Speech Separation Using an Asynchronous Fully Recurrent Convolutional Neural Network

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent advances in the design of neural network architectures, in particular those specialized in modeling sequences, have provided significant improvements in speech separation performance. In this work, we propose to use a bio-inspired architecture called Fully Recurrent Convolutional 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*. In contrast to the traditional approach updating stages in parallel, we propose to first update the stages one by one in the bottom-up direction, then fuse information from adjacent stages simultaneously and finally fuse information from all stages to the bottom stage together. Experiments showed that this asynchronous updating scheme achieved significantly better results with much fewer parameters than the traditional synchronous updating scheme on speech separation. In addition, the proposed model achieved competitive or better results with high efficiency as compared to other state-of-the-art approaches on two benchmark 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 [15, 16, 14], 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, called *multi-scale fusion (MSF)* methods, such as in FurcaNeXt [29] or Sudo-RM-RF [24], and provide impressive results on the speech separation task.

Evidence from observations of sensory systems of mammals show them to utilize MSF 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 reflects a basic structural fact in the brain and does not show the dynamics of the sensory system. Specifically, given an external stimulus, it is unclear the information through which connections arrives at a stage (or a specific neuron) earlier than the information through

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

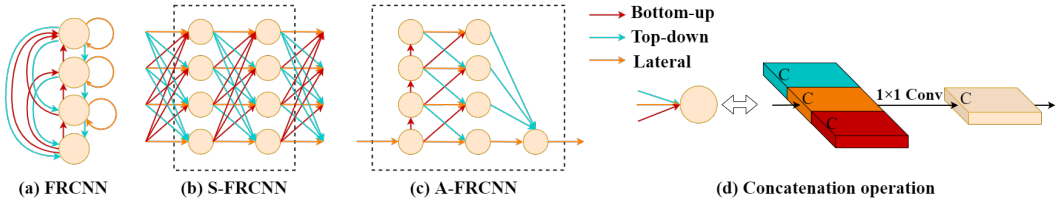

Figure 1: The structure of FRCNN and typical updating schemes. The number of stages $S = 4$. (a) The structure of the FRCNN. Every node denotes a stage. (b) Synchronous updating scheme in one block [13]. (c) The proposed asynchronous updating scheme in one block. The dashed box in each subfigure indicates the basic building block for constructing a complete RNN (see Figure 4). (e) Multi-scale information fusion for an example stage receiving three types of inputs.

other connections. In fact, given a stimulus, the neurons along a sensory hierarchy do not fire simultaneously like shown in Figure 1b; 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 [11] named *recurrent convolutional neural network (RCNN)* that has lateral recurrent connections only. The model with the synchronous updating scheme (Figure 1b) is called the synchronous FRCNN or S-FRCNN, which was studied in a previous work [13].

We propose an asynchronous FRCNN (Figure 1c), denoted by A-FRCNN, in which the information first passes through stages one by one in the bottom-up direction, then fuses between adjacent stages in parallel, and finally fuses together with skip connections to the bottom stage. In the S-FRCNN, the information transmission from the bottom stage to any upper stage then back to the bottom stage is too fast: one step upward and one step downward (Figure 1b). In contrast, in the A-FRCNN, the information starting from the bottom stage goes through more processing steps before it goes back to the bottom stage, which is advantageous for comprehensive MSF. Increasing the depth of a model is one of the keys for the success of deep learning.

In experiments, the A-FRCNN achieved competitive results with the state-of-the-art (SOTA) method DualPathRNN [14] on the WHAM! dataset, and even better results on the Libri2Mix dataset. Moreover, A-FRCNN 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 [16]). 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 [15]. Conv-TasNet [16] has been proposed to solve this problem, replacing RNN with Temporal Convolutional Networks, much faster to train. In addition, DualPathRNN [14] uses a simple module in place of the encoder, and performs on those shorter sequences parallel intra-chunk and global inter-chunk RNN processing, for a large gain in performance.

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 MSF 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. For instance, Sudo-RM-RF [24] uses a U-Net structure [20], obtaining good results on smaller-sized models. FurcaNeXt [29], a variant of Conv-TasNet [16], uses multiple branch learning methods to improve the performance of speech separation. In addition, MSGT-TasNet [30] uses Transformer [25] to capture features of different scales for speech separation.

