Speaker Counting and Separation From Single-Channel Noisy Mixtures

Srikanth Raj Chetupalli, *Member, IEEE*, and Emanuël A. P. Habets, *Senior Member, IEEE*

***Abstract*—We address the problem of speaker counting and separation from a noisy, single-channel, multi-source, recording. Most of the works in the literature assume mixtures containing two to five speakers. In this work, we consider noisy speech mixtures with one to five speakers and noise-only recordings. We propose a deep neural network (DNN) architecture, that predicts a speaker count of zero for noise-only recordings and predicts the individual clean speaker signals and speaker count for mixtures of one to five speakers. The DNN is composed of transformer layers and processes the recordings using the long-time and short-time sequence modeling approach to masking in a learned time-feature domain. The network uses an encoder-decoder attractor module with long-short term memory units to generate a variable number of outputs. The network is trained with simulated noisy speech mixtures composed of the speech recordings from WSJ0 corpus, and noise recordings from the WHAM! corpus. We show that the network achieves 99% speaker counting accuracy and more than 19 dB improvement in the scale-invariant signal-to-noise ratio for mixtures of up to three speakers.**

***Index Terms*—Source separation, speaker counting, attractors, transformers.**

I. Introduction

Speech presence detection, suppression of undesired noises, and estimation of the speaker signals, from noisy speech recordings containing one or more speakers, are the three most important tasks for any speech analytics application. Speech mixtures containing overlap of two or more speakers are found to be a bottleneck for the success of speech diarization, automatic speech recognition and speaker verification. A separate-and-process strategy is found to benefit these tasks, especially in time regions containing speaker overlap. Data driven approaches using deep neural networks, in particular, have been successful for single-channel speech separation. Traditional approaches commonly assume the speech recording to be a mixture of two or more concurrent speakers. In long-form recordings, there exist time regions where none of the speakers are active, or more commonly regions where a single speaker is speaking, and also regions of multi-speaker overlap. The number of speakers in the mixture is often unknown. It is desirable to have a single system, that can detect speech presence, count the number of speakers, and separate individual sources from a mixture signal with unknown number of speakers.

Early DNN approaches to source separation considered masking in the short-time Fourier transform domain [1], [2]. Recent advances showed that masking in a learned time-feature domain gives better separation than a short-time Fourier transform (STFT) based approach provided the window duration is sufficiently small, which is also desirable for low-latency/online source separation [3]. Further, dual-path processing to model the long-time and short-time relations in the data is found to be beneficial [4]. Convolutional layers are used in [3], recurrent layers in [4], transformers in [5], [6], and a combination of recurrent and transformer layers in [7], to compose the DNN separator. Strategies such as incorporating auxiliary speaker loss are also studied [8]. The number of speakers in the mixture is assumed to be known, a-priori, in [1], [2], [3], [4], [5], [6], [7], [8].

Several works [9], [10], [11], [12] have addressed the separation of an unknown number of speakers (mixtures of more than two speakers). In [9], a recursive multi-pass source extraction strategy is proposed for a STFT-domain masking network. The network, trained on two speaker mixtures, was found to generalize to mixtures of zero to two speakers. A recursive source separation scheme, using a one-vs-rest training strategy, was proposed in [10]. The approach was shown to generalize to an unseen number of sources. In the multi-decoder approach of [11], several decoders, one for each possible number of speakers, were used with a common encoder and trained along with a speaker-counting head. In [12], multiple architectures, one for each possible speaker count, were trained and a voice activity based criterion is used to infer the number of speakers. The above works require multiple forward passes [10], multiple decoders [11] or multiple models [12]. To overcome these limitations, we proposed an architecture in [13], in which the speaker counting is achieved using an encoder-decoder-attractor (EDA) module [14] and the speech separation module uses a network of transformer layers configured to model short-time, long-time and cross-speaker relationships.

Previous works on source separation focused mainly on clean speech mixtures, but in real recordings, the mixtures are often noisy and have reverberation. WHAM! [15] and WHAMR! [16] datasets were proposed to develop source separation algorithms for noisy and reverberant mixtures. Joint separation and enhancement was studied in [17], [18], but they assume a fixed

Manuscript received 29 September 2022; revised 9 March 2023; accepted 9 April 2023. Date of publication 20 April 2023; date of current version 3 May 2023. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Marc Delcroix. *(Corresponding author: Srikanth Raj Chetupalli.)*

The authors are with the International Audio Laboratories, A Joint Institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer IIS, 91058 Erlangen, Germany (e-mail: srikanth.chetupalli@iis.fraunhofer.de; emanuel.habets@audiolabs-erlangen.de).

Digital Object Identifier 10.1109/TASLP.2023.3268572

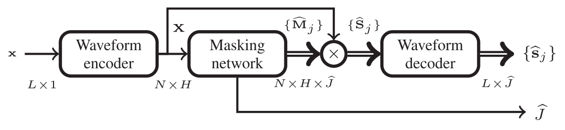

Fig. 1. Source separation system for unknown number of speakers.

number of sources. The approach in [12], which uses a different model for each speaker count, was recently extended to noisy and reverberant mixtures in [19].

In this paper, we develop a unified approach to the tasks of speech detection, speaker counting and separation from a noisy single-channel recording. The joint task is posed as that of "predicting zero or more speakers, and estimating the individual speaker signals when the predicted speaker count is greater than zero". The work uses the DNN architecture we recently proposed in [13]. Unlike [13] where the DNN is trained on two or more speaker mixtures, we train it on noise-only signals (zero speakers), noisy single-speaker recordings, and noisy multi-speaker mixtures in the present work. We show that, the proposed architecture (i) predicts zero speaker count for a noise-only recording, (ii) auto-encodes/enhances the single-speaker recordings, and (iii) estimates the individual source signals for noisy multi-speaker recordings. Further, we study the generalization capabilities of the trained network for different, practical evaluation conditions and provide insights into the feature representations computed by the DNN at various layers, which provide cues to efficient architecture designs in the future.

II. System Overview

Let $\mathbf{x}$ be the vector representation of a recording with a duration of L samples,

$$\mathbf{x} = \sum_{j=1}^{J} \mathbf{s}_j + \mathbf{v}, \tag{1}$$

where J is the number of speakers, $\mathbf{s}_j$ is the j^{th} source signal and $\mathbf{v}$ is the noise signal. The source signals and noise signal are assumed to be mutually uncorrelated. The goal in this work is to, (i) predict $J = 0$ when $\mathbf{x} = \mathbf{v}$, i.e., when the speech is absent, and (ii) predict J and estimate $\mathbf{s}_j, \forall j$, when $\mathbf{x}$ follows (1) with $J \geq 1$,

$$\widehat{J}, \{\widehat{\mathbf{s}}_j\}_{j=1}^{\widehat{J}} = f_{\boldsymbol{\theta}}(\mathbf{x}). \tag{2}$$

We consider the supervised learning approach using a DNN in this paper and $f_{\boldsymbol{\theta}}(.)$ represents the function learned by the DNN architecture having parameters $\boldsymbol{\theta}$.

We consider masking, using a DNN, in the learned time-feature (TF) domain [3], as shown in Fig. 1. The waveform encoder, with a learnable analysis filter-bank, converts the time-domain signal ($\mathbf{x}$) into the TF domain ($\mathbf{X}$). The masking network, using the EDA block in the architecture, predicts the number of speakers $\widehat{J}$ and a mask for each predicted speaker $\widehat{\mathbf{M}}_j$ (when $\widehat{J} > 0$). The predicted masks are multiplied with the mixture TF representation and input to the waveform decoder. The decoder synthesis filter-bank reconstructs the source signals

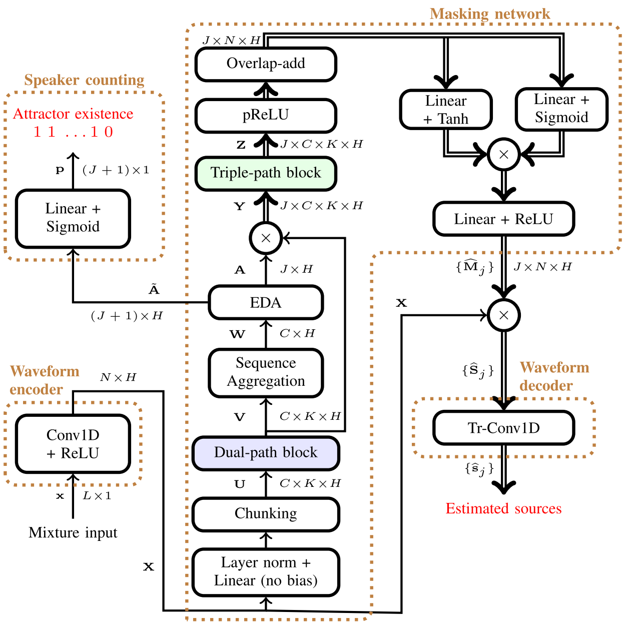

Fig. 2. Block diagram of the SD-SepEDA architecture.

