Context-Aware Speech Generation Using BiLSTM-Based Neural Networks

K. Jeevan Reddy[1]; Dr. Syed Jahangir Badashah[2];
S. Tharun Govind[3]; M. Dinesh[4]; K. Bindusri[5]

[1,2] Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology, Assistant Professor, 501301, Hyderabad, India

[3,4,5] Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology, Undergraduate, 501301, Hyderabad, India

Publication Date: 2025/07/11

Abstract: According to recent studies, feed-forward Deep neural networks (DNNs) perform better than text-to-speech (TTS) systems that use decision-tree clustered context-dependent hidden Markov models (HMMs) [1, 4]. The feed-forward aspect of DNN-based models makes it difficult to incorporate the long-span contextual influence into spoken utterances. Another typical strategy in HMM-based TTS for establishing a continuous speech trajectory is using the dynamic characteristics to constrain the production of speech parameters [2]. Parametric time-to-speech synthesis is used in this study by capturing the co-occurrence or correlation data between any two points in a spoken phrase using time aware memory network cells. Based on our experiments, a combination of DNN and BLSTM-RNN is the best system to use. Upper hidden layers of this system use a bidirectional RNN structure of LSTM, the low layers use a simple, one way structure followed by additional layers. On objective and subjective metrics, it surpasses both the traditional decision-based tree HMM's and the DNN-TTS system. Dynamic limitations are superfluous since the BLSTM-RNN TTS produces very smooth speech trajectories.

Keywords: *Bidirectional Long Short-Term Memory(BLSTM), Deep_Neural_Network, Recurrent_Neural_Network, Statistical Parametric Voice Synthesis, hidden Markov Model.*

How to Cite: K. Jeevan Reddy; Dr. Syed Jahangir Badashah; S. Tharun Govind; M. Dinesh; K. Bindusri (2025) Context-Aware Speech Generation Using BiLSTM-Based Neural Networks. *International Journal of Innovative Science and Research Technology*, 10(7), 270-277. https://doi.org/10.38124/ijisrt/25jul358

I. INTRODUCTION

Data Analysis using Parameters A few of the ways in which the TTS Synthesis system excels above the competition are its compact size, durability, and adaptability to different voice types [1]. Because of these benefits, parametric TTS synthesis based on GMM-HMM has become the standard for many speech synthesis applications, especially on mobile devices [2]. By treating the spoken audio as random-like series of the sound detail vectors, a parametric hidden Markov model (HMM) is able to accurately describe their development. For HMM-based voice recognition, many methods have been devised, such as context-dependent modeling, speaker adaption, and grouping states for speech creation. They have also proven effective in generating the speech parameter trajectories in HMM-based TTS. Concatenation mistakes are rare in unit-selective synthesis, even though trained HMMs often provide very smooth trajectories. One issue is that when training HMMs statistically, parameter trajectories are unduly smoothed, which might make the synthetic speech seem less natural than it really is.

Recent DNN research has focused on improving vocoder-based voice synthesis. In their proposal to use DNN for voice synthesis, Zen et al. [3,23] addressed the shortcomings of the traditional HMM-based method, including decision-tree based contextual state grouping. Given an equivalent number of model parameters, it demonstrates that, when modeling the link between the input texts and their associated acoustic properties, a DNN-based strategy may perform better than an HMM-based technique. Qian et al. [4] investigated training parameters, including activation functions and weights initialization in "pre-training," for DNN based TTS synthesis using a modest size corpus, which is more typical for parametric TTS training. In their DNN-based synthesis, Lu et al. [5] used continuous or binary data such as phone numbers, letters, and Vector Space Models (VSMs) as input characteristics. Either "frame" or "state" were the predicted outputs.

One may model the simultaneous distribution of auditory and linguistic input in speech synthesis using stacked restricted-Boltzmann-machines (RBMs) in a Deep Belief Network (DBN). In order to avoid over-fitting during the discriminative "fine-tuning" phase of speech recognition, these networks may identify a region of the weight-space that is conducive to generative "pre-training" of the input data [6]. The reason RBM's mode is used to represent the spectrum envelop distribution for each HMM state in [8] is because it produces higher-quality produced voices compared to GMM's mean. With contrast to its HMM-based alternative, DNN sidesteps the issues associated with contextual state grouping based on decision trees. Improving context forecasting for testing is the goal of this approach, which employs extended context states linked to more general ones. In addition to compactly modeling very complicated mapping functions, DNN effectively represents high-dimensional and linked characteristics. Unfortunately, DNN is still flawed since it relies on simplistic modeling units such as states or frames. Particularly targeted contexts are used as input characteristics to capture co-articulation effects and imitate actual intonation. Synthesizing smooth parameter trajectories for speech is possible by mixing dynamic model parameters with their static equivalents [2].

