On the Robustness of Deep Learning-Based Speech Enhancement

Amit S. Chhetri, Philip Hilmes, Mrudula Athi, Nikhil Shankar
Amazon Inc., USA

***Abstract*—In this paper, we present the design of a robust deep neural network based speech enhancement (DNNSE) solution for joint noise reduction and dereverberation under real-world acoustic conditions. This makes our proposed solution suitable for smart-speaker products that encounter a wide variety of acoustic challenges during their real-world deployment. We provide a systematic introduction to the acoustic challenges involved in real-world products and perform a detailed analysis to compare DNNSE models using metrics such as short-time objective intelligibility (STOI), scale-invariant signal-to-distortion-ratio (SI-SDR), and generalizability to unseen acoustic conditions. We then develop a robust DNNSE solution along with a robust training procedure that are well suited for the acoustic challenges specified in the paper. Through detailed analysis, we demonstrate that our DNNSE solution performs and generalizes better than a baseline solution that is 5x-6x times larger.**

***Index Terms*—Deep neural network, speech enhancement, densely connected network, convolutional recurrent network, convolutional encoder-decoder**

I. Introduction

A common problem addressed by speech enhancement (SE) systems on smart speakers is the mitigation of noise and reverberation from microphone signal captures. These products need to operate under increasingly challenging acoustic conditions which has increased the demands on the performance of speech enhancement (SE) systems for Voice over Internet Protocol (VoIP) and automatic speech recognition (ASR) applications. Traditional methods used for single-microphone SE have performance limitations, since these methods use simpler models that do not generalize well to dynamic and challenging acoustic conditions [1]–[3].

Recently, end-to-end (e2e) deep neural network based speech enhancement (DNNSE) techniques have been found to perform very well in non-stationary noise environments, since a well-designed DNN can approximate an arbitrarily complex nonlinear function by incorporating higher-order statistics of the signal [4]–[11] (and references therein). Many of the proposed e2e solutions operate in the Short-Time Fourier Transform (STFT) domain. For example in [6], an ideal ratio mask (IRM) is first estimated and then applied to the noisy speech spectrogram to perform enhancement. Alternatively, other variants use training targets based on a direct representation of the speech signal [8]–[11]. Two leading approaches [10], [11] have been proposed based on a convolutional recurrent network (CRN) architecture that uses a combination of convolutional encoder-decoder (CED) structure and recurrent units. In [10], the magnitude spectrogram of the speech is used as the training target of the CRN (hereby called CRN-M network). In [11], the complex spectrogram is directly used as the training target (hereby called CRN-C network) and shown to perform better than the CRN-M solution. Other approaches are either non-causal or use a significant amount of look-ahead that prevent real-time deployment [12], [13]; therefore, these methods are not pursued for this work.

The focus of this work is on the general development of robust single-microphone DNNSE solution that can jointly perform noise reduction (NR) and dereverberation and can be trained e2e. Higher emphasis is given to the robustness of the DNNSE solution than a particular network architecture, as this allows us to design network architectures that are more suitable for real-world conditions. We are interested in DNNSE solutions that are causal and amenable for real-time deployment (i.e., they don't require a look-ahead window during inferencing). The DNNSE solution should exhibit robustness to a wide dynamic range of speech and noise levels, and it should be robust to variations in speaker type, noise type (stationary/non-stationary), microphone gain levels, SNR levels, room reverberation, and sudden changes in speech and noise characteristics. It is also desirable to control the dereverberation characteristics (independent of the NR performance) in order to fine-tune the overall performance of the DNNSE solution. Lastly, it is desirable to choose DNN architectures that not only perform well for the conditions they are trained for, but also exhibit robustness to unseen acoustic conditions.

To address the above robustness requirements, we aim to design a DNNSE solution that uses fewer parameters and exhibits better robustness to unseen acoustic conditions than the baseline CRN-C solution in [11]. We first show that while the CRN-C solution performs quite well under controlled acoustic conditions, its performance drops significantly for unseen acoustic conditions, which emphasizes our point on using robustness as an important criterion for network design. Early efforts with the CRN-C solution suggested that merely reducing the number of parameters of the network also resulted in a loss of performance. Thus, there was a need to fundamentally improve the structure of the network, so that parameter efficiency becomes a key virtue of the new design. Towards this, we extended the CRN-C network to the CRN-D (convolutional recurrent network with dense connectivity) network that combines the benefits of the CRN architecture with the densely-connected convolutional neural networks (DCNNs) [14] and trained it with a robust training procedure. As we

will show, the CRN-D architecture is not only more robust to unseen acoustic conditions, but it also performs better than the CRN-C architecture that is 5x-6x times larger. This work has the following contributions:

- We provide a systematic introduction to acoustic challenges for real-world deployment of SE solutions. We highlight the key acoustic parameters that need to be met to build a robust DNNSE solution.
- We propose a robust training procedure for building a DNNSE solution that generalizes well to real-world conditions. Our training procedure also allows us to exert fine-grained control on the dereverberation performance.
- We conduct a detailed analysis that helps us compare different DNNSE models w.r.t. various metrics such as the short-time objective intelligibility (STOI) [15], scale-invariant signal-to-distortion-ratio (SI-SDR) [16], and generalizability to unseen acoustic conditions. Note that both STOI and SI-SDR metrics indicate the efficacy of speech enhancement systems, with higher values indicating a better performance.
- We develop a robust CRN-D solution that demonstrates significantly improved generalizability over the CRN-C solution on unseen acoustic conditions. Our work and results emphasize that robustness to real-world conditions is an important criteria for designing DNNSE solutions.