($\widehat{\mathbf{s}}_j, \forall\, j \in [1, \widehat{J}]$), independently. We refer to the proposed DNN architecture as "Speech Detector and Separator using Encoder-Decoder-Attractors (SD-SepEDA), in the following sections.

III. SD-SepEDA Separator

A block diagram of the proposed SD-SepEDA architecture is shown in Fig. 2. The individual blocks in the architecture are described in this section.

A. Time-Feature Representation

The L dimensional mixture signal $\mathbf{x}$ is fed to the waveform encoder, which generates a non-negative, sub-sampled, time-feature representation $\mathbf{X}$. It is composed of a 1D convolution layer (Conv1D) with H filters followed by a rectified linear unit (ReLU) activation. An over-complete representation using short-time windows is found to benefit source separation [3], hence, we choose a kernel size equivalent to 2 ms (window length) with a stride of 1 ms (hop size),

$$\mathbf{X} = \text{ReLU}(\text{Conv1D}(\mathbf{x})),\ \mathbf{X} \in \mathbb{R}_{+}^{N \times H}, \tag{3}$$

where N is the number of time-frames.

B. Masking Network

Input processing: The masking network input $\mathbf{X}$ is first passed through a layer-norm (LN) layer [20], followed by a linear layer without bias. It is then segmented into chunks of size K frames with a 50% overlap between successive chunks. In the present work, we choose $K = 250$ which corresponds to a chunk size of 250 ms and a hop of 125 ms.

Dual-path block: Dual-path block, shown in Fig. 3, models the short-time and long-time relations in the input using transformers [21]. The block composition is identical to the SepFormer block proposed in [5]. The intra-chunk transformer performs

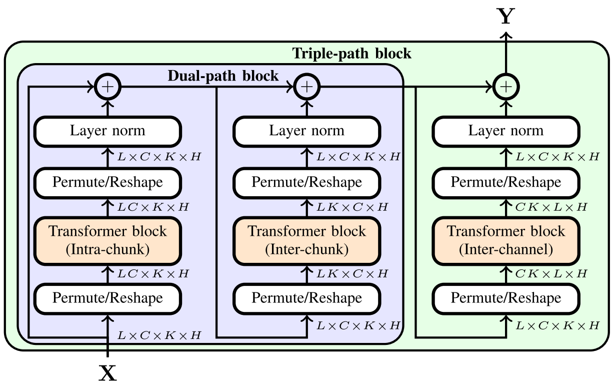
Y
Triple-path block
Dual-path block
Layer norm
Permute/Reshape
Transformer block (Intra-chunk)
Transformer block (Inter-chunk)
Transformer block (Inter-channel)
$L\times C\times K\times H$
$LC\times K\times H$
$LK\times C\times H$
$CK\times L\times H$
X

Fig. 3. Block diagram of the dual-path and triple-path blocks. L, C, K, and H denote the number of channels, number of chunks, chunk size, and feature dimension, respectively. $L = 1$ for the dual-path block and $L = \hat{J}$ for the triple-path block in Fig. 2.

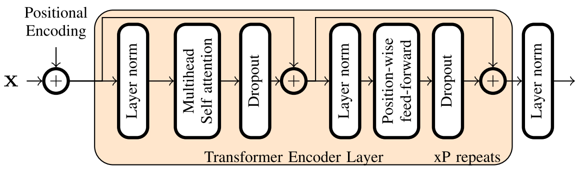

Fig. 4. Block diagram of the transformer block.

attention across the time-steps with-in a chunk, while treating the chunk dimension as the batch. In contrast, the inter-chunk transformer performs attention across the chunks at every time-frame. The axes of the inputs and outputs of the transformers are appropriately permuted and reshaped as shown in Fig. 3. Skip connections are implemented across the intra-chunk and inter-chunk transformers after passing the transformer outputs through a LN layer.

The transformers comprise of a stack of transformer encoder layers and optional position encoding, as shown in Fig. 4. The input feature matrix is, optionally, added with a positional encoding matrix and input to a stack of P transformer encoder layers followed by a LN layer. We use the sinusoidal position encoding in this work. The composition of the transformer encoder layer, shown in Fig. 4, closely follows the definition in [21], with pre-normalization and dropout. In this work, we use $P = 4$ layers in the intra-chunk transformer and $P = 2$ layers in the inter-chunk transformer.

Attractor generation and speaker counting: The purpose of the attractor generation module is to generate a conditioning vector for each speaker in the mixture signal and facilitate speaker counting. The attractors are generated on chunk-level aggregated representations, as they capture speaker characteristics better than the fine frame level features. A typical choice for the intra-chunk sequence aggregation is mean pooling, which gives equal importance to all the time frames including the speech silences. In this work, we consider the self-attentive weighted subspace projection strategy shown in Fig. 5, which weights the samples based on their importance as computed by the attention weights. Such a strategy is found to encode the sequence characteristics effectively and improve the final

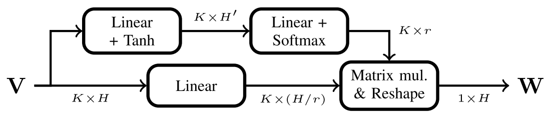

Fig. 5. Block diagram of the sequence aggregation module.

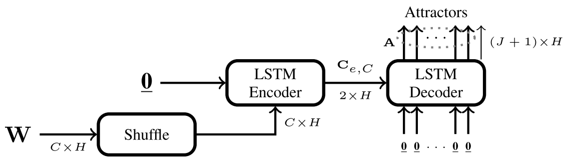

Fig. 6. Block diagram of the EDA module.

performance, for example, in the context of speaker verification in [22]. The processing steps are,

$$\mathbf{V}' = \mathrm{Tanh}(\mathrm{Linear}(\mathbf{V})),\ \mathbf{V}' \in \mathbb{R}^{K\times H'}, \tag{4}$$

$$\boldsymbol{\alpha} = \mathrm{Softmax}(\mathrm{Linear}(\mathbf{V}'))),\ \boldsymbol{\alpha} \in \mathbb{R}^{K\times r}, \tag{5}$$

$$\mathbf{W} = \mathrm{Flatten}(\boldsymbol{\alpha}^T \mathrm{Linear}(\mathbf{V})),\ \mathbf{W} \in \mathbb{R}^{1\times H}. \tag{6}$$

Here, r different weighted linear combinations of a low-dimension (H/r) projection of the input $\mathbf{V}$ are concatenated to generate the summary vector. The weights are computed, from a higher dimension projection of $\mathbf{V}$ ($H' = 4H$), as shown in the top section of Fig. 5.

The EDA module, shown in Fig. 6, is composed of an LSTM encoder and an LSTM decoder. The operation of the EDA module is similar to the method proposed in [14]. The input $\mathbf{W}$ of chunk level representations is first shuffled along the chunk dimension to avoid learning the sequence related information and model only the speaker attributes. The shuffled representations are fed to the LSTM encoder. The state $\mathbf{c}_{e,C}$ of the encoder at the last step (chunk C), comprising of the cell state and the memory state, encodes a file-level summary of the speakers in the recording, i.e.,

$$\sim, \mathbf{c}_{e,C} = \text{LSTM encoder}(\text{shuffle}(\mathbf{W})). \tag{7}$$

The state $\mathbf{c}_{e,C}$ is used as the state at step-0 of the LSTM decoder. The LSTM decoder is run for $J + 1$ steps for a recording with J speakers. In step-j, it takes a vector of zeros as the input and generates an embedding $\mathbf{a}_j$, referred to as the "attractor". The $J + 1$ attractors are fed to the attractor existence detection module. In addition, the first J attractors are multiplied element-wise with the DPB output $\mathbf{V}$ to generate J parallel channels, i.e.,

$$\mathbf{Y}_j = \mathbf{a}_j \odot \mathbf{V}, \forall j \in \{1, 2, \ldots, J\}, \tag{8}$$

where $\mathbf{Y} = [\mathbf{Y}_1 \ldots, \mathbf{Y}_J]$.

