JULY 27 2020

Efficient two-microphone speech enhancement using basic recurrent neural network cell for hearing and hearing aids[a]

Special Collection: Machine Learning in Acoustics

Nikhil Shankar ; Gautam Shreedhar Bhat; Issa M. S. Panahi

J. Acoust. Soc. Am. 148, 389–400 (2020)
https://doi.org/10.1121/10.0001600

JASA ARTICLE

Efficient two-microphone speech enhancement using basic recurrent neural network cell for hearing and hearing aids[a)]

Nikhil Shankar,[b)] Gautam Shreedhar Bhat, and Issa M. S. Panahi
Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, Texas 75080, USA

ABSTRACT:
This work presents a two-microphone speech enhancement (SE) framework based on basic recurrent neural network (RNN) cell. The proposed method operates in real-time, improving the speech quality and intelligibility in noisy environments. The RNN model trained using a simple feature set—real and imaginary parts of the short-time Fourier transform (STFT) are computationally efficient with a minimal input-output processing delay. The proposed algorithm can be used in any stand-alone platform such as a smartphone using its two inbuilt microphones. The detailed operation of the real-time implementation on the smartphone is presented. The developed application works as an assistive tool for hearing aid devices (HADs). Speech quality and intelligibility test results are used to compare the proposed algorithm to existing conventional and neural network-based SE methods. Subjective and objective scores show the superior performance of the developed method over several conventional methods in different noise conditions and low signal to noise ratios (SNRs). © *2020 Acoustical Society of America.*
https://doi.org/10.1121/10.0001600

(Received 26 March 2020; revised 11 June 2020; accepted 27 June 2020; published online 27 July 2020)

[Editor: Zoi-Heleni Michalopoulou] Pages: 389–400

I. INTRODUCTION

Speech-based applications or systems often deal with speech corrupted by different types of background noise and interferences. In recent years, the demand for noise-free clean speech using hearing aid devices (HADs), hands-free communication systems, automatic speech recognition, and recording systems has drastically increased (Moore *et al.*, 2017). The performance of these devices depends on the quality and intelligibility of speech extracted and enhanced from the noisy speech signal. A speech enhancement (SE) method aims at reducing the background noise and obtaining speech signals of high perceptual quality in harsh noisy environments.

Understanding the speech and words in the presence of various and dominant background noise is difficult for people with normal hearing and is unacceptable for those with moderate-to-severe hearing loss and the HAD users. Reports from the National Institute on Deafness and Other Communication Disorders (NIDCD) point out that close to 15% of adults experience some kind of hearing loss in the United States. It is also reported that over 360×10^6 people in the entire world suffer from hearing loss (Quick Statistics, 2019). To address these problems, researchers, medical professionals, and designers have been offering many alternatives and possible solutions such as the application of appropriate HADs and cochlear implants (CIs). However, the working of many HADs and CIs can deteriorate in the presence of different, and often strong, environmental noise. Due to the physical design constraints, these devices often lack the computational power to run complex signal processing algorithms (Klasen *et al.*, 2007; Kuo *et al.*, 2008).

Single-channel SE algorithms usually operate in the time-frequency domain. The spectral subtraction method introduced by Boll (1979) focuses on subtracting short-time noise magnitude spectrum from noisy speech magnitude spectrum. Other statistical model-based approaches developed by Ephraim and Malah (1985) have been successful in the removal of background noise at high signal-to-noise ratio (SNR) levels. Many recent algorithms such as the optimally modified log-spectral amplitude (OM-LSA) (Cohen and Berdugo, 2002) and non-negative matrix factorization (NMF) (Wilson *et al.*, 2008) can achieve better noise suppression and SE at the cost of higher computational complexity. Some computationally efficient algorithms are proposed in Lotter and Vary (2005) and Wolfe and Godsill (2003) that rely on maximum *a posteriori* (MAP) estimation. Due to dependencies on the type of background noise, *a priori* knowledge, and inaccurate estimation of the noise spectrum, almost all of these methods introduce speech distortions in the form of musical noise, especially at low SNRs. Theoretically, multi-channel SE algorithms cause minimal speech distortion when compared to single-channel SE methods. Recent progressions include microphone array-based SE techniques for better noise suppression (Benesty *et al.*, 2008; Hoffman *et al.*, 1994).

Multi-channel SE algorithms are usually based on spatial filtering. Spatial filtering helps to boost the desired source signal and simultaneously attenuates the interferences from other directions. In real-life environments, beamforming can control the working of a microphone array system. Fixed beamformers have static filter coefficients and

[a)]This paper is part of a special issue on Machine Learning in Acoustics.
[b)]Electronic mail: Nikhil.Shankar@utdallas.edu, ORCID: 0000-0003-1963-126X.

signal independent spatial response. The simplest example is the delay-and-sum beamformer (Dudgeon, 1977). The most widely employed adaptive beamformer is the linearly constrained minimum variance (LCMV) beamformer. The minimum variance distortionless response (MVDR) beamformer (also known as the Capon beamformer) (Capon, 1969) is a particular case of the LCMV beamformer. In these multi-channel methods, an increase in the number of microphones escalates the cost, computational complexity, and power consumption. Compromising on these limitations, dual-channel (2 microphones) methodologies (Shankar *et al.*, 2020; Shankar *et al.*, 2018) could be considered which would provide a favorable solution for real-time implementation to improve the quality and intelligibility of the enhanced speech signal. Since the dual-channel SE algorithms remove spatial noise along with the additive noise, they can outperform single-channel SE methods. However, these traditional approaches tend to work on certain assumptions and cannot show good performance for the non-stationary types of background noise.

Machine learning and neural networks have gained increasing popularity for solving multivariable, complex, nonlinear problems. They can also be used to solve many audio/speech signal processing problems. Application of deep neural networks (DNN) for SE has shown significant results in the past few years. The main concept is to train complex models using a large amount of data, often off-line, instead of depending on any *a priori* conditions and approximations. Neural networks perform non-linear mappings from input to output data and are trained based on the target application. The variety of neural network techniques that are present vary mainly in terms of operations, connections between nodes, and the procedures used for training.

Deep learning techniques are classified into mask approximation- and signal approximation-based approaches. A feed-forward neural network (FNN) is proposed in Xu *et al.* (2015) to estimate the ideal binary mask (IBM) from noisy input speech. Compared to mask-based techniques, the signal-based approximations minimize the difference between estimated and target gain (Wang and Wang, 2013). In Xu *et al.* (2014b), a DNN-based SE trained on the log magnitude spectrum reduces a significant amount of noise without any introduction of musical noise. A direct feature estimating DNN architecture is presented in Xu *et al.* (2014a) to predict the log power spectra features of clean speech. Recent developments in the convolutional neural network (CNN) make them useful for SE by inputting spectrogram features to train the model (Du and Huo, 2008). CNN-based methods make use of untied weights shared among multiple inputs. Fully convolutional neural network (FCN)-based end-to-end SE utilizing raw data as the input has been considered (Fu *et al.*, 2018). Other features such as Mel frequency cepstral coefficients (MFCC) are used along with the magnitude spectrum for the CNN-based SE approach (Bhat *et al.*, 2019). Comparing CNNs with recurrent neural network (RNN) techniques, RNNs offer much more complexity since they do not have weight sharing. However, RNNs are most suited for time series data, as they can use their internal memory to process random sequences of input data, hence, making RNNs ideal for text and speech analysis.

Researchers have invested a significant amount of time and effort into long short-term memory (LSTM) networks, which are special kinds of RNNs (Keshavarzi *et al.*, 2018). The main reason is that these networks are designed to avoid long-term dependency problems. LSTM-RNN (Healy *et al.*, 2019; Sun *et al.*, 2017; Weninger *et al.*, 2014) and a combination of convolutional and LSTM networks (Tan *et al.*, 2019) outperform other neural networks for SE at lower SNRs. Even though they achieve better results, an increase in the number of hidden layers and the use of LSTMs come with an increase in the complexity and operating cost. This acts as a limitation for real-time operation. Without sacrificing much on the performance of speech enhancement, a slightly less complex neural network approach is proposed in this paper using basic RNN cells. That is, a simple RNN cell, unlike LSTMs, comprises just a multiplication process between the input and previous output, passing through an activation function without any additional operations.