We explore the possibility of using RNNs with bidirectional Long Short Term Memory (LSTM) cells [9–13] as a model for TTS synthesis since they may possibly receive input from any point in the feature sequence. By retrieving the hidden states that unfolded before to or subsequent to RNNs, a complete temporal framework may be constructed. Similar to DNNs, RNNs may develop deep spatial structures via layer stacking.

II. LITERATURE SURVEY

The possibilities of this technology in everyday life are truly remarkable! If you're looking to polish your English, there's this fantastic software developed by Jose and his crew.

It's designed to help folks correct their pronunciation using standard phonetics. Now, shifting gears to Sivakumar and his team—they did some interesting research comparing the pros and cons of speech recognition systems with different vocabulary sizes. They set up a bunch of test scenarios, like noisy backgrounds and jumbled words, to really show how a language model can boost the accuracy of speech-to-text.

We've got Yogita and her team, who created a system for digitizing sound across various languages. They used a method called Mel-Frequency Cepstral Coefficient (MFCC), along with a Minimum Distance Classifier and Support Vector Machine (SVM) algorithms, to refine speech processing. Some researchers also found out how to use the open-source Sphinx 4 tool to turn actual Bengali speech into written text.

They reported an accuracy rate of 71.7% with their algorithm. Then there's Wan, who took a different path by developing a process to summarize English documents based on semantic associations. He claims that this new method is more effective and precise than others. Many people actually prefer LDA for topic-sensitive document classification because it's pretty straightforward for measuring similarity.

Their findings could really shake things up for others trying to nail down the best strategies for their projects. They even experimented with a multi-objective optimization model for selecting words. And guess what? They employed a human-learning optimization technique!

III. SYSTEM DESIGN AND METHODOLOGY

➢ *Deep Bidirectional-LSTM(DBLSTM)*

$$h_t = \mathscr{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = W_{hy}h_t + b_y \quad (2)$$

At this instance bias vectors, which make up the hidden state vectors, reflect the weight matrices, which are composed of both input and hidden vectors. For hidden nodes, the nonlinear activation function can be expressed as ^. $\mathscr{H}$ is frequently the hyperbolic/sigmoid curve function in regular RNNs. However, RNNs are unable to explain extended connections in sequence data due to the gradient disappearing issue. Figure 1 shows the long short term memory (LSTM) network, which is one way to fix the issues with regular RNNs [11]. This network can represent signals with low- and high-frequency components by hand-constructing an internal memory cell. The following routines are used to implement *h* in LSTM [12]:

$$i_t = \sigma(W_{xi}\,x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (3)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (4)$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (5)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (6)$$

$$h_t = o_t tanh(c_t) \quad (7)$$

in where sigmoid is the relevant function, # is the output gate symbol, and is the input gate symbol.

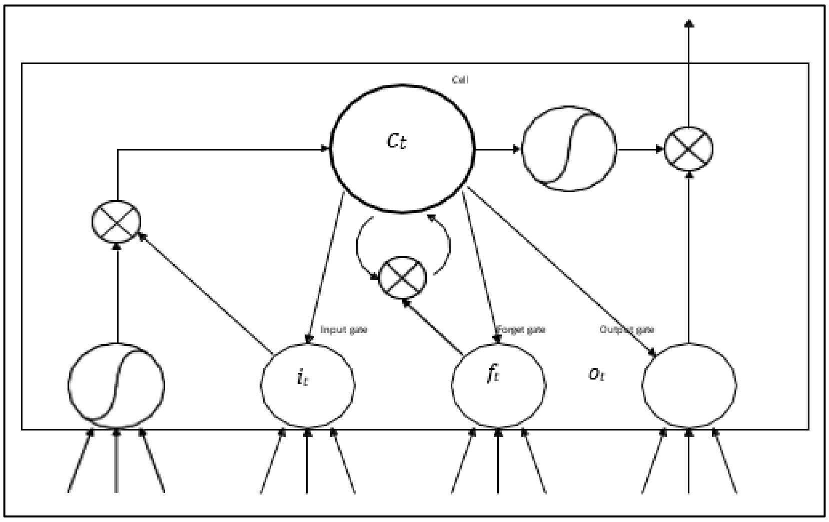

Fig.1. Long Short Term Memory Bidirectional RNN

[13] may be able to access both of the scenarios shown in Figure 2. It tells the difference between the forward state sequence (h%□) and the backward state sequence ($\overleftarrow{h}$%) of the hidden layer. This approach is iterative.

