INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Multi-path RNN for hierarchical modeling of long sequential data and its application to speaker stream separation

Keisuke Kinoshita[1], *Thilo von Neumann*[2], *Marc Delcroix*[1], *Tomohiro Nakatani*[1], *Reinhold Haeb-Umbach*[2]

[1]NTT Corporation, Japan
[2]Paderborn University, Germany

Abstract

Recently, the source separation performance was greatly improved by time-domain audio source separation based on dual-path recurrent neural network (DPRNN). DPRNN is a simple but effective model for a long sequential data. While DPRNN is quite efficient in modeling a sequential data of the length of an utterance, i.e., about 5 to 10 second data, it is harder to apply it to longer sequences such as whole conversations consisting of multiple utterances. It is simply because, in such a case, the number of time steps consumed by its internal module called inter-chunk RNN becomes extremely large. To mitigate this problem, this paper proposes a multi-path RNN (MPRNN), a generalized version of DPRNN, that models the input data in a hierarchical manner. In the MPRNN framework, the input data is represented at several (≥ 3) time-resolutions, each of which is modeled by a specific RNN sub-module. For example, the RNN sub-module that deals with the finest resolution may model temporal relationship only within a phoneme, while the RNN sub-module handling the most coarse resolution may capture only the relationship between utterances such as speaker information. We perform experiments using simulated dialogue-like mixtures and show that MPRNN has greater model capacity, and it outperforms the current state-of-the-art DPRNN framework especially in online processing scenarios.
Index Terms: speech separation, neural networks

1. Introduction

Automatic meeting analysis is one of the essential technologies required for realizing, e.g. communication agents that can follow and respond to our conversations. Source separation is one of the important sub-tasks for the meeting analysis.

A considerable number of source separation techniques have been proposed, based on emerging deep learning technologies, such as Deep Clustering (DC) [1], and Permutation Invariant Training (PIT) [2, 3]. DC and its related technologies [4, 5] can be viewed as two-stage algorithms. They first encode an input signal into an embedding space based on a pretrained neural network (NN), and obtain embedding vector(s) for each time frame [5] or time-frequency bin [1, 4]. Then, to obtain source separation masks or separated signals, these embedding vectors are clustered by means of e.g. K-means clustering, given the correct number of speakers. Since the clustering step is usually used only in the inference stage, there is a mismatch in the processing flow between the test and training stage. Thus, it is difficult for this type of approaches to optimize the total system for source separation metrics. In contrast, PIT and its related approaches [6, 7] are single-stage algorithms, which let NNs directly estimate separated signals without the clustering step. Such approaches make it possible not only to optimize the entire system for source separation, but also to jointly optimize the source separation module with other tasks, such as a source counting task [8–10] and an ASR task [11–13]. Motivated by such an advantage of the single-stage algorithms, this paper proposes an extension for that approach.

The PIT-based approach has greatly advanced recently and achieved the state-of-the-art performance by incorporating (1) an idea of time-domain audio source separation network (TasNet) framework [6] and (2) an advanced network architecture called dual-path recurrent neural network (DPRNN) [7]. The DPRNN framework utilizes RNNs to model a long sequential input in a very simple way. It first splits the input sequence into short chunks and interleave two RNNs, an intra-chunk RNN and an inter-chunk RNN, for local and global modeling, respectively. In a DPRNN block, the intra-chunk RNN first processes the local chunks independently, and then the inter-chunk RNN aggregates the information from all the chunks to perform utterance-level processing. While this algorithm is found to be more efficient than e.g., dilated temporal convolutional NNs in modeling sequential data whose length is about an utterance, i.e., about 5 to 10 second data, it may pose a problem for modeling even longer sequential data. In such a case, the number of time steps consumed by the inter-chunk RNN becomes extremely large, leading to a performance degradation.

Since the modeling of long sequential data is essential when separating long meeting-like data (i.e., speaker *stream* separation), this paper proposes to generalize the DPRNN framework to enable efficient modeling of much longer data. The proposed network architecture, which we call multi-path RNN (MPRNN), models the input sequential data in a hierarchical manner. In other words, in the MPRNN framework, the input data is represented at several (≥ 3) time-resolutions, each of which is then modeled by a specific RNN sub-module. The RNN sub-module dealing with the finest resolution may model temporal relationship within a phoneme, while the RNN sub-module handling the most coarse resolution may capture the relationship between utterances such as speaker information.

