

Compute and memory efficient universal sound source separation

Efthymios Tzinis · Zhepei Wang ·
Xilin Jiang · Paris Smaragdis

Received: date / Accepted: date

Abstract Recent progress in audio source separation led by deep learning has enabled many neural network models to provide robust solutions to this fundamental estimation problem. In this study, we provide a family of efficient neural network architectures for general purpose audio source separation while focusing on multiple computational aspects that hinder the application of neural networks in real-world scenarios. The backbone structure of this convolutional network is the SUccessive DOwnsampling and Resampling of Multi-Resolution Features (SuDoRM-RF) as well as their aggregation which is performed through simple one-dimensional convolutions. This mechanism enables our models to obtain high fidelity signal separation in a wide variety of settings where a variable number of sources are present and with limited computational resources (e.g. floating point operations, memory footprint, number of parameters and latency). Our experiments show that SuDoRM-RF models perform comparably and even surpass several state-of-the-art benchmarks with significantly higher computational resource requirements. The causal variation of SuDoRM-RF is able to obtain competitive performance in real-time speech separation of around 10dB scale-invariant signal-to-distortion ratio improvement (SI-SDRi) while remaining up to 20 times faster than real-time on a laptop device.

Keywords Audio source separation · low-cost neural networks · deep learning · real-time processing

Efthymios Tzinis
University of Illinois at Urbana-Champaign
E-mail: etzinis2@illinois.edu

Zhepei Wang
University of Illinois at Urbana-Champaign

Xilin Jiang
University of Illinois at Urbana-Champaign

Paris Smaragdis
University of Illinois at Urbana-Champaign & Adobe Research

1 Introduction

The advent of the deep learning era has enabled the effective usage of neural networks towards single-channel source separation with mask-based architectures [12]. Recently, end-to-end source separation in time-domain has shown state-of-the-art results in a variety of separation tasks such as speech separation [24, 22], universal sound separation [15, 37] and music source separation [5]. The separation module of ConvTasNet [24] and its variants [15, 37] consist of multiple stacked layers of depth-wise separable convolutions [33] which can aptly incorporate long-term temporal relationships. Building upon the effectiveness of a large temporal receptive field, a dual-path recurrent neural network (DPRNN) [22] has shown remarkable performance on speech separation. Demucs [5] has a refined U-Net structure [31] and has shown strong performance improvement on music source separation. Specifically, it consists of several convolutional layers in each a downsampling operation is performed in order to extract high dimensional features. A two-step approach has been introduced in [36] and showed that universal sound separation models could be further improved when working directly on the latent space and learning the ideal masks on a separate step.

Despite the dramatic advances in source separation performance, the computational complexity of the aforementioned methods might hinder their extensive usage across multiple devices. Specifically, many of these algorithms are not amenable to, e.g., embedded systems deployment, or other environments where computational resources are constrained. Additionally, training such systems is also an expensive computational undertaking which can amount to significant costs.

Several studies, mainly in the image domain, have introduced more efficient architectures in order to overcome the growing concern of large models with high computational requirements. Models with depth-wise separable convolutions [33] have shown strong potential for several image-domain tasks [4] while significantly reducing the computational requirements. Thus, several variants such as MobileNets [11] have been proposed for deep learning on edge devices. However, convolutions with a large dilation factor might inject several artifacts and thus, lightweight architectures that combine several dilation factors in each block have been proposed for image tasks [26]. More recent studies propose meta-learning algorithms for optimizing architecture configurations given specific computational resource and accuracy requirements [42, 3].

Despite the recent success of low-resource architectures in the image domain, little progress has been made towards proposing efficient architectures for audio tasks and especially source separation. In [13] a WaveRNN is used for efficient audio synthesis in terms of floating point operations (FLOPs) and latency. Other studies have introduced audio source separation models with reduced number of trainable parameters [22, 25, 23] and binarized models [16]. Modern approaches, mainly in speech enhancement and music source separation, have been focusing on developing models which are capable of real-time inference. Specifically, a temporal convolutional network for real time speech

enhancement has been proposed in [27] while the latest state-of-the-art performance has been obtained by a real-time variation of Demucs for online speech denoising [6]. In a similar sense real-time music source separation models have been proposed in [9] and a system capable of real-time speech separation from background music has been implemented in [14].

