A DUAL-STAGED CONTEXT AGGREGATION METHOD TOWARDS EFFICIENT END-TO-END SPEECH ENHANCEMENT

Kai Zhen[1,2], *Mi Suk Lee*[3], *Minje Kim*[1,2]

[1]Indiana University, Luddy School of Informatics, Computing, and Engineering, Bloomington, IN
[2]Indiana University, Cognitive Science Program, Bloomington, IN
[3]Electronics and Telecommunications Research Institute, Daejeon, South Korea
zhenk@indiana.edu, lms@etri.re.kr, minje@indiana.edu

ABSTRACT

In speech enhancement, an end-to-end deep neural network converts a noisy speech signal to a clean speech directly in the time domain without time-frequency transformation or mask estimation. However, aggregating contextual information from a high-resolution time domain signal with an affordable model complexity still remains challenging. In this paper, we propose a densely connected convolutional and recurrent network (DCCRN), a hybrid architecture, to enable dual-staged temporal context aggregation. With the dense connectivity and cross-component identical shortcut, DCCRN consistently outperforms competing convolutional baselines with an average STOI improvement of 0.23 and PESQ of 1.38 at three SNR levels. The proposed method is computationally efficient with only 1.38 million parameters. The generalizability performance on the unseen noise types is still decent considering its low complexity, although it is relatively weaker comparing to Wave-U-Net with 7.25 times more parameters.

Index Terms— End-to-end, speech enhancement, context aggregation, residual learning, dilated convolution, recurrent network

1. INTRODUCTION

Monaural speech enhancement can be described as a process to extract the target speech signal by suppressing the background interference in the speech mixture in the single-microphone setting. There have been various classic methods, such as spectral subtraction [1], Wiener-filtering [2] and non-negative matrix factorization [3], to remove the noise without leading to objectionable distortion or adding too much artifacts, such that the denoised speech is of decent quality and intelligibility. Recently, the deep neural network (DNN), a data-driven computational paradigm, has been extensively studied thanks to its powerful parameter estimation capacity and correspondingly promising performance [4][5][6][7].

DNNs formulate monaural speech enhancement either as mask estimation [8] or end-to-end mapping [9]. In terms of mask estimation, DNNs usually take acoustic features in time-frequency (T-F) domain to estimate a T-F mask, such as ideal binary mask (IBM) [10], etc. In comparison, both the input and output of end-to-end speech enhancement DNNs can be T-F spectrograms, or even time domain signals directly without any feature engineering.

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (2017-0-00072, Development of Audio/Video Coding and Light Field Media Fundamental Technologies for Ultra Realistic Tera-media).

In both mask estimation and end-to-end mapping DNNs, dilated convolution [11] serves a critical role to aggregate contextual information with the enlarged receptive field. Gated residual network (GRN) [12] employs dilated convolutions to accumulate context in temporal and frequency domains, leading to a better performance than a long short-term memory (LSTM) cell-based model [13]. In end-to-end setting, WaveNet [14] and its variations also adopt dilated convolution in speech enhancement.

For real-time systems deployed in resource-constrained environment, however, the oversized receptive field from dilated convolution can cause a severe delay issue. Although causal convolution can enable real-time speech denoising [15], it performs less well comparing to the dilated counterpart [12]. Besides, when the receptive field is too large, the amount of padded zeroes in the beginning of the sequence and a large buffer size for online processing can be a burdensome spatial complexity for a small device. Meanwhile, recurrent neural networks (RNN) can also aggregate context through a frame-by-frame processing without relying on the large receptive field. However, the responsiveness of a practical RNN system, such as LSTM [13], comes at the cost of the increased number of model parameters, which is neither as easy to train nor resource-efficient. There has been effort to apply dilated DenseNet [16] or a hybrid architecture to source separation [17], the mechanism to enable dual-staged context aggregate through the heterogeneous model topology has not been addressed.