The attractor existence detection module is similar to a binary classifier and composed of a Linear-Sigmoid layer. The module is trained to predict a 1 or 0 depending on whether the attractor should correspond to a speaker or otherwise. For a recording with J speakers, the module is expected to predict a 1 for the attractors $\{\mathbf{a}_1, \ldots, \mathbf{a}_J\}$ and a 0 for $\mathbf{a}_{J+1}$.

During training, for a recording with J speakers, we extract $J + 1$ attractors. During inference, if the number of speakers is

Algorithm 1: EDA Processing With Speaker Count Estimation.

```
Input: $\mathbf{W}$
$\sim, \mathbf{c}_{e,C}$ = LSTM encoder(input=$\mathbf{W}$, state=$\underline{\mathbf{0}}$)
Attractor matrix $\mathbf{A} = [\,]$
$j = 0$
$\mathbf{a}_j, \mathbf{c}_{d,j}$ = LSTM decoder(input=$\underline{\mathbf{0}}$, state=$\mathbf{c}_{e,C}$)
while Linear-Sigmoid($\mathbf{a}_j$) $\geq 0.5$ do
    $\mathbf{A}$ = Concatenation($\mathbf{A}, \mathbf{a}_j$)
    $j \leftarrow j + 1$
    $\mathbf{a}_j, \mathbf{c}_{d,j}$ = LSTM decoder(input=$\underline{\mathbf{0}}$, state=$\mathbf{c}_{d,j-1}$)
end while
$\widehat{J}$ = length($\mathbf{A}$)
return attractors $\mathbf{A}$ and speaker count $\widehat{J}$
```

not known a-priori, the attractors are generated sequentially as described in Algorithm 1.

Triple-path block: Triple-path block (TPB) extends the dual-path block processing with an additional inter-channel transformer, as shown in Fig. 3. In this block, the intra-chunk and inter-chunk transformers model the J channels independently. In the inter-channel transformer, the intra- and inter-chunk dimensions are treated as batch size and the attention is applied along the channel dimension. We do not use position encoding in the inter-channel transformer, since the channel order is arbitrary. The inter-channel is also bypassed for recordings with $J = 1$. The inter-channel transformer is similar to the strategy implemented in [23] for multi-channel speech enhancement.

Mask prediction: The mask prediction follows the steps from the SepFormer architecture [5]. The 4D output of the triple-path block is passed though a pReLU layer and overlap-add method is used to invert the chunking operation. The signals are then fed to a Linear-Tanh layer and a Linear-Sigmoid layer in parallel and the corresponding outputs are multiplied element-wise to compute a gated output. Finally, a Linear-ReLU layer predicts the source masks $\widehat{\mathbf{M}}_j, \forall j \in \{1, 2, \ldots, J\}$ for the predicted sources. TF representations for the separated sources are obtained as, $\widehat{\mathbf{S}}_j = \widehat{\mathbf{M}}_j \odot \mathbf{X}, \forall j \in \{1, 2, \ldots, J\}$.

C. Signal Reconstruction

The waveform decoder uses a 1D transpose-convolution layer (Tr-Conv1D) with the same number of filters, kernel size, and stride as that of the waveform encoder, discussed in Section III-A,

$$\widehat{\mathbf{s}}_j = \text{Tr-Conv1D}(\widehat{\mathbf{S}}_j),\ \widehat{\mathbf{s}}_j \in \mathbb{R}^{L\times 1}, \forall\, j \in \{1, 2, \ldots, J\}. \quad (9)$$

D. Loss Function

To train the DNN, we use a weighted combination of the signal estimation loss $\mathcal{L}_{\text{signal}}$ and the attractor existence loss $\mathcal{L}_{\text{attr}}$, i.e.,

$$\mathcal{L} = \begin{cases} \mathcal{L}_{\text{signal}} + \eta\mathcal{L}_{\text{attr}}, & \text{if } J \geq 1; \\ \mathcal{L}_{\text{attr}}, & \text{otherwise}, \end{cases} \quad (10)$$

where $\eta > 0$ is the weight value.

The negative of the scale-invariant signal-to-noise ratio (SI-SNR), averaged over the speakers in the mixture, computed in a permutation-invariant manner [24] is used for $\mathcal{L}_{\text{signal}}$. Let $\mathcal{P}$ denote the set of all possible permutations of the estimated source signals. The signal estimation loss $\mathcal{L}_{\text{signal}}$ is then computed as,

$$\mathcal{L}_{\text{signal}} = \min_{p\in\mathcal{P}} \frac{1}{J} \sum_{j=1}^{J} -\text{SI-SNR}(\mathbf{s}_j, \widehat{\mathbf{s}}_{p(j)}). \quad (11)$$

The SI-SNR measure, for a reference and estimated signal pair $(\mathbf{s}, \widehat{\mathbf{s}})$, is defined as [3], [25],

$$\text{SI-SNR}(\mathbf{s}, \widehat{\mathbf{s}}) = 20 \log \frac{\|\alpha\mathbf{s}\|}{\|\alpha\mathbf{s} - \widehat{\mathbf{s}}\|}, \quad (12)$$

where $\alpha =< \mathbf{s}, \widehat{\mathbf{s}} > / < \mathbf{s}, \mathbf{s} >$ is the scale parameter. For a mixture with J speakers, $\mathcal{L}_{\text{attr}}$ is computed as

$$\mathcal{L}_{\text{attr}} = \frac{1}{J+1} \left(\sum_{j=1}^{J} \text{BCE}(1, p_j) + \text{BCE}(0, p_{J+1}) \right), \quad (13)$$

where $\mathbf{p} = [p_1, p_2, \ldots, p_{J+1}]$ is the Linear-Sigmoid layer output, $0 \leq p_j \leq 1, \forall j$. To compensate for the scale differences between $\mathcal{L}_{\text{signal}}$ and $\mathcal{L}_{\text{attr}}$, we choose $\eta = 10$, unless otherwise stated.

IV. Experimental Setup

A. Datasets

We conducted the experiments on noisy-speech mixtures, simulated using single-speaker recordings from the WSJ0 corpus [26] and noises from the WHAM! corpus [15].

WSJ0-Mix dataset: WSJ0-JMix is a synthetic dataset of J speaker mixtures ($J \in \{2, \ldots, 5\}$), composed using clean WSJ0 recordings [26] and created as defined in [2] using an open-source Python tool.[1] To create the mixture, (i) the clean signals were normalized, (ii) scaled with gain values sampled randomly, and (iii) summed and then normalized such that the peak amplitude of the mixture signal is 0.9. Let g_1, g_2 be gain values uniformly sampled from the range [0,2.5] dB. The gains for the individual speaker signals for two to five speaker mixtures are $\{g_1, -g_1\}$, $\{g_1, -g_1, 0\}$, $\{g_1, g_2, -g_1, -g_2\}$, $\{g_1, g_2, -g_1, -g_2, 0\}$ respectively.

For each J, the dataset has 20 K, 5 K and 3 K recordings in the train, validation and test splits, respectively. These three splits have disjoint speakers. We used the "min" duration mode of the dataset (as defined in [2]) for the experiments in this paper, which also guarantees that the mixture signals have a full-overlap between the speakers except for the intra-recording silences. We refer to the original recordings from WSJ0 as single-speaker mixtures (WSJ0-1Mix). The train, validation and test splits of WSJ0-1Mix have 8769, 3557 and 1770 recordings respectively. We refer to the pool of 1-5 speaker mixtures as the WSJ0-Mix dataset.

WHAM!-WSJ0-Mix dataset: We paired the mixtures from the WSJ0-Mix dataset with noise signals sampled randomly from

[1] [Online]. Available: https://github.com/mpariente/pywsj0-mix

the WHAM! corpus. We used the same train, validation and test splits of the raw recordings defined in [15] to sample the noise signals for the corresponding splits of WSJ0-Mix. The noise signals were added to the speech mixtures at a mixture-signal-to-noise ratio (MSNR) sampled randomly from the range 30–40 dB, for all the splits of the dataset. Additionally, we evaluated the trained models using a wider range of MSNR values in Section VI. For the training examples, we paired each mixture with a different noise sample with a different MSNR value in each epoch. For validation and test, each mixture from WSJ0-Mix was paired with a unique noise sample with a unique MSNR value at all epochs.