II. Signal Model

The microphone signal at discrete time index n is composed of reverberant speech $x_s(n)$ mixed with additive noise $v_a(n)$ (e.g. ambient noise, sensor noise, and mechanical noise):

$$x(n) = x_s(n) + v_a(n), \tag{1}$$

where $x_s(n) = h_s * s(n)$, with $s(n)$ denoting the clean anechoic speech signal and h_s denoting the room impulse response (RIR) for speech (hereby called speech RIR). Eqn. 1 can be written in the frequency domain as:

$$X(l,k) = X_s(l,k) + V_a(l,k) = H_s(l,k)S(l,k) + V_a(l,k). \tag{2}$$

Here, l and k denote the frame index and frequency bin index, respectively. The signal spectrum is computed using a 512-point DFT filterbank with a 75% overlap and a Hann window.

Note that for real-world deployment, the DNNSE solutions need to operate robustly under dynamic acoustic conditions. These include the following specifications: (a) the SNR of the signal varies from -5 dB to 20 dB, (b) the speech and noise levels can vary from -70 dBFS to -5 dBFS, (c) the room reverberation varies from 60 ms to 500 ms, and (d) the additive noise can have stationary/non-stationary characteristics and vary from being band-limited (e.g. tones) to broadband.

III. Robust DNNSE Training Procedure

The state-of-the-art (SOTA) DNNSE solutions employ a sequence-to-sequence technique, where a model is trained to transform a noisy input spectrogram to a clean spectrogram on a per-frame basis. To ensure a causal system, the network is allowed access to only the current and past frames of the input spectrogram. Figure 1 depicts the overall DNNSE system where the neural network generates complex-valued masks that are applied to the noisy spectrogram to generate the enhanced spectrogram. Note that while the masks are generated as an output of the DNN, the DNN itself is trained e2e by comparing the enhanced speech output with the target signal. Note also that a side benefit of estimating the masks is that they can be used as a proxy for voice activity detection (VAD), which is often useful for many audio processing algorithms.

A. Loss Function

The DNN is trained in a supervised manner in which we provide a target signal for the network to achieve. For this, we utilize a database comprising of anechoic speech samples, noise files, and speech RIRs, using which the training samples are generated on-the-fly as per the following equation:

$$X(l,k) = g(\, H(l,k)S(l,k) + \alpha V_a(l,k)) \,. \tag{3}$$

Here, α controls the signal-to-noise ratio (SNR) and g controls the linear gain on the training sample. In this work, the values of α and g are chosen such that the SNR can be varied from -5 dB to 15 dB, and the overall signal level can vary from -70 dBFS to -5 dBFS. Note that without loss in generality, we set the levels of anechoic speech samples (i.e., $s(n)$) to -40 dBFS, as it helped the training proceed more smoothly. Note also that on-the-fly sample generation provides a large sample diversity (order of billion combinations) that offers natural regularization and prevents overfitting by the models.

We now formulate the basic loss function for optimizing the DNN parameters, which will be further refined in later sections. Referring Figure 1, the network produces complex mask $M_C(l,k)$ at the output of the final layer of the form

$$M_C(l,k) = M_R(l,k) + jM_I(l,k), \tag{4}$$

where $M_R(l,k)$ and $M_I(l,k)$ are the real and imaginary components, respectively, of $M_C(l,k)$. The target signal for the training is set to $X_s(l,k)$ (the clean speech signal) with real and imaginary components denoted as $X_{s,R}(l,k)$ and $X_{s,I}(l,k)$, respectively. The complex mask is then applied to $X(l,k)$ to produce the real $\hat{X}_{s,R}(l,k)$ and imaginary $\hat{X}_{s,I}(l,k)$ components of the estimated speech spectrogram:

$$\hat{X}_{s,R}(l,k) = M_R(l,k)X_R(l,k) - M_I(l,k)X_I(l,k), \tag{5}$$

$$\hat{X}_{s,I}(l,k) = M_R(l,k)X_I(l,k) + M_I(l,k)X_R(l,k), \tag{6}$$

where $X_R(l,k)$ and $X_I(l,k)$ are the real and imaginary components, respectively, of $X(l,k)$. Training is performed using a mini-batch of N samples, and the loss during training is computed by averaging over the mini-batch samples. For this, we first compute the real and imaginary component loss

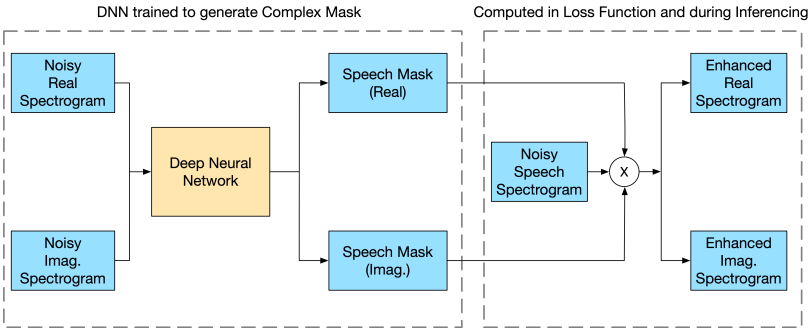

Fig. 1: Overall block diagram for the DNNSE system.