To achieve efficient end-to-end monaural speech enhancement, we propose a densely connected convolutional and recurrent network (DCCRN), which conducts dual-level context aggregation. The first level of context aggregation in DCCRN is achieved by a dilated 1D convolutional neural network (CNN) component, encapsulated in the DenseNet architecture [18]. It is followed by a compact gated recurrent unit (GRU) component [19] to further utilize the contextual information in the "many-to-one" fashion. Note that we also employ a cross-component identical shortcut linking the output of DenseNet component to the output of GRU component to reduce the complexity of the GRU cells. We also propose a specifically designed training procedure for DCCRN that trains the CNN and RNN components separately, and then finetune the entire model. Experimental results show that the hybrid architecture of dilated DenseNet and GRU in DCCRN consistently outperforms other CNN variations with only one level of context aggregation on untrained speakers. Our model is computationally efficient and provides reasonable generalizability to untrained noises with only 1.38 million parameters.

We describe the proposed method in Section 2, and then provide experimental validation in Section 3. We conclude in Section 4.

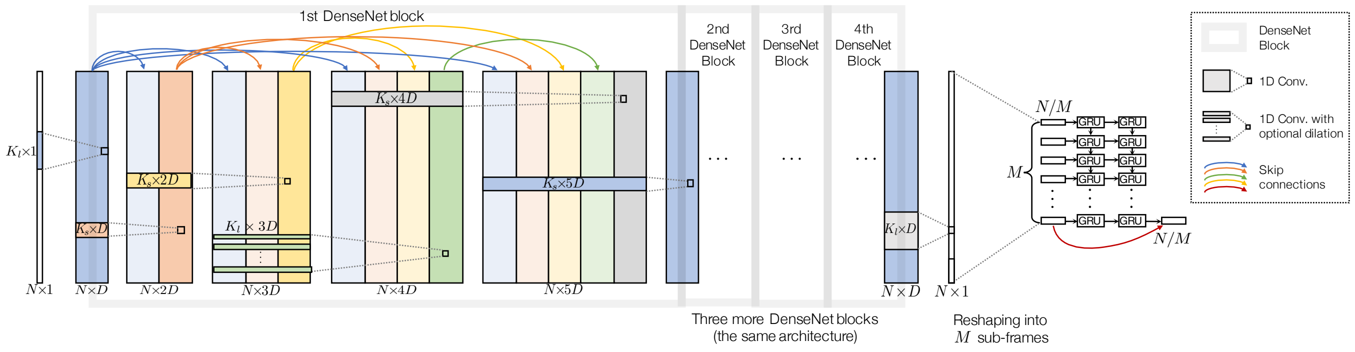

Fig. 1: A schematic diagram of the DCCRN training procedure including dilated DenseNet and GRU components.

2. MODEL DESCRIPTION

2.1. Context aggregation with dilated DenseNet

Residual learning has become a critical technique to tackle the gradient vanishing issue when tuning a deep convolutional neural network (CNN), such that the deep CNN can achieve better performance but with a lower model complexity. ResNet illustrates a classic way to enable residual learning by adding identical shortcuts across bottleneck structures [20]. Although the bottleneck structure includes direct paths to feedforward information from earlier layers to later layers, it does not extend to its full capacity of the information flow. Therefore, ResNet is usually found to be accompanied by a gating mechanism, a technique heavily used in RNNs, such as LSTM or GRU, to further facilitate the gradient propagation in convolutional networks [12].

In comparison, DenseNet [18] resolves the issue by redefining the skip connections. The dense block differs from the bottleneck structure in that each layer takes concatenated outputs from all preceding layers as its input, while its own output is fed to all subsequent layers (Figure 1). Consequently, DenseNet requires fewer model parameters to achieve a competitive performance.

In fully convolutional architectures, the dilated convolution is a popular technique to enlarge the receptive field to cover longer sequences [14], which has shown promising results in speech enhancement [12]. Because of the lower model complexity, dilated convolution is considered as a cheaper alternative to the recurrence operation. Our model adapts this technique sparingly with a receptive field size that does not exceed the frame size.