$$\overrightarrow{h}_t = \mathcal{H}(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \quad (8)$$

$$\overleftarrow{h}_t = \mathcal{H}(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \quad (9)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y \quad (10)$$

A deep-bidirectional-RNN may have been built by forming many RNN_hidden layers. In the iterative process, the two way hidden states, denoted as $\overline{h}$□n, $\overleftarrow{h}$bn, respectively, are substituted for each hidden state.

$$\overrightarrow{h}^n = \mathcal{H}(W_{\overrightarrow{n+1}\,\overrightarrow{n}} \overrightarrow{h}^{n-1} + \overline{\overline{W}}_{\overrightarrow{n}\,\overrightarrow{n}} \overrightarrow{h}^{n} + b^n) \quad (11)$$

$$\overleftarrow{h}^n = \mathcal{H}(W_{\overleftarrow{n+1}\,\overleftarrow{n}} \overleftarrow{h}^{n-1} + W_{\overleftarrow{n}\,\overleftarrow{n}} \overleftarrow{h}^{n} + b^n) \quad (12)$$

$$y_t = \overline{\overline{W}}_{\overrightarrow{N}} \overrightarrow{h}^N + W_{\overleftarrow{N}} \overleftarrow{h}^N + b_y \quad (13)$$

The DBLSTM is the product of merging LSTM with deep bidirectional RNN. In order to mimic the long-span deep feature representation, it combines the finest features of DNN and LSTM.

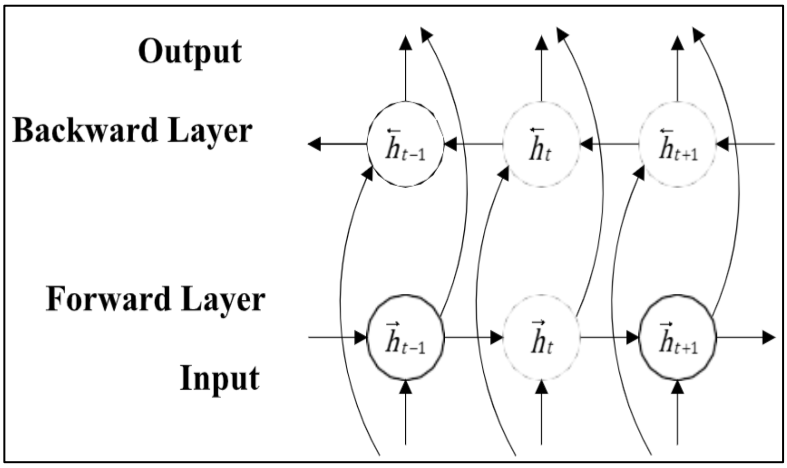
Output
Backward Layer
Forward Layer
Input
$\overleftarrow{h}_{t-1}$
$\overleftarrow{h}_t$
$\overleftarrow{h}_{t+1}$
$\overrightarrow{h}_{t-1}$
$\overrightarrow{h}_t$
$\overrightarrow{h}_{t+1}$

Fig.2. Bidirectional RNN

- *Technical Specifications*

- Operating System: Windows XP or Fedora Core-I
- Software: jdk version 1.4
- x86 class processor
- 32 MB RAM
- 1 GB Hard Disk Space
- Speaker connected to the computer

- *DBLSTM-RNN based TTS Synthesis*

Words, learn their phonetics, and then use the articulators to produce speech. So it is an ongoing substances' shifting compositions. When given language data, the deep layered hierarchical network (DBLSTM-RNN) may generate speech that sounds frighteningly like human speech. Since a TTS synthesis system often receives an entire phrase as input, it would be imprudent ignoring wide-range context from both past and future words. We suggest synthesizing TTS using DBLSTM-RNN. Figure 3 shows the structure of the TTS synthesis utilizing DBLSTM-RNN.