functions using the mean absolute error (MAE) between the network output and the target signal:

$$\begin{aligned}&\mathscr{L}_R(\,X_s(l,k),\hat{X}_s(l,k)) \\ &= \frac{1}{gKLN}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L} \mid X^i_{s,R}(l,k) - \hat{X}^i_{s,R}(l,k)\mid, \\ &\mathscr{L}_I(\,X_s(l,k),\hat{X}_s(l,k)) \\ &= \frac{1}{gKLN}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{L} \mid X^i_{s,I}(l,k) - \hat{X}^i_{s,I}(l,k)\mid.\end{aligned} \tag{7}$$

Here $i = 1, 2, \cdots, N$ is an index over the number of samples of a mini-batch. Note that we also normalize the component loss functions by the gain g that we applied in Eqn. 3 as it makes the loss function independent of gain variations; this is part of the proposed robust training procedure and it has been found to help the training converge more smoothly. The overall loss function is the sum of the component loss functions: $\mathscr{L}(\,X_s,\hat{X}_s) = \mathscr{L}_R(\,X_s,\hat{X}_s) + \mathscr{L}_I(\,X_s,\hat{X}_s)\,.$

B. Dereverberation through network training

To train the network to perform dereverberation on the input noisy and reverberant speech signal, we provide dereverberated counterpart of the input signal as the target signal during training. For this, we start off with a database containing original speech RIRs for training (denoted by h_s), and for each speech RIR, we generate a dereverberated speech RIR counterpart (denoted by h_d) using the equation below:

$$h_d(m) = \begin{cases} h_s(m) & \text{for } m < m_0 \\ h_s(m)\exp[-a(m-m_0)] & \text{for } m \geq m_0, \end{cases} \tag{8}$$

where m_0 denotes the sample corresponding to the largest absolute value of h_s and a denotes the exponential decay factor that controls how quickly h_d decays beyond m_0. The benefit of this method is that it results in RIRs that closely mimic naturally occurring RIRs with faster decay time than that of h_s. Ad hoc methods such as arbitrary truncation of the speech RIR are not desirable because they introduce artifacts in the target speech signal. Further, Eqn. (8) is fully parameterized by a single parameter a, which makes it easier to control the effective length of h_d. Typical values of a are in the range of [0.001, 0.005]. Lastly, we provide the STFT of the dereverberated speech signal as the target signal for optimization:

$$\text{Target Signal: } X_d(l,k) = \text{STFT}\left\{x_d(n) \triangleq h_d * s(n)\right\} \tag{9}$$

C. Ideal Speech/Noise Segmentation for Loss Function

During training, the network is presented with a wide variety of noisy signals. These signals have two distinct parts: a part where both speech and noise are present (called the speech-active region), and a part where only noise is present (called the speech-inactive region). For the speech-active region, minimization of the loss function translates to maximizing the SNR of the input signal. For the speech-inactive region, minimization of the loss function translates to maximizing the noise reduction on the input signal. For the SE problem, there's an implicit tradeoff between maximization of SNR and NR levels. However, during the training procedure, the network is provided with samples that have varying proportions of speech-active and speech-inactive regions. Consequently, the convergence process slows down as the network has to continuously balance the tradeoff between SNR and NR improvements. Thus, it is important to have a mechanism to directly control this tradeoff during training.

Towards this, we propose to use ideal speech/noise segmentation for the training samples. We first mark the speech-active and speech-inactive regions for each of the N training samples of a mini-batch using a VAD algorithm. Let $L_{i,s}$ and $L_{i,v}$ denote the number of speech-active and speech-inactive frames, respectively, for the ith training sample; then $L_i = L_{i,s} + L_{i,v}$ denotes the total number of frames for the ith training sample. Next, we define the total number of speech-active and speech-inactive frames for the mini-batch as:

$$L_s = \sum_{i=1}^{N} L_{i,s}, \qquad L_v = \sum_{i=1}^{N} L_{i,v}. \tag{10}$$

The loss function for the speech-active region is defined as:

$$\mathscr{L}_s(\,X_d,\hat{X}_d) = \mathscr{L}_{R,s}(\,X_d,\hat{X}_d) + \mathscr{L}_{I,s}(\,X_d,\hat{X}_d)\,, \tag{11}$$

where $\hat{X}_d$ is the network's approximation for X_d and

$$\mathscr{L}_{R,s}(X_d, \hat{X}_d) \quad (12)$$
$$= \frac{1}{gKL_sN} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{L_s} | X^i_{d,R}(l,k) - \hat{X}^i_{d,R}(l,k) |,$$
$$\mathscr{L}_{I,s}(X_d, \hat{X}_d)$$
$$= \frac{1}{gKL_sN} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{L_s} | X^i_{d,I}(l,k) - \hat{X}^i_{d,I}(l,k) |,$$

where it is understood that the summation over l in Eqn. 12 is conducted only for speech-active frames. Next, by substituting L_s with L_v and computing the terms in Eqns. (11) and (12) for speech-inactive frames, we obtain $\mathscr{L}_v(X_d, \hat{X}_d)$, the loss function for the speech inactive region. We now combine the speech-active and speech-inactive component loss functions into an overall training loss function:

$$\mathscr{L}(X_d, \hat{X}_d) = \lambda \mathscr{L}_s(X_d, \hat{X}_d) + (1-\lambda)\mathscr{L}_v(X_d, \hat{X}_d), \quad (13)$$

where $\lambda \in [0, 1]$ is a constant that determines the proportion of speech-active and speech-inactive loss functions in the overall loss function. In this work, we set $\lambda = 0.5$.