2. Proposed approach: Multi-path RNN

The proposed MPRNN is a generalized version of the previously proposed DPRNN [7], which aims to model long temporal context in a sequential input. In this section, we will explain the core steps in the proposed MPRNN framework, clarifying its relationship to DPRNN.

Similarly to DPRNN, MPRNN consists of mainly two stages; segmentation, and a core processing realized using a MPRNN block composed of multiple RNNs. The segmentation stage splits a sequential input into chunks and forms a tensor that stores the chunks in a hierarchical manner. The tensor is then passed to an MPRNN block. The MPRNN block is the core of this framework, and models temporal relationships in the input sequential data, from short-term (e.g., phoneme level) through middle-term (e.g., an utterance level) to long-term (e.g.,

http://dx.doi.org/10.21437/Interspeech.2020-2388

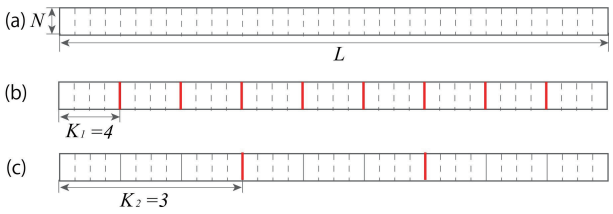

Figure 1: *Results of segmentation module in the MPRNN framework: (a) a shape of the input sequence $\boldsymbol{W}$ to an MPRNN block. (b) Finest-level segmentation with the chunk length of 4 ($K_1 = 4$). (c) Coarse level segmentation based on (b), with the chunk length of 3 ($K_2 = 3$). Red lines correspond to boundaries of the chunks.*

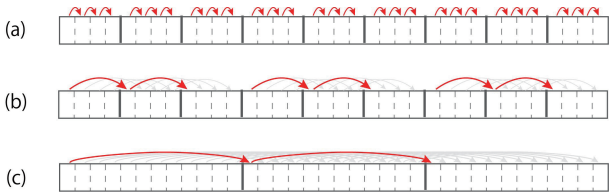

Figure 2: *Behavior of RNN sub-modules in an MPRNN block: (a) RNN modeling the finest temporal relationship, (b) RNN modeling the medium-level temporal relationship, (c) RNN modeling the coarse temporal relationship. For the sake of simplicity, uni-directional processing is used for this figure.*

inter-utterance level) relationships. The MPRNN blocks can be stacked on top of each other to enable deep modeling. An output from the last MPRNN block is transformed back to a sequence data by performing an inverse operation of the segmentation stage and overlap-add.

2.1. Hierarchical segmentation

Let us first denote the input single-channel speech signal in the time domain as $\boldsymbol{w} \in \mathbb{R}^{1\times T}$. Before the segmentation stage, $\boldsymbol{w}$ is converted with an encoder convolutional neural network [6] to $\boldsymbol{W} \in \mathbb{R}^{N\times L}$ where N is the feature dimension and L is the number of time frames (see Fig. 1 (a)). Then, the segmentation stage splits $\boldsymbol{W}$ into hierarchical chunks as depicted in Fig. 1 (b) and (c). Fig. 1 (b) shows the finest-level segmentation applied to the input sequence $\boldsymbol{W}$. Denoting the length of each chunk in this finest-level segmentation as K_1, we obtain $S_1 = L/K_1$ chunks on this segmentation level[1]. Similarly, following the finest segmentation, we apply another segmentation, i.e., more coarse level segmentation on top of the finest segmentation as in Fig. 1 (c). On this segmentation level, a chunk contains several finest-level chunks. The length of each chunk on this segmentation level is K_2 indicating the number of the finest chunks included in a chunk. On this segmentation level, we obtain $S_2 = L/(K_1 \cdot K_2)$ chunks.

Based on this hierarchical segmentation, we can finally form a tensor $T \in \mathbb{R}^{N\times K_1\times K_2\times S_2}$. Conceptually, by repeating the above hierarchical segmentation M times, we can obtain an $(M+2)$-dimensional tensor $T \in \mathbb{R}^{N\times K_1\times K_2\times\cdots\times K_M\times S_M}$. For the sake of simplicity, we assume $M = 2$ case, i.e. $T \in \mathbb{R}^{N\times K_1\times K_2\times S_2}$, for the following explanation.