2.2 Lateral and Top-Down Recurrent Connections

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] [19], and in 2015 they were introduced into the CNN, resulting the RCNN [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 was proposed in 2016 [13]. If a hierarchical model has recurrent connections, the neurons can be updated in different orders. In [13], only the conventional synchronous updating scheme was presented. However, no evidence has shown that this is how the neural system works, or it outperforms asynchronous updating schemes on engineering tasks. In fact, in [12], it was shown that an asynchronous updating scheme for RCNN outperformed the synchronous updating scheme on an image segmentation task. We here propose a novel asynchronous scheme for the FRCNN that achieved better results with fewer parameters than the synchronous scheme on speech separation.

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 $\mathbf{x} \in R^{1\times T}$:

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

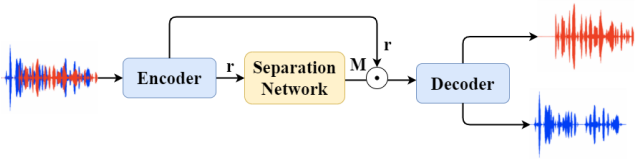

Figure 2: The overall pipeline for speech separation.

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 [16], 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 as the separation network. It can be represented by a graph with nodes denoting stages and edges denoting connections. Figure 1a shows an example with $S = 4$ stages. In biological terms, every node corresponds to 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. 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 bottom-up and top-down connections can be realized by downsampling (e.g., max pooling or convolution with stride>1) and upsampling (e.g., interpolation or transposed convolution) operations, respectively. The lateral connections can be realized by standard convolution.

3.2.2 Updating Schemes in the Micro-level

To run a complex recurrent neural network (RNN) one needs to first determine the updating order of the neurons. This order determines the RNN *unfolding* or *unrolling* scheme. A commonly used approach is to update all neurons simultaneously. In the case of FRCNN as shown in Figure 1a, it corresponds to updating all stages synchronously. This scheme is depicted in Figure 1b [13], and denoted by S-FRCNN. 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. In the present work, we propose an efficient updating scheme A-FRCNN, as shown in Figure 1c.

In the proposed A-FRCNN, we first sequentially update the stages in the bottom-up direction, then update them simultaneously by fusing information from adjacent stages, and finally, fuse information from all stages to the bottom stage. This entire process is repeated several times (Sec. 3.2.4). The two types of fusions can be viewed as local and global fusions, respectively. Considering their biological counterparts, the connections between the same two stages use the same operation and parameters (if any) since they represent the same operation at different time, e.g., the vertical upward connection and the oblique upward connection between stage 3 and stage 4.

The A-FRCNN is adapted from the S-FRCNN in a step-by-step manner.

1. Inspired by the structure of the U-Net [20], we design a bottleneck structure at the bottom stage as shown in Figure 3a. All upper stages exchange information between different blocks through the bottom stage. This design increases the number of steps including down-sampling and up-sampling operations for processing the input coming at the highest resolution. This block is denoted by Control 1.

2. The block Control 1 has too many connections which make the model inefficient in both parameters and computation. The bottom-up skip-connections and top-down skip-connections are symmetric, and may be redundant. Therefore we remove the bottom-up skip-connections, which results in the block shown in Figure 3b, denoted by Control 2.

3. It is too fast to fuse the information across non-adjacent stages through top-down skip-connections in the block Control 2. One possible way to represent an increasing firing delay from widely separated units would be to fuse the information across adjacent stages first, then across non-adjacent stages. This change increase the shortest path from high stages to the bottom stage. In addition, to save parameters and computation, we only keep the top-down skip-connections to the bottom stage and removed other top-down skip-connections. We also remove the vertical downward connections because the top-down stage-by-stage fusion has already been performed through the oblique downward connections. This is made possible by the delayed global fusion; otherwise, the stages would become disconnected after removing the vertical downward connections. We then obtain the A-FRCNN (Figure 1c).

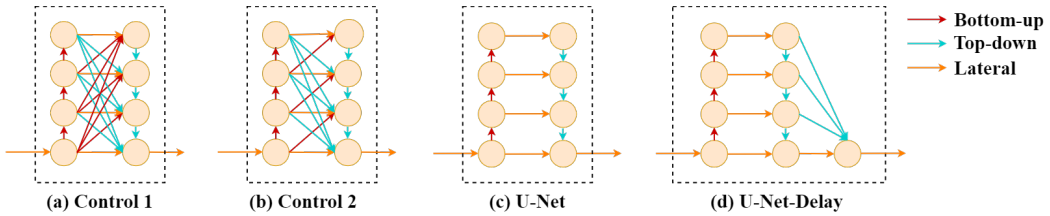

Figure 3: Various asynchronous updating schemes in the micro-level.