IV. Baseline DNNSE System

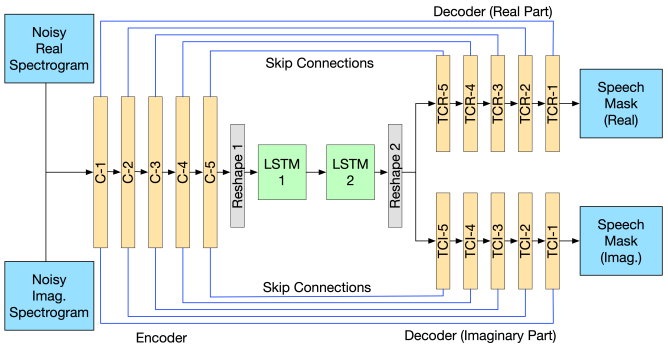

Fig. 2: Architecture block diagram for the baseline CRN-C network.

For the baseline solution, we employ the CRN-C architecture of [11] (refer Figure 2 and Table I) in which two LSTM (Long Short Term Memory) units are stacked between the encoder and decoder convolutional layers. The main idea here is that the convolutional layers would utilize the local information (e.g. spectral) while the recurrent units would utilize the long-term context in the input data (e.g. temporal). Skip connections are placed between tensor-conformal encoder and decoder layers to promote gradient flow. Note that we use the labels C, TCR, and TCI to denote convolutional, transpose convolutional real, and transpose convolutional imaginary layers, respectively.

For all convolutional and transpose convolutional layers (except the output layers), the CRN-C uses a batch normalization (BN) layer followed by an exponential linear unit (ELU) nonlinearity. In the output layer, it uses a BN layer followed by linear activation. The number of convolutional kernels is kept symmetric; the number of kernels is gradually increased in the encoder, while it is gradually decreased in the decoder. For larger context along the frequency direction, a stride of 2 along the frequency dimension is used for all convolutional and transpose convolutional layers. In Table I, for each layer, we provide the dimension of the input and output tensors in the form of *frequencyBins*×*timeSteps*×*featureMaps*. The layer configuration is provided in the form of *kernelFrequency*×*kernelTime, (strideFrequency*×*strideTime), numberOfFilters*.

TABLE I: Architectural details for the baseline CRN-C network.

Layer Name	Input Size	Layer Configuration	Output Size
C-1	161xTx2	3x1, (2x1), 16	80xTx16
C-2	80xTx16	3x1, (2x1), 32	39xTx32
C-3	39xTx32	3x1, (2x1), 64	19xTx64
C-4	19xTx64	3x1, (2x1), 128	9xTx128
C-5	9xTx128	3x1, (2x1), 256	4xTx256
Reshape 1	4xTx256	Default	1024xT
LSTM 1	1024xT	Default	1024xT
LSTM 2	1024xT	Default	1024xT
Reshape 2	1024xT	Default	4xTx256
TCI-5, TCR-5	4xTx512	3x1, (2x1), 128	9xTx128
TCI-4, TCR-4	9xTx256	3x1, (2x1), 64	19xTx64
TCI-3, TCR-3	19xTx128	3x1, (2x1), 32	39xTx32
TCI-2, TCR-2	39xTx64	3x1, (2x1), 16	80xTx16
TCI-1, TCR-1	80xTx32	3x1, (2x1), 1	161xTx2

Note that the baseline architecture with 17 M parameters uses a 320-point DFT filterbank with 50% overlap and Hann window [11]. To bring parity with our choice of 512-point DFT filterbank for the CRN-D models, we trained another CRN-C model that uses a 512-point filterbank, but we reconfigured the number of CNN filters so that the new model uses 20 M parameters, which is close to the number of parameters used in the baseline architecture. The number of filters used for C-1, C-2, C-3, C-4, and C-5 layers are 10, 20, 40, 80, and 160, respectively, and the dimension of input tensor is 257xTx2.

V. Robust CRN-D Architecture

The baseline CRN-C model can be trained to perform well under controlled acoustic conditions. However, we found a few shortcomings which rendered CRN-C unsuitable for real-world applications: (a) it is a large model (17 M parameters), which makes it difficult to deploy it for real-time solutions, (b) reducing the number of parameters also resulted in performance loss, and (c) as we show later, the CRN-C model does not generalize well to unseen acoustic conditions. To address these, we develop the CRN-D architecture that improves upon the CRN-C architecture as follows:

1) The CRN-D architecture makes use of dense layers (instead of standard convolutional layers) in the encoder and decoder structure. Dense layers were first introduced for the DCNN architecture in [14], and the authors showed that this architecture achieved (or improved) SOTA results by using a relatively smaller model; the dense connections promote feature reuse, allow late fusion, and alleviate vanishing gradient problem. The dense convolutional layers in the CRN structure allows us to train deeper networks more effectively for the SE problem.
2) Instead of using separate decoders for the real and imaginary parts of the spectrum, we used a single decoder for the CRN-D architecture. This makes sense conceptually because the

real and imaginary parts of the mask are naturally correlated. This also helped in reducing the model complexity without any loss in performance (based on our empirical studies).

3) To further reduce the computational complexity, we replaced the LSTM units with gated recurrent units (GRUs). Based on empirical studies, this did not result in any loss in performance for the speech enhancement task.

Next, we present the concept of a dense block before presenting the CRN-D network.