2.2. Processing in an MPRNN block

The output from the segmentation stage, $\boldsymbol{T}$, is then passed to an MPRNN block. An MPRNN block contains $M+1$ sub-modules each of which models specific temporal relationships in the data by using a (bidirectional) RNN. In order to process the four-dimensional tensor $T \in \mathbb{R}^{N\times K_1\times K_2\times S_2}$, we need 3 sub-modules explained below. The first, second and third RNN sub-modules aim to roughly capture temporal relationships within a phoneme, those within an utterance (e.g., inter-phoneme relationships), and those between utterances (e.g., speaker information), respectively.

[1]We actually divided the input data into overlapping chunks as in the original DPRNN [7], but here for the sake of simplicity, we explain a processing flow of MPRNN by assuming the non-overlapping chunks (i.e., corresponding to $P = K$ case in [7]).

For the RNN sub-module that models the finest temporal relationship, the input tensor $\boldsymbol{T}$ is reshaped to a three-dimensional tensor $\bar{\boldsymbol{T}}_1 \in \mathbb{R}^{N\times K_1\times(K_2\cdot S_2)}$. In $\bar{\boldsymbol{T}}_m$ $(m = 1, 2, 3)$, the first, second and third dimensions correspond to the input feature size, the number of time steps, and number of samples/examples, in terms of RNN [14]. Using Python-like expressions and notations for matrix operation as in [7], the finest-level RNN processes the input data $\bar{\boldsymbol{T}}_1$ as:

$$\bar{\boldsymbol{U}}_1 = [f_1(\bar{\boldsymbol{T}}_1[:,:,i]),\ i = 1,\ldots,K_2\cdot S_2], \tag{1}$$

where $\bar{\boldsymbol{U}}_1 \in \mathbb{R}^{H\times K_1\times(K_2\cdot S_2)}$ is the output of the RNN, H is the dimension of the hidden layer in the RNN, $f_1(\cdot)$ is the mapping function defined by an RNN, and $\bar{\boldsymbol{T}}_1[:,:,i] \in \mathbb{R}^{N\times K_1}$ is the sequence defined by the index i. In other words, the module $f_1(\cdot)$ processes all $K_2 \cdot S_2$ examples independently in parallel. Figure 2 (a) shows how $f_1(\cdot)$ processes the data according to Eq. (1). A linear fully-connected (FC) layer is then applied to transform the feature dimension of $\bar{\boldsymbol{U}}_1$ back to that of $\bar{\boldsymbol{T}}_1$

$$\hat{\boldsymbol{U}}_1 = [\boldsymbol{G}_1\bar{\boldsymbol{U}}_1[:,:,i] + \boldsymbol{m}_1,\ i = 1,\ldots,K_2\cdot S_2] \tag{2}$$

where $\hat{\boldsymbol{U}}_1 \in \mathbb{R}^{N\times K_1\times(K_2\cdot S_2)}$ is the transformed feature, $\boldsymbol{G}_1 \in \mathbb{R}^{N\times H}$ and $\boldsymbol{m}_1 \in \mathbb{R}^{N\times 1}$ are the weight and bias of the FC layer. Finally, the output from this sub-module is formed with a residual connection and layer normalization as:

$$\hat{\boldsymbol{T}}_2 = \bar{\boldsymbol{T}}_1 + LN(\bar{\boldsymbol{U}}_1), \tag{3}$$

where $LN(\cdot)$ represents a layer normalization [7]. $\hat{\boldsymbol{T}}_2$ will be used as an input to the following RNN sub-module. From the explanation of the remaining RNN sub-modules, we omit the processing concerning the FC layer, layer normalization and the residual connection, as it is similar to Eq. (3).

For the RNN sub-module that models a medium-level temporal relationship (see Fig. 2 (b)), the tensor obtained in the previous sub-module, $\hat{\boldsymbol{T}}_2 \in \mathbb{R}^{N\times K_1\times(K_2\cdot S_2)}$, is reshaped to $\bar{\boldsymbol{T}}_2 \in \mathbb{R}^{N\times K_2\times(K_1\cdot S_2)}$. Then, it is processed as:

$$\bar{\boldsymbol{U}}_2 = [f_2(\bar{\boldsymbol{T}}_2[:,:,i]),\ i = 1,\ldots,K_1\cdot S_2], \tag{4}$$

Similarly, for the RNN sub-module that models a long-term temporal relationship (see Fig. 2 (c)), the tensor obtained from the previous RNN sub-module is reshaped to $\bar{\boldsymbol{T}}_3 \in \mathbb{R}^{N\times S_2\times(K_1\cdot K_2)}$. Then, it is processed as:

$$\bar{\boldsymbol{U}}_3 = [f_3(\bar{\boldsymbol{T}}_3[:,:,i]),\ i = 1,\ldots,K_1\cdot K_2], \tag{5}$$

Figure 3 shows an overall processing diagram in an MPRNN block containing $M+1$ RNN sub-modules.