We use $\mathcal{F}^{(l)}$ to denote a convolution operation between the input $\boldsymbol{X}^{(l)}$ and the filter $\boldsymbol{H}^{(l)}$ in the l-th layer with a dilation rate $\gamma^{(l)}$:

$$\boldsymbol{X}^{(l+1)} \leftarrow \mathcal{F}^{(l)}(\boldsymbol{X}^{(l)}, \boldsymbol{H}^{(l)}, \gamma^{(l)}) \tag{1}$$

$$\boldsymbol{X}^{(l+1)}_{\tau,d} = \sum_{n+k\gamma^{(l)}=\tau} \boldsymbol{X}^{(l)}_{n,d} \boldsymbol{H}^{(l)}_{k,d}, \tag{2}$$

where n, τ, d, k are the indices of the input features, output features, channels, and filter coefficients, respectively. Note that k is an integer with a range $k \leq \lfloor K/2 \rfloor$, where K is the 1D kernel size. In our system we have two kernel sizes: $K_s = 5$ and $K_l = 55$. As DCCRN is based on 1D convolution, the tensors are always in the shape of (features)$\times$(channels). Zero padding keeps the number of features the same across layers. Given the dilation rate $\gamma^{(l)} > 1$ [11], the convolution operation is defined in (2) with the dilation being activated.

In DCCRN, a dense block combines five such convolutional layers. In each block, the input to the l-th layer is a channel-wise concatenated tensor of all preceding feature maps in the same block, thus substituting (1) with

$$\boldsymbol{X}^{(l+1)} \leftarrow \mathcal{F}^{(l)}\Big(\big[\boldsymbol{X}^{(l)}, \boldsymbol{X}^{(l-1)}, \cdots, \boldsymbol{X}^{(l_b)}\big], \boldsymbol{H}^{(l)}, \gamma^{(l)}\Big), \tag{3}$$

where $\boldsymbol{X}^{(l_b)}$ denotes the first input feature map to the b-th block. Note that in this DenseNet architecture, $\boldsymbol{H}^{(l)}$ grows its depth accordingly, i.e., $\boldsymbol{H}^{(l)} \in \mathbb{R}^{K\times(l-l_b+1)D}$ with a growing rate D, the depth of $\boldsymbol{X}^{(l_b)}$. In the final layer of a block, the concatenated input channels collapse down to D, which forms the input to the next block. The first dense block in Figure 1 depicts this process. We stack four dense blocks with the dilation rate of the middle layer in each block to be 1, 2, 4 and 8, respectively. Different from the original DenseNet architecture, we do not apply any transition in-between blocks, except for the very first layer, prior to the stacked dense blocks, expanding the channel of the input from 1 to D, and another layer right after the stacked dense blocks to reduce it back to 1. This forms our fully convolutional DenseNet baseline. In all the convolutional layers, we use leaky ReLU as the activation.

2.2. Context aggregation with gated recurrent network

DCCRN further employs RNN layers following the dilated DenseNet component (Figure 1). Among LSTM and GRU, two most well-known RNN variations, DCCRN chooses GRU for its reduced computational complexity compared to LSTM. The information flow within each unit is outlined as follows:

$$\boldsymbol{h}(t) = \big(1 - \boldsymbol{z}(t)\big) \odot \boldsymbol{h}(t-1) + \boldsymbol{z}(t) \odot \tilde{\boldsymbol{h}}(t) \tag{4}$$

$$\tilde{\boldsymbol{h}}(t) = \tanh\Big(\boldsymbol{W}_h \boldsymbol{x}(t) + \boldsymbol{U}_h\big(\boldsymbol{r}(t) \odot \boldsymbol{h}(t-1)\big)\Big) \tag{5}$$

$$\boldsymbol{z}(t) = \sigma\big(\boldsymbol{W}_z \boldsymbol{x}(t) + \boldsymbol{U}_z \boldsymbol{h}(t-1)\big) \tag{6}$$

$$\boldsymbol{r}(t) = \sigma\big(\boldsymbol{W}_r \boldsymbol{x}(t) + \boldsymbol{U}_r \boldsymbol{h}(t-1)\big), \tag{7}$$

where t is the index in the sequence. $\boldsymbol{h}$ and $\tilde{\boldsymbol{h}}$ are the hidden state and the newly proposed one, which are mixed up by the update gate $\boldsymbol{z}$ in a complementary fashion as in (4). The GRU cell computes the tanh unit $\tilde{\boldsymbol{h}}$ by using a linear combination of the input $\boldsymbol{x}$ and the gated previous hidden state $\boldsymbol{h}$ as in (5). Similarly, the gates are estimated using another sigmoid units as in (6) and (7). In all linear operations, GRU uses corresponding weight matrices, $\boldsymbol{W}_h, \boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{U}_h, \boldsymbol{U}_z, \boldsymbol{U}_r$. We omit bias terms in the equations.

