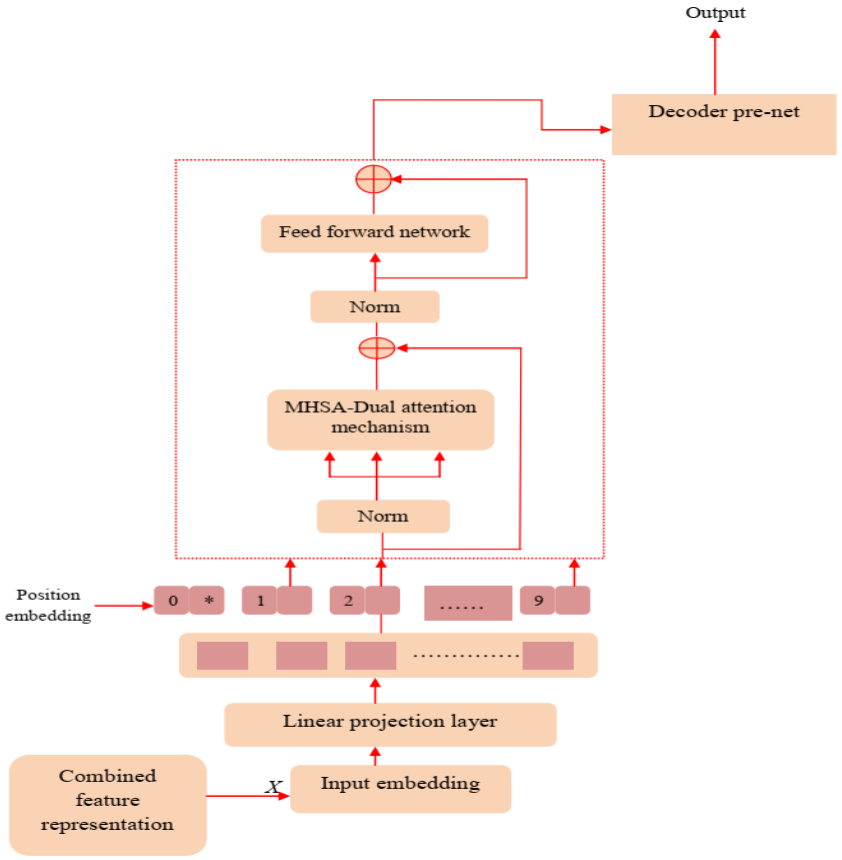


Figure 3: Proposed Transformer encoder model

3.3.1 Adaptive cheetah optimization algorithm

The Adaptive Cheetah Optimisation (ACO) algorithm is used to optimise the transformer encoder. This approach optimizes the transformer encoder's model parameters to improve the device's ability to produce precise and superior mel-spectrograms. The ACO algorithm determines the transformer encoder's optimal weights and configurations by effectively identifying the parameter space. The CO algorithm is a novel meta-heuristic optimisation method that takes its cues from cheetah hunting techniques [23]. It has many benefits, including fast convergence; minimal parameter modifications needed, and simplified computations. The three main phases of the algorithm include searching, waiting, and attacking.

1. Initialization

In the Cheetah Optimisation Algorithm, a population consists of a collection of potential solutions denoted by the locations of individual cheetahs inside the search region. Important hyperparameters like learning rate (L_r), batch size (B_s), number of attention heads (H), and number of layers (l_N) are considered during the process to ensure the model is effectively fine-tuned. Every member of the population is a potential solution to the optimisation issue at hand. Each cheetah's location inside the search area corresponds to the solution it provides. To get the best solution, the algorithm moves such individuals around iteratively.

$$W = \{w_1, w_2, w_3, \ldots\ldots, w_n\} \tag{15}$$

Where, W is the population and w_n represents the n^{th} solution.

$$w_1 = [L_r, B_s, H, l_N] \tag{16}$$

In which, τ represents the current iteration; $\gamma \in (0,1)$

2. Fitness calculation

Fitness is evaluated by this method using the error rate. The following is a description of the fitness calculation:

$$Fitness = \frac{1}{E_R + \in} \tag{17}$$

Where, $E_R \rightarrow$Error rate of the model

$\in \rightarrow$A small constant that is inserted to prevent division by zero

In order to improve the score, the fitness function minimizes the error rate because a lower error rate corresponds to a better fitness score.