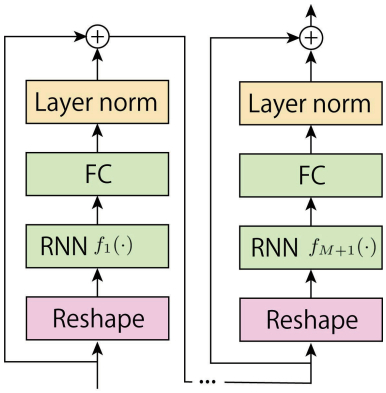

Figure 3: *Processing in an MPRNN block. This block can be stacked to enable deep modeling.*

2.3. The relation to DPRNN

While MPRNN performs the hierarchical segmentation as explained in section 2.1, segmentation in DPRNN is not hierarchical. It performs the segmentation only one time to obtain the segmentation result depicted in Fig. 1 (b). Thus the obtained tensor in DPRNN is the size of $\mathbb{R}^{N \times K_1 \times S_1}$, which is the special case of MPRNN with $M = 1$. Accordingly, DPRNN requires only 2 RNN sub-modules in a DPRNN block which is referred to as inter-chunk and intra-chunk RNNs [7]. On the other hand, MPRNN utilizes M+1 sub-modules.

When modeling a long mixture such as meeting-like data, the number of time steps consumed by the inter-chunk RNN in DPRNN is going to be extremely large, degrading its performance because LSTM is in general not capable of handling a very long sequence [6, 7]. On the other hand, MPRNN has extra RNN sub-modules such as $f_3(\cdot)$ that model only long-term temporal relationship in the data (Fig.2 (c)). By skipping the large number of time samples for modeling such a long-term temporal relationship, MPRNN explicitly circumvent the modeling of extremely large number of time-steps. Therefore, we argue that MPRNN is more appropriate to model very long sequential data containing multiple utterances.

3. Separation framework

To construct separation networks based on MPRNN and DPRNN, we insert the DPRNN and MPRNN modules into the TasNet framework [6]. Specifically, to better understand the behavior of DPRNN and MPRNN, we prepared the following two types of frameworks based on TasNet.

3.1. 2-output framework

The first framework, hereafter referred to as 2-output framework, provides the same outputs as [6], i.e. the models estimate speech signals for all speakers at once. We simply replaced the convolutional separation module in [6] with the DPRNN or MPRNN modules, keeping the structure of the encoder and decoder unchanged.

3.2. 1-output framework

Recently, iterative approaches to source separation that output one speaker at a time [8–10] have received increased interest as they allow to perform separation for an unknown number of speakers or track speakers along long recordings [9]. We believe that the proposed MPRNN scheme would be particularly suited for such iterative schemes as it may allow better tracking a speaker over a long recording.

As a preliminary investigation of MPRNN for iterative separation, we investigate a second separation framework, referred to as 1-output framework. Here, the network provides a single output speech signal, which consists of one of the speaker arbitrarily chosen by the network. Then, the separated signal for the other speaker is obtained by subtracting the output signal from the mixture in the time domain. Although we limit our investigation to 2-speaker cases in the following experiments, such a scheme could be easily extended to iterative source separation schemes to cope with more speakers [8, 10].

4. Experimental procedures

We now evaluate MPRNN in comparison to DPRNN.

4.1. Data

For experiments, we generated data based on WSJ0 [15]. The training data set for this experiment comprises 20000 examples. Each example was generated to simulate 30 second dialogue, such that every 5 second frame contains zero speaker with a probability of 25 %, one speaker with a probability of 50 %, and two speakers with a probability of 25 %, respectively. Development data set comprises 1000 examples, and has similar characteristics to the training data. For evaluation, we generated two types of evaluation data that differ in length. The first evaluation data set contains 3000 examples each of which is 30 second long, while the second evaluation data set consists of 3000 examples each of which is 120 second long. Both evaluation data sets were generated by following the same speaker overlap probability used for the training data.

The evaluation metric used in the experiments are SDR [16]. The sampling frequency was 8 kHz.