A. Dense Block

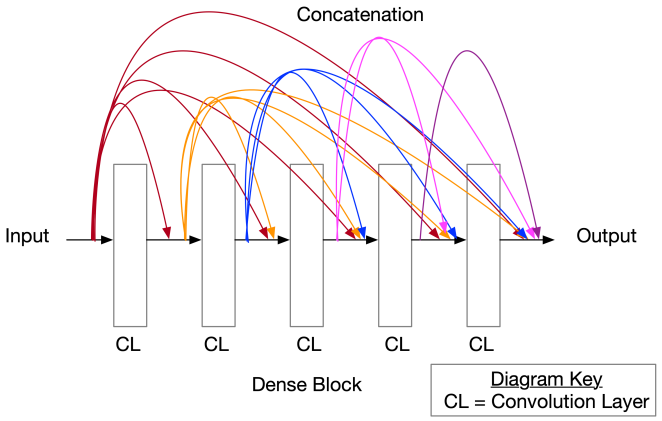

Fig. 3: Illustration of dense block used in CRN-D architecture.

Figure 3 illustrates a dense block with five convolutional layers, where the input to a given layer is a concatenation of the feature maps of all preceding layers [14]. Every convolutional layer in the dense block performs a AxB 2-D causal convolution in the encoder and a causal transpose convolution in the decoder. This ensures that the CRN-D can be used for real-time deployment. We have used values of $A = 3$ (across frequency) and $B = 1$ (across time) to ensure causality. The dense block has a growth rate of G which is the number of feature maps produced by each convolutional layer. In this work, we set $G = 48$.

B. Overall CRN-D Architecture

The CRN-D architecture is presented in Figure 4 (and Table II). Here, two GRU units are placed between the encoder and decoder that are composed of dense convolutional layers. Further, every stage in the encoder and decoder consists of a dense block followed by a transition layer. The dense block performs causal convolutions with filters of size $(3, 1)$ and a stride of $(1, 1)$. Every convolution in the dense block is followed by an ELU nonlinearity. The transition layer performs a CxD 2-D convolution and produces H feature maps. In the transition layer, we have used $C = 3$, $D = 1$ and $H = 48$ and the convolutions are performed with a stride of $(2, 1)$. Except for the output layer, every convolution in the transition layer is followed by an ELU non-linearity. In the output transition layer, a linear activation function is used. The decoder is composed of 2-D transpose convolutions in every stage.

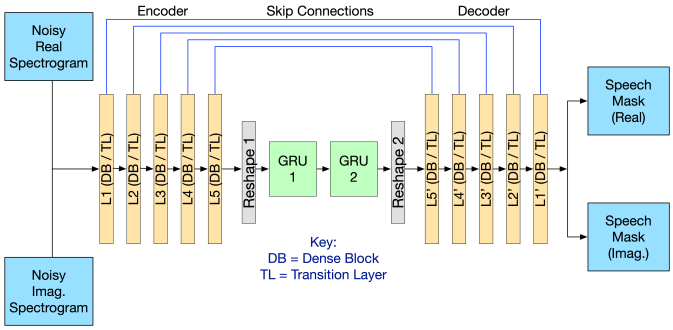

Fig. 4: Architecture block diagram for the CRN-D network.

TABLE II: Architectural details for the CRN-D network.

Layer	Input Dimension	Output Dimension
L1	257xTx2	128xTx48
L2	128xTx48	63xTx48
L3	63xTx48	31xTx48
L4	31xTx48	15xTx48
L5	15xTx48	7xTx48
Reshape 1	7xTx48	336xT
GRU 1	336xT	336xT
GRU 2	336xT	336xT
Reshape 2	336xT	7xTx48
L5´	7xTx96	15xTx48
L4´	15xTx96	31xTx48
L3´	31xTx96	63xTx48
L2´	63xTx96	128xTx48
L1´	128xTx96	257xTx2

TABLE III: Summary of models (A through H) trained to evaluate performance across architectures and acoustic conditions. Model size is specified in millions (M) of parameters.

Model	Architecture	Model size	FFT size	Gain varn.	Reverb varn.	Value of a
A	CRN-C	17.43 M	320	No	No	N/A
B	CRN-C	20.33 M	512	No	No	N/A
C	CRN-D	2.93 M	512	No	No	N/A
D	CRN-D	2.93 M	512	Yes	No	N/A
E	CRN-D	2.93 M	512	Yes	Yes	0.0
F	CRN-D	2.93 M	512	Yes	Yes	0.001
G	CRN-D	2.93 M	512	Yes	Yes	0.002
H	CRN-D	2.93 M	512	Yes	Yes	0.004

VI. Training and Experimental Setup

In order to train the network, we used an internal database with the following specifications:

1) The training set consisted of approximately 20,000 speech files, 3,100 noise files, and 9,000 speech RIRs. The speech files were acquired from WSJ1 database with a balance between male and female voices [17]. The noise files represented a wide variety of environmental sounds and were acquired from [18]. The speech RIRs were acquired from an internal database with RT60 length varying from 60 ms to 500 ms.
2) The test set consisted of approximately 4,800 speech files, 1,050 noise files, and 350 speech RIRs.
3) The validation set consisted of approximately 4,000 speech files, 650 noise files, and 350 speech RIRs.

Note that there was no overlap between the train, test, and validation sets. Further, care was taken to not use the same speakers from WSJ1 database across the train, test, and validation sets. Lastly, for the speech RIRs, we also generated

TABLE IV: STOI and SI-SDR scores for models A through H for Acoustic Condition 1.