We also created noise-only recordings, sampled from WHAM!, and refer to them as 0-speaker mixtures. We sampled 20 K, 5 K and 3 K examples for the train, validation and test splits. Each example in the 0-speaker mixtures set has a duration of 4 s. We refer to the cumulative set of 0-speaker mixtures and the 1–5 speaker mixtures of WSJ0-Mix added with WHAM! noises as the WHAM!-WSJ0-Mix dataset.

The mixture signal duration was limited to a maximum of 15 s during training and not limited for the test recordings. The duration of the mixture recordings in the test splits varied from 1.62 s to 13.87 s.

B. Training Details

The SpeechBrain [27] platform was used to train the models. Adam optimizer [28] was used with an initial learning rate of 1.5×10^{-4} and a batch size of 1. The learning rate was fixed for the first 20 epochs and halved later if the validation SI-SNR is not increasing for two consecutive epochs. A loss threshold of -30 dB was applied to the negative SI-SNR loss function. The norm of the gradients was clipped to 5. Automatic fixed precision was used to increase the training speed. Time domain speech perturbation [29] was used as the audio augmentation scheme with perturbation factors 95%, 100% and 105%. The model parameters corresponding to the best validation SI-SNR were used for the final evaluation.

C. Performance Measures

We measured the performance of the proposed approach using SI-SNR improvement (SI-SNRi), SDR improvement (SDRi) [30], and speaker-counting accuracy (SCA) measures. The number of sources is known and fixed during training, but the estimated number of sources during inference can be different from the ground truth. When the number of sources was under-estimated, an all-zero signal was used as the pseudo-estimate for the sources not accounted for in the output, to compute the SI-SNRi measure. On the other hand, if the number of sources was over-estimated, the subset of sources with the maximum average SI-SNRi were used for the performance metric computation. In all the cases, the performance measures were computed for the optimal pairing of estimated and reference sources, i.e., accounting for the permutation ambiguity. Computation of SDR involves a projection of the estimated signal into the desired signal and interfering speaker signal subspaces [30]. This is not possible if either the desired signal or its estimate have all zeros. Hence, SDRi was calculated only for recordings with correct speaker count estimation. We used the open source Python tool in [31] for the computation of SDRi measure. SCA was calculated as the percentage of test recordings with correctly estimated speaker count ($\widehat{J} = J$).

TABLE I
PERFORMANCE FOR WHAM!-WSJ0-MIX TEST SETS

#Srcs	SCA %	SI-SNRi (dB)			SDRi (dB)	
		$\widehat{J}$	J	$\widehat{J}=J$	J	$\widehat{J}=J$
0	99.97	-	-	-	-	-
1	100	1.23	1.23	1.23	1.27	1.27
2	99.93	20.00	20.02	20.04	20.23	20.25
3	98.93	18.64	18.96	19.00	19.19	19.23
4	90.53	13.88	15.96	16.32	16.19	16.54
5	75.4	9.78	13.97	14.54	14.18	14.74

$\widehat{J}$ and J denote the results with estimated and ground-truth speaker count used during inference, respectively. The column $\widehat{J} = J$ shows the performance of the system across the samples for which the estimated and ground-truth speaker counts are equal.'-': data not available.

TABLE II
SPEAKER CONFUSION MATRIX FOR WHAM!-WSJ0-MIX TEST SETS

		Estimated #Srcs ($\widehat{J}$)						
		0	1	2	3	4	5	6
#Srcs (J)	0	99.97	0.03	0	0	0	0	0
	1	0	100	0	0	0	0	0
	2	0	0.07	99.93	0	0	0	0
	3	0	0	1	98.93	0.07	0	0
	4	0	0	0.03	9.23	90.5	0.23	0
	5	0	0	0	0.1	24.53	75.37	0

V. Results

We evaluated the proposed SD-SepEDA model using (i) the WHAM!-WSJ0-Mix test sets which denote the matching condition for the training, and (ii) the WSJ0-Mix test sets without noise. For each condition, we evaluated the model with speaker count estimation (as described in Algorithm 1) and with ground-truth speaker count (as in the training stage).

Table I shows the signal estimation performance for the WHAM!-WSJ0-Mix test sets. For $J = 0$, we found that only 1 out of the 3000 test samples is identified as having a speaker ($\widehat{J} = 1$). For $J = 1$, the input MSNR is high (sampled from the range $[30 - 40]$ dB) and hence the improvement is less. However, the results indicate that the model is capable of signal enhancement. For $J > 2$, the metrics reflect mainly the source separation quality, and we see that the performance degrades with an increase in J. The incorrect speaker count estimation contributes significantly to the performance degradation for $J > 3$, since inference using the ground-truth speaker count for the same model achieves better performance. Table I also shows the average performance across the recordings with correct speaker count estimation only. The SI-SNRi and SDRi measures are comparable on these recordings with correct estimated speaker count. Table II shows the speaker confusion matrix. We see that the model has a tendency to under-estimate the number of speakers. We observed that the model does not estimate more

TABLE III
PERFORMANCE FOR WSJ0-MIX TEST SETS AND COMPARISON WITH THE BASELINE SYSTEMS

#Srcs	SD-SepEDA						MulCAT [12]			SepEDA [13]			Recursive SS [10]
	SCA %	SI-SNRi (dB)			SDRi (dB)		SCA %	SI-SNRi (dB)		SCA %	SI-SNRi (dB)		SI-SNRi (dB)
		$\widehat{J}$	J	$\widehat{J}=J$	J	$\widehat{J}=J$		J	$\widehat{J}$		J	$\widehat{J}$	J
1*	100	48.67	48.67	48.67	48.57	48.57	-	-	-	-	-	-	-
2	99.93	20.14	20.16	20.18	20.39	20.40	84.6	20.1	18.6	99.80	21.1	21.1	14.8
3	98.77	18.64	19.02	19.07	19.25	19.30	69.0	16.9	14.6	97.00	18.6	18.4	12.6
4	90.37	13.87	16.00	16.36	16.22	16.58	47.5	12.9	11.5	90.17	14.7	14.4	10.2
5	74.43	9.63	13.99	14.58	14.19	14.79	92.3	10.6	10.4	96.87	12.1	11.6	-

The columns $\widehat{J}$ and J denote the results with estimated and ground-truth speaker count used during inference, respectively. The column $\widehat{J}=J$ shows the performance of the system across the samples for which the estimated and ground-truth speaker counts are equal. '*': For single-speaker recordings, reconstruction SI-SNR and SDR are reported. '-': data not available.

TABLE IV
SPEAKER CONFUSION MATRIX FOR WSJ0-MIX TEST SETS

		Estimated #Srcs ($\widehat{J}$)						
		0	1	2	3	4	5	6
#Srcs (J)	1	0	100	0	0	0	0	0
	2	0	0.07	99.93	0	0	0	0
	3	0	0	1.2	98.77	0.03	0	0
	4	0	0	0.03	9.43	90.37	0.17	0
	5	0	0	0	0.1	25.47	74.43	0

than 5 speakers in general (the maximum number of speakers in the train set). For $0-3$ speakers, the SCA obtained may be satisfactory for most practical applications.

The performance obtained for the clean WSJ0-Mix test sets is shown in Tables III, IV. For $J=1$, the table shows the reconstruction SI-SNR (no-noise and no-interfering speakers). We see that the reconstruction SI-SNR is close to 50 dB and the SCA is 100%. For the mixture recordings ($J>1$), the performance trends are similar to the WHAM!-WSJ0-Mix test set results shown in Tables I and II.

Table III also shows the performance of three baselines, the recursive separation system in [10], the MulCAT approach of [12] which uses multiple models, and our previously reported SepEDA model [13]. We note that the baseline systems in [12], [13] work only for two or more speaker mixtures ($J \geq 2$) as they are trained on two-five speaker mixtures, unlike the proposed model which works for $J \geq 0$. The method in [10], in theory, can detect single speaker recordings. But the single speaker recordings are not considered in the experiments in [10]. Comparing with the proposed SD-SepEDA, we see that SD-SepEDA has better performance than the current best architectures reported in the literature. Compared to the SepEDA model reported in [13], we see a slight degradation in performance for the condition where the speaker count is estimated. For the known number of sources condition, the proposed model is found to be better for $J>2$. The speaker counting is also better for $J<5$ for the proposed model compared to SepEDA model of [13].