4.2. Details of tested DPRNN and MPRNN

4.2.1. Model configuration

For DPRNN, we used the following hyper-parameters. Window size was 16. Segment length K and hop size were set at 100 and 50, respectively. The input feature size N and the dimension of the hidden layer H were 64 and 128. The number of the DPRNN blocks used for this experiment was 5. With these settings, the number of time-steps consumed by the inter-chunk RNNs in DPRNN are 600 for 30 second data, and 2400 for 120 second data, respectively.

For MPRNN, we evaluated the case where the number of hierarchical segmentation was two, i.e. $M = 2$. Segment length K_1 and K_2 were set at 100 and 60, and corresponding hop sizes were set at 50 and 30, respectively. Other parameters were kept the same as the DPRNN setting for a fair comparison. The number of MPRNN blocks was set to 3 to have a total number of parameters similar to the baseline DPRNN model containing 5 DPRNN blocks.

We train all models for 150 epochs with utterance-level permutation invariant training [3] to maximize scale-dependent signal-to-distortion ratio (SD-SDR). The network was trained on a whole utterance of 30 seconds, rather than a part of an utterance. Adam is used as the optimizer, with an initial learning rate of 0.001. For all configurations, we used the model that achieved the best performance on the validation set.

Table 1: *Performance of offline DPRNN and MPRNN for 30 and 120 second data, in the 1-output and 2-output frameworks*

Model		SDR (dB)	
Separator	Framework	30 sec.	120 sec.
DPRNN	2-output	19.61	16.15
	1-output	19.32	15.75
MPRNN	2-output	19.39	15.90
	1-output	**19.72**	**16.26**

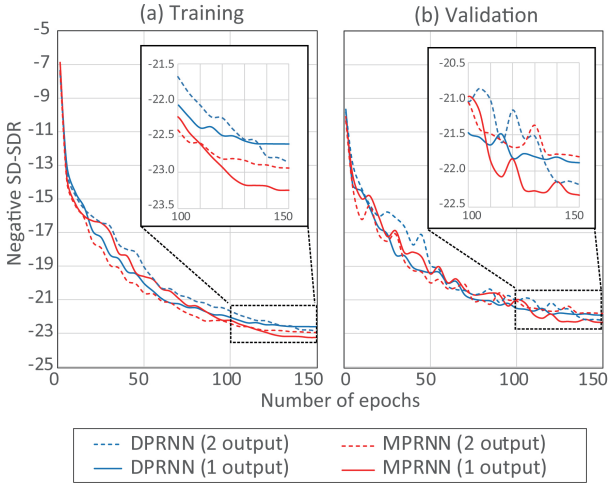

Figure 4: *Loss curves for (a) training and (b) validation data.*

4.2.2. *Latency*

When processing relatively long data, online processing is often needed. To clarify how the algorithm latency affects the performance, we prepared offline and online settings for DPRNN and MPRNN models.

In case of the offline DPRNN, we used BLSTM networks for both inter-chunk and intra-chunk RNNs. For the online DPRNN, we used BLSTM networks for intra-chunk RNNs, and LSTM networks for inter-chunk RNNs.

Similarly, in case of the offline MPRNN, we use BLSTM networks for all RNNs from $f_1(\cdot)$ to $f_3(\cdot)$. For online MPRNN, we use BLSTM for $f_1(\cdot)$ and $f_2(\cdot)$, and LSTM for $f_3(\cdot)$. By doing so, the online DPRNN and MPRNN models have the same number of parameters.

With the aforementioned parameters, the online MPRNN model has an algorithmic delay of about 1.5 seconds, while the online DPRNN model works with the delay of 100 ms. To make a fair comparison with online MPRNN, we investigated increasing the delay for DPRNN by delaying the output, but we found that DPRNN would not train well in such cases. Therefore, in our investigation, we employ the online DPRNN model with the delay of 100 ms as the best performing online DPRNN model.

5. Results

5.1. Offline processing

Table 1 shows the separation results obtained with offline DPRNN and offline MPRNN in case of 2-output and 1-output frameworks for 30 and 120 second data sets. The MPRNN with the 1-output framework slightly outperforms the other models.

To further analyze the performance and capacity of each model, we show loss curves for the training and validation data in Fig. 4. Looking at the loss curves for the training data, we found that the MPRNN models tend to converge to lower SD-SDR values, which may suggest greater model capacity of MPRNNs. Note that, the number of parameters for the offline DPRNN and MPRNN models are 2.17M and 1.95M, respectively. In other words, the greater model capacity of MPRNN does not come from the size of the model, but comes from the inherent structure of the model.