In this study, we propose a novel efficient neural network architecture for audio source separation while following a more holistic approach in terms of computational resources that we take into consideration (FLOPs, latency and total memory requirements). Our proposed model performs SUccessive DOWn-sampling and Resampling of Multi-Resolution Features (SuDoRM-RF) using depth-wise convolutions. By doing so, SuDoRM-RF exploits the effectiveness of iterative temporal resampling strategies [7] and avoids the need for multiple stacked dilated convolutional layers [24]. We also propose improved versions of the aforementioned architecture with significant benefits in terms of computational resource requirements as well as causal variations where online inference is available. We report a separation performance comparable to or even better than several recent state-of-the-art models on speech, environmental and universal sound separation tasks with significantly lower computational requirements. Our experiments suggest that SuDoRM-RF models a) could be deployed on devices with limited resources, b) be trained significantly faster and achieve good separation performance and c) scale well when increasing the number of parameters. Our code is available online ¹.

2 Sudo rm -rf network architecture

On par with many state-of-the-art approaches in the literature [24, 36, 22, 5], SuDoRM-RF performs end-to-end audio source separation using a mask-based architecture with adaptive encoder and decoder basis. We have extended our basic model in order to also remove the mask estimation process by introducing SuDoRM-RF++ that directly estimates the latent representations of the sources in the adaptive front-end domain. First, we describe all the modules which are needed for both architectures and describe extensively the inference path for our basic SuDoRM-RF architecture. In Figure 1, both architectures are shown. Consequently we also present the extensions of our original SuDoRM-RF model including: its improved version SuDoRM-RF++ (Section 2.4), a variant of SuDoRM-RF++ including group communication [23] (Section 2.5) as well as its causal variation C-SuDoRM-RF++ (Section 2.6).

The input is the raw signal from a mixture $\mathbf{x} \in \mathbb{R}^T$ with T samples in the time-domain. First, we feed the input mixture \mathbf{x} to an encoder \mathcal{E} in order to obtain a latent representation for the mixture $\mathbf{v}_{\mathbf{x}} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{C_{\mathcal{E}} \times L}$. Consequently the latent mixture representation is fed through a separation module \mathcal{S} which estimates the corresponding masks $\hat{\mathbf{m}}_i \in \mathbb{R}^{C_{\mathcal{E}} \times L}$ for each one of the N sources $\mathbf{s}_1, \dots, \mathbf{s}_N \in \mathbb{R}^T$ which constitute in the mixture. The estimated latent representation for each source in the latent space $\hat{\mathbf{v}}_i$ is retrieved

¹ Code: https://github.com/etzinis/sudo_rm_rf

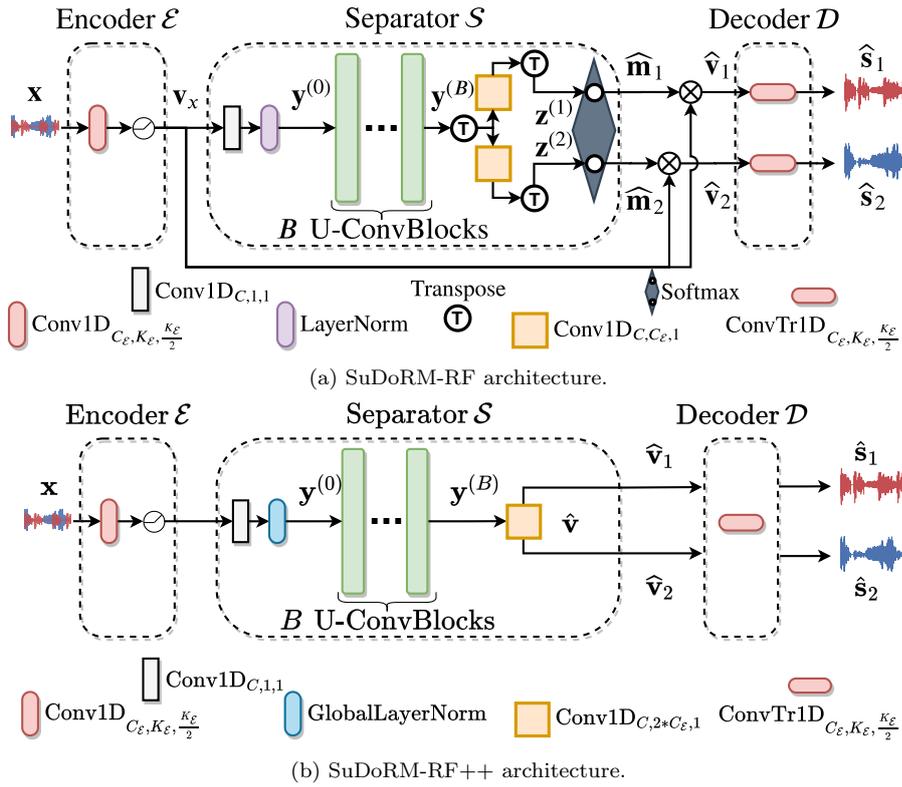


Fig. 1: SuDoRM-RF and SuDoRM-RF++ architectures for separating two sources.