$$f_t=(W_{xf}x_t+W_{hf}h_{t-1}+W_{cf}c_{t-1}+b_f) \quad (14)$$

$$c_t=f_tc_{t-1}+i_ttanh(W_{xc}x_t+W_{hc}h_{t-1}+b_c) \quad (15)$$

$$o_t=(W_{xo}x_t+W_{ho}h_{t-1}+W_{co}c_t+b_o) \quad (16)$$

$$h_t=o_ttanh(c_t) \quad (17)$$

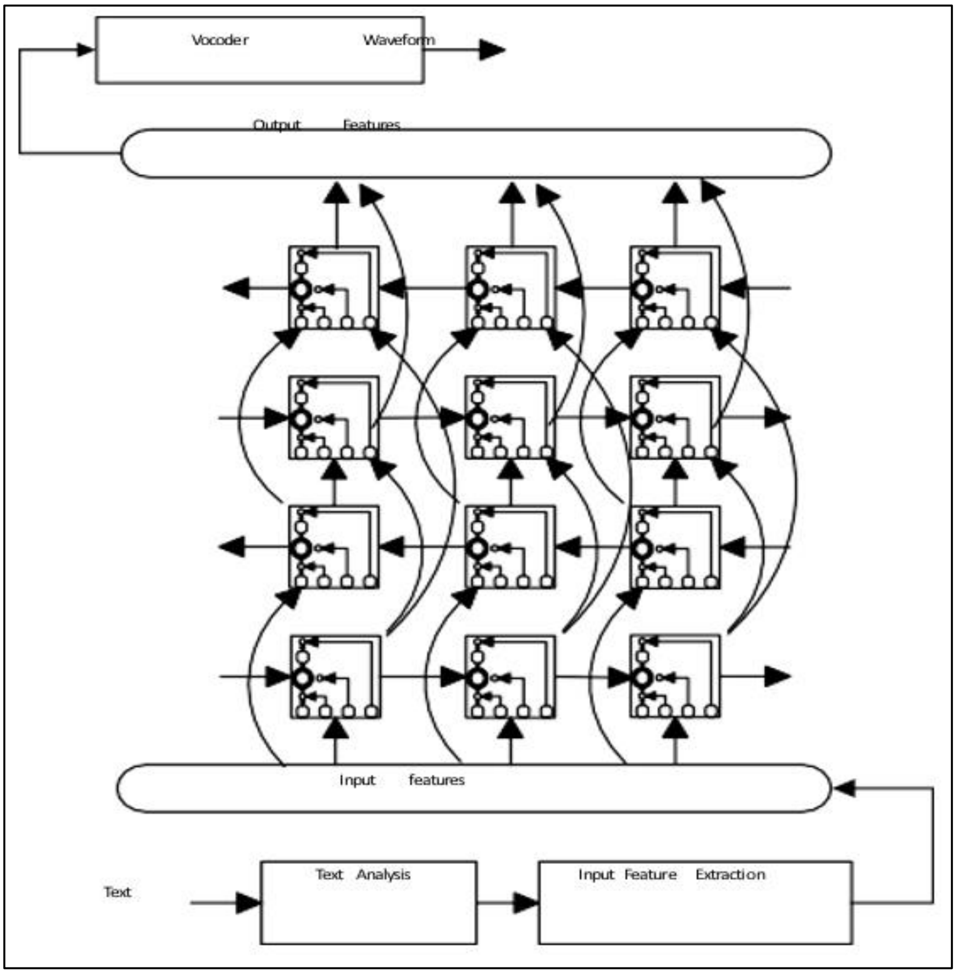

Fig.3.DBLSTM-RNN based TTSsynthesis

The DBLSTM-RNN based TTS synthesis takes input features from several rich contexts. In categorical situations, binary information may take the form of phone numbers, word POS labels, or TOBI labels; in numerical circumstances, it can take the shape of the sentence's word count or the current phone frame's position. Features of sound, such as the frequency range of audible sounds and vital frequency (FF). You may use a well-trained HMM model to time-align the input and output attributes frame-by-frame. If the input-output alignment is uncertain, RNN may be a valuable tool for modeling sequential data in Sequence Transduction [15] and Connectionist Temporal Classification [14]. The objective of learning DNN weights from training data using input/output pairs is to reduce the mismatch between the mapped and intended outputs as much as possible.

