Speech Separation Using Fully Recurrent CNNs with Delayed Top-Down Integration Through Skip-Connections

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent advances in the design of network architectures, in particular those specialized in modeling sequences, have provided significant improvements in speech separation performance. In this paper, we propose to use a bio-inspired architecture called Multi-Stage Recurrent Neural Network (FRCNN) to solve the separation task. This model contains *bottom-up*, *top-down* and *lateral* connections to fuse information processed at various time-scales represented by *stages*. We present different unfolding schemes, allowing for the network to update those stages in a sequential fashion, in addition to the traditional approach updating stages in parallel. Extensive experiments showed one of the proposed models to achieve competitive or better results on speech separation with high efficiency as compared to other state-of-the-art approaches, on the Libri2mix, WSJ0-2Mix and WHAM! datasets.

1 Introduction

Speech separation aims to extract individual speeches from a mixture of speeches of multiple speakers. It is an important preprocessing step for speech recognition in noisy environment. Recent development of speech separation methods at the waveform level has aroused researchers' interest [16, 17, 15], avoiding the traditional representation of STFT amplitude and phase used in so-called time-frequency (TF) domain methods [6, 8]. Among these so-called time-domain methods, some presented mechanisms fuse information processed at various time scales, such as in FurcaNeXt [24] or Sudo-RM-RF [26], and provide impressive results on the speech separation task.

Evidence from observations of sensory systems of mammals show them to utilize this multi-scale information fusion in their processing. For instance, the visual system includes multiple processing stages (from lower functional areas such as the lateral geniculate nucleus to higher functional areas such as the inferior temporal cortex), which process different scales of information [1]: the higher the stage, the coarser the scale. Those mechanisms and areas have also been identified and located in the auditory system [1]. More importantly, physiological and anatomical studies have revealed abundant recurrent synaptic connections within the same stage (also called *lateral connections*) and bottom-up/top-down synaptic connections between stages [3]. This structure is illustrated in Figure 1a. The intra-stage and inter-stage connections bring different scales of sensory information together and each stage performs information fusion.

However, Figure 1a merely shows a basic structural fact in the brain and does not show the dynamics of the sensory system. Specifically, given an external stimuli, it is unclear the information through which connections arrives at a stage (or a specific neuron) earlier than the information through other connections. In fact, given a stimulus, the neurons along a sensory hierarchy do not fire

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

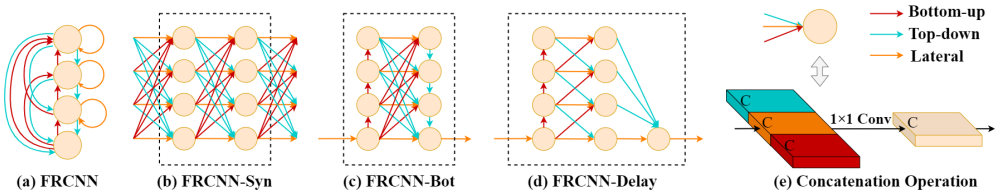

Figure 1: The structure of FRCNN and various micro unfolding schemes. The number of stages $S = 4$. (a) The structure of the FRCNN. Every node denotes a stage. (b-h) Different unfolding schemes in one block. The dashed box in each subfigure indicates the basic building block for constructing a complete RNN (see Figure 3). (i) Multi-scale information fusion for an example stage receiving three types of inputs.

simultaneously as illustrated in Figure 1b [13]; and the time at which a neuron starts to fire depends on a variety of factors including the neuron type, the stage at the sensory pathway, the number of the dendrites connected to it and the morphology of these dendrites. This precludes the possibility of faithfully replicating the sensory system to obtain an excellent artificial neural network (ANN). Nevertheless, the history of ANN development indicates that getting inspiration from the brain is enough to make great progress if task-specific techniques are combined. We investigate here if there exists an asynchronous updating scheme for the structure shown in Figure 1a that provides improvement for speech separation performance.

We will use the convolution operation to implement the bottom-up, top-down and lateral connections as shown in Figure 1a. Therefore we call the model a *fully recurrent convolutional neural network (FRCNN)*. This name emphasizes the presence of both lateral and top-down recurrent connections in the model, distinguishing the model from an existing model [] named *recurrent convolutional neural network (RCNN)* that has lateral recurrent connections only.

For speech separation, the output of the model should be connected to the bottom stage as it has the highest resolution. The major problem with the synchronous updating scheme [13] is that the information transmission from the bottom stage to any upper stage then back to the bottom stage is too fast: one step bottom-up and one step top-down (Figure 1b). To fuse the information at multiple scales comprehensively, we need to increase the number of processing steps before the information goes back to the bottom stage. To this end, we propose a bottleneck model FRCNN-Bot in which the stages are updated first in the bottom-up direction then in the top-down direction (Figure 1c), inspired by the U-net structure []. A simplified and improved model FRCNN-Delay is further proposed, which features delayed top-down integration of all stage through skip-connections. Through extensive experiments, we found that FRCNN-Delay achieved competitive results with the state-of-the-art (SOTA) method DualPathRNN [15] on WSJ0-2mix and WHAM! datasets, and even better results on Libri2Mix dataset. Moreover, FRCNN-Delay was 3 times faster in both training and inference than DualPathRNN.