3. Updation process

i. ***Searching phase:*** **-** The cheetah watches its surroundings with alert eyes, using its hunting skills to find the best prey based on the dynamics of the surrounding environment. At this point, the mathematical model is described as follows:

$$C_{i,j}^{\tau+1} = C_{i,j}^{\tau} + rand \times \ell_{i,j}^{\tau} \tag{18}$$

$$\ell_{i,j}^{\tau} = 0.001 + \frac{\tau}{\tau_{\max}}\left(U_L - L_L\right) \tag{19}$$

Where $C_{i,j}^{\tau}$ denotes the current position of cheetah-*i* at iteration τ, $C_{i,j}^{\tau+1}$ represents updated position of cheetah-*i* at iteration $\tau+1$, *rand* is a random number selected from the interval (0, 1), $\ell_{i,j}^{\tau}$ indicates random step length; U_L *and* L_L represents the upper and lower bounds of variable, respectively; τ and $\tau_{\max}$ denote the current and maximum iteration numbers.

ii. ***Sitting and waiting phase:*** - Every motion a cheetah makes when hunting has the danger of alerting and possibly shocking its prey, which could result in an escape. Cheetahs wait patiently for prey to approach them from a distance by remaining low or hiding in bushes. The cheetah keeps its location constant during the sitting and waiting phase, which can be expressed numerically as follows:

$$C_{i,j}^{\tau+1} = C_{i,j}^{\tau} \tag{20}$$

iii. ***Attacking phase:*** - The secret to a cheetah's skill is timing the attacks on prey perfectly, using their exceptional speed and quickness. During the attacking phase, they close the distance swiftly and carefully disrupt the target with speed. Attacks, whether alone or in a group, need strategic placement according to the motions of the prey and the dynamics of the group. Here is the stage's mathematical representation:

$$C_{i,j}^{\tau+1} = C_{H,j}^{\tau} + M_{i,j} \cdot N_{i,j}^{\tau} \tag{21}$$

$$M_{i,j} = \left|rand_{i,j}\right|^{\exp\left(\frac{rand_{i,j}}{2}\right)} \times \sin(2\pi\, rand_{i,j}) \tag{22}$$

$$N_{i,j}^{\tau} = C_{q,j}^{\tau+1} - C_{i,j}^{\tau} \tag{23}$$

Where, $C_{H,j}^{\tau}$ denotes the position of the prey, $M_{i,j}$ *and* $N_{i,j}^{\tau}$ stand for the turning and interaction factors, respectively, and $rand_{i,j}$ represents a value selected at random from a normal distribution. Furthermore, the symbols $C_{q,j}^{\tau+1}$ *and* $C_{i,j}^{\tau}$ represent the locations of cheetahs *q* and *i* at iteration τ, respectively.

Tent chaotic mapping

Following the introduction of Tent chaotic mapping [24], the Tent chaotic map was used to initialize the cheetah instead of the randomly generated approach used in the regular CO. As a result, the equation (20) might be changed as follows:

$$C_{i,j}^{\tau+1} = C_{i,j}^{\tau} + T_c^{\tau} \tag{24}$$

$$T_c^{\tau} = \begin{cases} \dfrac{T_c^{\tau-1}}{\gamma}, & T_c^{\tau-1} \in [0, \gamma] \\ \dfrac{(1-T_c^{\tau-1})}{(1-\gamma)}, & T_c^{\tau-1} \in [\gamma, 1] \end{cases} \tag{25}$$

Dynamic weighting factor

In this phase, the cheetah updates its position continuously using the dynamic weighting factor λ. λ is given a greater value at the start of the iteration, which enables the cheetah to carry out a successful global search. As the iteration comes to a close, λ gradually drops in an adaptive manner. Consequently, the following modification could be made to Eq. (9):

$$C_{i,j}^{\tau+1} = C_{H,j}^{\tau} + \lambda_{i,j} \cdot N_{i,j}^{\tau} \tag{26}$$

$$\lambda = \frac{e^{4(1-\delta)} - e^{-4(1-\delta)}}{\left[e^{2(1-\delta)} + e^{-2(1-\delta)}\right]^2}, \quad \delta = \frac{\tau}{\tau_{\max}} \tag{27}$$

$$N_{i,j}^{\tau} = C_{q,j}^{\tau} - C_{i,j}^{\tau} \tag{28}$$

With these modifications, the fundamental CO can achieve superior search characteristics and balance the phases of exploration and exploitation by avoiding local traps or optima, leading to a global solution.

4. Termination

Equations (17) through (23), when applied to the algorithm, are repeated iteratively. The best solution found during the procedure is returned by the ACOA algorithm at completion of the execution.

Algorithm 1: Adaptive Cheetah Optimization algorithm

Input: learning rate (L_r), batch size (B_s), number of attention heads (H), and number of layers (l_N)

Output: Optimal output

Start
Initialize the learning population of solutions, iterations, and population size
Use Tent chaotic map to initialize the cheetahs' positions in the search space for better diversity using equation (25)
Arrange cheetahs in the search area at random.
For (each iteration)
Calculate fitness for each cheetah using equation (17)
Update cheetah positions using a step-length and random factor using equations (18), (20) and (21)
if (no prey detected)
Keep cheetah's position constant
Else
Adjust positions based on prey location and turning factor
Adapt Tent chaotic map and λ for better global and local search balance using equations (24) and (26)
Repeat until max iterations or target fitness is reached.
Return the best solution
End

In our system, the output mel-spectrogram is processed by the opt-transformer encoder to create the encoded representation. The next step is to convert the encoded representation into an audio waveform.