Mean-Square-Error, disparity between the model results along with the (GT)ground truth, is what RNN training is all about. When training RNNs, backward propagation through the time is a common technique. Unfolded network ready for back--propagation training, BPTT gradually converts the RNN into a feed-forward network. A deep bidirectional LSTM employs the BPTT method of back-propagation by use of layers on both the forward and backward hidden nodes. To train a DNN, one uses a back-propagation technique in conjunction with a stochastic gradient descent algorithm. This methodology uses a "mini-batch" selection method to randomly choose frames from data used to help the modern learning weights. It is common practice to calculate weight gradients constantly during speech while training RNNs. The RNN weights are updated simultaneously for each update by randomly selecting tens of utterances. The procedure is accelerated since this enables parallelization.

By analyzing the text input, a trained DBLSTM-RNN may create an input feature vector, which it can then utilize to create output vectors during synthesis. With the help of the contextual label, HMM-based TTS is able to enter the learning-trees and get the context based HMM sequences of state. Parameter generation module uses corresponding-means and measure of joint variable of HMM quantities to build smooth parameter trajectories for speech based on the dynamic information. As part of DNN-based TTS, the sound characteristic extractor may be used to create steady speech feature paths that meet the requirements of both static and dynamic features. A vocalization marker will be set according to a DNN decision boundary. This is accomplished by using pre-computed (global) variances for all training data and mean vectors that reflect the DNN's predicted output features. Our proposed DBLSTM-RNN based TTS synthesis makes use of RNN's parameter generation module, which it uses for sequential problem modeling. Timbre contour, Loudness level, core pitch, and voicing marker are the static output properties of the RNN that we immediately feed into a vocoder in order to generate the final voice waveform.

With DBLSTM-RNN's better learning capabilities, TTS voice synthesis should be improved. To put it another way, multi-layered architectures may be able to depict changes across great distances with relative ease [16].K-tier deep design, in contrast to the low depth network design and parameter rich model, may enable the functions to be more compactly represented and increase performance. The decision tree, often used in HMM-based TTS to group together related context-dependent states, is an example of a shallow architecture. According to the conditional independence requirement, every observation is completely dependent on the states that generate them. It's possible that HMMs, similar to RNNs and DNNs, may overcome this limitation. Inputted with either the FTT or HMM state, DNN is limited to simulating the connection between text and voice at the frame level and can only generate speech by alternating between the FTT states. By using the same infinite states as input, random NNs may learn from nearby frames, extract several hidden states from a single input, and avoid limitations imposed by discrete input states.

IV. EXPERIMENTS

- *Experimental Setup*

A collection of American English samples spoken by innate female speakers is used to administer our examinations. There is a wealth of prosodic and phonetic data in this corpus. Five hours' worth of training utterances make up the corpus, with an additional 200 utterances used for testing purposes. Every 5 milliseconds, the speech signals are shifted, and every 25 milliseconds, they are windowed. The samples are taken at 16 kHz. A 40th-order LPC is created, both dynamically and statically. Examples of phonetic and prosodic contexts include quin-phones, syllable and word placements within phrases and sentences, word and phrase length, syllable stress, TOBI, and POS of words.

One Gaussian diagonal covariance output distribution is used for each of the five states in the most basic TMS phone models based on HMM. The prosodic and phonetic scenarios employed in the role of a query collection for constructing trees which are choice trees. (MDL) Minimal Description Length criterion is being used to decide when to stop condition for grouping the states in a decision tree development to strike balance the amount of training data with complexity of the models. Two training stages—highest probability criterion and lowest output power strategy—are used to fine-tune the HMM parameters. To reduce the generation error between the original parameter trajectories of the training data and the synthetic ones, it tweaks the HMM parameters constrained by the traditional EM technique [18].

For numerical linguistic settings, 319 of the 355 dimensions in the input feature vector of the DNN-based TTS are binary features, while the remaining dimensions are characteristics for categorical linguistic contexts. Includes voicing indicator flag, logF0, LSP, gain, and their time varying versions; the output feature vector has 127 dimensions. The voicing of the present frame is indicated by the binary feature known as voiced/unvoiced flag. The interpolation of F0 is accomplished using an exponential decay function [19] to fill in the gaps in voiceless speech. Removing 80% of the quiet frames reduces computational effort and ensures that the training data is balanced [3]. Researchers found that excluding silence frames from DNN training helped prevent them from being very adept at speech recognition tasks including the silence label. The input and output characteristics are centered and scaled into one. We employ a back-propagation procedure and a stochastic gradient descent method based on "mini-batch" to train the weights.