Note that the sequential fusion method in the third step is more biologically plausible than the synchronous fusion method since the connections between non-adjacent stages are longer than those between adjacent stages, while signal transmission through connections takes time.

The proposed A-FRCNN block is closely related to the U-Net [20] (Figure 3c). To show the advantage of delayed global fusion, we add top-down skip-connections to the U-Net block and obtain a new block, denoted by U-Net-Delay (Figure 3d). We compared the performances of these blocks in experiments.

3.2.3 Multi-scale Information Fusion inside Blocks

The blocks depicted in Figures 1 and 3 are RNN blocks, and 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 1d illustrates this process. This process performs multi-scale information fusion.

3.2.4 Unfolding Methods in the Macro-level

Figures 1 and 3 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 time steps". At the macro-level, the FRCNN can be 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 4a). This method is formulated as follows:

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

where $f(\cdot)$ denotes a block shown in Figures 1 and 3, $R(t)$ 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 4b). Formally,

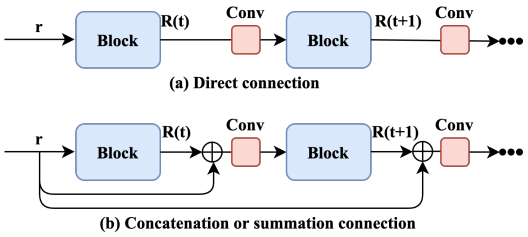

Figure 4: Macro-level unfolding schemes of the FRCNN. Every blue box corresponds to a dashed box in Figures 1 and 3. The pink boxes in a model denote 1×1 convolutions with shared weights.

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

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., A-FRCNN and the blocks shown in Figure 3, we directly use the unfolding methods as described above. For the multi-input-multi-output block, i.e., S-FRCNN, we apply these unfolding methods for each input-output pair corresponding to the same stage. It should be noted that Figure 1b only illustrate the intermediate blocks of S-FRCNN unfolding scheme. 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.

3.3 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) [26]. 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 [17]. Random extracts were selected for different speakers and mixed with uniformly sampled Loudness Units relative to Full Scale (LUFS) [21] between -25 and -33 dB. Random noise samples were 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 used train-100 as the training set unless otherwise specified. We used the same test set to compare different methods.

WHAM! [28]. This dataset was constructed by combining the speech signals of different speakers in the Wall Street Journal corpus, similar to a simple dataset WSJ0-2Mix [6]. Since the WSJ0-2Mix dataset is free of noise, it does not match the real scene speech separation situation. In contrast, WHAM! added noise to WSJ0-2Mix, which was recorded in scenes such as cafes, 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 were 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, the number of channels C in every stage was set to 512 and the SC method was used for unfolding in the macro-level. Whenever $C \neq N$, a 1×1 convolutional layer was added between the encoder and the separation network.

We designed two methods to realize the connections shown in Figures 1 and 3.

- Method A: The bottom-up and top-down connections were realized by convolution (kernel size 5 and stride 2) and the PixelShuffle technique [22] (kernel size 5), respectively. The PixelShuffle technique was shown to be better than other upsampling techniques for image super-resolution reconstruction. The lateral connections were realized by 1×1 convolution.
- Method B: The bottom-up connections were realized by the convolution operation with kernel size 5 and appropriate strides. 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. The

Table 1: Comparison of different FRCNN updating schemes on the Libri2Mix test set. The values are presented as "Method A / Method B". Each model was unfolded for 8 time steps.

Model	SI-SNRi	SDRi	Params (M)	Time (s)
S-FRCNN	$12.60_{\pm 0.05}$ / $12.08_{\pm 0.05}$	$13.01_{\pm 0.10}$ / $12.49_{\pm 0.05}$	9.75/ 9.61	1.35 / 1.12
Control 1	$15.00_{\pm 0.06}$ / $14.55_{\pm 0.04}$	$15.47_{\pm 0.06}$ / $15.00_{\pm 0.03}$	8.27 / 8.03	1.41 / 0.96
Control 2	$15.06_{\pm 0.01}$ / $14.58_{\pm 0.04}$	$15.44_{\pm 0.02}$ / $15.03_{\pm 0.04}$	6.57 / 6.43	1.16 / 0.81
A-FRCNN (ours)	$\mathbf{15.24}_{\pm 0.04}$ / $\mathbf{15.47}_{\pm 0.04}$	$\mathbf{15.70}_{\pm 0.04}$ / $\mathbf{15.93}_{\pm 0.04}$	6.21 / 6.13	1.07 / 0.72
S-FRCNN (light)	- / $11.80_{\pm 0.03}$	- / $12.15_{\pm 0.02}$	- / 6.38	- / 0.97

top-down connections were realized by interpolation. The lateral connections were realized by simply copying the feature maps.