To investigate further, we studied the distribution of source-wise SI-SNRi for the mixture recordings ($J>1$) in Fig. 7. For $J=2$ and $J=3$, peak in the distribution is observed beyond 20 dB, and most of the recordings have more than 15 dB SI-SNRi. A significant left-leaning tail is observed for $J>3$, though the peaks are beyond 15 dB SI-SNRi. This indicates that, for $J>3$, a subset of speakers in the mixture are correctly estimated, and the remaining speakers may be confused. Fig. 8 shows the effect of the median pitch frequency of individual speakers on the separation quality for the two-speaker mixtures. The frame-level pitch values were computed using the Parselmouth Python tool [32] and the median was computed over the voiced speech regions. We see that, closer to the diagonal, the separation is poor, i.e., the difference of median pitch frequency between the two speakers is small. There are two clusters with poor separation, in the low and high frequency regions, which correspond to same gender speaker mixtures. We also studied

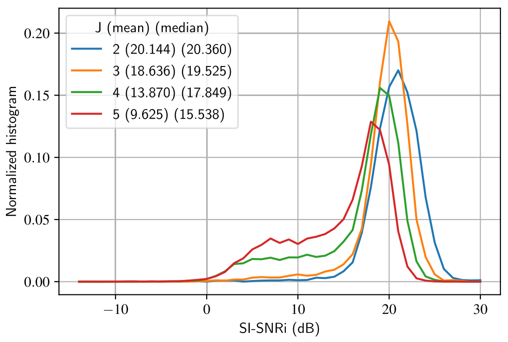

Fig. 7. Histogram of source-wise SI-SNRi for WSJ0-Mix test sets.

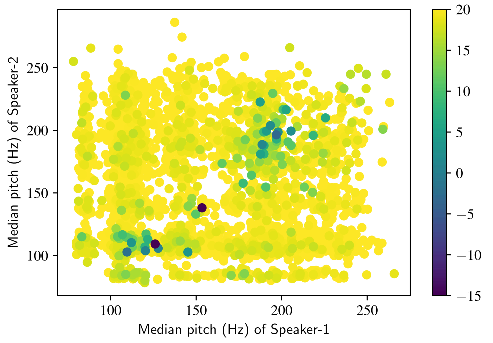

Fig. 8. Scatter plot of median pitch frequencies of speakers in two-speaker mixtures (WSJ0-2Mix test set). The colorbar (clipped to the range $[-15, 20]$ dB) indicates the corresponding SI-SNRi value.

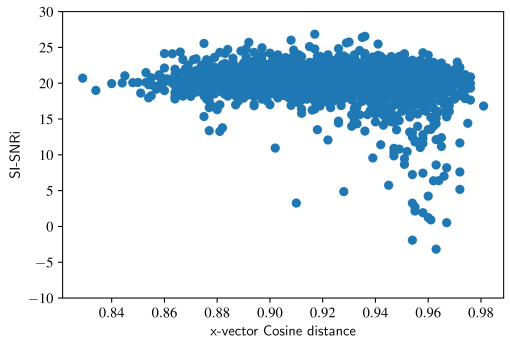

Fig. 9. SI-SNRi versus x-vector Cosine similarity between speakers for the two-speaker mixtures (WSJ0-2Mix test set).

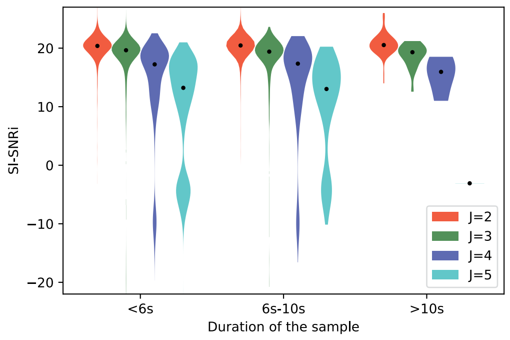

Fig. 10. SI-SNRi distribution for the test samples in WSJ0-Mix test sets divided into three groups by duration. The black dot shows the median value.

the relation between SI-SNRi and the x-vector cosine similarity between the speakers for the two speaker mixtures. For x-vector extraction, we used the pre-trained extractor available in the SpeechBrain framework [27].[2] Fig. 9 shows the scatterplot of cosine similarity against the SI-SNRi. We see that almost all recordings which have a poor SI-SNRi (<15 dB) have a higher cosine similarity, but the vice-versa is not true. The analysis shows that the architecture may be using not just timbre and pitch but also other information such as the speech onset instances for source separation.

Next, we study the performance as a function of the duration of the test samples. Fig. 10 shows the distribution of SI-SNRi of the test samples divided into three groups by duration for the mixtures of two-five speakers. The median performance is similar for the different duration groups for a given speaker count. However, the worst-case performance is improved as the duration of the sample increases, which can be seen for the mixtures with more than three speakers in Fig. 10.

We measured the computational complexity of the proposed architecture using the number of GigaFLOPs estimated using the open-source flop-counter tool.[3] For a 3 s recording, the complexity is found to be $\{70.38, 125.84, 171.04, 216.26, 261.49\}$ GigaFLOPs for 1-5 speaker recordings, respectively.

[2][Online]. Available: https://huggingface.co/speechbrain/spkrec-xvect-voxceleb

[3][Online]. Available: https://github.com/facebookresearch/fvcore

TABLE V
PERFORMANCE FOR DIFFERENT MSNR CONDITIONS

#Srcs	Speaker counting accuracy (%)				SI-SNRi (dB)			
	[0-10]	[10-20]	[20-30]	[30-40]	[0-10]	[10-20]	[20-30]	[30-40]
1	4.69	68.31	100.00	100	-9.13	2.37	1.92	1.23
2	2.33	58.06	99.90	99.93	7.22	15.14	18.90	19.99
3	1.93	58.86	99.03	98.93	8.89	15.76	18.29	18.64
4	1.83	55.73	93.03	90.53	7.01	13.89	14.36	13.88
5	94.23	97.00	82.06	75.40	3.68	12.01	10.79	9.78

VI. ROBUSTNESS AND GENERALIZATION ANALYSIS

In the following, we study the generalization capabilities of the proposed SD-SepEDA model for different MSNR conditions, SIR variation, and for reverberant recordings.

MSNR variation: The test sets for this analysis use the noisy speaker mixtures, but with different MSNR values compared to the WHAM!-WSJ0-Mix test sets discussed in Section IV-A. The MSNR values were sampled from three different ranges $[0-10]$ dB, $[10-20]$ dB, $[20-30]$ dB and compared with the training data setting of $[30-40]$ dB range. As shown in Table V, the SCA degrades with decreasing MSNR except for $J=5$, which is also reflected in the SI-SNRi values. At low MSNRs, the speaker count estimated is either 0 (i.e., no speech) or 5 (maximum predicted by the model). This also justifies the better SCA obtained at low MSNRs for $J=5$. For the $[20-30]$ dB condition, the performance is comparable to the $[30-40]$ dB case.

SIR variation: For the analysis in this section, we used the WSJ0-Mix test sets with $2-5$ speakers, but with different relative gains for the mixed signals, which translate to differences in the SIR. The training dataset as defined in Section IV-A has gains g_1, g_2 sampled from the range $[0-2.5]$ dB. For the evaluation in this section, we sampled the gains from three ranges $[0-5]$ dB, $[0-10]$ dB, $[0-15]$ dB. The SCA and the separation accuracy degrade with increase in SIR range, as shown in Table VI. We observed that the speaker count is under-estimated for higher SIR cases, i.e., the weak sources are not identified by the model. The performance for the 5 dB condition is also comparable to the train condition (2.5 dB), shown in Table I. Table VI also shows the results for the known number of sources case. The performance degradation is less with the change in the SIR, and closer to the results reported in Table I. This shows that the separation architecture is robust to different SIR conditions, while the EDA module is sensitive to the SIR variation.

Reverberant test set: In this section, we study the performance of the SD-SepEDA model trained on WSJ0-WHAM!-Mix dataset on reverberant speech. We created a reverberant test set using the simulated RIRs and the WSJ0-Mix test sets. The RIRs were simulated using the pyroomacoustics [33] Python tool, closely following the procedure and the parameters used to create the WHAMR! dataset [16]. Each example in the WSJ0-Mix test sets was paired with a different set of source RIRs. The sources were first convolved with their corresponding RIRs, scaled according to the SIR value, and added to create the reverberant mixture. We used the SIR values from WSJ0-Mix definitions to scale the sources. We considered three different

TABLE VI
PERFORMANCE FOR DIFFERENT SIR CONDITIONS. THE COLUMN LABELS SHOW THE UPPER LIMIT OF THE RANDOM GAINS APPLIED TO SOURCES

#Srcs	#Srcs estimated								#Srcs known			
	Speaker counting accuracy (%)				SI-SNRi (dB)				SI-SNRi (dB)			
	2.5 dB	5 dB	10 dB	15 dB	2.5 dB	5 dB	10 dB	15 dB	2.5 dB	5 dB	10 dB	15 dB
2	99.93	99.90	98.10	74.60	20.14	20.24	19.05	7.90	20.16	20.27	19.71	16.25
3	98.77	98.30	87.10	62.40	18.64	18.80	15.97	9.82	19.02	19.33	19.61	19.14
4	90.37	85.60	49.80	21.50	13.87	13.79	7.14	-1.45	16.00	17.01	17.25	15.12
5	74.43	64.47	27.53	12.93	9.63	8.66	3.27	-2.54	13.99	14.65	15.15	13.57

For a setting with the limit x dB, the worst case SIR for the individual sources across all samples is $-2x$ dB.