A feature vector is used as input in DBLSTM-RNN-based TTS, much as in DNN-based TTS. The difference is that the output feature vector does not include the dynamic versions of voiced/unvoiced flag, logF0, LSP, or gain; instead, it remains with 43 dimensions. In addition, we can solely use the information from the present frame data as the feed without surrounding levels. Not eliminating quiet levels during RNN training preserves the continuity of acoustic properties within a phrase. We use the CURRENT RNN machine learning library [21] in our research.

To create dynamic constrained smooth features, the testing utterances are fed into a module that creates parameters together with the outputs of the HMM and DNN. Applying this step to RNN outputs is not required.

To address the problem of over-smoothing in statistical parametric modeling and the resulting "muffled" speech, a technique called tonal landmark uplift [24] is being utilized. The last stage involves feeding the generated speech parameters into an LPC synthesizer, which will then generate waves of synthetic speech.

- *Training DBLSTM*

Training DBLSTM-RNN still requires much more computing power than training DNN, even with the use of GPGPU. In comparison to DNN, DBLSTM-RNN uses a much larger number of model parameters, although having similar total quantity of hidden nodes and hidden layers in each layer. In order to keep the parameter count comparable to the HMM and DNN models, we run our tests by replacing and wrapping the context layers of BLSTM. Preliminary results on a 1,000-word training corpus imply that DBLSTM-RNN's deeper structures don't always attain better performance, even when "layer-wise BP" is used during training [20]. We postulate that, particularly when faced with inadequate training data, DBLSTM-RNN's inherent temporal and spatial depth leads to an incorrect gradient descent computation.

To train TTS systems using HMM, DNN, and DBLSTM-RNN, you may follow these steps:

- Applying Hidden Markov Models (HMM) with MDL=1 to LSP and F0 decision trees Two, a DNN A with six hidden layers and 512 hidden nodes
- Utilizing a 1024-nodesper layer in one of DNN_B's three hidden layers
- Hybrid_A combines DNN with BLSTM-RNN. In a feed-forward configuration, the first three 512-node hidden layers use sigmoid activation functions; the fourth, a bidirectional RNN with LSTM, has 256-ahead and behind processors.
- Hybrid_B: this variant is quite similar to Hybrid_A, but instead of two bottom-level hidden layers using sigmoid activation functions, it has two upper-level layers using BLSTM-RNN, with 256-ahead and behind processors, respectively.