2 Related Work

2.1 Speech Separation Methods

A typical speech separation method is to model different sources in the TF domain. First, the short-time Fourier transform (STFT) calculates the TF representation of the mixed sound. Second, the subsequent process approximates the clean spectrogram of each source from the mixed spectrogram and uses the inverse STFT (iSTFT) to synthesize the source waveform, as in DPCL [6], uPIT [9] etc.

So-called time-domain methods were also proposed for speech separation, making use of non-STFT encoders for extracting meaningful representations out of the waveform (in a bio-inspired fashion [4], or fully learned over training [17]). While RNN were naturally used to perform speech separation given the sequential aspect of the input representation, they often require long training and inference time [16]. Conv-TasNet [17] has been proposed to solve this problem, replacing RNN with Temporal Convolutional Networks, much faster to train. In addition, DualPathRNN [15] uses a simple segmentation in place of the encoder, and performs on those shorter sequences parallel

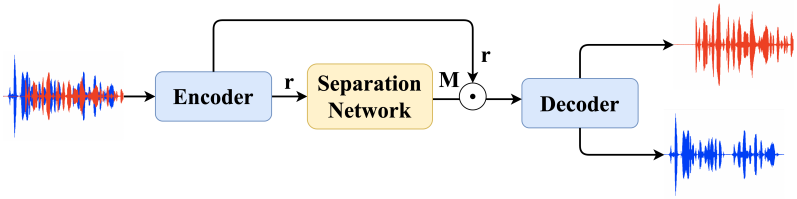

Figure 2: The overall pipeline for speech separation.

intra-chunk and global inter-chunk RNN processing, for a large gain in performance and processing time.

The time-domain methods mentioned above usually take as input the original waveform and process it in a sequential manner, without changing the time scale of analysis. Recently, some models have been proposed for speech separation, where different branches adopt different time resolutions for the processing of their respective inputs, then the outputs are fused with some rule or dedicated module. We call such models *multi-scale fusion (MSF) models*. For instance, Sudo-RM-RF [26] uses a U-Net structure [22], obtaining good results on smaller-sized models. FurcaNeXt [24], a variant of Conv-TasNet [17], uses multiple branch learning methods to improve the performance of speech separation. In addition, MSGT-TasNet [31] uses Transformer [27] to capture features of different scales for speech separation.

2.2 MSF Models for Computer Vision

We discussed MSF models for speech separation above. Now we briefly mention some popular MSF models used in the computer vision field. These MSF models often use multiple branches to process different scales of information then integrate the outputs using feature map concatenation or summation method. These include the U-net [22], the hourglass model [18], the interlinked CNN [32] and the Feature Pyramid Network (FPN) [14]. They have been widely used in computer vision applications such as image segmentation, object detection and human pose estimation.

The lateral and top-down recurrent connections have been modeled by ANN researchers for a long time. In 1990s recurrent connections were introduced into the multi-layer Perceptrons [5] [21], and in 2015 they were introduced into the CNN [11][12]. In 2000s top-down connections were introduced into unsupervised deep learning models [7] [10]. A general framework with both lateral and top-down recurrent connections were proposed in 2016 [13], but only the synchronous updating scheme was presented. We conjecture that different updating schemes may result in different performances as we notice that an asynchronous updating scheme for RCNN performed better than the synchronous updating scheme on an image segmentation task [12]. This servers as another motivation for the current research.

3 Methods

3.1 Overall pipeline

An algorithm dedicated to speech separation aims to extract the individual speech signal of different speakers from a mixture. We denote the waveform of the mixture as $\mathbf{x} \in R^{1\times T}$:

$$\mathbf{x} = \sum_{i=1}^{C} \mathbf{s}_i + \sigma \tag{1}$$

where $\mathbf{s}_i \in R^{1\times T}$ denotes the waveform of speaker i, $\sigma \in R^{1\times T}$ denotes the noise signal, T denotes the number of samples of the signal, and C denotes the number of speakers. The task is to estimate $\mathbf{s}_i$ from $\mathbf{x}$ for all i.

We use the same pipeline as Conv-TasNet [17], as shown in Figure 2. It consists of an encoder, a separation network and a decoder. The encoder divides $\mathbf{x}$ into K overlapping segments $\overline{\mathbf{x}}_k \in R^{1\times L}$ and transforms each segment into a feature vector $\overline{\mathbf{r}}_k \in R^{1\times N}$:

$$\overline{\mathbf{r}}_k = \overline{\mathbf{x}}_k \mathbf{U}_{\mathrm{e}} \tag{2}$$

where $\mathbf{U}_\mathrm{e} \in R^{L\times N}$ is a weight matrix. The two steps can be realized by a trainable 1D convolution with kernel $\mathbf{U}_\mathrm{e}$ and an appropriate stride.

The separation network receives $\overline{\mathbf{r}}_k$ to estimate a mask $\mathbf{M}_i \in R^{1\times N}$ for speaker i. The mask is then used to extract the corresponding source. The detailed structure of the separation network is introduced in Section 3.2.