3.4 Waveform Synthesis: HiFi-GAN Vocoder

The mel-spectrogram that is generated is subsequently transformed into an auditory signal by means of a pretrained HiFi-GAN vocoder. Using GAN as the core generative method, HiFi-GAN incorporates two discriminators, the generator, and other components to efficiently convert the spectrum generated by the acoustic model into audio of excellent quality. An effective generator and multi-period and multi-scale discriminators are features of the high-fidelity neural vocoder HiFi-GAN [25], which is based on generative adversarial networks (GAN). After receiving a mel-spectrogram, the generator upsamples the

data to match the temporal resolution of the intended raw waveform using transposed convolution neural networks and multi-receptive field fusion (MRF) modules. A customized kernel and dilation sizes for each of the receptive fields where the input feature is to be captured make up a multiple residual block model (MRF). Discover how to fool the generator's discriminators. Least Square GAN states that the discriminators pick up the skill of telling the difference between generated and ground truth speech, while the generator learns how to control them. The generator L_G has three losses as its training objective: a mel-spectral L1 loss L_{mel}, an adversarial loss $L_{g.adv}$, a feature matching loss [15] L_{fm}, and so on.

$$L_G = L_{g.adv} + \lambda_{fm} L_{fm} + \lambda_{mel} L_{mel} \tag{29}$$

Where, λ_{fm} and λ_{mel} are denotes the loss balancing hyperparameters.

This adversarial process improves the quality of generated speech, gradually increasing its accuracy and naturalism. The combined impact of these losses, as controlled by certain hyperparameters, contributes to voice output that is clear, high-quality, and natural-sounding, closely matching the desired target audio.

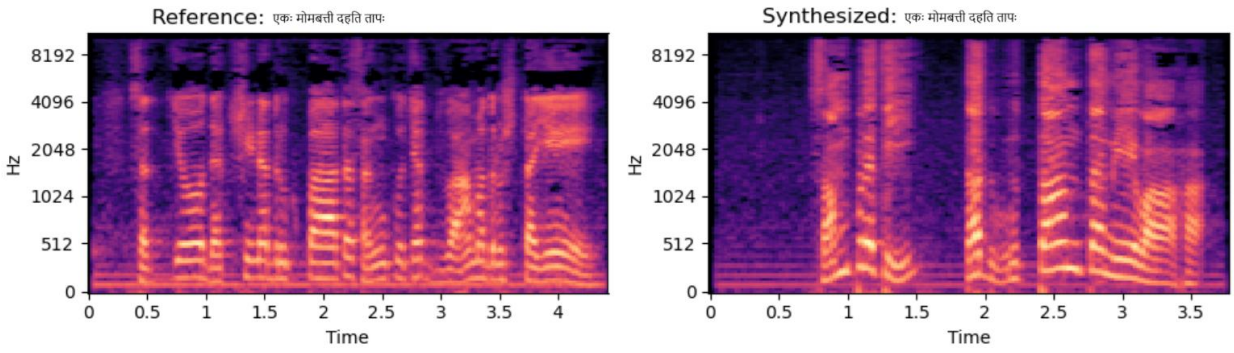
4. Results and Discussion

The advanced TTS synthesis method for Sanskrit yields major improvements in speech quality and naturalness. Evaluation metrics such as Word Error Rate (WER%), Mel-Cepstral Distortion (MCD dB), Speech Emotion Recognition Score (SECS), Mean Opinion Score (MOS), and Speaker Mean Opinion Score (SMOS) show that the system captures and reproduces the target speaker's vocal features accurately. Python is used to implement the proposed methodology. The proposed methodology presented in this study should be implemented using a high-performance processor, such as an Intel Core i9 or AMD Ryzen 9, and a system with 32GB or more of RAM.

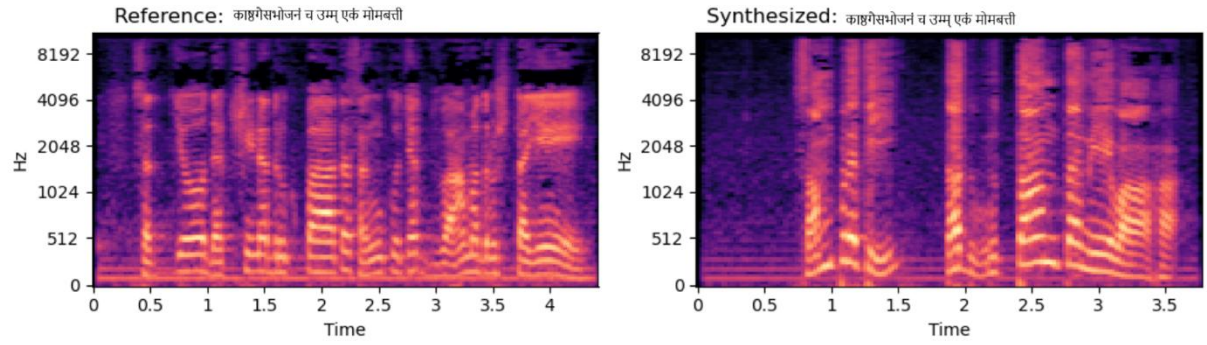
Dataset description: - The Vāksañcayaḥ - Sanskrit voice corpus has about 78 hours of data and 45,953 sentences recorded at a 22 KHz sampling rate. The majority of the material consists of readings from numerous Sanskrit literature books covering many Śāstras. There are also modern tales, radio programs, extempore speeches, etc.