V. EVALUATION RESULTS AND ANALYSIS

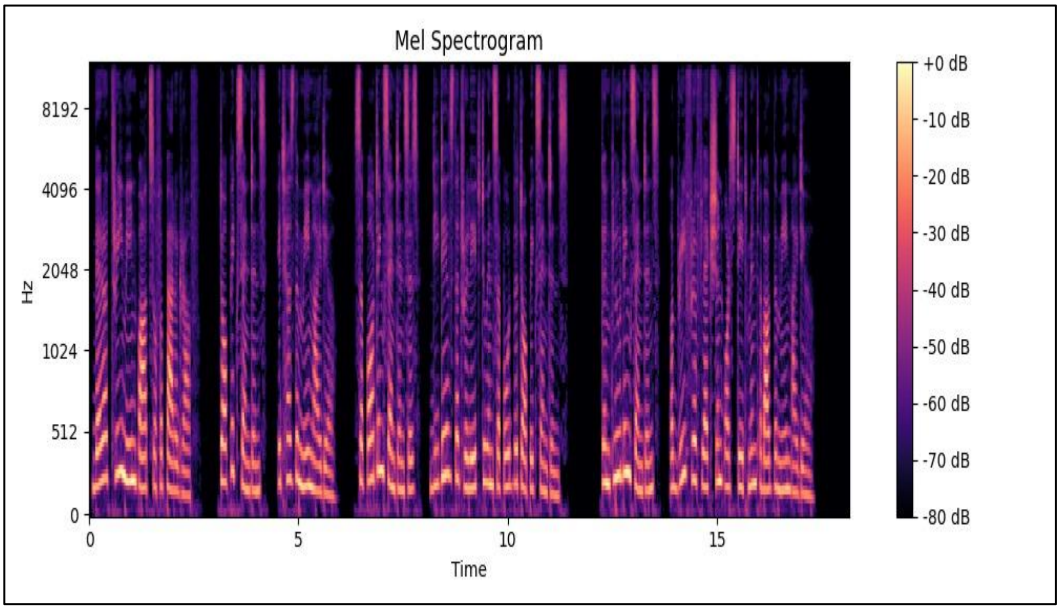

Fig.4. Mel Spectrogram Output

Based on the results of the tests, we evaluate three TTS systems using both measurable and personal assessments. One way to objectively synthesis quality check by contrasting the distorted synthesized speech to the original speaker's legitimate test utterances using oracle state durations, which are created by forcing the alignment of real speech. Finding the root-mean-squared error (RMSE), spectrum standard deviation in log spectral distance (LSD), and amount of voiced/unvoiced (V/U) switching mistakes are all desirable results. Once MDL reaches 1, we no longer use sound pitch state grouping trees that train HMM. Our prior practice run has shown that HMM inner adjustments are taught from a learning point of view and then changed via error reduction learning, objective measurements are poorer with either bigger or smaller MDL parameters. We discovered that all three objective measures stayed close to the same while training DNNs with three or six hidden layers, each with 512 or 1024 nodes [...]. via analyzing and comparing the outcomes of several quantities to maximize performance on spectra by using the hybrid system to increase the LSD of created and naturally occurring spectra trajectories by more than 0.1 dB. Similar to DNN, hybrid employs both simulated and actual F0 trajectories to calculate the root-mean-squared error.

VI. FINDINGS AND PERFORMANCE ASSESSMENT

➢ *Outcomes of HMM, DNN and Hybrid trainings*

As a subjective criterion, we assess the two selected systems' ability to synthesize pairs of spoken phrases using an AB preference test. The outcomes of the objective metrics collected during the Hybrid, DNN, and HMM training periods as shown in Table 1.

Table 1. The Results of Objective Measures of Different Configurations (the Number of System Parameters) in HMM, DNN and Hybrid Trainings

Measures Model	LSD (dB)	V/U Error rate	F0 RMSE (Hz)
HMM (2.89M)	3.74	5.8%	17.7
DNN_A (1. 55M)	3.73	5.8%	15.8

DNN_B (2.59 M)	3.73	5.9%	15.9
Hybrid_A (2.30M)	3.61	5.7%	16.4
Hybrid_B (3.61M)	3.54	5.6%	15.8

To further subjectively assess the efficacy of HMM, DNN, and Hybrid systems, perceptual tests are available. Sixty people took part in the study by completing three AB preference tests on Amazon Mechanical Turk. In order to synthesize, we randomly selected utterances from the testing set and employed the best baseline HMM system (MDL=1), DNN system (DNN_B), and hybrid systems (Hybrid_A and Hybrid_B) [22]. Each person wears a headset and tests fifty pairings. I can give you three options: Three potential results exist: There are three potential outcomes: 1) the first is superior, 2) the second is superior, and 3) neither is preferred nor neutral; in other words, it's hard to determine which sentence is better. Image 4 displays the preference-based ratings. At the $p < 0.001$ level, the findings show that the Hybrid system's synthesized speech performs much better than the top HMM and DNN systems. The DNN system and the HMM system both have preference ratings of 20%, however this one is 55% higher and 59% higher, respectively. The following URLs provide examples of synthetic speech: http://research.microsoft.com/en-us/projects/dnntts/default.aspx.

VII. DISCUSSION

The prospect of more expressive and realistic synthetic voices is quite exciting, and that's all made possible by text-to-speech (TTS) technology. I mean, imagine if the sounds produced were super smooth, without all those annoying artifacts and choppy bits. It'd be like listening to a human voice. That's definitely something that needs work. And speaking of sounding human, these TTS systems could really up their game by incorporating prosody. What's that? Well, it's basically the melody of speech—intonation, stress patterns, rhythm.

If they could get that down, they could express emotions such as surprise, happiness, or even worry. I believe if they had context-sensitive pronunciation rules for difficult things like dates, numbers, acronyms, and those nasty homographs, it would make synthetic speech so much more understandable and clearer in our daily lives. Ahead of us, I imagine quite possibly that we will get TTS systems with a whole range of customizable voices.

We may have the choice of different ages, genders, dialects, and even styles! And how about graphical avatars or "faces" that progress in sync with the speech? It'd provide for more immersive conversation, without a doubt. But it's not merely sounding nice or looking trendy—practicality would be involved too. Having the ability to export and import files in PDF, Word, and HTML formats might truly be helpful in professional as well as education-based environments. For people who have to read text to speech on a daily basis, that would be saving them loads of time. And let's not mention user control. People ought to be able to control the tempo and pace of the reading. This way, those who are still learning the language can go at their own pace, and others who simply want to skim can make it faster. It's all about making it work for everybody.