In this paper, we present an efficient approach of using the basic RNN cells for enhancing speech in the presence of background noise using the two microphones of a smartphone. The smartphone is considered as an example to prove the real-time working of the proposed method. It should be noted that the smartphone is used as a stand-alone processing platform, without any external component or device, for implementing and running our proposed SE algorithm in real-time. We propose that the real and imaginary part of the frequency domain signal be used as the primary input feature for the model. The proposed method works in real-time on a frame by frame processing of the data with a minimal input-output delay and can be implemented on any other processing platform (edge device).

Another possible solution proposed in this work is the use of popular smartphones to capture the noisy speech data, process the signal, perform complex computations using the SE algorithm, and pass on the enhanced speech signal to the HADs through wired or wireless connection (Bhat *et al.*, 2017; Karadagur Ananda Reddy *et al.*, 2017), hence, making the proposed application on the smartphone an efficient assistive tool for the HAD users. However, the proposed application can also be used by a normal hearing user with the help of wired or wireless earphones/headphones. Thus, a computationally efficient RNN architecture is developed for SE using a simple but efficient input feature set and its real-time implementation on the smartphone without the help of any external hardware components. The proposed SE algorithm can act as a vital component in the signal processing pipeline consisting of other blocks like adaptive feedback cancellation and dynamic range compression. The objective evaluations and subjective test scores of the proposed RNN-based SE method signify the operational capability of the developed approach in several different noisy environments and low SNRs. For the objective test of speech quality measurement, we use the perceptual evaluation of speech quality (PESQ) (Rix *et al.*, 2001). Coherence speech intelligibility index (CSII) (Loizou, 2013) and short-time objective intelligibility (STOI) (Taal *et al.*, 2011) are used to measure the intelligibility of speech.

The paper focuses on comparing the performance of the proposed method with other existing statistical-based and deep learning-based SE algorithms. First, we compare the proposed algorithm with a conventional single microphone log-MMSE-based SE (Ephraim and Malah, 1985). A dual microphone SE algorithm Shankar *et al.* (2018) operating in real-time on a smartphone is also considered for our comparison, thus, proving the use of the proposed method over conventional and real-time SE methods. Since the proposed method is based on neural networks, it is also compared with existing DNN-SE methods such as FCN (Fu *et al.*, 2018) and CNN-based denoising autoencoder (CNN-DAE) (Kounovsky and Malek, 2017). This comparison shows the performance improvement of the proposed RNN model architecture over CNN architecture for real-time SE. Additional test results also compare the computational complexity and performance of the proposed RNN method against a baseline LSTM method.

The remainder of this paper is as follows: The signal model used in the proposed algorithm and the proposed RNN architecture is briefly described in Sec. II. Section III presents objective and subjective assessments. Section IV presents the real-time implementation of the developed method on a smartphone. The conclusion is in Sec. V.

II. PROPOSED RNN-BASED SPEECH ENHANCEMENT

The proposed SE pipeline is described in this section. The block diagram of the proposed algorithm shown in Fig. 1 represents the real-time usability and application of the proposed method using a smartphone and HAD. We consider a simple signal model with the noisy signal received by the ith microphone ($i = 1, 2$) written as

$$y_i(t) = s_i(t) + w_i(t) = s(t - \tau_i) + w_i(t), \tag{1}$$

where $y_i(t)$, $s_i(t)$, and $w_i(t)$ are noisy input speech, clean speech, and noise signals, respectively, picked up by the ith microphone at time t. We assume audio/acoustic plane wave arriving at the microphones. Thus, $\tau_i = (i-1)\tau_0 \cos\theta_d$ is the relative time delay between the ith microphone and reference sensor, and $\tau_0 = \delta \backslash c$ with δ being the distance between the two microphones (13 cm in case of smartphone), and c is the speed of sound in free air. The incidence angle of the target speech source is θ_d. $s_1(t)$ is considered to be the clean speech captured by the reference microphone. All the signals are considered to be real and zero-mean. The input noisy speech is transformed to frequency domain by taking short-time Fourier transform (STFT) and re-written as

$$\begin{aligned} Y_i(\omega_k) &= S_i(\omega_k) + W_i(\omega_k) \\ &= e^{-j(i-1)\omega_k \tau_0 \cos(\theta_d)} S_1(\omega_k) + W_i(\omega_K), \end{aligned} \tag{2}$$

where $Y_i(\omega_k)$, $S_i(\omega_k)$, and $W_i(\omega_k)$ are the Fourier transforms of $y_i(t)$, $s_i(t)$, and $w_i(t)$, respectively. This is the discrete version of the STFT by sampling the frequency variable ω at N uniformly spaced frequencies (i.e., $\omega_k = 2\pi k/N$, $k = 0, 1, \ldots, N-1$). The frequency bins are represented by k, and N is the STFT size.

A. Primary input and output features

The mathematical representation of the STFT of the noisy input signal is a complex number consisting of both real and imaginary parts. The real and imaginary parts of Eq. (2) are used as primary input features for the proposed RNN-based dual-channel SE. Computing the real and imaginary part of the noisy and clean speech recordings is part of the training approach. The input features from both the channels are concatenated together to form an input vector of dimension $2C(F+1) \times 1$. Where $F = N/2$ and N is the STFT size, $C = 2$ is the number of input channels (microphones).

$$\mathit{Input\ feature\ vector}(\boldsymbol{x}) = \begin{bmatrix} Real(Y_1(\omega_0)) \\ \vdots \\ Real(Y_1(\omega_F)) \\ Imag(Y_1(\omega_0)) \\ \vdots \\ Imag(Y_1(\omega_F)) \\ Real(Y_2(\omega_0)) \\ \vdots \\ Real(Y_2(\omega_F)) \\ Imag(Y_2(\omega_0)) \\ \vdots \\ Imag(Y_2(\omega_F)) \end{bmatrix}. \tag{3}$$

Since the second half of the STFT signal is a complex conjugate of the first half, we consider only the first half of the frequency domain data. Figure 2 shows the entire working

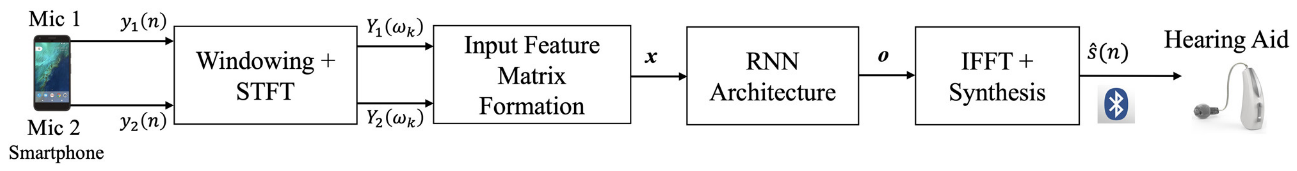

FIG. 1. (Color online) Block diagram representation of the proposed two-microphone RNN-based SE method.

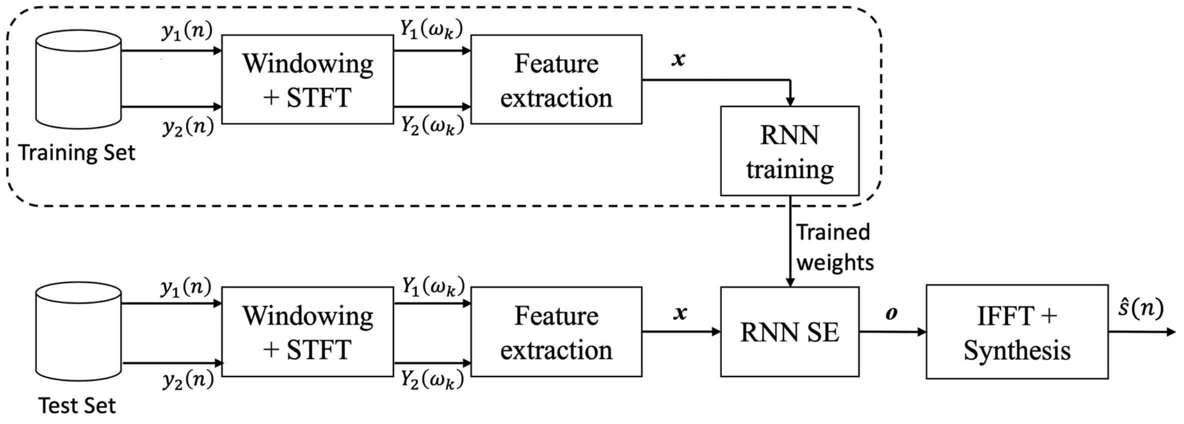

FIG. 2. SE system describing the training (dashed lines) and the testing phase.