4.1 Experimental results

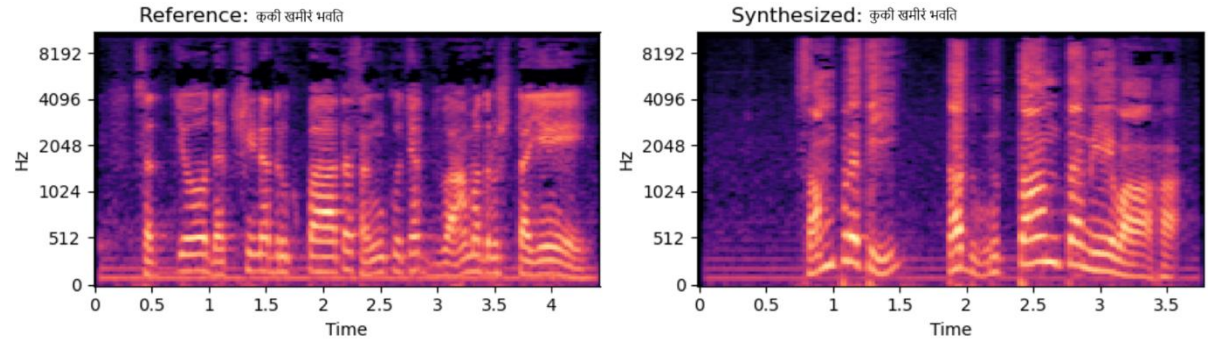
The experimental results demonstrate the effectiveness of the advanced TTS synthesis system for Sanskrit. The analysis indicates that the synthesized audio closely replicates the frequency and temporal features of the original speech, demonstrating a high level of closeness in the output. Performance measurements demonstrate constant progress throughout the training phase, with notable decreases in loss and error rates, demonstrating the model's ability to learn and adapt efficiently.


Reference: एकः मोमबत्ती दहति तापः
Synthesized: एकः मोमबत्ती दहति तापः
8192
4096
2048
1024
512
0
Hz
0 0.5 1 1.5 2 2.5 3 3.5 4
Time

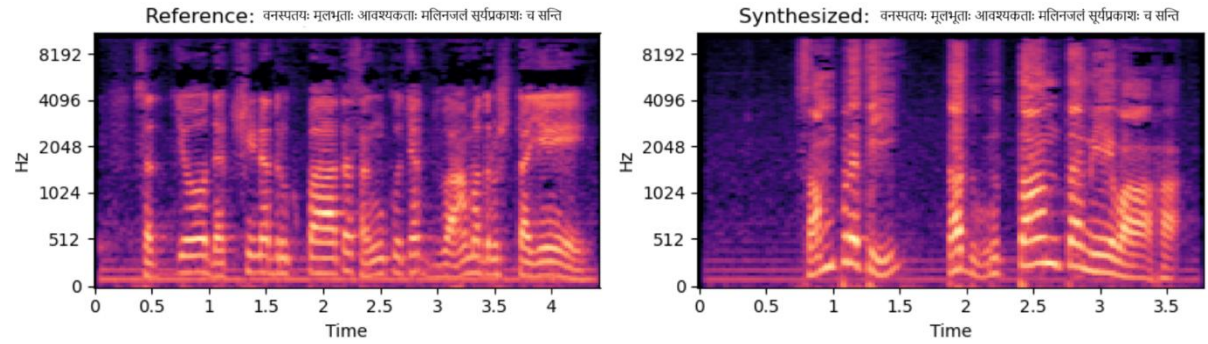
(a)


Reference: काष्ठगेसभोजनं च उम्म् एकं मोमबत्ती
Synthesized: काष्ठगेसभोजनं च उम्म् एकं मोमबत्ती
8192
4096
2048
1024
512
0
Hz
0 0.5 1 1.5 2 2.5 3 3.5 4
Time

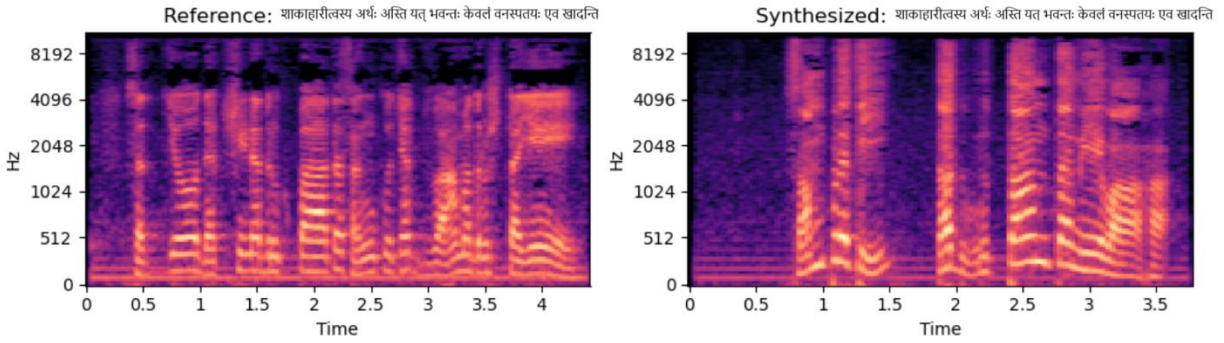
(b)