All convolutions were depthwise separable convolutions. In Method A all connections had trainable parameters, resembling plastic synapses in biological systems. In Method B, only the bottom-up connections had parameters, and it is therefore less biologically plausible. However, Method B is more parameter efficient and computing efficient.

We trained all models for 200 epochs on 3-second utterances for Libri2Mix and 4-second utterances for wham! 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.

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 models can be found in the Supplementary Materials. This project is licensed under the MIT License.

5 Results

We used the scale-invariant signal-to-noise ratio improvement (SI-SNRi) and signal-to-distortion ratio improvement (SDRi) [26] as the evaluation metrics to measure the speech separation accuracies of models. In Secs. 5.1 and 5.2, we report the mean$\pm$std of these metrics over 5 models trained from different random seeds. We report inference time on CPU, indicated by "Time" in tables throughout the paper. It was calculated by processing a four seconds audio on CPU then averaged over 1000 trials.

5.1 Comparison of Micro-level Updating Schemes

We compared the performances of the micro-level updating schemes on Libri2Mix. The results are presented in Table 1. By using either Method A or Method B for implementing the connections (Sec. 4.2), the proposed A-FRCNN scheme performed better than the S-FRCNN scheme and the two control schemes. By comparing the results of S-FRCNN and Control 1, we see that the bottleneck design brought a large improvement on the SI-SNRi and SDRi metrics. Control 1 and Control 2 achieved similar SI-SNRi and SDRi values indicating that the bottom-up skip-connections are indeed redundant. The delayed global fusion further improved the results of Control 2. From S-FRCNN to Controls 1 and 2 and A-FRCNN, the number of connections decreased; as a result, the amount of parameters and inference time also decreased.

For implementing the connections in the four models in Table 1, Method B obtained similar or even better results with fewer parameters and less inference time than Method A. Therefore we adopted Method B in later experiments. All results reported in what follows were obtained with Method B.

The S-FRCNN had much more parameters than the A-FRCNN. We then reduced C to 412 and obtained a new model, S-FRCNN (light), which had similar number of parameters to the A-FRCNN. Its results were even worse than the original S-FRCNN (Table 1), suggesting that the poor performance of the S-FRCNN was not due to overfitting caused by large amount of parameters.

The U-Net structure did not yield good results (Table 2). With delayed global fusion, the U-Net-Delay block yielded better results. For a fair comparison with the A-FRCNN, we changed the number of

Table 2: Comparison with the U-Net and U-Net-Delay on the Libri2Mix test set. Each model was unfolded for 8 time steps.

Model	SI-SNRi	SDRi	Params (M)	Time (s)
U-Net	$11.38_{\pm 0.01}$	$11.81_{\pm 0.01}$	4.02	0.39
U-Net-Delay	$13.16_{\pm 0.05}$	$13.61_{\pm 0.05}$	5.06	0.60
U-Net (large)	$11.47_{\pm 0.01}$	$11.91_{\pm 0.01}$	6.13	0.69
U-Net-Delay (large)	$12.06_{\pm 0.01}$	$12.52_{\pm 0.01}$	6.36	0.88
A-FRCNN (ours)	$\mathbf{15.47}_{\pm 0.04}$	$\mathbf{15.93}_{\pm 0.04}$	6.13	0.72

Table 3: Performance of different models on two datasets. 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 SDRi values marked with '*' indicate that they were not reported in the original papers but obtained by using the Asteroid toolkit [18]. The toolkit is licensed under the MIT licence.