of the training and testing phase. The training phase is shown in dashed lines. The real and imaginary parts of the respective channels are concatenated together as shown in Eq. (3). This input feature vector is then fed as input to the RNN architecture. Similar to Eq. (3), the output feature vector for the single-channel clean speech was also obtained, as shown in Eq. (4). The output vector in Eq. (4) behaves as a label for training the RNN model. The size of the output vector is $2(F+1)\times 1$. Finally, the estimated clean speech real-imaginary frequency domain values from the RNN model are used for time-domain signal reconstruction by taking the inverse fast Fourier transform (IFFT). Usage of real and imaginary values as input features helps in achieving a distortionless reconstruction. Overall, the noisy frequency domain signals act as input to the model to obtain an estimate of clean frequency domain speech as the output of the RNN model.

$$Output\ feature\ vector(\boldsymbol{o}) = \begin{bmatrix} Real(S_1(\omega_0)) \\ \vdots \\ Real(S_1(\omega_F)) \\ Imag(S_1(\omega_0)) \\ \vdots \\ Imag(S_1(\omega_F)) \end{bmatrix}. \tag{4}$$

B. RNN architecture for speech enhancement

This paper proposes a novel RNN architecture, which uses basic RNN cells to lower the complexity of the model. RNNs consist of at least one feedback connection, allowing us to model sequential data. However, due to the vanishing gradient problem, it is difficult to train them (Bengio *et al.*, 1994). The dynamic behavior of the RNN cell to use its internal state memory for processing sequences makes it very reliable for speech analysis. For this work, the entire architecture is interpreted as a filter in the frequency domain for enhancing speech in noisy environments.

The proposed model consists of basic RNN cells stacked together to form a RNN layer. The output from the RNN layer is then flattened and connected to a fully connected layer. This is then connected to a non-linear output layer in the end. The RNN layer comprises the basic RNN cells stacked on top of each other, and the cells are wrapped together into a single-layer cell. Each basic RNN cell can consist of R number of hidden units or neurons.

Activation functions are used in the hidden layers to help the neural network learn complicated, non-linear relations between the input and the output labels. Rectified linear unit (ReLU) is selected as the activation function, which acts as a solution for the abovementioned vanishing gradient problem (Maas *et al.*, 2013). ReLU is given by $\sigma(.)$:

$$\sigma(a) = max\{a, 0\}. \tag{5}$$

In RNN, a sequence of input vector v can be processed by applying a recurrence formula at every time frame t:

$$\boldsymbol{h_t} = \sigma(\boldsymbol{W_{hh}}\boldsymbol{h_{t-1}} + \boldsymbol{W_{vh}}\boldsymbol{v_t}). \tag{6}$$

$\boldsymbol{h}$ is taken to be the hidden vector, where $\boldsymbol{h_t}$ is for the new state (current state), and $\boldsymbol{h_{t-1}}$ is for the previous state. $\boldsymbol{v_t}$ is the input vector for the current state. The above equation shows that the current state depends on the previous state. $\boldsymbol{W_{hh}}$ is the weight parameters between the previous and the current hidden state. $\boldsymbol{W_{vh}}$ is the weights between the input sequence at time t (current state) and the hidden state. The dimensions of the abovementioned vectors depend on the input feature set and the number of hidden neurons in the RNN cell. The parameters for all time steps remain the same when RNN is trained, and the gradient at each output layer depends on the current time step calculation as well as on all previous time steps. This is called backpropagation through time (BPTT).

A fully connected layer is present right before the output layer, comprising D nodes. The linear activation function is used to map the predicted output features. The RNN SE architecture is further explained in Sec. III, based on experimental analysis.

III. EXPERIMENTAL EVALUATION AND TEST RESULTS

A. Dataset

For offline training of the proposed RNN model, the 2-channel noisy speech files are created by using the image-source model (ISM) (Lehmann *et al.*, 2007). The noise and the speech sources are separated by different angles from 0° to 180° with a resolution of 30° (0°, 30°, 60°, 90°, 120°, 150°, and 180°). First, the noise source location was fixed and the speech source was varied to achieve the angle separation between the two sources. Later on, the noise source location was varied by fixing the speech source location at 0°. Two different room sizes are assumed, and the two-microphone array is positioned at the center of the room. The size of the two rooms considered for generating the data is $5\,\mathrm{m}^3$ and $10\,\mathrm{m}^3$, respectively. The distance between the microphones is 13 cm (similar to the distance between the two microphones on the smartphone). Three different SNRs of -5, 0, and $+5$ dB are considered, with sampling frequency set to 16 kHz. The clean speech dataset used for training the model is a combination of TIMIT and LibriSpeech corpus (Panayotov *et al.*, 2015). DCASE 2017 challenge dataset (Mesaros *et al.*, 2017) is used as the noise dataset, which consists of 15 different types of background noise signals. The 15 types of noise are further categorized into 3 important types of noise, namely machinery, traffic, and multi-talker babble. There are around 300 noise files per type, which are commonly seen in real-life environments. In addition to these noise types, 20 different Pixel 1 smartphone recordings of realistic noise are collected and half of which are used for testing phase only. Finally, the clean speech and noise files are randomized and selected to generate the simulated noisy speech.

In addition to the abovementioned dataset, real recorded data is collected using a Pixel 3 smartphone placed on the center of a round table in 3 different rooms. The setup is as follows: Five loudspeakers are equally spaced and placed around the smartphone to generate a diffuse noise environment with one speaker playing clean speech and the rest playing noise. The 5 loudspeakers play the clean speech sequentially to make sure that the speech source direction is not fixed. The distance between the smartphone and the loudspeaker is set to be 0.6 m and 2.4 m in room 1, and 1.3 m and 0.92 m in rooms 2 and 3, respectively. The distance between the smartphone and the loudspeaker is varied to make sure that the recorded database is a collection of both near and far end speakers. The dimensions for rooms 1, 2, and 3 are $7\,\mathrm{m} \times 4\,\mathrm{m} \times 2.5\,\mathrm{m}$, $6.5\,\mathrm{m} \times 5.5\,\mathrm{m} \times 3\,\mathrm{m}$, and $5\,\mathrm{m} \times 4.5\,\mathrm{m} \times 3\,\mathrm{m}$, respectively. The reverberation time (RT60) for rooms 1, 2, and 3 is measured to be around 400, 350, and 300 ms, respectively. The abovementioned clean speech and noise files were played in the loudspeakers during data collection. To generate the clean speech labels for training the model, the noise files and the clean speech files are recorded separately on the smartphone, and then added together to generate noisy speech at different SNR. This additional dataset for training helped in increasing the realistic use and robustness of the real-time application.

B. RNN architecture for experiments

The proposed RNN architecture developed for experimental analysis has an input layer, 4 basic RNN cells stacked upon each other to form a single RNN layer, 1 fully connected layer, and an output layer. The proposed architecture remains the same for both offline and real-time evaluations.

The audio signal is sampled at 16 kHz with a frame size of 16 ms (50% overlap) and a 512-point (N) STFT is computed. Due to complex-conjugate property in STFT, the first 257, i.e., $(N/2+1)$, real and imaginary values are considered. The real and the imaginary values are then arranged on top of each other, thus, leading to 514 (257 real and 257 imaginary values) input features per channel. As we are performing dual-channel SE, the input feature vector will consist of real and imaginary values from both the channels, leading to an input matrix of size 1028×1. This is as shown in Eq. (3). The 4 basic RNN cells comprise of $R = 100$ neurons each and are stacked upon each other. The stacking of RNN cells together can be further understood by referring to Fig. 3. The number for R was fixed after many trials with different values and comparing the performances for each. The fully connected layer, after the RNN layer, consists of $D = 1024$ neurons. STFT of the single-channel clean speech is computed and used to set output labels for training the

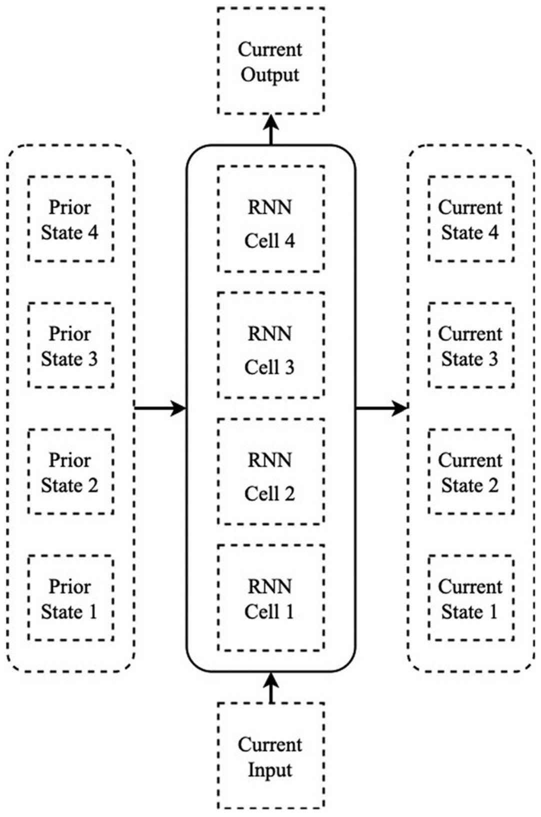

FIG. 3. Stacking of basic RNN cells together.