Reference: कुकी खमीरं भवति
Synthesized: कुकी खमीरं भवति
8192
4096
2048
1024
512
0
Hz
0 0.5 1 1.5 2 2.5 3 3.5 4
Time

(c)


Reference: वनस्पतयः मूलभूताः आवश्यकताः मलिनजलं सूर्यप्रकाशः च सन्ति
Synthesized: वनस्पतयः मूलभूताः आवश्यकताः मलिनजलं सूर्यप्रकाशः च सन्ति
8192
4096
2048
1024
512
0
Hz
0 0.5 1 1.5 2 2.5 3 3.5 4
Time

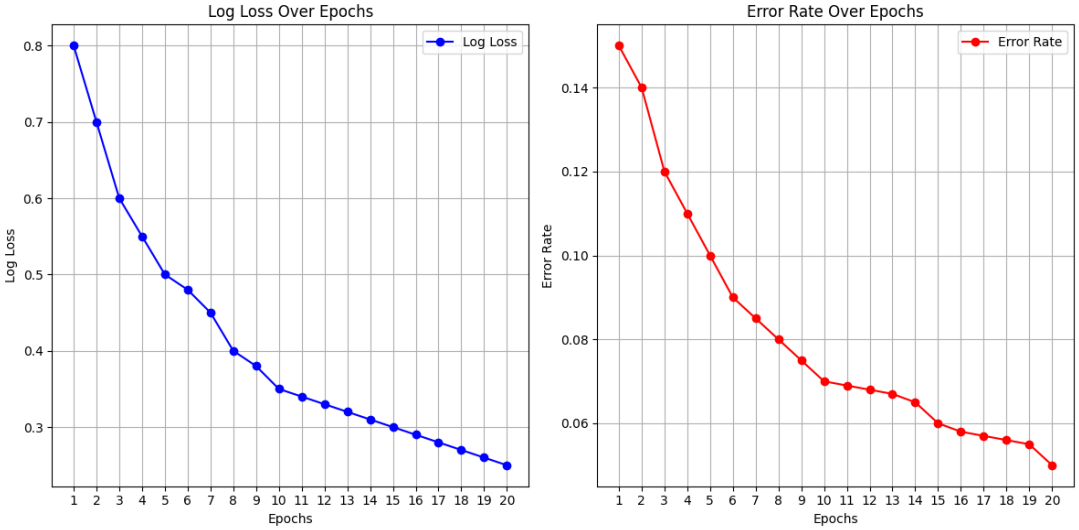
(d)



(e)

Figure 4: Comparison of Reference and Synthesized Spectrograms for Various Phrases in the TTS System

Figure 4 displays different spectrograms that compare synthesized and reference audio for various phrases proposed Sanskrit TTS system. The reference spectrogram is on the left, and the synthesized version is on the right, in each pair of spectrograms that are shown side by side. The comparison shows how well the TTS technology replicates normal speech patterns. In subplot (a), the synthesized audio nearly matches the reference in terms of frequency and time, with just minor changes in strength. Subplot (b) demonstrates that the synthesized output captures the general structure of the phonemes, with minor deviations in the higher frequency ranges. In subplot (c), the reference and synthesized spectrograms are very similar, with only very small variations in the transition between phonemes. Subplot (d) shows an excellent matching between the two versions, especially in the low-frequency areas, but some smoothing is visible in the synthesized version. Subplot (e) shows a largely accurate reproduction of the reference audio. Overall, the comparisons indicate that the TTS system is effective at producing natural-sounding speech, with only modest changes in frequency strength and transitions.



(a) **(b)**

Figure 5: (a) Log Loss over Epochs (b) Error Rate over Epochs

The two graphs in the figure 5 show the model's performance increase over 20 epochs, as measured by (a) log loss and (b) error rate. Training of the model is improving

the accuracy of its probability predictions, as seen by the left graph where the log loss steadily decreases from 0.8 in the first epoch to about 0.25 by the 20th epoch. This declining trend begins earlier in the epochs (1–5), indicating quick learning at first, and then it becomes more gradual as the model converges. The error rate is also plotted on the right graph, falling to approximately 0.015 by epoch 20 from a starting point of nearly 0.145 in the first epoch. Both graphs depict the model's successful learning process, with performance improving as the epochs progress.