The decoder reconstructs the waveform segment

$$\overline{\mathbf{s}}_{i,k} = (\overline{\mathbf{r}}_k \odot \mathbf{M}_i)\mathbf{U}_\mathrm{d}^\mathsf{T} \tag{3}$$

where $\mathbf{U}_\mathrm{d} \in R^{L\times N}$ is a weight matrix and $\mathbf{U}_\mathrm{d}^\mathsf{T}$ is the transpose of $\mathbf{U}_\mathrm{d}$, and $\odot$ stands for element-wise multiplication. The estimated waveform $\widehat{\mathbf{s}}_i$ is obtained by summing K overlapping segments $\overline{\mathbf{s}}_{i,k}$. The two steps can be realized by a 1-D transposed convolution operation.

3.2 Separation Network

3.2.1 Structure of FRCNN

We use the FRCNN [13] as the separation network. It is a graph with nodes denoting stages and edges denoting connections. Figure 1a shows an example with $S = 4$ stages. In biological terms, to every node corresponds a set of neurons in a certain stage in the sensory pathway, e.g., the inferior colliculus in the auditory pathway. In our model, every node corresponds to a convolutional layer. Different nodes process different scales of the input information. The higher the node, the coarser the information. There are three types of connections: bottom-up, top-down and lateral connections. The bottom-up and top-down connections are realized by downsampling (e.g., max pooling or convolution with stride>1) and upsampling (e.g., interpolation or transposed convolution) operations, respectively. Note that both bottom-up and top-down connections can be between adjacent stages and non-adjacent stages: in the latter case, the connections are called *skip-connections*. The lateral connections mimic recurrent synapses among neurons in the same sensory area, and can be implemented by canonical convolution (with stride 1), as in [11]. For simplicity, in this study, we copy the feature maps without processing for the next time step.

3.2.2 Unfolding Schemes in the Micro-level

To make this model work in practice, one needs to unfold its different time steps. A commonly used approach is to update the stages in Figure 1a synchronously. This is the unfolding scheme depicted in Figure 1b, used in the original FRCNN work [13]. However, if the stages are allowed to be updated asynchronously, there will be a large number of possible unfolding schemes. For example, without considering the skip connections, we can update the stages one by one in the upward direction then update them one by one in the downward direction, which is the case depicted in Figure 1c. In the present work, we explore seven typical unfolding schemes, named FRCNN-I to FRCNN-VII, as shown in Figure 1.

The first two schemes have been described above. Note that FRCNN-II does not have skip-connections (Figure 1c). In FRCNN-III we add top-down skip connections (Figure 1d), and in FRCNN-IV we add both top-down and bottom-up skip connections (Figure 1e). In these schemes, due to the presence of vertical connections between adjacent stages, the S stages have to be updated sequentially in the upward direction then in the downward direction. We are interested in the performances of the models in which all S stages are updated synchronously, so we change the vertical connections between adjacent stages to be cross connections and obtain FRCNN-V, FRCNN-VI and FRCNN-VII correspondingly (Figures 1f, 1g and 1h). To prevent the nodes in the first columns in FRCNN-V and FRCNN-VI from being isolated, in FRCNN-V we add lateral connections to the input of these

3.2.3 Multi-scale Information Fusion inside Blocks

To make the models depicted in Figures 1b-1h correspond to unfolding schemes as in e.g. RNN, the nodes in the same horizontal row should belong to the same stage (or in biological terms, the same set of neurons in a sensory area). To ensure this, for a node receiving K inputs from different nodes, we concatenate the feature maps from these source nodes in the channel dimension, resulting in KC feature maps (in this study we use C feature maps for every stage), then use a 1×1 convolution to change the number of feature maps to C. Figure 1i illustrates this process. Clearly, this process

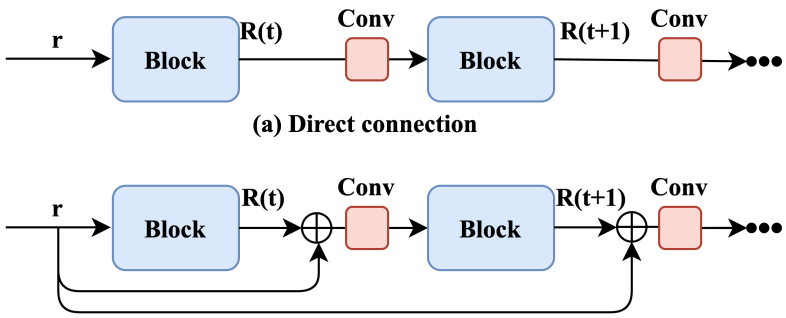

Figure 3: Macro-level unfolding schemes of FRCNN. Every blue box corresponds to a dashed box in Figure 1. The pink boxes in a model denote 1×1 convolutions with shared weights.

performs multi-scale information fusion. It is seen that there are many such nodes in every unfolding scheme, indicating every unfolded structure is a multi-scale information fusion model.