RNN model. Similar to the input-feature vector, we generate an output-feature vector as shown in Eq. (4). Since the output matrix is of size 514×1, the output layer has 514 neurons. ReLU is used as an activation function for the 4 RNN cells and the fully connected layer. Whereas, the linear activation function is used for the final output layer, which predicts real and imaginary values of the enhanced speech signal. The proposed network architecture is illustrated in Fig. 4. The proposed model has nearly 0.9×10^6 parameters.

For training the RNN model, the Adam optimization algorithm (Kingma and Ba, 2014) with mean squared error loss function is used. We initialize the training vectors, which include weights and biases for the nodes with a truncated normal distribution of zero mean and 0.05 standard deviation. With a learning rate of 10^{-6} and an appropriate batch size of 500, the model is trained for 15 epochs. The complete modeling and training for offline evaluations are carried out using Tensorflow in Python (Google TensorFlow, 2019). Chameleon cloud computing is used to train the proposed RNN model.

C. Offline objective test results

The dataset used for the offline training is explained in Sec. III A. PESQ varies from −0.5 to 4.5, with 4.5 being high speech quality. The calculation of PESQ is measured as follows: The original clean and deteriorated signals are equalized to a normal level of listening and filtered through a filter with a response equivalent to a telephone system. The signals are then time-synchronized and processed through an auditory transform to obtain the loudness spectra. Finally, to determine the subjective quality ranking, the difference in loudness between the signals is measured and averaged over time and frequency. CSII varies from 0 to 1, with 1 being high intelligibility. The speech intelligibility index is used as the basis for the measurement of CSII. The scoring is dependent on the signal-to-distortion ratio term, which is computed using the coherence between the input and output signals. The higher the STOI score, the better the intelligibility of speech. STOI generates the correlation coefficient between the clean reference signal and a noisy/distorted version of the reference signal. Figure 5 shows the plots of PESQ, CSII, and STOI versus different SNRs of −5, 0, and +5 dB for the three noise types. The results are the average of 10 HINT sentences that are not seen by the model mixed randomly with 10 unseen noise recordings. Machinery, traffic, and multi-talker babble noise types are considered. The size of the room considered for generating the test data is $7.5\,\text{m}^3$. This is to make sure that the test set (validation set) is completely different when compared to the data used for training the proposed model. It should be noted that the HINT database is not considered for model training, and the above test set is used for all our experimental analysis.

As mentioned in Sec. I, the proposed RNN-based SE is compared with noisy speech, statistical-based SE methods (Ephraim and Malah, 1985; Shankar *et al.*, 2018), and DNN-based SE methods (Fu *et al.*, 2018; Kounovsky and Malek, 2017). In (Shankar *et al.*, 2018), a two-channel MVDR beamformer is used as an SNR booster to a single channel SE, and the beamformer output acts as input to the SE. The FCN model (Fu *et al.*, 2018) trained on raw audio data has 8 convolutional layers with zero padding. Wherein, the convolutional layers consist of 30 filters with a filter size of 55 in each, apart from the last layer, which has only 1 filter. The resulting FCN network has nearly 0.3×10^6 parameters. The CNN-DAE model (Kounovsky and Malek, 2017)

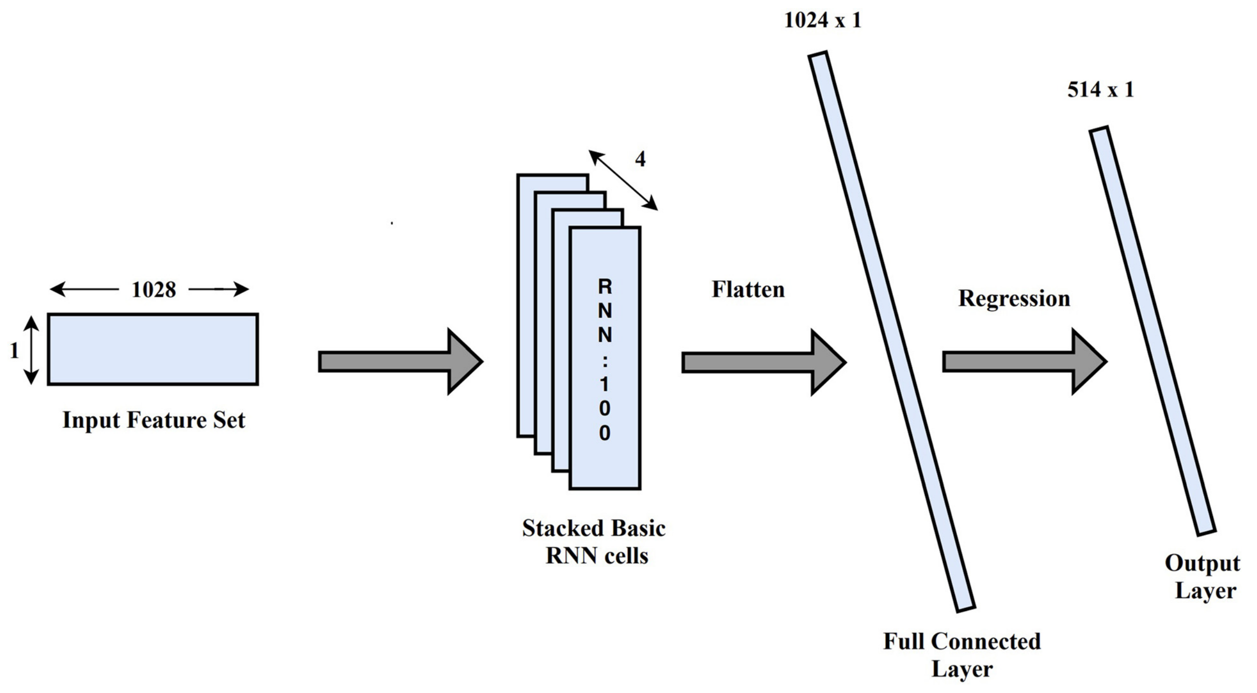

FIG. 4. (Color online) Proposed RNN architecture for SE.