4.2 Comparative analysis

In this section, a comparative analysis is presented to evaluate the performance of the proposed TTS system against existing models like Tacotron and Transformer encoder. The focus is on examining key metrics such as word error rates, mel-cepstral distortion, and speaker effectiveness to highlight improvements in speech synthesis accuracy and audio quality. Additionally, listener evaluations, including MOS and SMOS scores, provide insights into the naturalness and similarity of the synthesized speech to human speech. The analysis also explores classification accuracy by comparing true positive rates across different false positive rates, offering a comprehensive assessment of the proposed system's effectiveness in advancing TTS systems.

Table 1: Performance Comparison of GCNN and GRU Models in TTS Systems

Model	MCD(dB)↓	SECS↑
GCNN only	2.56	0.806
GRU only	2.48	0.814
GCNN+GRU (Proposed)	2.13	0.851

Table 1 illustrates the results of an ablation study evaluating the impact of various model elements on speaker control in the proposed TTS system. The GCNN model has an MCD (mel-cepstral distortion) of 2.56 dB and an SECS (Speaker Encoder Cosine Similarity) score of 0.806, whereas the GRU model has a slightly higher MCD of 2.48 dB and SECS of 0.814. With the lowest MCD of 2.13 dB and the greatest SECS score of 0.851, the proposed model, which combines both GCNN and GRU, performs better. According to these outcomes, combining the GRU and GCNN models improves speaker similarity, decreases distortion, and improves speaker control over employing a single model.

Table 2: Comparative Evaluation of TTS Models Using Key Metrics (Existing Models vs. Proposed Model)

Model	WER (%)↓	MCD(dB)↓	SECS↑	MOS↑	SMOS↑
Tacotron	6.59	5.835	0.7907	3.12	3.54
Transformer encoder	6.08	5.439	0.8120	3.8	3.71
Opt-transformer encoder (Proposed)	5.73	4.965	0.8970	4.98	3.89

A comparative analysis of TTS models based on important metrics is shown in Table 2, which also highlights the performance of the suggested optimised transformer encoder and existing models. The suggested model's WER is 5.73%, which is substantially lower than Tacotron and the Transformer encoder's WERs of 6.59% and 6.08%, respectively, suggesting better speech synthesis accuracy. Furthermore, the suggested model shows improved audio quality over existing models with a decreased MCD of 4.965 dB. Notably, the suggested model outperforms Tacotron and the Transformer encoder with a higher SECS score of 0.8970. Furthermore, the MOS of 4.98 demonstrates a significant improvement in perceived speech quality, while the SMOS of 3.89 suggests that listeners consider the synthesized speech to be more comparable to actual human speech than previous models. Overall, the optimized transformer encoder exceeds standard models on all major measures, demonstrating its effectiveness in TTS systems. Tacotron has greater word error rates because to its ineffective handling of speech, which affects audio quality and listening experience. The Transformer encoder improves slightly in accuracy, but it still problems with audio clarity and speaker efficacy. In contrast, the suggested optimised transformer encoder exceeds these existing models, displaying improved accuracy and audio quality. This improvement increases speaker effectiveness and audience satisfaction, demonstrating its usefulness in text-to-speech systems.

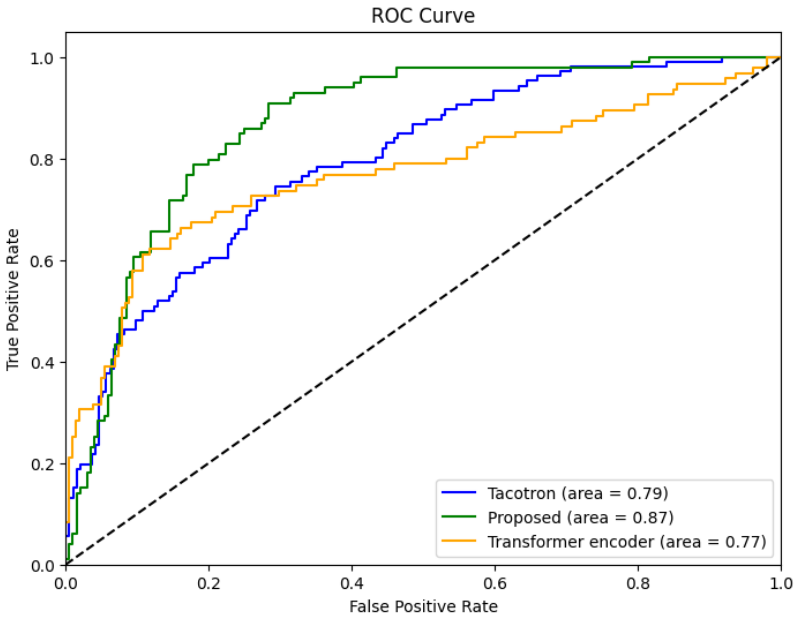


Figure 6: Comparison of ROC curve for proposed and existing models

Figure 6 illustrates a ROC curve that compares the True Positive Rate (TPR) vs False Positive Rate (FPR) of three models: Tacotron, Transformer encoder, and the proposed model. With an Area Under the Curve (AUC) of 0.8, the proposed model (green curve) performs most efficiently, considerably exceeding the Transformer encoder (orange curve) of AUC =0.77 and the Tacotron (blue curve) of AUC = 0.79. The model's ability to differentiate between positive and negative classes improves with the curve's proximity to the top-left corner. All models outperform the random classifier

shown by the dashed diagonal line with AUC = 0.5. When compared to the other two models, the proposed model demonstrates greater true positive rates throughout a variety of false positive rates, indicating that it provides more accurate classification.