Model	Libri2Mix		WHAM!		Params (M)	Time (s)
	SI-SNRi	SDRi	SI-SNRi	SDRi		
DPCL++	5.94*	6.61*	-	-	13.6	0.37
uPIT-BLSTM-ST	7.55*	8.22*	-	-	92.7	0.77
Chimera++	6.29*	6.99*	9.95	-	32.9	0.66
BLSTM-TasNet	7.91*	8.68*	9.84	-	23.6	2.90
Conv-TasNet	12.22	12.68	12.72	-	5.6	0.39
Conv-TasNet {train-360}	**14.7**	**15.1**	12.72	-	5.6	0.39
Two-Step TDCN	12.04*	12.45*	-	-	8.63	1.85
MSGT-TasNet (Light)	-	-	12.30	-	37.36	3.37
MSGT-TasNet (Dense)	-	-	13.10	-	66.79	5.56
SudoRM-RF 0.25x	10.84*	11.25*	10.38*	10.80*	0.79	0.24
SudoRM-RF 0.5x	12.20*	12.61*	11.80*	12.21*	1.42	0.33
SudoRM-RF 1.0x	13.54*	13.95*	12.89*	13.31*	2.66	0.53
SudoRM-RF 2.5x	13.98*	14.40*	13.69*	**14.10***	6.42	1.15
DualPathRNN	14.10*	14.57*	**13.71***	14.09*	2.69	4.66
A-FRCNN-4	$12.61_{\pm 0.01}$	$13.06_{\pm 0.02}$	$11.95_{\pm 0.02}$	$12.30_{\pm 0.02}$	6.13	0.33
A-FRCNN-8	$15.47_{\pm 0.04}$	$15.93_{\pm 0.04}$	$13.37_{\pm 0.02}$	$13.75_{\pm 0.02}$	6.13	0.72
A-FRCNN-16	$\mathbf{16.65}_{\pm 0.03}$	$\mathbf{17.15}_{\pm 0.03}$	$\mathbf{14.53}_{\pm 0.02}$	$\mathbf{14.84}_{\pm 0.03}$	6.13	1.51
A-FRCNN-16 (light)	$15.27_{\pm 0.02}$	$15.73_{\pm 0.02}$	$13.30_{\pm 0.03}$	$13.72_{\pm 0.03}$	2.47	1.21

channels C from 512 to 684 and 580 in U-Net and U-Net-Delay blocks, respectively, and obtained the U-Net (large) and U-Net-Delay (large) blocks. The two blocks had similar amount of parameters to the A-FRCNN. With more parameters, the U-Net-Delay (large) tended to overfit the training data and yielded worse test results than the original U-Net-Delay. Even in this case, U-Net-Delay (large) yielded better results than the U-Net (large), verifying the effectiveness of the delayed global fusion. Nevertheless, they both yielded much lower SI-SNRi and SDRi values than the A-FRCNN.

5.2 Comparison with Existing Models

We compared A-FRCNN with some popular models for speech separation. Some models work in the time-frequency domain: DPCL++ [6], uPIT-BLSTM-ST [9] and Chimera++ [27]. Some models work in the time-domain: BLSTM-TasNet [15], Conv-TasNet [16], Two-Step TDCN [23], MSGT-TasNet [30], SudoRM-RF [24] and DualPathRNN [14]. 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 A-FRCNN by unfolding for 4, 8, 16 times in the macro-level. In addition, we tested a model, named A-FRCNN-16 (Light), on Libri2Mix by setting $C = 324$ to make the number of parameters similar to that of DualPathRNN. The comparison results on the two benchmark datasets are presented in Table 3.

On the Libri2Mix dataset, we had the following observations. First, Conv-TasNet trained on a large training set train-360 achieved the best separation results measured by SI-SNRi and SDRi 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, A-FRCNN achieved better results at the cost of increasing inference time. Fourth, both A-FRCNN-8 and A-FRCNN-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, A-FRCNN-16 (Light) outperformed DualPathRNN.

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

On the WHAM! dataset, again, DualPathRNN achieved the best results among existing models. In comparison, A-FRCNN-16 obtained better results on WHAM!.

We noted that DualPathRNN was the slowest method. On Libri2Mix, it took 6 days on 8 GPUs for training while A-FRCNN-16 took 2 days and A-FRCNN-16 (Light) took 0.8 days. Morever, its inference on CPU was about 6$\times$ slower than A-FRCNN-8 and 3$\times$ slower than A-FRCNN-16 (Table 3). This limits its applications.

5.3 Ablation Study

The experiments were on the Libri2Mix dataset. We studied the influence of the number of stages S (Figure 1c) by fixing the unfolding method to SC. The results was the best with $S = 5$ (Table 4). We then compared the results with different unfolding methods (Figure 4) by fixing $S = 5$ and found that the SC method was the best (Table 5). In the two experiments the A-FRCNN was unfolded for 8 time steps. We therefore used $S = 5$ and the SC method in all other experiments. Only the results with this setting are average results over 5 different runs in Tables 4 and 5; and we did not train models with other settings for multiple times considering the small standard deviations in previous tables.