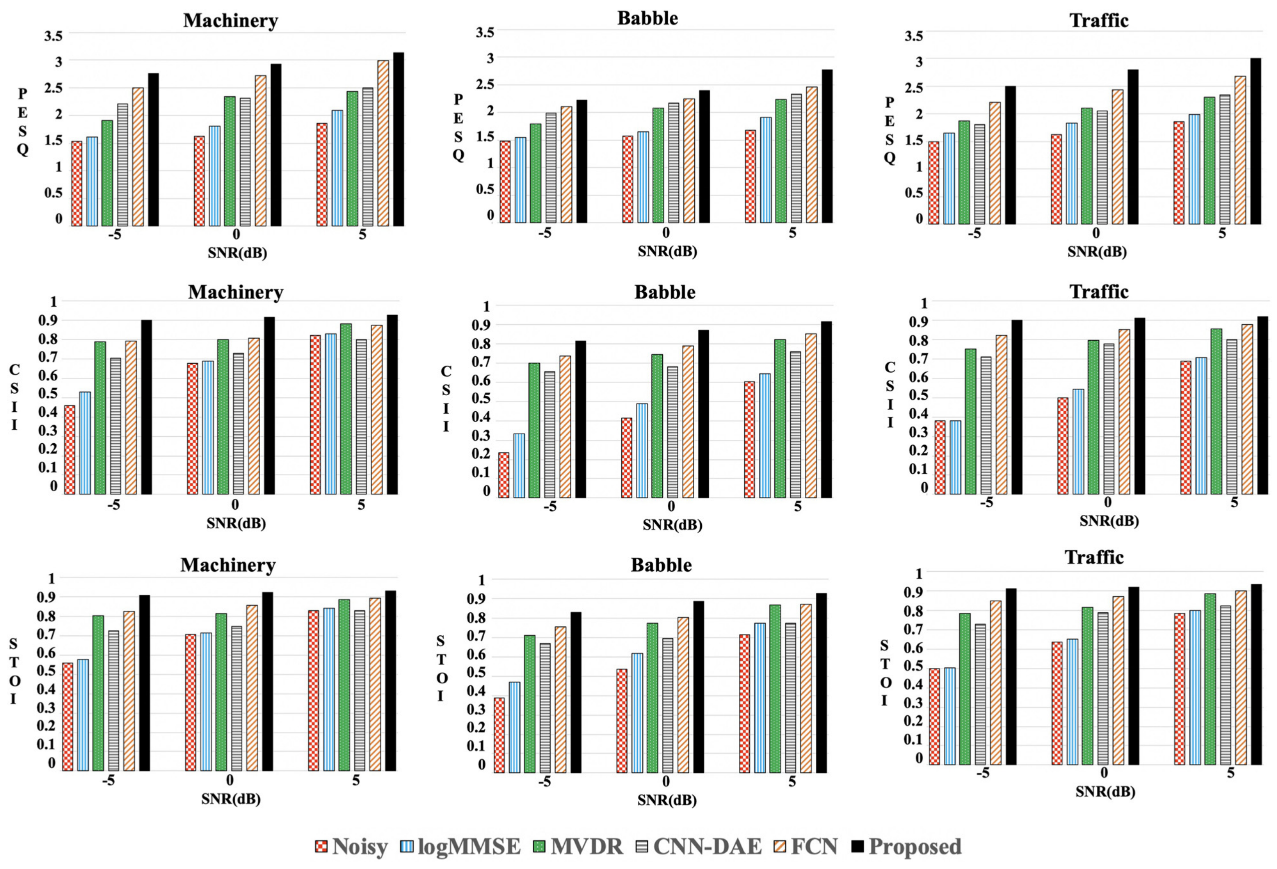

FIG. 5. (Color online) Average PESQ, CSII, and STOI scores of unseen data for different SE methods. Machinery, babble, and traffic noise at different SNR levels of −5, 0, and +5 dB are considered.

has 2 convolutional layers with a max-pooling layer in between them and 2 fully connected layers. The first and second convolutional layer has 52 and 78 feature maps, respectively, with a kernel size of 5×1. Max-pooling by factor 3 is considered. The fully connected layers have 1024 neurons, followed by an output layer with 129 neurons to predict the clean speech spectrum. The resulting CNN-DAE network has nearly 4.5×10^6 parameters. The proposed method gives better values of PESQ, CSII, and STOI, as shown in Fig. 5 for babble, machinery, and traffic noise types. The CSII and STOI scores for the proposed method are closer to 1 for different noise types, indicating minimal speech distortion. Objective results obtained show that the proposed method achieves substantial noise suppression.

D. Dropout training

Regularization techniques are used in the field of deep learning to solve the problem of overfitting in neural networks. Overfitting generally means possible mismatches between the training and testing accuracies. Dropout is a type of regularization technique that refers to dropping out some of the hidden units or neurons from a layer during training (Xu *et al.*, 2015). Dropout can also be used to improve the generalization capability of the neural network. When training the model, dropout randomly omits p percentage of the neurons in the hidden layer, averaging the model to avoid overfitting. This operation might reduce the performance of the model on training data, but improves the robustness in varying noise type cases that are not seen in the training data. However, overfitting of the proposed model leads to random variations in the objective scores. To prove that the proposed model is not overfitting, we consider different dropouts of 10%, 20%, and 30%. Clean speech files from HINT (not seen by the model) mixed with unseen machinery noise at 0 dB SNR are considered and tested at the different dropouts. Table I shows the objective measures for different dropouts. It is seen that there is not much performance loss in the values of PESQ, CSII, and STOI. Thus, we can say that there is no overfitting in the proposed RNN model for SE. The scores are slightly reduced when dropout is increased to 30%. This is mainly due to the decrease in

TABLE I. Illustration of the dropout effect on the proposed RNN-based SE model for machinery noise condition at 0 *d*B SNR.

Dropout	PESQ	CSII	STOI
0%	2.991	0.925	0.932
10%	2.958	0.919	0.927
20%	2.931	0.913	0.921
30%	2.895	0.898	0.902

TABLE II. Comparison of proposed RNN-based SE model with LSTM baseline model for different noise conditions at 0 dB SNR.

Objective measure	Noise type (SNR 0 dB)	RNN	LSTM
PESQ	Babble	2.399	2.545
	Machinery	2.931	3.012
	Traffic	2.795	2.888
CSII	Babble	0.869	0.881
	Machinery	0.913	0.921
	Traffic	0.91	0.918
STOI	Babble	0.885	0.898
	Machinery	0.921	0.93
	Traffic	0.92	0.928

the number of neurons in the neural network. For objective measures shown in Fig. 5 and for real-time implementation of the model, a dropout of 20% is considered.

E. Comparison of RNN with LSTM

In this experiment, we compare the performance of RNNs with LSTMs. Using the same input features, two different models are evaluated and compared. First, we consider an LSTM baseline method that uses 4 LSTM layers with 100 neurons each instead of the 4 basic RNN cells present. The rest of the model architecture remains the same. The input and output features for training the model will be the input (dual-channel) and output feature vector given in Eqs. (3) and (4), respectively. This method is then compared with the proposed RNN model for SE. The training data explained in Sect. IIIA remained the same for training both the methods. Table II shows the performance of the proposed method versus the two-channel LSTM output. The same validation set used for offline evaluation is considered at an SNR of 0 dB for this experiment. The PESQ, CSII, and STOI results shown in Table II suggest that the performance of the proposed RNN model is on par with that of the LSTM model for stationary noise types. The presence of additional gates in the LSTM layer leads to additional mathematical operations on the inputs, resulting in better scores for both stationary and non-stationary noise types. However, there is an increase in the number of trainable parameters present in the baseline LSTM model by approximately one million parameters when compared to the number of trainable parameters present in the proposed basic RNN model, thus increasing the cost of computations and complexity. Due to the following limitations, we have considered basic RNN cells in the proposed method over LSTMs.

F. Effect of number of neurons and unseen SNR

1. Number of neurons effect

In deep learning, the size of the model can be reduced by reducing the number of hidden layers or by reducing the number of neurons in each hidden layer. It is mentioned that the number of neurons in each basic RNN cell is fixed to be $R = 100$. However, this can be varied, and by reducing it, we can reduce the size of the model. Reduction in the number of neurons in each RNN cell may degrade the performance of the proposed RNN-based SE. However, having a very large R will cause the model to overfit and also increase the computational complexity of the entire application. Hence, an ideal model has to be proposed and implemented after several experiments. Figure 6 shows the performance comparison of two models with a different number of neurons in the RNN cells. A model is designed similar to that of the proposed RNN model, but with R set to be equal to 10 instead of 100. The results obtained show that the objective scores for the proposed RNN-based SE with 100 neurons outperform the model with 10 neurons. Noisy speech degraded by all three types of noise is considered at 0 dB SNR. The performance trend of the noise types at different SNR remained the same. The number of trainable parameters present when $R = 100$ is 10 times more when compared to $R = 10$. Further increasing R will lead to a drastic increase in the number of trainable parameters in the model ($R = 200$ has 20 times more parameters when compared to $R = 10$). Besides, there was no major improvement in the objective results when R was 100 and 200, therefore, we fixed the number of neurons per RNN cell to be 100.

2. Unseen SNR effect

The proposed RNN model is trained at 3 different values of SNR (i.e., −5, 0, and +5 dB). Now we evaluate the performance of these proposed models with new unseen SNR. Since the application of the proposed RNN model is a real-time enhancement of speech, varying SNR is common. Thus, it is important to have a robust model that performs well for varying SNR in real-world noisy conditions. To examine the effect of unseen SNR, the models trained at different SNRs of −5, 0, and +5 dB are tested with noisy speech at −2.5, +2.5, and +7.5 dB, respectively. Table III shows the performance of the proposed method against noisy speech in these challenging conditions. Similar to previous experiments, HINT clean speech degraded with machinery noise at −5, 0, and +5 dB SNRs are considered for experimental analysis. In detail, the neural

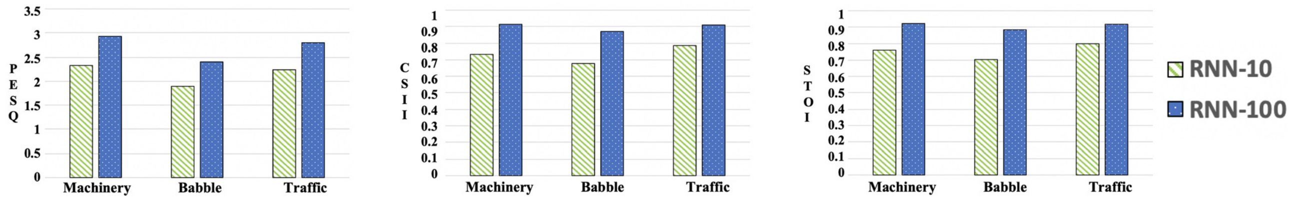

FIG. 6. (Color online) Performance evaluation of different number of hidden neurons in RNN cell (10 vs 100) for speech degraded by machinery, babble, and traffic noise at 0 dB SNR.