4.3 Comparison with published work

In order to evaluate the efficacy of the proposed optimized transformer encoder for Sanskrit TTS synthesis, several of well-known speech synthesis techniques were chosen for comparison in this study. In Section 2, these reference models, which include the Transferable Monotonic Aligner (TMA) [11], the Tacotron 2-based neural vocoder [12], the Neural Codec Language Model (VALL-E) [13], and MOSNet [18], were examined and thoroughly explained.

Table 3: Comparing published works with various metrics

Ref	Technique	WER (%) ↓	MCD(dB)↓	SECS↑	MOS↑	SMOS↑
Yinghao Aaron Li and Cong Han et al., [11]	Transferable Monotonic Aligner (TMA)	8.26	-	-	2.43	-
Ye Jia et al., [12]	Neural vocoder based on Tacotron 2	10.14	-	-	3.65	-
Chengyi Wang et al., [13]	Neural codec language model (called VALL-E)	5.9	-	-	2.66	3.45
Rishabh Jain et al.,[18]	MOSNet	15.27	-	-	-	-
Opt-transformer encoder (Proposed)		5.73	4.965	0.8970	4.98	3.89

The table 3 presents findings that illustrate how well the optimised transformer encoder-based TTS model performs in comparison to other methods. The suggested model produces a WER of 5.73%, which is much lower than the WER of models like MOSNet with15.27%, Tacotron 2-based neural vocoder with10.14%, and TMA with 8.26%. Furthermore, as seen by its MCD of 4.965 dB, the suggested model has exceptional audio quality. The suggested system is noteworthy for its exceptional ability to capture speaker similarity, as demonstrated by its high SECS of 0.8970, which surpasses all other models. By achieving a MOS of 4.98 in perceptual assessments, the suggested model outperforms methods like VALL-E, which only obtain a MOS of 2.66, on the basis of overall naturalness and quality of the synthesized speech. With a score of 3.89, the SMOS indicates the model's superior ability to replicate the speaker's features compared to VALL-E, which scored 3.45. With significant gains in several important speech synthesis parameters, these findings clearly show the efficacy of the optimized transformer encoder and establish it as a very accurate and affordable Sanskrit TTS system.

5. Conclusion

The purpose of this study is to present an advanced and innovative TTS system specifically designed for Sanskrit, a language with immense historical and cultural significance that faces many challenges in speaking today. The suggested system efficiently solves the complexity of Sanskrit

phonetics by applying an optimized transformer encoder in conjunction with techniques such as G2P conversion, GCNN for local feature extraction, and GRU for temporal analysis. The application of the Adaptive Cheetah Optimisation algorithm to increase transformer model performance, as well as the HiFi-GAN vocoder for high-quality audio waveform creation, considerably improves the naturalness and speaker similarity of synthesized speech. The efficiency of the proposed technique has been evaluated using the Vāksañcayaḥ - Sanskrit Speech Corpus dataset. The system shows significant advances in generating accurate, human-like speech when measured by important metrics such as WER, MCD, SECS, and MOS and SMOS. These enhancements demonstrate the efficacy of the optimized transformer encoder and the integration of G2P, GCNN, GRU, and HiFi-GAN vocoders in addressing the problems of Sanskrit speech synthesis. The results suggest that this method has the potential to considerably enhance the accessibility and preservation of the Sanskrit language by providing a modern and efficient approach for producing highly accurate and natural speech. In the future, the suggested method could be enhanced by investigating transfer learning between languages with similar phonemes. Furthermore, research that uses pretrained language models is required to increase training performance and achieve greater alignment across identical resource instances.

References

[1] Harrison, K. David, (2007). When Languages Die: The Extinction of the World's Languages and The Erosion of Human Knowledge, Oxford University Press Local Language Speech Technology Initiative website http://llsti.org/

[2] I. Goodfellow et al., *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org

[3] Dunaev, A., 2019. A Text-to-Speech system based on Deep Neural Networks. *Bachelor Thesis*.

[4] Alastalo, A., 2021. Finnish end-to-end speech synthesis with Tacotron 2 and WaveNet.

[5] Rama, GL Jayavardhana, A. G. Ramakrishnan, R. Muralishankar, and R. Prathibha. "A complete text-to-speech synthesis system in Tamil,"In Proc. 2002 IEEE Workshop on Speech Synthesis, pp. 191-194. IEEE, 2002.

[6] Kumar, H. R. S., J. K. Ashwini, B. S. R. Rajaram and A. G. Ramakrishnan. "MILE TTS for Tamil and Kannada for Blizzard Challenge 2013." In Blizzard Challenge Workshop, vol. 2013.