3.2.4 Unfolding Schemes in the Macro-level

Figures 1b-1h show single blocks of the entire unfolding schemes. An entire unfolding scheme usually consists of multiple such blocks with tied weights. If there are B blocks in total, we say "FRCNN is unfolded for B times". At the macro-level, the FRCNN is unfolded by simply repeating these blocks along time such that the output of one block is the input of the next block. To fuse the multi-scale information more exhaustively, we add a 1×1 convolution between two consecutive blocks (Figure 3a). This method is formulated as follows:

$$R(t+1) = f(\varphi(R(t))), \tag{4}$$

where $f(\cdot)$ denotes a block shown in Figures 1b-1h, $R(t)$ denotes the output of the block at time step t and φ denotes 1×1 convolution. This is called the *direct connection (DC)* method. Another idea is to integrate the input of the model with the output of every block via feature map concatenation or summation before sending to the next block. This rule was used in constructing the recurrent CNN in a previous study [11]. Again, we add a 1×1 convolution for more exhaustive information fusion (Figure 3b). Formally,

$$R(t+1) = f(\varphi(R(t) \oplus \mathbf{r})) \tag{5}$$

where $\mathbf{r}$ denotes the input feature maps and $\oplus$ denotes concatenation or summation of two sets of feature maps. This is called the *concatenation connection (CC)* or *summation connection (SC)* depending on which feature map integration method is used.

For single-input-single-output blocks, i.e., FRCNN-{II,III,IV,VI,VII}, we directly use the unfolding methods as described above. For multi-input-multi-output blocks, i.e., FRCNN-{I,V}, we apply these unfolding methods for each input-output pair corresponding to the same stage.

It should be noted that Figures 1b and 2f only illustrate the intermediate blocks of FRCNN-I and FRCNN-V unfolding schemes. In the beginning of unfolding we use downsampling to obtain different scales of feature maps, and in the end of unfolding we use up-sampling to fuse different scales of feature maps.

3.2.5 Relation to Existing Models

Some unfolding schemes of FRCNN are closely related to existing models. FRCNN-II and FRCNN-V have similar structures to SudoRM-RF [26] and iCNN [32], respectively, but FRCNN-II and FRCNN-V use shared weights in different blocks as they correspond to unfolding schemes of an RNN. If the top-down skip-connections in FRCNN-III are replaced with lateral connections to get multi-scale output, this model becomes equivalent to the FPN [14], widely used in computer vision.

3.3 Training Method

When the FRCNN is unfolded through time with any scheme, it becomes a feedforward model and we use the standard BP algorithm to train it. The object is to maximize the scale-invariant signal-to-noise ratio (SI-SNR) [28]. It is also a metric to evaluate the performance of speech separation methods. SI-SNR for each speaker is defined as

$$\text{SI-SNR} = 10 \times \log_{10} \left(\frac{\|\mathbf{A}_{\text{target}}\|_2^2}{\|\mathbf{e}_{\text{noise}}\|_2^2} \right), \tag{6}$$

where

$$\mathbf{A}_{\text{target}} = \frac{\langle \widehat{\mathbf{s}}, \mathbf{s} \rangle \, \mathbf{s}}{\|\mathbf{s}\|_2^2}, \quad \mathbf{e}_{\text{noise}} = \widehat{\mathbf{s}} - \mathbf{A}_{\text{target}}. \tag{7}$$

In above equations, $\mathbf{s} \in R^{1 \times T}$ denotes the original single-source speech, $\widehat{\mathbf{s}} \in R^{1 \times T}$ denotes the estimated speech, $\langle \cdot, \cdot \rangle$ denotes inner product, and $\|\cdot\|_2^2$ denotes l_2-norm. Scale invariance is ensured by normalizing and zero-mean before the calculation.

It is worth noting that the estimated speech signal and the real clean speech signal do not necessarily have the same speaker arrangement order. This is called the *label permutation problem* and we use the utterance-level label permutation invariant training (uPIT) [9] to solve this problem.

4 Experimental Settings

4.1 Dataset

Libri2Mix [2]. This dataset was constructed using train-100, train-360, dev, and test set in the LibriSpeech dataset [19]. Random extracts are selected for different speakers and mixed with uniformly sampled Loudness Units relative to Full Scale (LUFS) [23] between -25 and -33 dB. Random noise samples are added, with loudness uniformly sampled between -38 and -30 LUFS. There are two training set, 212 hours and 58 hours, generated from LibriSpeech train-360 set and train-100 set respectively, namely train-360 and train-100. We use train-100 as the training set unless otherwise specified. But we use the same test set to compare different methods.

WSJ0-2Mix [6]. This dataset contains a 30-hour training set, a 10-hour validation set and a 5-hour test set. It was generated by combining the speech signals of different speakers in the Wall Street Journal corpus. The speaker signals are randomly selected and mixed with a Signal-to-Noise Ratio uniformy sampled between -5dB and 5dB.

WHAM! [30]. This dataset was constructed using similar speech signals as in WSJ0-2Mix, and real environmental noise that were recorded in scenes such as coffee shops, restaurants and bars. The speaker mixtures are mixed with noise with a SNR uniformly sampled between -6dB and 3dB. . Due to the existence of noise, WHAM! is more challenging for speech separation than WSJ0-2Mix.