TABLE III. PESQ, CSII, and STOI results in unseen SNR condition with machinery noise. The proposed models were trained at −5, 0, and +5 dB SNR and then tested at −2.5, +2.5, and +7.5 dB, respectively.

Objective measure	Method	SNR (−2.5 dB)	SNR (2.5 dB)	SNR (7.5 dB)
PESQ	FCN	2.294	2.404	2.756
	RNN-Proposed	**2.574**	**2.732**	**2.994**
CSII	FCN	0.735	0.789	0.845
	RNN-Proposed	**0.804**	**0.865**	**0.901**
STOI	FCN	0.741	0.795	0.856
	RNN-Proposed	**0.814**	**0.869**	**0.912**

network model trained for −5 dB SNR machinery noise is tested with a noisy speech at −2.5 dB SNR. Similar to this, 0 dB and +5 dB SNR models are also tested. A slight tradeoff is seen in the performance when tested with unseen SNR, but the models show significant PESQ, CSII, and STOI scores in Table III when compared to the FCN approach. Different types of noise maintained the same performance trend as that of machinery noise.

G. Subjective results

Along with objective measures, we performed the mean opinion score (MOS) tests on 15 normal hearing male and female subjects. The subjects were presented with noisy speech and enhanced speech using the proposed, log-MMSE, MVDR beamformer, CNN-DAE, and FCN SE methods at 0 dB SNR. The same validation test database used for objective evaluation was considered for subjective testing. The subject was instructed to score for the different audio files in the range 1 to 5 based on the following criteria: 5 being outstanding speech quality and imperceptible distortion level, 4 for good quality of speech with a minimal degree of distortion, 3 having a reasonable level of distortion for equal speech content, 2 for poor speech quality with a lot of disturbances and uneven distortions, and 1 for having the least quality of speech with intolerable distortion level. Figure 7 shows the average subjective test results of 15 subjects and the scores illustrate the effectiveness of the proposed RNN-based SE method. Simultaneously, we also perceptually evaluated speech quality and intelligibility. Even though the CNN-DAE approach achieves significant noise suppression based on objective measures, the MOS was low due to the presence of speech distortions. A detailed description of the scoring procedure is explained in ITU (1996). The enhanced audio files using the proposed method can be found in (SSPRL, 2019). Furthermore, spectrogram plots are shown in Fig. 8. Clean speech degraded with machinery noise at SNR of 0 dB is considered. The plots consist of clean speech, noisy speech, and the proposed RNN-based algorithm enhanced speech.

IV. REAL-TIME IMPLEMENTATION

A. Smartphone implementation

The proposed RNN-based SE method is a real-time approach that can be used on any processing platform (e.g., smartphone). For real-time implementation, the Pixel 3 smartphone running the Android operating system (OS) is considered. Android OS is one of the most widely adopted mobile OSs, and the system allows developers to use stereo microphones on the smartphone. TensorFlow Lite provides

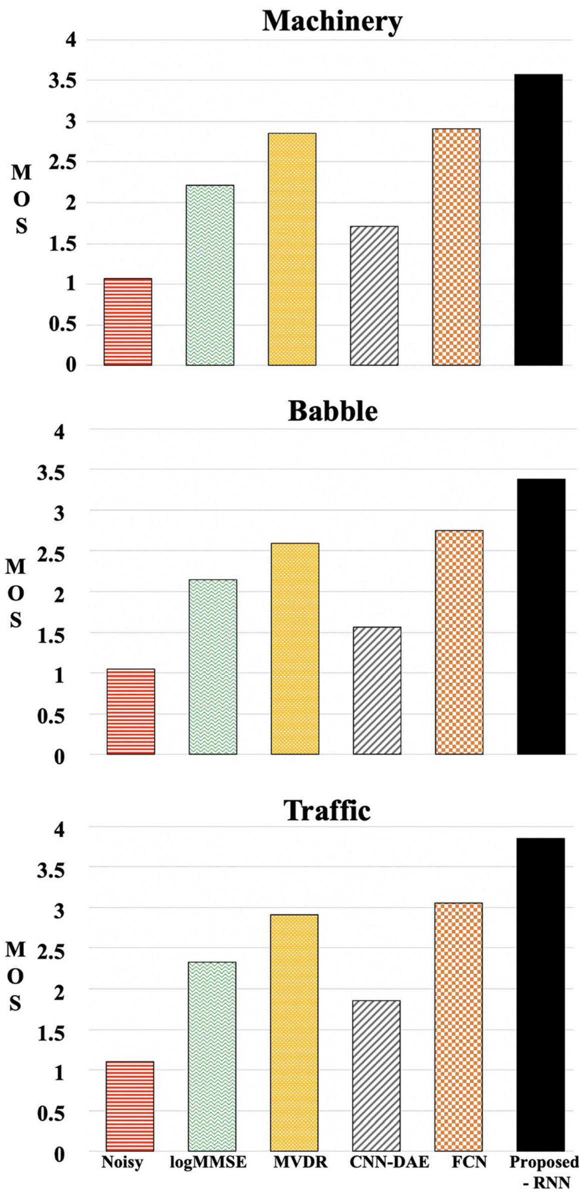

FIG. 7. (Color online) Subjective MOS for different noise conditions at 0 dB SNR.

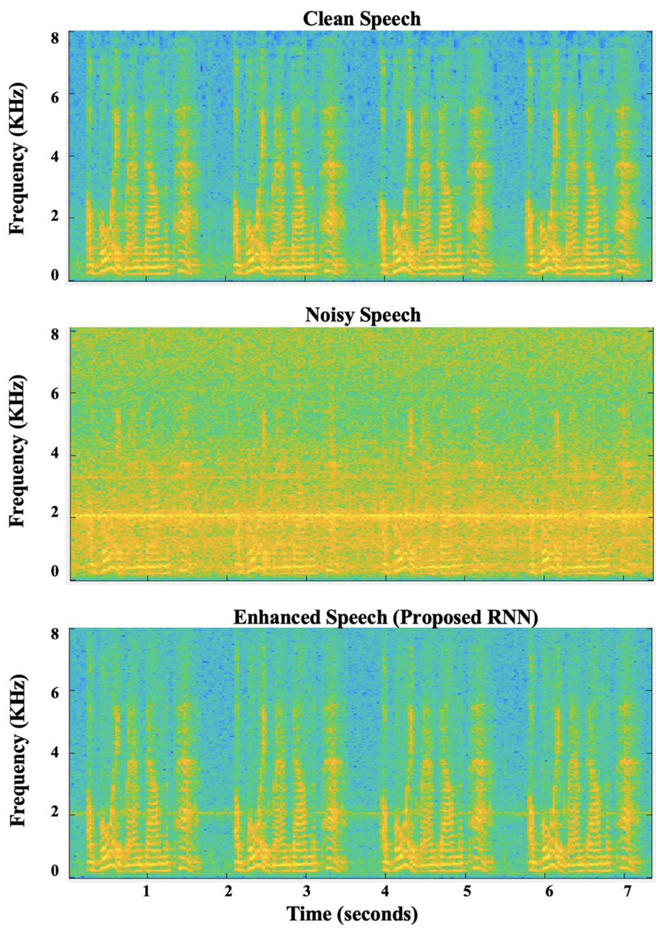

FIG. 8. (Color online) Spectrogram plots of clean speech, noisy speech (machinery noise, SNR 0 dB), and enhanced speech using proposed RNN model.