TABLE VII
SPEAKER COUNTING ACCURACY AND SI-SNRI (DB) MEASURE FOR THE REVERBERANT TEST SET

#Srcs	Speaker counting accuracy (%)						Direct component reference						Reverberant reference					
	Low		Medium		High		Low		Medium		High		Low		Medium		High	
	A	R	A	R	A	R	A	R	A	R	A	R	A	R	A	R	A	R
1	95.20	**99.32**	51.86	**98.59**	11.47	**87.63**	**16.13**	16.04	**7.95**	6.43	**5.04**	2.03	46.97	**72.53**	28.97	**71.18**	11.07	**63.59**
2	95.93	**97.47**	43.60	**94.30**	6.60	**75.70**	**13.95**	12.31	**9.30**	7.43	**7.45**	5.59	**17.49**	16.17	10.10	**13.33**	4.45	**10.64**
3	**92.20**	84.23	35.73	**78.13**	2.86	**51.90**	**15.00**	10.18	**10.75**	6.84	**6.79**	6.21	**16.52**	11.76	**10.73**	9.67	4.58	**8.62**
4	**83.66**	70.17	34.57	**65.13**	3.53	**42.97**	**12.50**	8.71	**10.18**	6.90	4.55	**6.71**	**13.03**	9.48	**9.81**	8.40	2.84	**7.88**
5	75.30	**75.93**	**95.10**	80.10	**93.23**	93.07	**8.98**	7.73	**8.46**	6.31	0.02	**6.36**	**9.15**	8.14	**8.05**	7.09	1.35	**6.91**

The columns 'A' and 'R' show the performance for the models trained on anechoic and reverberated WSJ0-WHAM!-Mix datasets, respectively.

reverberation time ranges, low: $0.1 - 0.3$ s, medium: $0.2 - 0.6$ s, and high: $0.4 - 1.0$ s, similar to [16]. The performance measures were computed with (i) the direct path component, and (ii) the reverberant source, at the microphone as the reference signal.

For comparison, we also trained the SD-SepEDA model on reverberated WSJ0-WHAM!-Mix dataset. The reverberated training examples were created dynamically. The reverberation time for the training set was sampled uniformly from the range $0.1 - 1.0$ s, and the remaining parameters are sampled similar to the test set, described above. The model is trained to predict the reverberant sources given the reverberant mixture inputs, since our goal is to separate sources and not dereverberation.

Table VII shows the results. For the model trained on anechoic speech, we see that the speaker counting is significantly affected by the reverberation. We observed that the speaker count is over-estimated in the reverberant cases. The better SCA accuracy for the five speaker test set is misleading, because the network is observed to not predict more than five speakers. The SI-SNRi also degraded with the reverberation level. For "low" reverberation condition, SI-SNRi with reverberant reference is better compared to the direct component reference, indicating that the network estimates the reverberant sources at the output. This can be observed clearly in the single-speaker test set, where the SI-SNRi is closer to the clean speech results, shown in Table. I. Training the model with reverberant speech significantly improves the speaker-counting performance for the medium and high reverberant cases. The SI-SNRi performance for the high reverberation case is also improved. However, the performance for the low and medium reverberation cases is degraded compared to the model trained on non-reverberant mixtures. This is due to the training set which has reverberation times sampled from the wider range of $0.1 - 1.0$ s, the model performance is improved for high reverberation recordings at the cost of degradation for low reverberation cases.

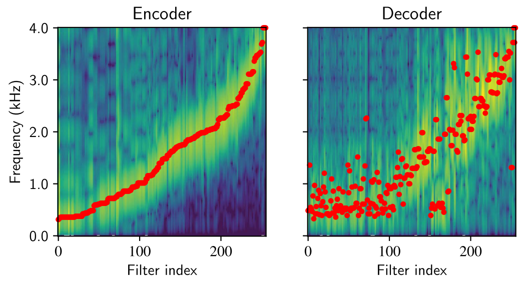

Fig. 11. Frequency response of the encoder and decoder filters. The red-dots correspond to the filter center frequencies.

VII. DISCUSSION

We study the processing of a speech mixture by the network, using a synthetic 2-speaker mixture with partial speaker overlap ($\approx 50\%$ overlap). This is to understand the attention schemes computed by the different transformer layers in the network. We note that the proposed network is trained and evaluated in Section V with fully overlapping speaker mixtures and partial overlap mixture is used here only for illustration purposes. The source signals are from the WSJ0 test set.

Waveform encoder-decoder: The order of the encoder filters as a function of frequency is random, since the network architecture or the training procedure do not encourage the encoder filters to be in sorted frequency order. Fig. 11 shows the frequency response of the encoder and decoder filters, sorted by the center frequencies of the encoder filters. We see that the learned center-frequencies are on a warped scale, as it is also observed in other works [3]. Contrary to the expectation, the

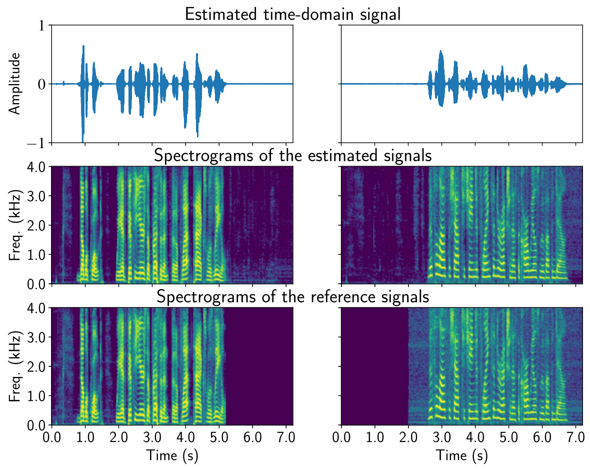

Fig. 12. Time domain signal, spectrograms of the estimated sources and their corresponding reference signals.

decoder filter at the same index as the encoder filter was not at the same frequency. The waveform encoder-decoder are not found to be perfect reconstruction type, i.e., passing the encoder output directly through the decoder does not reconstruct the original signal. However, there exists a mask to reconstruct the single-speaker signal, as we have seen in the results section for the single-speaker recordings.

We observed that the waveform-encoder output has 50% zeros on average. A closer look at the Conv1D filter weights showed that for several filters their negative version is also a filter, i.e., if $\mathbf{c}$ is a filter then $-\mathbf{c}$ is also a filter (hence 50% zeros). This shows that if the waveform-encoder output is computed as

$$\mathbf{X} = \text{ReLU}\left(\text{Concatenate}\left(\text{Conv1D}(\mathbf{x}), -\text{Conv1D}(\mathbf{x})\right)\right),$$

the number of encoder filters could possibly be halved. A further investigation is beyond the scope of this paper.

We also observed that some filters are repeated, i.e., have same weights, indicating a redundant representation. This repetition of filters at the same frequency may be helping with the separation of sources with similar characteristics (same pitch range), since the masks can be disjoint between speakers for features of the same frequency filter. For the single-speaker recordings, we observed that the obtained masks have 60% non-zeros on average, which confirms the redundant representation by the encoder.

Signal estimation: Fig. 12 shows the spectrograms of the estimated sources and their corresponding reference signals. For the example shown, there is no leakage across the speaker channels during voiced speech regions. However, the silence/non-speech regions of one channel may leak into the other in practice.

Intra-chunk attention: Fig. 13 shows the intra-chunk attention computed by the four transformer layers in DPB and TPB. We see that the attention-weights are concentrated around the main-diagonal and the spread around the diagonal increases with the layer index. This shows that the intra-chunk transformers attend to the current and neighboring frames only. The transformers accumulate temporal context through successive layers, similar to the dilated convolution blocks in ConvTasNet [3]. The spread around the diagonal is higher in the TPB for the two output channels, where the network's goal is to generate speaker specific outputs.