4.2 Implementation Details

The encoder and decoder in the speech separation pipeline are respectively a 1-D convolutional layer and a 1-D transposed convolutional layer. We set their kernel size to 21, stride to 10, and channel number to 512, i.e., $L = 21$ and $N = 512$. Unless otherwise specified, the number of stages S was set to 5 and the number of channels C in every stage was set to 512. Whenever $C \neq N$, a 1×1 convolutional layer was added between the encoder and the separation network. We used the convolution operation (kernel size 5 and stride 2) to perform downsampling. For example, one operation was used for $2\times$ down-sampling and two consecutive operations were used for $4\times$ down-sampling, and so on. All convolutions were depthwise separable convolution.

We trained all models for 200 epochs on 3-second utterances for Libri2Mix and 4-second utterances for WSJ0-2Mix with 8K Hz sampling rate. Batch size was set to 8. The initial learning rate of Adam optimizer was 1×10^{-3}, and it decayed to $1/3$ of the previous rate every 40 epochs. During training, gradient clipping with a maximum l_2-norm of 5 was used. For each architecture, we picked the best model based on its results on the validation set, then performed testing.

Table 1: Performance of FRCNN-VI on the Libri2Mix test set with different unfolding schemes in the macro-level. DC, CC and SC denote respectively direct connection, concatenation connection and summation connection.

Method	SI-SNRi	SNRi	Params	Inference time (CPU)
DC	12.78	11.38	6.13M	0.70s
CC	14.26	14.83	6.65M	0.96s
SC	15.26	15.76	6.13M	0.72s

Table 2: Performance of FRCNN on the Libri2Mix test set with different unfolding schemes in the micro-level. Each model is unfolded for 8 steps.

Model	SI-SNRi	SNRi	Params	Inference time (CPU)
FRCNN-I	12.10	12.52	9.61M	1.12s
FRCNN-II	11.36	11.75	4.02M	0.39s
FRCNN-III	13.05	13.48	5.06M	0.60s
FRCNN-IV	14.11	14.56	7.21M	0.90s
FRCNN-V	11.43	11.85	6.40M	0.51s
FRCNN-VI	15.26	15.76	6.13M	0.72s
FRCNN-VII	15.59	16.07	6.15M	1.01s
FRCNN-II (Dense)	11.42	11.80	6.13M	0.69s
FRCNN-III (Dense)	12.01	12.45	6.36M	0.88s

All experiments were conducted on a server with Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz and GeForce RTX 1080 Ti 11G $\times 8$. The Pytorch implementation of the FRCNN model can be found in the Supplementary Materials.

5 Results

We used the scale-invariant signal-to-noise ratio improvement (SI-SNRi) and signal-to-noise ratio improvement (SNRi) [28] as the evaluation metrics to measure the speech separation accuracies of models.

5.1 Comparison of Macro-level Unfolding Schemes

We first compared the three macro-level unfolding schemes shown in Figure 1 on Libri2Mix. We chose FRCNN-VI as the basic recurrent block and set the number of unfolding times to 8, i.e., there were 8 identical FRCNN-VI blocks in the unfolded model. The SC method achieved the best results (Table 1). Note that the CC method resulted in more parameters than the DC and SC methods. Therefore in the rest of experiments, we used the SC method as the macro-level unfolding scheme.

5.2 Comparison of Micro-level Unfolding Schemes

We compared the performance of the seven micro-level unfolding schemes on Libri2Mix. We unfolded each scheme for 8 time steps using the SC method. The results are presented in Table 2.

Comparison of FRCNN-{II,III,IV} indicated that the separation accuracy measured by SI-SNRi and SNRi gradually improved by adding top-down and botton-up skip-connection in turn at the cost of more model parameters and longer inference time. We observed similar trends from FRCNN-V to FRCNN-VI and FRCNN-VII. In addition, FRCNN-{V,VI,VII} obtained higher separation accuracies than FRCNN-{II,III,IV}, respectively. Note that FRCNN-{V, VI} had more parameters than FRCNN-{II,III}. For a fair comparison, we changed the number of channels C from 512 to 684 and 580

Table 3: Performance of FRCNN-VI on the Libri2Mix test set with different number of stages S.

S	SI-SNRi	SNRi	Params
3	13.77	14.52	4.02M
4	14.38	14.85	5.08M
5	15.26	15.74	6.13M
6	14.39	14.88	7.19M

Table 4: Performance of different models on the Libri2Mix, WSJ0-2Mix and WHAM! dataset. Conv-TasNet{train-360} indicates that the model was trained on Libri2Mix{train-360}. All other models were trained on Libri2Mix{train-100}, a smaller training set, to evaluate the results on Libri2Mix. The SI-SNRi and SNRi values marked with '*' indicate that they were not reported in the original papers but obtained by using the Asteroid toolkit [20]. A four seconds audio was used to compute the inference time on CPU averaged over 1000 trials.