Table 4: Test results of the A-FRCNN with different number of stages S.

S	SI-SNRi	SDRi	Params
3	13.77	14.52	4.02M
4	14.38	14.85	5.08M
5	**15.47**	**15.93**	6.13M
6	14.39	14.88	7.19M

Table 5: Test results of A-FRCNN with different unfolding methods in the macro-level.

Method	SI-SNRi	SDRi	Params	Time (s)
DC	12.78	11.38	6.13M	0.70s
CC	14.26	14.83	6.65M	0.96s
SC	**15.47**	**15.93**	6.13M	0.72s

5.4 Transfer to DualPathRNN

The DualPathRNN uses LSTM-based intro-chunks and inter-chunks repeatedly to fuse information. We used our proposed MSF method in A-FRCNN to fuse 5 multi-scale intro-chunks and sent the result to the inter-chunk in one block, and repeated multiple blocks with tied weights. The new model obtained better results than the DualPathRNN, further demonstrating the merit of the proposed MSF method. See Supplementary Materials for details.

6 Conclusion

We investigated an asynchronous updating scheme for a bio-inspired FRCNN architecture for solving the speech separation task. The resulted model A-FRCNN achieved competitive or even better results with the SOTA model DualPathRNN on two benchmark datasets. It was 3 times faster than DualPathRNN in both training and inference. Moreover, the multi-scale information fusion method of the A-FRCNN can boost the performance of the DualPathRNN.

This work has some limitations. First, we only considered two speakers in the mixed speech and did not consider the more natural setting where the number of speakers is unknown. Second, the proposed method was only applied to speech separation, and its performance on other applications is unknown.

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*. The MIT Press, 2001.

[4] David Ditter and Timo Gerkmann. A multi-phase gammatone filterbank for speech separation via tasnet. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 36–40, 2020.

[5] Benito Fernandez, G. Parlos, Alexander, and K. Tsai, Wei. Nonlinear dynamic system identification using artificial neural networks (ANNs). In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 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 *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (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. In *Proceedings of the International Speech Communication Association (Interspeech)*, pages 545–549, 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. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 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 (ICML)*, pages 609–616, 2009.

[11] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[12] Ming Liang, Xiaolin Hu, and Bo Zhang. Convolutional neural networks with intra-layer recurrent connections for scene labeling. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 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] Yi Luo, Zhuo Chen, and Takuya Yoshioka. Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 46–50, 2020.

[15] Yi Luo and Nima Mesgarani. Tasnet: time-domain audio separation network for real-time, single-channel speech separation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 696–700, 2018.

[16] Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1256–1266, 2019.

[17] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[18] 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 *Proceedings of the International Speech Communication Association (Interspeech)*, 2020.

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

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

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

[22] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[23] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. Two-step sound source separation: Training on learned latent targets. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35, 2020.

[24] Efthymios Tzinis, Zhepei Wang, and Paris Smaragdis. Sudo rm-rf: Efficient networks for universal audio source separation. In *Proceedings of the Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2020.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[26] Emmanuel Vincent, Rémi Gribonval, and Cédric Fevotte. Performance measurement in blind audio source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.

[27] Zhong-Qiu Wang, Jonathan Le Roux, and John R Hershey. Alternative objective functions for deep clustering. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 686–690, 2018.

[28] 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. In *Proceedings of the International Speech Communication Association (Interspeech)*, 2019.

[29] Liwen Zhang, Ziqiang Shi, Jiqing Han, Anyan Shi, and Ding Ma. Furcanext: End-to-end monaural speech separation with dynamic gated dilated temporal convolutional networks. In *Proceedings of the International Conference on Multimedia Modeling (MMM)*, pages 653–665. Springer, 2020.

[30] Yucheng Zhao, Chong Luo, Zheng-Jun Zha, and Wenjun Zeng. Multi-scale group transformer for long sequence modeling in speech separation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3251–3257, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
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)? [Yes] We provide the code for our main experiments, the data generation code, and instructions for running it in the Supplemental Materials.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The objective function for training was described in Sec. 3.3; the data split is specified in Sec. 4.1; the hyperparameters are described in 4.2; how they were chosen is specified in Sec. 5.3

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the mean and standard deviation of the results of our proposed model and its variants over 5 training runs.

(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)? [Yes] The specific hardware configurations used in this work were described in Sec. 4.2

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? [Yes]

(b) Did you mention the license of the assets? [Yes] MIT License

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] The data we used is purposely designed for public usage.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

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? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]