However, the loss curves for the validation data do not always coincide with the training loss curves. Specifically, the superiority of MPRNNs for the training data did not necessarily translate to the improvement for the validation and evaluation data. This mismatch may suggest that MPRNNs overfit to the training data. This overfitting issue will be revisited in our future work, by e.g., increasing the amount of the training data, and improving the model training scheme.

Table 2: *Performance of online DPRNN and MPRNN for 30 and 120 second data, in the 1-output and 2-output frameworks*

Model		SDR (dB)	
Separator	Framework	30 sec.	120 sec.
DPRNN	2-output	15.69	13.22
	1-output	16.73	14.01
MPRNN	2-output	17.70	14.62
	1-output	**17.90**	**14.82**

5.2. Online processing

Table 2 shows the results obtained with the online DPRNN and online MPRNN models. In this case, the MPRNN models outperform the DPRNN models, capitalizing on its RNN submodules that model long-term temporal relationships. However the performance of the DPRNN models should be interpreted carefully since the algorithmic delay between the DPRNN and MPRNN models are different (See section 4.2.2). Comparing the performance difference between the offline and online models, we see that the performance degradation for the online MPRNN models are smaller than with the online DPRNN models, showing the potential of MPRNN for block-online source separation task [9].

Looking at the results with the 120 second data set, we observe performance degradation for both DPRNN and MPRNN when the test data is much longer than the training data. However, the MPRNN still maintains its superiority and achieves high separation performance of $14.82\,\mathrm{dB}$.

6. Conclusions

This paper proposed an efficient sequence model, called MPRNN, and described its application to the source separation task. The proposed MPRNN is a generalization of DPRNN that achieves the state-of-the-art performance. In the MPRNN framework, the input data is represented at several time-resolutions. Then, the data in each resolution is modeled by a specific RNN sub-module. Experimental results suggests (1) MPRNN potentially has greater model capacity, and (2) it outperforms DPRNN especially in online processing scenarios.

7. References

[1] J. Hershey, Z. Chen, J. L. Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 31–35.

[2] D. Yu, M. Kolbæk, Z. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 241–245.

[3] M. Kolbæk, D. Yu, Z. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 10, pp. 1901–1913, Oct 2017.

[4] Y. Luo, Z. Chen, and N. Mesgarani, "Speaker-independent speech separation with deep attractor network," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 26, no. 4, pp. 787–796, 2018. [Online]. Available: https://doi.org/10.1109/TASLP.2018.2795749

[5] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," 2020, arXiv:2002.08933.

[6] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 27, no. 8, pp. 1256–1266, 2019.

[7] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: Efficient long sequence modeling fortime-domain single-channel speech separation," in *Proc. 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 46–50.

[8] K. Kinoshita, L. Drude, M. Delcroix, and T. Nakatani, "Listening to each speaker one by one with recurrent selective hearing networks," in *Proc. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5064–5068.

[9] T. von Neumann, K. Kinoshita, M. Delcroix, S. Araki, T. Nakatani, and R. Haeb-Umbach, "All-neural online source separation, counting, and diarization for meeting analysis," in *Proc. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 91–95.

[10] N. Takahashi, S. Parthasaarathy, N. Goswami, and Y. Mitsufuji, "Recursive speech separation for unknown number of speakers," in *Proc. Interspeech 2019*, 2019, pp. 1348–1352. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1749

[11] Y. Qian, X. Chang, and D. Yu, "Single-channel multi-talker speech recognition with permutation invariant training," *Speech Communication*, vol. 104, pp. 1–11, 2018.

[12] X. Chang, Y. Qian, K. Yu, and S. Watanabe, "End-to-end monaural multi-speaker asr system without pretraining," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6256–6260.

[13] T. von Neumann, K. Kinoshita, L. Drude, C. Boeddeker, M. Delcroix, T. Nakatani, and R. Haeb-Umbach, "End-to-end training of time domain audio separation and recognition," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7004–7008.

[14] L. Massaron and A. Boschetti, "Stock price prediction with LSTM," in *TensorFlow Deep Learning Projects: 10 real-world projects on computer vision, machine translation, chatbots, and reinforcement learning*. Packt Publishing, 2018, ch. 6, pp. 131–131.

[15] J. Garofolo, D. Graff, P. Doug, and D. Pallett, *CSR-I (WSJ0) Complete LDC93s6a*. Philadelphia, New Jersey: Linguistic Data Consortium, 1993.

[16] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech and Lang. Process.*, vol. 14, pp. 1462–1469, 2006.