Model	Libri2Mix		WSJ0-2Mix		WHAM!		Params	Inference Time(s)
	SI-SNRi	SDRi	SI-SNRi	SDRi	SI-SNRi	SDRi		
DPCL++	5.94*	6.61*	10.82	11.21	-	-	13.6M	0.37
uPIT-BLSTM-ST	7.55*	8.22*	9.82	10.00	-	-	92.7M	0.77
Chimera++	6.29*	6.99*	11.50	11.82	9.95	-	32.9M	0.66
BLSTM-TasNet	7.91*	8.68*	13.20	13.61	9.84	-	23.6M	2.90
Conv-TasNet	12.22	12.68	15.31	15.63	12.72	-	5.6M	0.39
Conv-TasNet {train-360}	**14.7**	**15.1**	15.31	15.63	12.72	-	5.6M	0.39
Two-Step TDCN	12.04*	12.45*	16.10	-	-	-	8.63M	1.85
MSGT-TasNet (Light)	-	-	16.80	17.10	12.30	-	37.36M	3.37
MSGT-TasNet (Dense)	-	-	17.00	17.30	13.10	-	66.79M	5.56
SudoRM-RF 0.25x	10.84*	11.25*	13.42	13.61	10.38*	10.80*	0.79M	0.24
SudoRM-RF 0.5x	12.20*	12.61*	15.43	15.63	11.80*	12.21*	1.42M	0.33
SudoRM-RF 1.0x	13.54*	13.95*	17.05	17.27	12.89*	13.31*	2.66M	0.53
SudoRM-RF 2.5x	13.98*	14.40*	17.35*	17.56*	13.69*	**14.10***	6.42M	1.15
DualPathRNN	14.10*	14.57*	**18.8**	**19.0**	**13.71***	14.09*	2.69M	4.66
FRCNN-VI-4	12.48	12.93	15.58	15.78	11.80	12.22	6.13M	0.33
FRCNN-VI-8	15.26	15.74	17.02	17.23	13.25	13.61	6.13M	0.72
FRCNN-VI-16	**16.54**	**17.03**	**18.21**	**18.54**	**14.3**	**14.72**	6.13M	1.51
FRCNN-VI-16 (Light)	15.25	15.71	-	-	-	-	2.47M	1.21

in FRCNN-{II,III}, respectively, and obtained FRCNN-II (Dense) and FRCNN-III (Dense). Table 2 shows that FRCNN-II (Dense) obtained similar results to FRCNN-V, but FRCNN-III (Dense) obtained worse results than FRCNN-VI. FRCNN-I did not yield good results though it had the most number of parameters.

Among the seven micro-level unfolding schemes, we found that FRCNN-VI had a good balance between prediction accuracy and model complexity. Therefore, in what follows, we mainly present the results of this scheme in comparison with existing speech separation models on all three benchmark datasets.

5.3 Influence of the Number of Stages

We compared the performance of FRCNN-VI with different number of stages S (Table 3). When S increased from 3 to 5, the separation results were better and better. However, when $S = 6$ the results started to deteriorate. Therefore we set $S = 5$ for FRCNN-VI in other experiments.

5.4 Comparison with Existing Models

We compared FRCNN-VI with some popular models for speech separation. Some models work in the time-frequency domain: DPCL++ [6], uPIT-BLSTM-ST [9] and Chimera++ [29]. Some models work in the time-domain: BLSTM-TasNet [16], Conv-TasNet [17], Two-Step TDCN [25], MSGT-TasNet [31], SudoRM-RF [26] and DualPathRNN [15]. SudoRM-RF has four variants which are labeled by appending 0.25x, 0.5x, 1.0x and 2.5x to the end of the name, indicating the variants consist of 4, 8, 16 and 40 blocks, respectively.

We tested three variants of FRCNN-VI by unfolding for 4, 8, 16 times in the macro-level. In addition, we tested a model, named FRCNN-VI-16 (Light), on Libri2Mix by setting $C = 324$ to make the number of parameters similar to that of DualPathRNN. The comparison results on the three benchmark datasets are presented in Table 4.

On the Libri2Mix dataset, we made the following observations. First, Conv-TasNet trained on a large training set train-360 achieved the best separation results measured by SI-SNRi and SNRi metrics among existing models. Second, among existing models trained on the smaller training set train-100, DualPathRNN achieved the best results. Third, with more unfolding times in the macro-level, FRCNN-VI achieved better results at the cost of increasing inference time. Fourth, both FRCNN-VI-8 and FRCNN-VI-16 achieved better results than existing models, including the Conv-TasNet version trained on the larger training set train-360. Finally, with similar amount of parameters to DualPathRNN, FRCNN-VI-16 (Light) outperformed DualPathRNN.

Some audio examples from Libri2Mix separated by different models including DualPathRNN and FRCNN-VI-16 are provided in Supplementary Materials. In most examples we found that separated speeches by FRCNN-VI-16 sounded clearer than those by other methods.

On WSJ0-2Mix and WHAM!, again, DualPathRNN achieved the best results among existing models. In comparison, FRCNN-VI-16 obtained competitive results on WSJ0-2Mix and better results on WHAM!.