Inter-chunk attention: Fig. 14 shows the inter-chunk attention computed by the two transformer layers of DPB and TPB. In DPB (Fig. 14(a)), the first layer has higher attention weights for the active speech chunks and the second layer has higher weights for chunks with non-speech. The attention weights are spread over all the time frames, indicating that the representations computed in DPB are also global. The attention pattern for the first layer appears like a superposition of two similarity matrices, corresponding to two different, overlapping time regions. The different heads of the transformer may be focusing on different time regions. So, in practice, the overall capacity of the network, in terms of the number of sources it can separate, may be limited by the number of heads in the inter-chunk transformer layers.

A different behavior is observed in the inter-chunk transformers of the TPB (Fig. 14(b), (c)). The attention weights in these are concentrated along the diagonal, indicating the modeling of relationships in neighboring chunks, i.e., relationships spanning several hundred milliseconds. In each channel, the second transformer layer attention weights are concentrated along the diagonal, only in the active speech regions of that particular channel and they are more spread-out during the silence regions. This indicates speaker-selective short-time modeling in the inter-chunk layers of TPB.

Inter-channel attention: Fig. 15 shows the intra-chunk mean and standard deviation of cross-channel weights (anti-diagonal of the self attention matrix), across the chunks for the two transformer layers. We see that the weights are always greater than zero, even for chunks which have only one active speaker, showing that there is information sharing across the two output channels. In the first layer, the cross-channel weights are mostly less than 0.5, but greater than 0.5 for the second layer. This is shows that the first layer is giving higher importance to intra-channel features and the second-layer is fusing the information from the second channel. Fig. 15 also shows that the attention weights have a smaller variance with-in a chunk.

Attractors: The EDA module generates 3 attractors for this example with $J = 2$. The existence probabilities for the generated attractors, for the example shown in Fig. 12, were $\{1.0, 1.0, 4.8e-7\}$. The module is found to predict the speaker existence with a very high confidence. We found this observation to be true for most of the test set, indicating over-fitting to the training data conditions. This could be a reason for the poor SCA in situations where one or more signals in the mixture are loud compared to the other signals, different from the train/test conditions of WHAM!-WSJ0-Mix dataset. The attractors are found to be recording specific. Hence, they do not necessarily contain speaker information useful for speaker identification or re-identification of a given speaker across the different blocks if the proposed method is applied in a block-wise manner to process long recordings.

Gating mechanism: The output of the triple-path block, after overlap-add, goes through a gating scheme before predicting

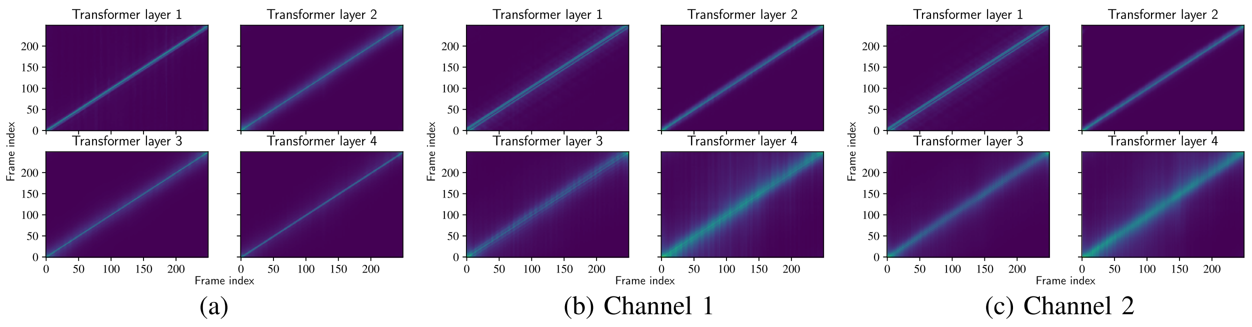

Fig. 13. Intra-chunk attention, averaged across the heads and the chunks, for the four transformer layers of (a) DPB, (b,c) two channels of TPB.

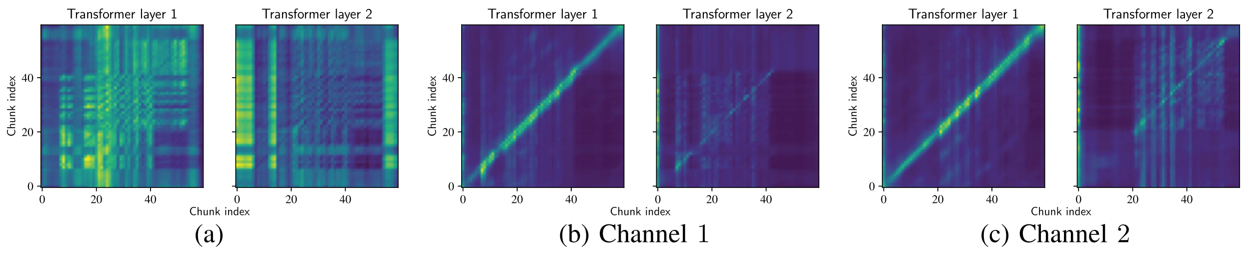

Fig. 14. Inter-chunk attention, averaged across the heads and with-in each chunk, for the two transformer layers of (a) DPB, (b), (c) two channels of TPB.

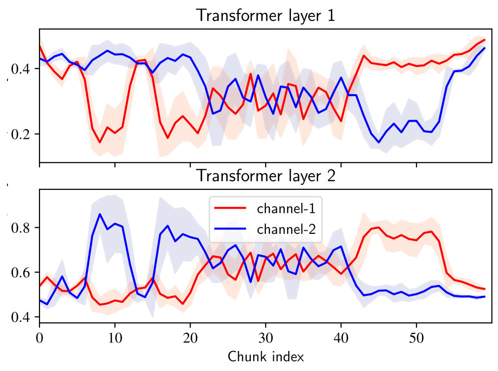

Fig. 15. Cross-channel attention weights (average across the heads and intra-chunk) in the inter-channel transformer block for the two transformer layers. The shaded region corresponds to the with-in chunk ± 1 standard deviation.

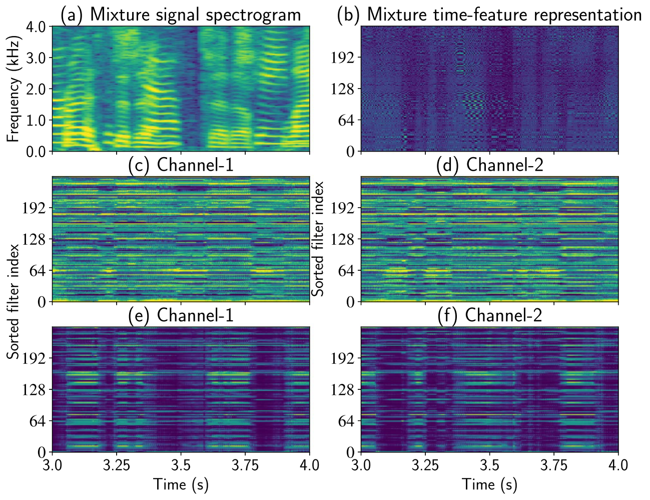

Fig. 16. A short segment of (a) Spectrogram and (b) waveform encoder output for the mixture signal. (c), (d) Linear-Tanh layer outputs and (e), (f) Linear-Sigmoid layer outputs for the two output channels.

the masks. Fig. 16 shows short segments of the Linear-Tanh and Linear-Sigmoid layer outputs, for the two speaker channels. We see that, while the output of the Linear-Tanh layer is dense, the Linear-Sigmoid layer output is sparse. It provides speaker-sensitive selection across the feature dimension and also suppresses the features during speech pauses. Finally, the matrices are multiplied element-wise and fed to a Linear-ReLU layer to predict the mask.

VIII. Ablation Experiments

In this section, we study the architecture and training choices. For this study, the architectures are trained on WSJ0-Mix dataset alone, i.e., 1-5 speaker mixtures and no-noise. Table VIII shows the SI-SNRi and SCA for $1-5$ speaker mixtures for different models. V0 shows the architecture trained with default parameters. Table III shows the same results, but the architecture was trained on 0-5 speaker mixtures (WHAM!-WSJ0-Mix dataset). The performance is better in Table VIII compared to Table III. The presence of noise-only samples and noise during training is found to deteriorate the speaker counting, especially for $J > 2$.