	STOI (%)					SI-SDR (dB)				
SNR (dB) / Model	Avg	-5	0	5	10	Avg	-5	0	5	10
Raw	82.4	69.1	78.9	87.2	93.0	2.6	-4.7	0.1	5.3	10.1
A	91.7	82.8	90.5	94.6	96.8	12.1	6.8	10.6	14.1	17.2
B	92.0	82.8	90.7	94.9	97.0	12.7	7.3	11.2	14.7	17.9
C	91.4	81.2	90.1	94.7	97.0	12.7	7.2	11.1	14.8	18.1
D	**92.2**	**82.9**	**91.1**	**95.2**	**97.3**	**13.6**	**8.0**	**12.0**	**15.6**	**18.9**
E	90.5	80.1	89.0	94.0	96.5	12.4	6.9	10.9	14.3	17.6
F	91.0	81.4	89.7	94.3	96.6	12.5	7.3	11.1	14.5	17.6
G	91.0	81.3	89.7	94.3	96.6	12.2	7.1	10.8	14.0	16.8
H	90.5	80.2	89.2	93.9	96.2	11.3	6.7	10.2	13.1	15.2

TABLE V: STOI and SI-SDR scores for models A through H for Acoustic Condition 2.

	STOI (%)					SI-SDR (dB)				
SNR (dB) / Model	Avg	-5	0	5	10	Avg	-5	0	5	10
Raw	82.4	68.5	79.5	87.5	92.9	2.7	-4.6	0.2	5.3	10.1
A	84.0	73.9	83.9	87.0	90.1	6.0	1.3	5.8	7.3	9.9
B	85.9	76.3	85.9	88.7	91.2	8.8	4.3	8.2	10.1	12.4
C	86.3	76.0	85.6	89.8	92.7	9.9	5.3	9.0	11.5	13.8
D	**92.3**	**82.5**	**91.3**	**95.4**	**97.4**	**13.7**	**8.0**	**12.0**	**15.7**	**18.9**
E	90.4	79.6	89.2	94.0	96.4	12.3	6.9	10.7	14.3	17.5
F	91.0	80.9	89.9	94.3	96.6	12.5	7.1	11.0	14.5	17.5
G	91.1	80.9	89.9	94.4	96.6	12.2	7.0	10.7	14.1	16.7
H	90.5	76.9	89.3	94.0	96.2	11.3	6.5	10.1	13.1	15.1

TABLE VI: STOI and SI-SDR scores for models A through H for Acoustic Condition 3.

	STOI (%)					SI-SDR (dB)				
SNR (dB) / Model	Avg	-5	0	5	10	Avg	-5	0	5	10
Raw	78.7	63.6	74.8	84.1	90.8	2.9	-4.7	0.2	5.5	10.7
A	64.0	54.4	62.4	67.8	72.3	-2.4	-5.3	-2.5	-1.7	0.6
B	69.1	58.8	67.7	73.0	76.8	1.7	-1.5	1.6	2.7	4.6
C	73.1	60.6	71.2	77.5	81.7	4.9	1.4	4.4	6.0	8.0
D	75.5	62.2	73.4	80.9	83.6	6.1	2.6	5.5	7.6	8.8
E	**86.8**	**73.6**	**84.3**	**91.4**	**95.1**	**11.5**	**6.1**	**9.9**	**13.4**	**17.1**
F	85.4	73.3	83.1	89.7	92.9	9.2	5.3	8.2	10.6	12.4
G	82.8	71.2	80.7	87.0	89.9	7.2	4.1	6.6	8.4	9.4
H	78.6	67.1	76.9	82.7	85.2	4.9	2.6	4.7	5.8	6.4

the dereverberated version of the RIRs (refer Section III-B) using values of a as 0.001, 0.002, and 0.004.

To perform a rigorous analysis on the speech enhancement architectures and models, we built and tested models under the following acoustic conditions:

1) Acoustic Condition 1: only SNR variations on the training samples.
2) Acoustic Condition 2: both SNR and gain variations are applied on the training samples.
3) Acoustic Condition 3: SNR, gain, and reverberation variations are used. The target signal is the clean reverberant speech signal.
4) Acoustic Condition 4: SNR, gain, and reverberation variations are used. The target signal is created by filtering speech signal with a dereverberated version of the impulse response using $a = 0.001$.
5) Acoustic Conditions 5 and 6 are same as Acoustic Condition 4, except that they use a as 0.002 and 0.004, respectively.

We used RMSprop as the optimization function in Keras, and the learning rate was gradually reduced from 0.001 to 0.00002. For each architecture, models were selected once the validation loss had sufficiently converged. Training was performed using the following general recipe: (a) Read 512 validation files and hold them fixed throughout the training process to track validation loss; (b) Generate N=512 samples (6-8 s long) on-the-fly for a given mini-batch using Eqn. 3. These samples should contain acoustic variations as per Acoustic Conditions 1-6 and parameters specified in Section II; (c) Generate the target signal for training by convolving the anechoic speech signal with dereverberated speech RIR; (d) To simulate the effect of training for long sequences, concatenate multiple training and validation samples. Additionally, Keras provides a *stateful* training option that allows the RNN states to be maintained across mini-batches (until the states are reset), which helps the network to generalize for rapidly changing acoustic conditions.

For performance evaluation, a test set was created for each of the six conditions specified above. The test set comprised of 7400 files uniformly distributed from -5 dB to 10 dB in steps of 1 dB (approximately 460 files per dB). Similar to the training and validation databases, these files were also created with SNR, reverberation, and gain variations. We used this test set to measure the STOI and SI-SDR metrics along with

TABLE VII: STOI and SI-SDR scores for models A through H for Acoustic Condition 4.