by multiplying element-wise an estimated mask $\hat{\mathbf{m}}_i$ with the encoded mixture representation \mathbf{v}_x . Finally, the reconstruction for each source $\hat{\mathbf{s}}_i$ is obtained by using a decoder \mathcal{D} to transform the latent-space $\hat{\mathbf{v}}_i$ source estimates back into the time-domain $\hat{\mathbf{s}}_i = \mathcal{D}(\hat{\mathbf{v}}_i)$. An overview of the SuDoRM-RF architecture is displayed in Figure 1a. The encoder, separator and decoder modules are described in Sections 2.1, 2.2 and 2.3, respectively. For simplicity of our notation we will describe the whole architecture assuming that the processed batch size is one. Moreover, we are going to define some useful operators of the various convolutions which are used in SuDoRM-RF.

Definition 1 $\text{Conv1D}_{C, K, S} : \mathbb{R}^{C_{in} \times L_{in}} \rightarrow \mathbb{R}^{C \times L}$ defines a kernel $\mathbf{W} \in \mathbb{R}^{C \times C_{in} \times K}$ and a bias vector $\mathbf{b} \in \mathbb{R}^C$. When applied on a given input $\mathbf{x} \in \mathbb{R}^{C_{in} \times L_{in}}$ it performs a one-dimensional convolution operation with stride equal to S as shown next:

$$\text{Conv1D}_{C, K, S}(\mathbf{x})_{i, l} = \mathbf{b}_i + \sum_{j=1}^{C_{in}} \sum_{k=1}^K \mathbf{W}_{i, j, k} \cdot \mathbf{x}_{j, S \cdot l - k}, \quad (1)$$

where the indices i, j, k, l denotes the output channel, the input channel, the kernel sample and, the temporal index, respectively. Note that without loss of generality and performing appropriate padding, the last dimension of the output representation would be $L = \lfloor L_{in}/S \rfloor$.

Definition 2 $\text{ConvTr1D}_{C,K,S} : \mathbb{R}^{C_{in} \times L_{in}} \rightarrow \mathbb{R}^{C \times L}$ defines a one-dimensional transpose convolution. Since any convolution operation could be expressed as a matrix multiplication, transposed convolution can be directly understood as the gradient calculation for a regular convolution w.r.t. its input [34].

Definition 3 $\text{DWConv1D}_{C,K,S} : \mathbb{R}^{C_{in} \times L_{in}} \rightarrow \mathbb{R}^{C \times L}$ defines a one-dimensional depth-wise convolution operation [33]. In essence, this operator defines $G = C_{in}$ separate one-dimensional convolutions $\mathcal{F}_i = [\text{Conv1D}_{C_G, K, S}]_i$ with $i \in \{1, \dots, G\}$ where $C_G = \lfloor C/G \rfloor$. Given an input $\mathbf{x} \in \mathbb{R}^{C_{in} \times L_{in}}$ the i th one-dimensional convolution contributes to $C_G = \lfloor C/G \rfloor$ output channels by considering as input only the i th row of the input as described below:

$$\text{DWConv1D}_{C,K,S}(\mathbf{x}) = \text{Concat}(\{\mathcal{F}_i(\mathbf{x}_i), \forall i\}), \quad (2)$$

where $\text{Concat}(\cdot)$ performs the concatenation of all individual one-dimensional convolution outputs across the channel dimension.

2.1 Encoder

The encoder \mathcal{E} architecture consists of a one-dimensional convolution with kernel size $K_{\mathcal{E}}$ and stride equal to $K_{\mathcal{E}}/2$ similar to [24]. Each convolved input audio segment of $K_{\mathcal{E}}$ samples is transformed to a $C_{\mathcal{E}}$ -dimensional vector representation where $C_{\mathcal{E}}$ is the number of output channels of the 1D-convolution. We force the output of the encoder to be strictly non-negative by applying a rectified linear unit (ReLU) activation on top of the output of the 1D-convolution. Thus, the encoded input mixture representation could be expressed as:

$$\mathbf{v}_{\mathbf{x}} = \mathcal{E}(\mathbf{x}) = \text{ReLU}(\text{Conv1D}_{C_{\mathcal{E}}, K_{\mathcal{E}}, K_{\mathcal{E}}/2}(\mathbf{x})) \in \mathbb{R}^{C_{\mathcal{E}} \times L}, \quad (3)$$

where the activation $\text{ReLU}(\cdot)$ is applied element-wise.