VIII. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This work aims to use BiLSTM-RNN in training a data driven model for the TTS. Nodes in the first or second internal processing layers of a DNN can be swapped out for bidirectional LSTM RNN nodes while maintaining the same amount of model parameters. The experiment's findings showing how the Hybrid BLSTM-RNN and DNN system performs better than both HMM and DNN in gathering complex details in a sentence. Our long-term goal is to study DBLSTM-RNN with a more comprehensive structure and a larger corpus.

It demonstrates with this whole experiment that a text-to-speech system actually works. Individuals can simply type what they want and then hear it read back to them. The system is pretty good at translating written words into spoken words, making digital information more usable and interactive. But the catch is, it's still not quite there yet when it comes tosounding natural. We might benefit from better prosody, intonation, and maybe a touch of emotional flavour to convey meaning and emotion truly behind the words. In front of us, I think, lies some interesting new developments—like context-based rules for inclusion of such things as dates, acronyms, and numbers.

That should further enhance accuracy. And, if they include some graphical avatars and let people customize their voice profiles, it could make the whole experience that much more interactive. And, being able to manage how fast or slow the speech plays may be attractive to different listening habits. They're going to take this platform to the next level, making it seem like an actual conversation rather than reading some text.

REFERENCES

[1]. H. Zen, K. Tokuda, and W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039-1064, 2009.

[2]. K. Tokuda, T. Kobayashi, T. Masuko, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Proc. ICASSP*, pp. 1315-1318, 2000.

[3]. H. Zen, A. Senior, and M. Senior, "Statistical parametric speech synthesis using deep neural networks," in *Proc. ICASSP*, pp. 8012-8016, 2013.

[4]. Y. Qian, Y.-C. Fan, W.-P. Hu, and F. K. Soong, "On the training aspects of deep neural network (DNN) for parametric TTS synthesis," in *Proc. ICASSP*, 2014.

[5]. H. Lu, S. King, and O. Watts, "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis," in *Proc. 8th ISCA Workshop on Speech Synthesis*, pp. 281-285, 2013.

[6]. G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82-97, 2012.

[7]. S. Kang, X. Qian, and H. Meng, "Multi-distribution deep belief network for speech synthesis," in *Proc. ICASSP*, pp. 7962-7966, 2013.

[8]. Z.-H. Ling, L. Deng, and D. Yu, "Modeling spectral envelopes using restricted Boltzmann machines for statistical parametric speech synthesis," in *Proc. ICASSP*, pp. 7825-7829, 2013.

[9]. A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, pp. 6645-6649, 2013.

[10]. A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. IEEE ASRU*, pp. 273-278, 2013.

[11]. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[12]. F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, pp. 115-143, 2003.

[13]. M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997.

[14]. A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, Pittsburgh, USA, 2006.

[15]. A. Graves, "Sequence transduction with recurrent neural networks," in *ICML Representation Learning Workshop*, 2012.

[16]. Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009.

[17]. K. Shinoda and T. Watanabe, "MDL-based context-dependent sub-word modeling for speech recognition," *J. Acoust. Soc. Jpn. (E)*, vol. 21, no. 2, pp. 79-86, 2000.

[18]. Y.-J. Wu and R. H. Wang, "Minimum generation error training for HMM-based speech synthesis," in *Proc. ICASSP*, 2006.

[19]. C. J. Chen, R. A. Gopinath, M. D. Monkowski, M. A. Picheny, and K. Shen, "New methods in continuous Mandarin speech recognition," in *Proc. EUROSPEECH*, 1997.

[20]. F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE ASRU*, 2011. [Online]. Available: http://sourceforge.net/projects/currennt/

[21]. H. Zen, "Deep learning in speech synthesis," *Proc. ISCA SSW8*, 2013. [Online]. Available: http://research.google.com/pubs/archive/41539.pdf

[22]. Z.-H. Ling, Y.-J. Wu, Y.-P. Wang, L. Qin, and R.-H. Wang, "USTC system for Blizzard Challenge 2006: An improved HMM-based speech synthesis method," in *Proc. Blizzard Challenge Workshop*, 2006.