First, we study the impact of weight η used for $\mathcal{L}_{\text{attr}}$ in the training loss function (rows V0-V2). $\eta = 10$ for the V0 model. SCA decreases with a decrease in η, which also reflects in the SI-SNRi values, for 4 and 5 speaker mixtures. For 2 and 3 speaker mixtures, SI-SNRi is marginally better for smaller η.

TABLE VIII
ABLATION EXPERIMENTS

ID	Model	SI-SNRi (dB)					Speaker Counting Accuracy %				
		1	2	3	4	5	1	2	3	4	5
V0	Base Architecture	70.96	20.32	19.07	15.70	13.22	**100.00**	**99.97**	99.27	96.43	91.87
V1	EDA loss scale $\eta = 1$	61.15	20.50	**19.37**	15.55	12.79	99.83	99.93	**99.50**	94.83	89.43
V2	EDA loss scale $\eta = 0.1$	62.73	**20.53**	19.31	15.31	11.43	99.89	99.93	99.20	93.33	80.33
V3	Sequence averaging	59.18	20.35	19.11	**15.83**	**13.67**	**100.00**	99.83	99.20	96.03	93.77
V4	EDA-no shuffle	66.38	20.11	18.75	15.71	13.43	**100.00**	99.80	98.87	95.77	**94.40**
V5	interChannel - intraChunk - interChunk	70.28	20.00	18.57	15.38	13.18	99.89	99.93	99.43	**96.63**	94.37
V6	interChannel - interChunk - intraChunk	**71.92**	19.77	17.75	14.03	10.87	**100.00**	99.87	98.87	95.73	86.97
V7	intraChunk - interChannel - interChunk	50.24	14.38	11.11	6.69	1.92	99.89	99.83	96.03	84.90	61.20

Model V3 in the table shows the results for a simpler scheme of averaging the intra-chunk features instead of the attentive sequence aggregation scheme discussed in Section II. We see that the performance is affected for single-speaker recordings but similar or better for the mixture signals.

The chunk-level features are shuffled prior to EDA in row V0. Row V4 shows the results for the model trained without shuffling. We see that the performance with shuffling is slightly better compared to EDA without shuffling.

Next, we study the ordering of the three transformers in the triple-path block. V0 configuration has the layers in the order intraChunk-interChunk-interChannel. Architectures V5, V6 and V7 show the results for different ordering of the blocks. The configuration V7 has a very poor performance compared to the other four. Placement of inter-channel transformer at the beginning (V5) and at the end (V0) has similar performance, though V0 is slightly better. The configuration in V6 has a slightly poor performance compared to V0 and V5.

The experiments show that the performance is sensitive to the ordering of the transformer layers. But the other training parameters have a marginal impact on the test performance.

IX. Conclusion

We proposed a DNN architecture to jointly estimate the speaker count and the individual sources from a single channel speech mixture of an unknown number of speakers. The network is trained with noise-only signals (i.e., no speakers), single-speaker signals and mixtures of up to five speakers. While the network does not generalize to unknown number of speakers, it achieves more than 99% speaker counting accuracy for input signals with zero to three speakers. The SI-SNR for recordings of one to three speakers is more than 19 dB. Through robustness analysis, we showed that the network generalizes to low-reverberation conditions and a higher range of speaker mixing ratios than those observed during training. We also showed that the network operates by building short-time, medium-time and global file-level representations at different blocks. Through the analysis, we provided insights for the design of compute efficient transformer architectures for source separation, for example, masked attention transformers can be used for all the intra-chunk transformers with a limited temporal context for each time frame.

Acknowledgment

The authors thank the Erlangen Regional Computing Center (RRZE) for providing computing resources and support. The authors would like to acknowledge the useful discussions with Edwin Mabande, Oliver Thiergart, and Soumitro Chakrabarty of the Communication Acoustics Group at Fraunhofer IIS, Erlangen, Germany.

References

[1] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 12, pp. 2136–2147, Dec. 2015.

[2] Y. Isik, J. L. Roux, Z. Chen, S. Watanabe, and J. R. Hershey, "Single-channel multi-speaker separation using deep clustering," in *Proc. Interspeech*, 2016, pp. 545–549.

[3] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[4] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: Efficient long sequence modeling for time-domain single-channel speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 46–50.

[5] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, "Attention is all you need in speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 21–25.

[6] J. Chen, Q. Mao, and D. Liu, "Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation," in *Proc, Interspeech*, 2020, pp. 2642–2646.

[7] Z. Zhang, B. He, and Z. Zhang, "TransMask: A compact and fast speech separation model based on transformer," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 5764–5768.

[8] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 2840–2849, 2021.

[9] K. Kinoshita, L. Drude, M. Delcroix, and T. Nakatani, "Listening to each speaker one by one with recurrent selective hearing networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 5064–5068.

[10] N. Takahashi, S. Parthasaarathy, N. Goswami, and Y. Mitsufuji, "Recursive speech separation for unknown number of speakers," in *Proc. Interspeech*, 2019, pp. 1348–1352.

[11] J. Zhu, R. A. Yeh, and M. Hasegawa-Johnson, "Multi-decoder DPRNN: Source separation for variable number of speakers," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 3420–3424.

[12] E. Nachmani, Y. Adi, and L. Wolf, "Voice separation with an unknown number of multiple speakers," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 7164–7175.

[13] S. R. Chetupalli and E. A. P. Habets, "Speech separation for an unknown number of speakers using transformers with encoder-decoder attractors," in *Proc. Interspeech*, 2022, pp. 5393–539.

[14] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and K. Nagamatsu, "End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors," in *Proc. Interspeech*, 2020, pp. 269–273.

[15] G. Wichern et al., “WHAM!: Extending speech separation to noisy environments,” in *Proc. Interspeech*, 2019, pp. 1368–1372.

[16] M. Maciejewski, G. Wichern, and J. L. Roux, “WHAMR!: Noisy and reverberant single-channel speech separation,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 696–700.

[17] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, “Joint separation and denoising of noisy multi-talker speech using recurrent neural networks and permutation invariant training,” in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2017, pp. 1–6.

[18] H. Shi, X. Chen, T. Kong, S. Yin, and P. Ouyang, “GLMSnet: Single channel speech separation framework in noisy and reverberant environments,” in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2021, pp. 663–670.

[19] S. E. Chazan, L. Wolf, E. Nachmani, and Y. Adi, “Single channel voice separation for unknown number of speakers under reverberant and noisy settings,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 3730–3734.

[20] J. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” in *Proc. Adv. NIPS Deep Learn. Symp.*, 2016, *arXiv:1607.06450*.

[21] A. Vaswani et al., “Attention is all you need,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, 2017, pp. 6000–6010.

[22] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, “Self-attentive speaker embeddings for Text-Independent speaker verification,” in *Proc. Interspeech*, 2018, pp. 3573–3577.

[23] A. Pandey, B. Xu, A. Kumar, J. Donley, P. Calamia, and D. Wang, “TPARN: Triple-path attentive recurrent network for time-domain multichannel speech enhancement,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 6497–6501.

[24] M. Kolbaek, D. Yu, Z.-H. Tan, and J. Jensen, “Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 10, pp. 1901–1913, Oct. 2017.

[25] Y. Luo and N. Mesgarani, “TaSNet: Time-domain audio separation network for real-time, single-channel speech separation,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 696–700.

[26] J. S. Garofolo et al., “CSR-I (WSJ0) complete LDC93S6A,” Web download, Philadelphia: Linguistic Data Consortium, 1993.

[27] M. Ravanelli et al., “SpeechBrain: A general-purpose speech toolkit,” 2021, *arXiv:2106.04624*.

[28] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic gradient descent,” in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[29] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, “Audio augmentation for speech recognition,” in *Proc. Interspeech*, 2015, pp. 3586–3589.

[30] E. Vincent, R. Gribonval, and C. Fevotte, “Performance measurement in blind audio source separation,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 4, pp. 1462–1469, Jul. 2006.

[31] C. Raffel et al., “MIR_EVAL: A transparent implementation of common MIR metrics,” in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2014, pp. 367–372.

[32] Y. Jadoul, B. Thompson, and B. d. Boer, “Introducing parselmouth: A python interface to praat,” *J. Phonetics*, vol. 71, pp. 1–15, 2018.

[33] R. Scheibler, E. Bezzam, and I. Dokmanic, “Pyroomacoustics: A python package for audio room simulation and array processing algorithms,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 351–355.