2.2 Separator

In essence, the separator \mathcal{S} module performs the following transformations to the encoded mixture representation $\mathbf{v}_{\mathbf{x}} \in \mathbb{R}^{C_{\mathcal{E}} \times L}$:

1. Projects the encoded mixture representation $\mathbf{v}_{\mathbf{x}} \in \mathbb{R}^{C_{\mathcal{E}} \times L}$ to a new channel space through a layer-normalization (LN) [1] followed by a point-wise convolution as shown next:

$$\mathbf{y}_0 = \text{Conv1D}_{C,1,1}(\text{LN}(\mathbf{v}_{\mathbf{x}})) \in \mathbb{R}^{C \times L}, \quad (4)$$

where $\text{LN}(\mathbf{v}_{\mathbf{x}})$ denotes a layer-normalization layer in which the moments used are extracted across the temporal dimension for each channel separately.

2. Performs repetitive non-linear transformations provided by B U-convolutional blocks (U-ConvBlocks) on the intermediate representation \mathbf{y}_0 . In other words, the output of the i th U-ConvBlock would be denoted as $\mathbf{y}_i \in \mathbb{R}^{C \times L}$ and would be used as input for the $(i + 1)$ th block. Each U-ConvBlock extracts and aggregates information from multiple resolutions which is extensively described in Section 2.2.1.
3. Aggregates the information over multiple channels by applying a regular one-dimensional convolution for each source on the transposed feature representation $\mathbf{y}_B^T \in \mathbb{R}^{L \times C}$. Effectively, for the i th source we obtain an intermediate latent representation as shown next:

$$\mathbf{z}_i = \text{Conv1D}_{C, C_\varepsilon, 1} (\mathbf{y}_B^T)^T \in \mathbb{R}^{C_\varepsilon \times L} \quad (5)$$

This step has been introduced in [36] and empirically shown to make the training process more stable rather than using the activations from the final block \mathbf{y}_B to estimate the masks.

4. Combines the aforementioned latent codes for all sources $\mathbf{z}_i \forall i \in \{1, \dots, N\}$ by performing a softmax operation in order to get mask estimates $\hat{\mathbf{m}}_i \in [0, 1]^{C_\varepsilon \times L}$ which add up to one across the dimension of the sources. Namely, the corresponding mask estimate for the i th source would be:

$$\hat{\mathbf{m}}_i = \text{vec}^{-1} \left(\frac{\exp(\text{vec}(\mathbf{z}_i))}{\sum_{j=1}^N \exp(\text{vec}(\mathbf{z}_j))} \right) \in \mathbb{R}^{C_\varepsilon \times L}, \quad (6)$$

where $\text{vec}(\cdot) : \mathbb{R}^{K \times N} \rightarrow \mathbb{R}^{K \cdot N}$ and $\text{vec}^{-1}(\cdot) : \mathbb{R}^{K \cdot N} \rightarrow \mathbb{R}^{K \times N}$ denotes the vectorization of an input tensor and the inverse operation, respectively.

5. Estimates a latent representation $\hat{\mathbf{v}}_i \in \mathbb{R}^{C_\varepsilon \times L}$ for each source by multiplying element-wise the encoded mixture representation $\mathbf{v}_\mathbf{x}$ with the corresponding mask $\hat{\mathbf{m}}_i$:

$$\hat{\mathbf{v}}_i = \mathbf{v}_\mathbf{x} \odot \hat{\mathbf{m}}_i \in \mathbb{R}^{C_\varepsilon \times L}, \quad (7)$$

where $\mathbf{a} \odot \mathbf{b}$ is the element-wise multiplication of the two tensors \mathbf{a} and \mathbf{b} assuming that they have the same shape.

2.2.1 U-convolutional block (U-ConvBlock)

U-ConvBlock uses a block structure which resembles a depth-wise separable convolution [33] with a skip connection as in ConvTasNet [24]. However, instead of performing a regular depth-wise convolution as shown in [4] or a dilated depth-wise which has been successfully utilized for source separation [24, 37, 36] our proposed U-ConvBlock extracts information from multiple resolutions using Q successive temporal downsampling and Q upsampling operations similar to a U-Net architecture [31]. More importantly, the output of each block leaves the temporal resolution intact while increasing the effective receptive field of the network multiplicatively with each temporal sub-sampling operation [21]. We postulate that this whole resampling procedure of extracting