The GRU component in this work follows a “many-to-one” map-

Algorithm 1 The feedforward procedure in DCCRN

```
1: Input: $N$ samples from the noisy utterance, $\boldsymbol{x}$
2: Output: The last $M/N$ samples of the denoised signal, $\hat{\boldsymbol{s}}$
3: DenseNet denoising: $\boldsymbol{X}^{(L)} \leftarrow \mathcal{D}(\boldsymbol{x}; \mathbb{W}^{\text{CNN}})$
4: Reshaping: $\bar{\boldsymbol{X}}^{(L)} \leftarrow \big[\boldsymbol{X}^{(L)}_{1:N/M}, \boldsymbol{X}^{(L)}_{N/M+1:2N/M}, \cdots,$
5:                  $\boldsymbol{X}^{(L)}_{N-N/M+1:N}\big]$
6: GRU denoising: $\hat{\boldsymbol{s}} \leftarrow \mathcal{G}(\bar{\boldsymbol{X}}^{(L)}; \mathbb{W}^{\text{RNN}})$
7: Post windowing: $\hat{\boldsymbol{s}} \leftarrow \text{Hann}(\hat{\boldsymbol{s}})$   {# at test time only}
```

ping style for an additional level of context aggregation. During training, it looks back M time steps and generates the output corresponding to the last time step. To this end, DCCRN reshapes the output of the CNN part, the $N \times 1$ vector, into M sub-frames, each of which is an N/M-dimensional input vector to the GRU cell. We have two GRU layers, one with 32 hidden units and the other one with N/M units to match the output dimensionality of the system. Furthermore, to ease the optimization and to limit the model complexity of the GRU layers, we pass the last N/M sub-frame output of the DenseNet component to the output of GRU component via a skipping connection, which is additive as in the ResNet architecture—the denoised speech is the sum of the output from both components. With the dilated DenseNet component well-tuned, its output will already be close to the clean speech, which leaves less work for GRU to optimize, as detailed in Section 2.5.

2.3. Data flow

During training, as illustrated in Figure 1, the noisy frame is first fed to the DenseNet component $\mathcal{D}(\boldsymbol{x}; \mathbb{W}^{\text{CNN}})$ (line 3 in Algorithm 1). It comprises of L consecutive convolutional layers that are grouped into four dense blocks, where $\mathbb{W}^{\text{CNN}} = \{\boldsymbol{W}^{(1)}, \cdots, \boldsymbol{W}^{(L)}\}$. The output frame of DenseNet, containing N samples, is then reformulated to a sequence of N/M dimensional vectors, $\bar{\boldsymbol{X}}^{(L)} \in \mathbb{R}^{N/M \times M}$, which serve as the input of the GRU component: $\hat{\boldsymbol{s}} \leftarrow \mathcal{G}(\bar{\boldsymbol{X}}^{(L)}; \mathbb{W}^{\text{RNN}})$. The cleaned-up signal $\hat{\boldsymbol{s}}$ corresponds to the final state of the GRU with the dimension of N/M.

At test time, the output sub-frame of DCCRN is weighted by Hann window with 50% overlap by its adjacent sub-frames. Note that to generate the last N/M samples, DCCRN only relies on the current and past samples, up to N within that frame, without seeing future samples, which is similar to causal convolution. Hence, the delay of DCCRN is the sub-frame size ($256/16,000 = 0.016$ second). If it were just for the DenseNet component only, such as those convolutional baselines compared in Section 3, the Hann window with the same overlap rate would still be applied, but the model output would be all N samples for the corresponding frame, instead of the last N/M samples.

Table 1 summarizes the network architecture. The current topology is designed for speech sampled at 16kHz.