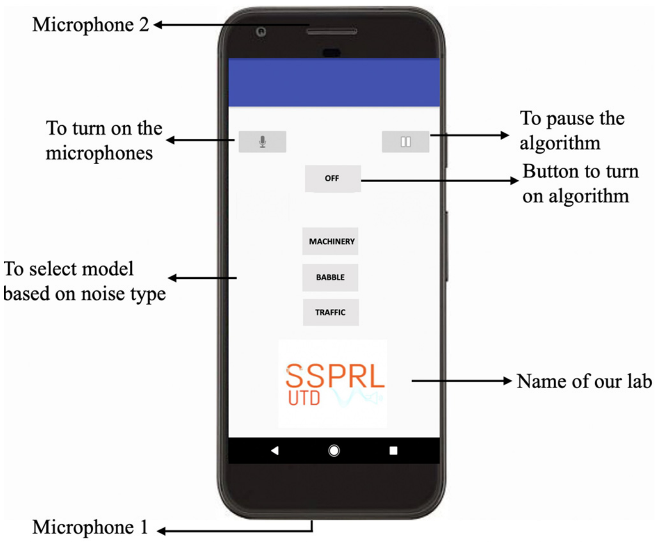

FIG. 9. (Color online) Screenshot of the developed real-time RNN-based SE application running on Pixel smartphone.

a C/C++ API for running deep learning models on Android-based platforms (Google TensorFlow Lite, 2019). Libraries such as the TensorFlow Lite converter and interpreter are used to compress and deploy the proposed model on the smartphone. After obtaining the trained model, the trained weights are frozen by removing backpropagation, training, and regularization layers. The final model with the required weights is saved to a file with.a pb extension and later used for real-time implementation.

The input frame size and STFT size remains the same as mentioned in Sec. III B. Each input frame is multiplied with a Hanning window, and overlap is considered between the frames with the help of an overlap-add technique that is widely used in SE (Benesty *et al.*, 2009). The two inbuilt microphones (13 cm apart) on the smartphone (as shown in Fig. 9) capture the audio signal; the signal is then enhanced, and the output signal (the clean speech) is transmitted to a wired or wireless headset. HADs can also be connected either through a wire or wirelessly through Bluetooth to the smartphone. The smartphone device considered for implementation has an M3/T3 HA compatibility rating and meets the requirements set by the Federal Communications Commission (FCC). Android Studio (Google Android Developer, 2019) is used for the real-time implementation of the proposed algorithm. An efficient stereo input/output framework (Küçük *et al.*, 2018) was used to carry out the real-time dual microphone input/output handling for the audio processing.

The input data on the smartphone is acquired at a 48 kHz sampling rate and then downsampled to 16 kHz by low-pass filtering and a decimation factor of 3. Figure 9 shows the screenshot of the proposed method implemented on the Pixel smartphone. Clicking on the Mic button can turn on the inbuilt microphones on the smartphone, and the pause button can turn off the microphones. Three models specific to three different types of noise (i.e., machinery, multi-talker babble, and traffic) are stored in the application on the smartphone. The hearing-aid user can select any one of the RNN models by simply clicking on the button corresponding to the name of the noise type, depending on the noisy environment they are in. When the button shown in the lower part of the screen is in "OFF" mode, no SE processing is carried out on the audio input, so the application performs simple audio playback through the smartphone. When it is in "ON" mode, the input feature vector is passed through the proposed and user-selected RNN model, which was described in Sec. III B. Then, the desired speech is extracted from the background noise and enhanced. The input feature vector passed through the frozen RNN model stored on the smartphone generates an output feature vector of size 514×1. After applying IFFT and reconstruction, the enhanced output speech signal so obtained is then transmitted to/played back to the HADs or earphones by either wire or wireless. It has to be noted that only outdoor noise types were considered for experimental evaluations. However, the proposed method can easily be expanded to numerous other indoor noise types and implemented on different stand-alone platforms (laptop) in a similar way to that discussed here.

B. Smartphone testing and computational complexity

The performance of the proposed RNN-based SE is tested on the Pixel 3 Android smartphone. Since the test

TABLE IV. Objective evaluation for unseen speech on smartphone and PC platform.

Testing platform	PESQ	CSII	STOI
PC (Simulation)	2.931	0.913	0.921
Smartphone	2.787	0.901	0.913

condition is in a real-time varying acoustic environment, the performance of the proposed model degrades on the smartphone when compared to the results from offline testing. For smartphone testing of the implemented application, HINT clean speech sentences are played by a loudspeaker and machinery noise is played in the background using the 5-speaker setup. The smartphone is placed in the center of the table with loudspeakers around the smartphone, as explained in Sec. III A. The distance between the smartphone and the loudspeaker is set to be 1 m and the room dimensions are $6.5\,\text{m} \times 5.5\,\text{m} \times 3\,\text{m}$ with reverberation time to be around 375 ms. The SNR is adjusted to be around 0 dB, and the microphones present on the android smartphone capture these signals. The captured input signal is processed by the frozen model stored in the application. The output enhanced speech is obtained and compared with the known clean speech signal to measure the quality and intelligibility of the output signal. Table IV shows the PESQ, CSII, and STOI scores of the proposed method for two test cases: the offline and the smartphone real-time test cases. The test results are on par with each other and show that the proposed SE method can be used for real-time applications on the smartphone. This also proves the working of the proposed model under an unseen position of the smartphone microphones in a new room.

The Central Processing Unit (CPU) and the memory usage of the proposed application reach around 45% and 1 GB, respectively, when selecting the RNN model on the smartphone. Once the model is selected by the user, the input data is continuously processed frame by frame using the frozen model in real-time, and the CPU and memory usage drops to around 25% and 450 MB, respectively. Since most of the smartphones have processors with 8–16 GB memory, the proposed dual-channel SE application consumes only around 4% of the memory. The audio latency for wireless input-output streaming through Bluetooth is measured to be 14–16 ms for Pixel 3. The proposed algorithm adds a processing delay of around 5 ms for a frame length of 16 ms. Therefore, in the case of wireless connection to the HADs, the total input-output delay of the proposed application is around 20 ms, which is negligibly perceived by the human ear in real-world conditions. As the processing delay (5 ms) is less than the frame length (16 ms), the proposed SE application is computationally efficient and runs seamlessly on a smartphone. For wired transmission, the total input-output delay is only the processing delay (5 ms).

Audio streaming is programmed for low power consumption. Java and C/C++ programming languages are combined on Android Studio. The proposed application is designed in Java, while the feature extraction and signal reconstruction are developed in C programming language. Since Java coding is not fast enough for audio processing, C/C++ is considered. Tensorflow API on android utilizes Java, which further increases the CPU consumption. The main reason for choosing the sampling rate as 48 kHz is that this sampling frequency gives minimum audio input-output delay. The sampling rate is then downsampled to 16 kHz for our application. Based on our experiments, the implemented application can run for approximately 5–6 h on Pixel 3 with a 2770 mAh battery.

V. CONCLUSION

We have proposed a dual-channel RNN-based speech enhancement application using a basic recurrent neural network cell. The proposed algorithm is operating in real-time with a low audio input-output latency. The developed method is implemented on the Android-based smartphone, proving the real-time usability of the proposed algorithm. This proposed smartphone-based speech enhancement application is computationally efficient and acts as an assistive hearing platform. Detailed analysis of the real-time implementation and the performance of the proposed method have been presented. The objective and subjective evaluations have verified the capability of the proposed method to be an apt option to use in real-world noisy environments. Furthermore, the presented method can be used for various real-time speech enhancement and noise reduction applications on different edge computing platforms.

ACKNOWLEDGMENTS

This work was supported by the National Institute of the Deafness and Other Communications Disorders (NIDCD) of the National Institutes of Health (NIH) under award number 1R01DC015430-04. The authors would like to thank the National Institute of the Deafness and Other Communication Disorders (NIDCD) of the National Institutes of Health (NIH) for their support. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

Benesty, J., Chen, J., and Huang, Y. (**2008**). *Springer Topics in Signal Processing Microphone Array Signal Processing* (Springer, Berlin Heidelberg), https://books.google.com/books?id=rFfF6BStEGIC.

Benesty, J., Chen, J., Huang, Y., and Cohen, I. (**2009**). *Noise Reduction in Speech Processing*, Vol. 2 (Springer Science & Business Media, Berlin).

Bengio, Y., Simard, P., and Frasconi, P. (**1994**). "Learning long-term dependencies with gradient descent is difficult," IEEE Trans. Neural Networks **5**(2), 157–166.

Bhat, G. S., Shankar, N., Reddy, C. K. A., and Panahi, I. M. S. (**2017**). "Formant frequency-based speech enhancement technique to improve intelligibility for hearing aid users with smartphone as an assistive device," in *Proceedings of IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT)*, pp. 32–35.