We noted that DualPathRNN was the slowest method. On Libri2Mix, it took 6 days on 8 GPUs for training while FRCNN-VI-16 took 2 days and FRCNN-VI-16 (Light) took 0.8 days. Morever, its inference on CPU was about 6$\times$ slower than FRCNN-VI-8 and 3$\times$ slower than FRCNN-VI-16 (Table 4). This limits its application in edge computing where GPU is unavailable.

5.5 Transfer to DualPathRNN

Interestingly, we found that the MSF method utilized by FRCNN-VI can be transfered to DualPathRNN to improve its performance. See Supplementary Materials for details.

6 Conclusion

We investigated in this work a bio-inspired FRCNN architecture for solving the speech separation task. The proposed model has multiple stages exchanging and integrating information of different scales through bottom-up, top-down and lateral connections. We explored different unfolding schemes by allowing the stages to be updated sequentially, and found one particular scheme to achieve competitive or even better results with the SOTA model DualPathRNN on several benchmark datasets. Moreover, it was 3 times faster than DualPathRNN in both training and inference.

Nowadays, new deep learning models are mainly developed based on *ad hoc* engineering techniques. In contrast, our study showed that brain-inspired structures can also help the design of powerful deep learning models.

References

[1] Mark Bear, Barry Connors, and Michael A Paradiso. *Neuroscience: Exploring the brain*. Jones & Bartlett Learning, LLC, 2020.

[2] Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. Librimix: An open-source dataset for generalizable speech separation. *arXiv preprint arXiv:2005.11262*, 2020.

[3] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

[4] D. Ditter and T. Gerkmann. A multi-phase gammatone filterbank for speech separation via tasnet. In *ICASSP*, pages 36–40, 2020.

[5] B. Fernandez, A.G. Parlos, and W.K. Tsai. Nonlinear dynamic system identification using artificial neural networks (ANNs). In *1990 IJCNN International Joint Conference on Neural Networks*, pages 133–141 vol.2, 1990.

[6] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *ICASSP*, pages 31–35, 2016.

[7] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[8] Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R Hershey. Single-channel multi-speaker separation using deep clustering. *arXiv preprint arXiv:1607.02173*, 2016.

[9] Morten Kolbæk, Dong Yu, Zheng-Hua Tan, and Jesper Jensen. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *TASLP*, 25(10):1901–1913, 2017.

[10] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616, 2009.

[11] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *CVPR*, June 2015.

[12] Ming Liang, Xiaolin Hu, and Bo Zhang. Convolutional neural networks with intra-layer recurrent connections for scene labeling. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[13] Qianli Liao and Tomaso Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016.

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017.

[15] Yi Luo, Zhuo Chen, and Takuya Yoshioka. Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation. In *ICASSP*, pages 46–50, 2020.

[16] Yi Luo and Nima Mesgarani. Tasnet: time-domain audio separation network for real-time, single-channel speech separation. In *ICASSP*, pages 696–700, 2018.

[17] Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *TASLP*, 27(8):1256–1266, 2019.

[18] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, pages 483–499, 2016.

[19] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *ICASSP*, pages 5206–5210, 2015.

[20] Manuel Pariente, Samuele Cornell, Joris Cosentino, Sunit Sivasankaran, Efthymios Tzinis, Jens Heitkaemper, Michel Olvera, Fabian-Robert Stöter, Mathieu Hu, Juan M. Martín-Doñas, David Ditter, Ariel Frank, Antoine Deleforge, and Emmanuel Vincent. Asteroid: the PyTorch-based audio source separation toolkit for researchers. In *Interspeech*, 2020.

[21] G.V. Puskorius and L.A. Feldkamp. Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297, 1994.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015.

[23] BS Series. Algorithms to measure audio programme loudness and true-peak audio level. 2011.

[24] Ziqiang Shi, Huibin Lin, Liu Liu, Rujie Liu, Jiqing Han, and Anyan Shi. Furcanext: End-to-end monaural speech separation with dynamic gated dilated temporal convolutional networks. *arXiv preprint arXiv:1902.04891*, 2019.

[25] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. Two-step sound source separation: Training on learned latent targets. In *ICASSP*, pages 31–35, 2020.

[26] Efthymios Tzinis, Zhepei Wang, and Paris Smaragdis. Sudo rm-rf: Efficient networks for universal audio source separation. In *MLSP*, pages 1–6, 2020.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[28] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *TASLP*, 14(4):1462–1469, 2006.

[29] Zhong-Qiu Wang, Jonathan Le Roux, and John R Hershey. Alternative objective functions for deep clustering. In *ICASSP*, pages 686–690, 2018.

[30] Gordon Wichern, Joe Antognini, Michael Flynn, Licheng Richard Zhu, Emmett McQuinn, Dwight Crow, Ethan Manilow, and Jonathan Le Roux. Wham!: Extending speech separation to noisy environments. *arXiv preprint arXiv:1907.01160*, 2019.

[31] Yucheng Zhao, Chong Luo, Zheng-Jun Zha, and Wenjun Zeng. Multi-scale group transformer for long sequence modeling in speech separation. In *IJCAI*, pages 3251–3257, 2020.