	STOI (%)					SI-SDR (dB)				
SNR (dB) / Model	Avg	-5	0	5	10	Avg	-5	0	5	10
Raw	76.2	61.9	72.4	81.5	87.9	1.6	-5.2	-0.3	4.0	7.9
A	64.9	55.7	63.6	68.6	72.5	-2.8	-5.5	-2.8	-2.0	-0.5
B	69.7	59.9	68.7	73.5	76.9	1.2	-1.8	1.3	2.1	3.5
C	73.2	61.2	71.5	77.4	81.1	4.2	1.1	3.9	5.2	6.5
D	76.3	63.2	74.4	81.4	83.9	5.7	2.4	5.2	7.0	7.8
E	84.7	72.4	82.6	89.0	92.3	8.4	4.9	7.7	9.7	11.0
F	**87.0**	**74.6**	**84.7**	**92.2**	**94.5**	**10.1**	**5.7**	**9.0**	**11.6**	**13.7**
G	86.1	73.7	83.8	90.4	93.6	9.3	5.2	8.3	10.8	12.5
H	83.4	70.8	81.4	87.8	90.7	7.2	3.9	6.7	8.4	9.3

TABLE VIII: STOI and SI-SDR scores for models A through H for Acoustic Condition 5.

	STOI (%)					SI-SDR (dB)				
SNR (dB) / Model	Avg	-5	0	5	10	Avg	-5	0	5	10
Raw	74.3	61.4	70.9	79.1	84.9	0.5	-5.8	-1.1	2.7	5.7
A	64.0	55.2	62.8	67.6	71.1	-3.7	-6.0	-3.5	-2.9	-1.8
B	68.5	59.2	67.6	72.2	75.3	0.1	-2.6	0.4	0.9	2.0
C	71.6	60.2	70.1	75.6	78.9	2.9	0.2	2.8	3.7	4.6
D	74.9	62.4	73.2	79.7	82.0	4.3	1.4	4.1	5.4	5.8
E	81.9	70.4	80.0	85.8	88.8	5.9	3.3	5.6	6.8	7.6
F	85.5	**73.4**	83.2	89.6	92.8	7.9	4.5	7.3	9.1	10.4
G	**85.7**	73.3	**83.4**	**89.9**	**93.1**	**8.3**	**4.6**	**7.5**	**9.6**	**11.0**
H	84.1	71.2	81.9	88.5	91.5	7.4	3.9	6.8	8.6	9.8

TABLE IX: STOI and SI-SDR scores for models A through H for Acoustic Condition 6.

	STOI (%)					SI-SDR (dB)				
SNR (dB) / Model	Avg	-5	0	5	10	Avg	-5	0	5	10
Raw	71.7	60.1	68.8	76.0	81.1	-1.1	-6.7	-2.3	0.9	3.1
A	62.4	54.1	61.5	65.9	69.0	-5.0	-7.0	-4.8	-4.3	-3.6
B	66.7	57.9	66.0	70.3	72.9	-1.5	-3.8	-1.2	0.9	0.0
C	69.4	58.6	68.1	73.2	76.1	1.1	-1.2	1.0	1.7	2.2
D	72.8	60.8	71.3	77.3	79.3	2.3	-0.1	2.3	3.1	3.3
E	78.5	67.8	76.9	82.1	84.4	3.2	1.3	3.1	3.8	4.3
F	82.6	71.2	80.6	86.5	89.4	5.0	2.6	4.8	5.9	6.5
G	**83.6**	**71.6**	**81.5**	**87.7**	**90.7**	5.8	**2.9**	5.4	6.8	7.6
H	83.2	70.3	81.1	87.6	**90.7**	**6.0**	**2.9**	**5.6**	**7.1**	**8.1**

measuring generalizability to unseen acoustic conditions.

We trained eight different models that cover Acoustic Conditions 1-6. These are specified in Table III along with the model size and acoustic variations used during training. Note that the digital latency incurred by model A is 20 ms while that for models B-H is 32 ms. This makes these solutions suitable for real-world deployment.

VII. Performance Analysis

We now evaluate the performance of models A through H for Acoustic Conditions 1 through 6. This helps us to understand how the models fare across various conditions when they are trained/not trained for that condition.

Table IV provides the measured STOI and SI-SDR scores for models A through H for Acoustic Condition 1 (i.e., only SNR variations but no gain or reverb variations). We note that all models provide an improvement over the raw microphone performance. For example, model A improves the STOI and the SI-SDR of raw microphone from 82.4% to 91.7% and from 2.6 dB to 12.7 dB, respectively. Note that model D that provides the best average STOI and SI-SDR improvement uses 3 M parameters compared to model B that uses 20 M parameters.

Table V summarizes the results for Acoustic Condition 2. Model D performs the best as it is trained for this condition. There's a slight loss in performance for models E, F, G, and H, but the performance degrades more significantly for models A, B, and C. This is mainly because models E, F, G, and H were trained with gain variations (in addition to reverb variations) in comparison with models A, B, and C that were not trained with gain variations. We note that model A performs poorly compared to model B. This suggests that the use of 512-point filterbank with 75% overlap is beneficial over the 320-point filterbank with 50% overlap; this is possibly because of increased frequency resolution with 512-point filterbank and that the increased overlap with 512-point filterbank (75% vs 50%) helps with generation of smoother masks. We also note that the average STOI and SI-SDR performance of model C is better than that of models A and B, which suggests that CRN-D architecture generalizes better to unseen acoustic condition even while using fewer model parameters.