[7] B S R Rajaram, H R Shiva Kumar, and A G Ramakrishnan, "MILE TTS for Tamil for Blizzard challenge 2014", In Blizzard Challenge Workshop, vol. 2014.

[8] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in INTERSPEECH, 2017.

[9] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A.

Saurous, Yannis Agiomyrgiannakis, and Yonghui. Wu, ”Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions,” In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2018.

[10] Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al., ”Deep voice: Real-time neural text-tospeech,” arXiv preprint arXiv:1702.07825, 2017

[11] Li, Y. A., Han, C., &Mesgarani, N. (2022). Styletts: A style-based generative model for natural and diverse text-to-speech synthesis. ArXiv preprint arXiv:2205.15439.

[12] Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., ... & Wu, Y. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis. Advances in neural information processing systems, 31.

[13] Wang, C., Chen, S., Wu, Y., Zhang, Z., Zhou, L., Liu, S., ... & Wei, F. (2023). Neural codec language models are zero-shot text to speech synthesizers. ArXiv preprint arXiv:2301.02111.

[14] Kumari, R., Dev, A., & Kumar, A. (2021). An efficient adaptive artificial neural networkbased text to speech synthesizer for Hindi language. Multimedia Tools and Applications, 80(16), 24669-24695.

[15] Ni, J., Wang, L., Gao, H., Qian, K., Zhang, Y., Chang, S., & Hasegawa-Johnson, M. (2022). Unsupervised text-to-speech synthesis by unsupervised automatic speech recognition. ArXiv preprint arXiv:2203.15796.

[16] Chen, M., Chen, M., Liang, S., Ma, J., Chen, L., Wang, S., & Xiao, J. (2019, September). Cross-Lingual, Multi-Speaker Text-To-Speech Synthesis Using Neural Speaker Embedding. In Interspeech (pp. 2105-2109).

[17] Li, N., Liu, Y., Wu, Y., Liu, S., Zhao, S., & Liu, M. (2020, April). Robutrans: A robust transformer-based text-to-speech model. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 05, pp. 8228-8235).

[18] Jain, R., Yihas been, M. Y., Bigioi, D., Corcoran, P., &Cucu, H. (2022). A text-to-speech pipeline, evaluation methodology, and initial fine-tuning results for child speech synthesis. IEEE Access, 10, 47628-47642.

[19] D. Min, D.B. Lee, E. Yang, S.J. Hwang, in *International Conference on Machine Learning*. Meta-StyleSpeech: multi-speaker adaptive text-tospeech generation (PMLR, 2021), p. 7748–7759

[20] Khan, A. and Sarfaraz, A., 2019. RNN-LSTM-GRU based language transformation. *Soft Computing*, *23*(24), pp.13007-13024.

[21] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization. (2016). arXiv preprint arXiv:1607.06450

[22] Shang, W., Chiu, J. and Sohn, K., 2017, February. Exploring normalization in deep residual networks with concatenated rectified linear units. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 31, No. 1).

[23] Sait, S.M., Mehta, P., Gürses, D. and Yildiz, A.R., 2023. Cheetah optimization algorithm for optimum design of heat exchangers. *Materials Testing*, *65*(8), pp.1230-1236.

[24] Fan, J., Li, Y. and Wang, T., 2021. An improved African vultures optimization algorithm based on tent chaotic mapping and time-varying mechanism. *Plos one*, *16*(11), p.e0260725.

[25] Kong, J., Kim, J. and Bae, J., 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, *33*, pp.17022-17033.

Authors:

Ms. Sabnam Kumari received the B.Tech degree in computer science and engineering from GGSIPU, New Delhi in 2011 and M.Tech. degree in computer science and engineering from MDU, Rohtak, Haryana, India, in 2013, and pursuing Ph.D. degree from the Department of Computer Science and Engineering, Deenbandhu Chhotu Ram University of Science and Technology (DCRUST), Murthal, Sonepat, Haryana, India, with nearly 12 years of experience in academic and research affairs. She has authored or co-authored more than 30 research papers in various international/national journals and conferences and 2 patents to her credit. She has guided 8 M.Tech. dissertations. She is a member of the International Association of Engineers (IAENG), Internet Society Chapter (ISOC) Delhi, Her research interests include Natural Language Processing, Text Analytics, Artificial Intelligence and Machine Learning.

Dr. Amita Malik earned her Ph.D. in Computer Engineering from the National Institute of Technology, Kurukshetra in 2010. She is presently working as Professor in the Department of Computer Science and Engineering at DCRUST, Murthal, Sonepat, India with nearly 24 years of teaching experience. She has published more than 60 research papers in various International /National Journals and Conferences of repute. She has guided 06 Ph.D thesis and 15 M.Tech dissertations. Her research interests include Mobile Ad hoc and Wireless Sensor Networks, Scale-Free Networks, Cloud and Edge Computing, Machine Learning, Text Analytics and Blockchain Technology