[32] Yisu Zhou, Xiaolin Hu, and Bo Zhang. Interlinked convolutional neural networks for face parsing. In *ISNN*, pages 222–231, 2015.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **[TODO]**
   (b) Did you describe the limitations of your work? **[TODO]**
   (c) Did you discuss any potential negative societal impacts of your work? **[TODO]**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[TODO]**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? **[TODO]**
   (b) Did you include complete proofs of all theoretical results? **[TODO]**
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[TODO]**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**

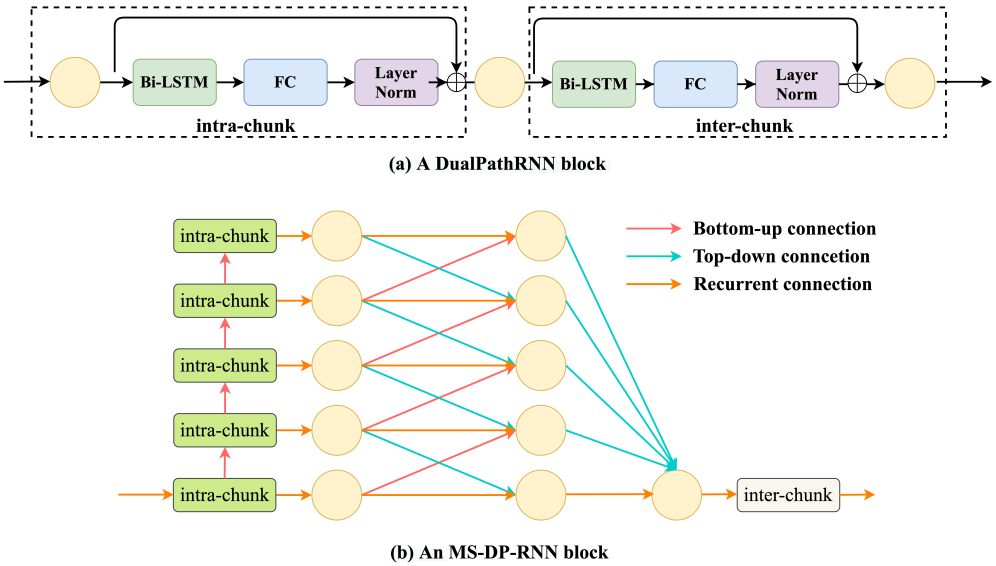

Figure 4: Structures of a DualPathRNN block (a) and an MS-DP-RNN block (b).

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? **[TODO]**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[TODO]**

A Appendix

A.1 Transfer to DualPathRNN

We were interested in whether the MSF method in FRCNN-VI is transferable to other architectures. We chose DualPathRNN for our experiment because it performed the best among existing models in our previous experiments.

The DualPathRNN has multiple blocks with untied weights. In each block, there is an RNN (called *intra-trunk RNN*) for processing multiple audio segments independently, and another RNN (called *inter-trunk RNN*) for aggregating the outputs of the intra-trunk RNN with all audio segments as inputs. The architecture is shown in Figure 4a.

We make S copies of the intra-trunk RNN which share weights and let them process the input in different scales sequentially, and then use the MSF method in FRCNN-VI to fuse the outputs of

these intra-trunk RNNs. See Figure 4b. The inter-trunk RNN is not altered. These constitute a new block. To save the number of parameters, we use tied weights in different blocks. We call this model *MS-DP-RNN*.

In this experiment, we set $S = 5$ again. As mentioned in the main text, training the DualPathRNN was time-consuming. The training time depended on the length of the input. The best result was achieved by setting the kernel size and stride in the encoder and decoder to 2 and 1, respectively. This is the configuration used in our previous experiments for comparison. The original paper of DualPathRNN [15] also reported the result with kernel size 18 and stride 8 on WSJ0-2Mix, which was much faster. We used this configuration in this experiment.

Table 5 show the result. The MS-DP-RNN obtained higher SI-SNRi values than DualPathRNN on the WSJ0-2Mix and Libri2Mix datasets. It indicates that the MSF method in FRCNN-VI can improve the performance of other models.

Model	WSJ0-2Mix	Libri2Mix	Parameters
DualPathRNN	15.9	13.88*	3.69M
MS-DP-RNN	**17.5**	**14.48**	2.90M

Table 5: The performance of DualPathRNN and MS-DP-RNN on the WSJ0-2Mix and Libri2mix test sets. The 2nd and 3rd columns are SI-SNRi values. '*' indicates that the SI-SNRi value was obtained by using the asteroid toolkit [20].

A.2 Sample Results

To give readers an impression of the separation results of different models, we randomly selected some sample audio files from the Libri2Mix [2] datasets and put in the *sample results* folder. The models presented include DPCL [6], uPIT [9], Conv-TasNet [17], SudoRMRF 1.0x [26], DualPathRNN [15] and FRCNN-VI-16. All separated voices were saved in a folder named by the model name. Since all mixtures of speeches were from two speakers, each folder has two subfolders, s1 and s2. We have created an offline web page for these samples *index.html*. Please open it with your explorer. In most examples we found that separated speeches by FRCNN-VI-16 sounded clearer than those by other methods.