2.4. Objective function

It is known that the mean squared error (MSE) itself cannot directly measure the perceptual quality nor the intelligibility, both of which are usually the actual metrics for evaluation. To address the discrepancy, the MSE can be replaced by a more intelligibly salient measure, such as short-time objective intelligibility (STOI) [21]. However, the improved intelligibility does not guarantee a better perceptual quality. The objective function in this work is defined in (8), which is still based on MSE, but accompanied by a regularizer that compares mel spectra between the target and output signals. The TF domain regularizer compensates the end-to-end DNN that would only operate in time domain, otherwise. Empirically, it is shown to achieve better perceptual quality, as proposed in [22].

$$\mathcal{E}(\boldsymbol{s}||\hat{\boldsymbol{s}}) = \text{MSE}(\boldsymbol{s}||\hat{\boldsymbol{s}}) + \lambda \text{MSE}\big(\text{Mel}(\boldsymbol{s})||\text{Mel}(\hat{\boldsymbol{s}})\big). \tag{8}$$

2.5. Model training scheme

We train the CNN and RNN components separately, and then finetune the combined network.

• ***CNN training:*** First, we train the CNN component to minimize the error $\arg\min_{\mathbb{W}^{\text{CNN}}} \mathcal{E}(\boldsymbol{y}||\mathcal{D}(\boldsymbol{x}; \mathbb{W}^{\text{CNN}}))$.

• ***RNN training:*** Next, we train the RNN part by minimizing $\arg\min_{\mathbb{W}^{\text{RNN}}} \mathcal{E}(\boldsymbol{s}||\mathcal{G}(\bar{\boldsymbol{X}}^{(L)}; \mathbb{W}^{\text{RNN}}))$, while $\mathbb{W}^{\text{CNN}}$ is locked.

• ***Integrative finetuning:*** Once both CNN and RNN components are pretrained, we unlock the CNN parameter and finetune both components to minimize the final error: $\arg\min_{\mathbb{W}^{\text{CNN}}, \mathbb{W}^{\text{RNN}}} \mathcal{E}\Big(\boldsymbol{s}||\mathcal{G}\Big(\mathcal{D}\big(\boldsymbol{x}; \mathbb{W}^{\text{CNN}}\big); \mathbb{W}^{\text{RNN}}\Big)$. Note that the learning rate for integrative finetuning should be smaller.

Table 1: Architecture of DCCRN: for the CNN layers, the data tensor sizes are represented by (size in samples, channels), while the CNN kernel shape is (size in samples, input channels, output channels). For the GRU layers, an additional dimension for the data tensors defines the length of the sequence, $M = 4$, while the kernel sizes define the linear operations (input features, output features). The middle layer of each dense block, marked by a dagger, is with larger kernel size $K_l = 55$ and an optional dilation with the rate of 1, 2, 4, and 8, for the four dense blocks, respectively.

Components	Input shape	Kernel shape	Output shape
Change channel	(1024, 1)	(55, 1, 32)	(1024, 32)
DenseNet	(1024, 32)	$\left.\begin{matrix}(5, 32, 32)\\(5, 64, 32)\\(55, 96, 32)^{\dagger}\\(5, 128, 32)\\(5, 160, 32)\end{matrix}\right] \times 4$	(1024, 32)
Change channel	(1024, 32)	(55, 32, 1)	(1024, 1)
Reshape	(1024, 1)	-	(4, 256, 1)
GRU	(4, 256, 1)	(256+32, 32)×3 (32+256, 256)×3	(256, 1)

3. EXPERIMENTS

3.1. Experimental setup

In this paper, the experiment runs on TIMIT corpus [23]. We consider two experimental settings. For the model training, we randomly select 1000 utterances from TIMIT training subset. 5 types of non-stationary noise (birds, cicadas, computer keyboard, machine guns and motorcycles) from [24] are used to create mixtures. Concretely, each clean signal is mixed with a random cut of each of these noise types at a SNR level randomly drawn from the set of integers with the range of $[-5, +5]$ dB. Therefore, 5,000 noisy speech samples, totaling 3.5 hours, are used for model training. At test time, we randomly select 100 unseen utterances from TIMIT test subset,