Table VI summarizes the results for Acoustic Condition 3. Model E performs the best as it is trained for this condition. The performance for models A, B, C, and D degrade significantly,

which suggests that reverberation is an important acoustic condition that needs to be factored in during training. Further, we note that models A and B perform much more poorly than model C, which again reinforces the observation that the CRN-C model does not generalize as well as the CRN-D model to unseen acoustic conditions. Model D performs better than model C because by using gain variations during training, it matches the training condition of model E much more closely than model C. Lastly, model F performs better than models G and H. This is because with a dereverberation factor of $a = 0.001$, its training condition is closer to model E than models G and H that use higher dereverberation factors of $a = 0.002$ and $a = 0.004$, respectively.

Table VII summarizes the results for Acoustic Condition 4. Model F performs the best as it was trained for this condition. The trend for models A through D is similar to our earlier observations. Models E and G show much less loss in performance than other models, which suggests that the performance of CRN-D architecture degrades gracefully and in proportion with the mismatch between training and test conditions. Further, given that model F performs better than models G and H, we infer that our training methodology allows us to apply fine-grained control on the level of dereverberation we seek to achieve through the trained models.

Table VIII summarizes the results for Acoustic Condition 5. Model G performs the best as it is trained for this condition. The trend for models A-D is similar to our earlier observations. We also note that models F and H perform much better than model E; this is mainly because the training conditions for models F and H are closer to that of model G than model E. Table IX provides the measured STOI and SI-SDR scores for models A through H for Acoustic condition 6. The observations drawn here are in line with those for Table VIII.

From the above analyses, we infer that while both the CRN-C and CRN-D models perform well for controlled acoustic conditions, the CRN-D architecture generalizes much better to challenging and unseen acoustic conditions even though it is 5x-6x times smaller than the CRN-C architecture. This suggests that the CRN-D architecture facilitates efficient use of parameters and thereby allows us to train deeper models more effectively. Our analyses also shows that our proposed training methodology is able to address the real-world acoustic variations for the SE problem, and it allows us to exert fine-grained control on the level of dereverberation achieved by the trained models. This is important as it allows us to fine-tune the overall performance of the trained DNNSE models. Lastly our results demonstrate that robustness is an important criterion for designing DNNSE solutions for real-world applications.

VIII. Conclusions

In this paper, we described the design a single microphone deep neural network based speech enhancement (DNNSE) solution for joint noise reduction and dereverberation on smart-speaker products that encounter a wide variety of acoustic challenges during their real-world deployment. We provided a systematic introduction to the acoustic challenges involved and performed a detailed analysis to compare DNNSE solutions for metrics like short-time objective intelligibility (STOI), scale-invariant signal-to-distortion-ratio (SI-SDR), and generalizability to unseen acoustic conditions. We then developed a robust DNNSE solution along with a robust training methodology that are well suited for the acoustic challenges parametrized in this paper. Through our detailed analysis, we demonstrated that our DNNSE solution performs and generalizes better than a baseline solution that is 5x-6x times larger. Our robust training framework can be applied to a wide variety of DNNSE architectures. Future work will involve further reduction of model size by improving architecture efficiency of the DNNSE solutions.

References

[1] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 443–445, 1985.

[2] J. Benesty, S. Makino, and J. Chen, *Speech enhancement*, Springer Science & Business Media, 2005.

[3] R. C. Hendriks, T. Gerkmann, and J. Jensen, "Dft-domain based single-microphone noise reduction for speech enhancement: A survey of the state of the art," *Synthesis Lectures on Speech and Audio Processing*, vol. 9, no. 1, pp. 1–80, 2013.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[5] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, 2018.

[6] Y. Wang, A. Narayanan, and D. Wang, "On training targets for supervised speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 22, no. 12, pp. 1849–1858, 2014.

[7] N. Das, S. Chakraborty, J. Chaki, N. Padhy, and N. Dey, "Fundamentals, present and future perspectives of speech enhancement," *International Journal of Speech Technology*, vol. 24, no. 4, pp. 883–901, 2021.

[8] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal processing letters*, vol. 21, no. 1, pp. 65–68, 2013.

[9] S.-W. Fu, T.-Y. Hu, Y. Tsao, and X. Lu, "Complex spectrogram enhancement by convolutional neural network with multi-metrics learning," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2017, pp. 1–6.

[10] K. Tan and D. Wang, "A convolutional recurrent neural network for real-time speech enhancement.," in *Interspeech*, 2018, pp. 3229–3233.

[11] K. Tan and D. Wang, "Complex spectral mapping with a convolutional recurrent network for monaural speech enhancement," in *International Conference on Acoustics, Speech and Signal Processing*, 2019.

[12] K. Tan, J. Chen, and D. Wang, "Gated residual networks with dilated convolutions for monaural speech enhancement," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 189–198, 2018.

[13] J. Lee, K. Kim, T. Shabestary, and H.-G. Kang, "Deep bi-directional long short-term memory based speech enhancement for wind noise reduction," in *2017 Hands-free Speech Communications and Microphone Arrays (HSCMA)*. IEEE, 2017, pp. 41–45.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[15] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[16] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "Sdr–half-baked or well done?," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 626–630.

[17] "Linguistic Data Consortium Philadelphia, USA: CSR-II (WSJ1) Complete.," https://catalog.ldc.upenn.edu/LDC94S13A, 1994.

[18] "Sound Ideas Ultimate Sound Effects Collection," https://www.sound-ideas.com/Collection/2/Sound-Effects-Collections, 2021.