Bhat, G. S., Shankar, N., Reddy, C. K. A., and Panahi, I. M. S. (**2019**). "A real-time convolutional neural network based speech enhancement for hearing impaired listeners using smartphone," IEEE Access **7**, 78421–78433.

Boll, S. (**1979**). "A spectral subtraction algorithm for suppression of acoustic noise in speech," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 4, pp. 200–203.
Capon, J. (**1969**). "High-resolution frequency-wavenumber spectrum analysis," in Proceed. IEEE **57**(8), 1408–1418.
Cohen, I., and Berdugo, B. (**2002**). "Microphone array post-filtering for non-stationary noise suppression," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. I-901–I-904.
Du, J., and Huo, Q. (**2008**). "A speech enhancement approach using piecewise linear approximation of an explicit model of environmental distortions," in *Proceedings of Ninth Annual Conference of the International Speech Communication Association*, September 6–8, Brisbane, Australia, pp. 569–572.
Dudgeon, D. E. (**1977**). "Fundamentals of digital array processing," Proceed. IEEE **65**(6), 898–904.
Ephraim, Y., and Malah, D. (**1985**). "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," IEEE Trans. Acoust. Speech Signal Process. **33**(2), 443–445.
Fu, S., Wang, T., Tsao, Y., Lu, X., and Kawai, H. (**2018**). "End-to-end waveform utterance enhancement for direct evaluation metrics optimization by fully convolutional neural networks," IEEE/ACM Trans. Audio Speech Lang. Process. **26**(9), 1570–1584.
Google Android Developer available at https://developer.android.com/studio/intro/index.html (Last viewed October 28, 2019).
Google TensorFlow available at https://www.tensorfow.org/ (Last viewed October 04, 2019).
Google TensorFlow Lite available at https://www.tensorflow.org/lite/ (Last viewed October 19, 2019).
Healy, E. W., Delfarah, M., Johnson, E. M., and Wang, D. (**2019**). "A deep learning algorithm to increase intelligibility for hearing-impaired listeners in the presence of a competing talker and reverberation," J. Acoust. Soc. Am. **145**(3), 1378–1388.
Hoffman, M. W., Trine, T. D., Buckley, K. M., and Van Tasell, D. J. (**1994**). "Robust adaptive microphone array processing for hearing aids: Realistic speech enhancement," J. Acoust. Soc. Am. **96**(2), 759–770.
ITU. (**1996**). "Subjective performance assessment of telephone band and wideband digital codecs," in *ITU-T Recommendation*, p. 830.
Karadagur Ananda Reddy, C., Shankar, N., Shreedhar Bhat, G., Charan, R., and Panahi, I. (**2017**). "An individualized super-gaussian single microphone speech enhancement for hearing aid users with smartphone as an assistive device," IEEE Signal Process. Lett. **24**(11), 1601–1605.
Keshavarzi, M., Goehring, T., Zakis, J., Turner, R. E., and Moore, B. C. (**2018**). "Use of a deep recurrent neural network to reduce wind noise: Effects on judged speech intelligibility and sound quality," Trends in Hearing **22**, 2331216518770964.
Kingma, D. P., and Ba, J. (**2014**). "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980.
Klasen, T. J., Van den Bogaert, T., Moonen, M., and Wouters, J. (**2007**). "Binaural noise reduction algorithms for hearing aids that preserve interaural time delay cues," IEEE Trans. Signal Process. **55**(4), 1579–1585.
Kounovsky, T., and Malek, J. (**2017**). "Single channel speech enhancement using convolutional neural network," in *Proceedings of IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pp. 1–5.
Küçük, A., Hao, Y., Ganguly, A., and Panahi, I. M. S. (**2018**). "Stereo i/o framework for audio signal processing on android platforms," J. Acoust. Soc. Am. **143**(3), 1955–1956.
Kuo, Y.-T., Lin, T.-J., Chang, W.-H., Li, Y.-T., Liu, C.-W., and Young, S.-T. (**2008**). "Complexity-effective auditory compensation for digital hearing aids," in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 1472–1475.
Lehmann, E. A., Johansson, A. M., and Nordholm, S. (**2007**). "Reverberation-time prediction method for room impulse responses simulated with the image-source model," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 159–162.
Loizou, P. C. (**2013**). *Speech Enhancement: Theory and Practice* (CRC Press, Boca Raton, FL).
Lotter, T., and Vary, P. (**2005**). "Speech enhancement by map spectral amplitude estimation using a super-gaussian speech model," EURASIP J. Adv. Signal Process. **2005**(7), 354850.
Maas, A. L., Hannun, A. Y., and Ng, A. Y. (**2013**). "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of ICML*, Vol. 30, p. 3.
Mesaros, A., *et al.* (**2017**). "DCASE 2017 challenge setup: Tasks, datasets and baseline system." *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events.*
Moore, A. H., Parada, P. P., and Naylor, P. A. (**2017**). "Speech enhancement for robust automatic speech recognition: Evaluation using a baseline system and instrumental measures," Comput. Speech Lang. **46**, 574–584.
Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (**2015**). "Librispeech: An ASR corpus based on public domain audio books," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5206–5210.
Quick Statistics available at https://www.nidcd.nih.gov/health/statistics/quick-statistics-hearing (Last viewed November 20, 2019).
Rix, A. W., Beerends, J. G., Hollier, M. P., and Hekstra, A. P. (**2001**). "Perceptual evaluation of speech quality (PESQ): A new method for speech quality assessment of telephone networks and codecs," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 749–752.
Shankar, N., Bhat, G. S., Reddy, C. K. A., and Panahi, I. M. S. (**2020**). "Noise dependent super gaussian-coherence based dual microphone speech enhancement for hearing aid application using smartphone," ArXiv preprint arXiv:2001.09571.
Shankar, N., Kucuk, A., Reddy, C. K. A., Bhat, G. S., and Panahi, I. M. S. (**2018**). "Influence of MVDR beamformer on a speech enhancement based smartphone application for hearing aids," in *Proceedings of 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 417–420.
SSPRL available at https://utdallas.edu/ssprl/hearing-aid-project/video-demonstration/speech-enhancement/dual-channel-rnn-based-speech-enhancement/ (Last viewed December 25, 2019).
Sun, L., Du, J., Dai, L.-R., and Lee, C.-H. (**2017**). "Multiple-target deep learning for LSTM-RNN based speech enhancement," in *Proceedings of Hands-free Speech Communications and Microphone Arrays (HSCMA), IEEE*, pp. 136–140.
Taal, C. H., Hendriks, R. C., Heusdens, R., and Jensen, J. (**2011**). "An algorithm for intelligibility prediction of time-frequency weighted noisy speech," IEEE Trans. Audio Speech Lang. Process. **19**(7), 2125–2136.
Tan, K., Zhang, X., and Wang, D. (**2019**). "Real-time speech enhancement using an efficient convolutional recurrent network for dual-microphone mobile phones in close-talk scenarios," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5751–5755.
Wang, Y., and Wang, D. (**2013**). "Towards scaling up classification-based speech separation," IEEE Trans. Audio Speech Lang. Process. **21**(7), 1381–1390.
Weninger, F., Hershey, J. R., Le Roux, J., and Schuller, B. (**2014**). "Discriminatively trained recurrent neural networks for single-channel speech separation," in *Proceedings of IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 577–581.
Wilson, K. W., Raj, B., Smaragdis, P., and Divakaran, A. (**2008**). "Speech denoising using nonnegative matrix factorization with priors," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4029–4032.
Wolfe, P. J., and Godsill, S. J. (**2003**). "Efficient alternatives to the Ephraim and Malah suppression rule for audio signal enhancement," EURASIP J. Adv. Signal Process. **2003**(10), 910167.
Xu, Y., Du, J., Dai, L., and Lee, C. (**2014a**). "Dynamic noise aware training for speech enhancement based on deep neural networks," in *Proceedings of Fifteenth Annual Conference of the International Speech Communication Association.*
Xu, Y., Du, J., Dai, L., and Lee, C. (**2014b**). "An experimental study on speech enhancement based on deep neural networks," IEEE Signal Process. Lett. **21**(1), 65–68.
Xu, Y., Du, J., Dai, L., and Lee, C. (**2015**). "A regression approach to speech enhancement based on deep neural networks," IEEE/ACM Trans. Audio Speech Lang. Process. **23**(1), 7–19.