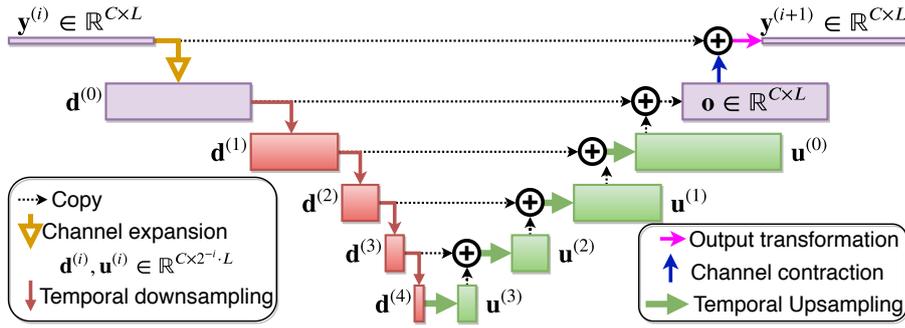


Fig. 2: U-ConvBlock architecture.

Algorithm 1: U-ConvBlock forward pass

```

Input:  $\mathbf{y}^{(i)} \in \mathbb{R}^{C \times L}$ 
Output:  $\mathbf{y}^{(i+1)} \in \mathbb{R}^{C \times L}$ 
// Expand channel dimensions
 $\mathbf{q} \leftarrow \text{PReLU}_{C_U} (\text{LN} (\text{Conv1D}_{C_U, 1, 1} (\mathbf{y}^{(i)})))$ ;
 $\mathbf{d}^{(0)} \leftarrow \text{PReLU}_{C_U} (\text{LN} (\text{DWConv1D}_{C_U, K_U, 1} (\mathbf{q})))$ ;
for  $i = 1; i++$ ; while  $i \leq Q$  do
    // Successive depth-wise downsampling
     $\mathbf{d}^{(i)} \leftarrow \text{LN} (\text{DWConv1D}_{C_U, K_U, S_U} (\mathbf{d}^{(i-1)}))$ ;
     $\mathbf{d}^{(i)} \leftarrow \text{PReLU}_{C_U} (\mathbf{d}^{(i)})$ ;
end
 $\mathbf{u}^{(Q)} \leftarrow \mathbf{d}^{(Q)}$ ;
for  $i = Q - 1; i--$ ; while  $i \geq 0$  do
    // Upsampling and adding multi-resolution features
     $\mathbf{u}^{(i)} \leftarrow \mathbf{d}^{(i)} + \mathcal{I}_{S_U} (\mathbf{u}^{(i+1)})$ ;
end
 $\mathbf{o} \leftarrow \text{LN} (\text{Conv1D}_{C, 1, 1} (\text{PReLU}_C (\text{LN} (\mathbf{u}^{(0)}))))$ ;
return  $\text{PReLU}_C (\mathbf{y}^{(i)} + \mathbf{o})$ ;
    
```

features at multiple scales combined with the efficient increase of the effective receptive field enables SuDoRM-RF models to outperform several convolutional architectures and perform in par with much more expensive recurrent and self-attention architectures [35]. An abstract view of the i th U-ConvBlock is displayed in Figure 2 while a detailed description of the operations is presented in Algorithm 1.

Definition 4 $\text{PReLU}_C : \mathbb{R}^{C \times L} \rightarrow \mathbb{R}^{C \times L}$ defines a parametric rectified linear unit (PReLU) [8] with C learnable parameters $\mathbf{a} \in \mathbb{R}^C$. When applied to an input matrix $\mathbf{y} \in \mathbb{R}^{C \times L}$ the non-linear transformation could be defined element-wise as:

$$\text{PReLU}_C (\mathbf{y})_{i,j} = \max (0, \mathbf{y}_{i,j}) + \mathbf{a}_i \cdot \min (0, \mathbf{y}_{i,j}) \quad (8)$$

Definition 5 $\mathcal{I}_M : \mathbb{R}^{C \times L} \rightarrow \mathbb{R}^{C \times M \cdot L}$ defines a nearest neighbor temporal interpolation by a factor of M . When applied on an input matrix $\mathbf{y} \in$

$\mathbb{R}^{C \times L}$ this upsampling procedure could be formally expressed element-wise as: $\mathcal{I}_M(\mathbf{u})_{i,j} = \mathbf{u}_{i, \lfloor j/M \rfloor}$

Definition 6 $\text{LN} : \mathbb{R}^{C \times L} \rightarrow \mathbb{R}^{C \times L}$ defines a parametric normalization layer [1] with learnable parameters $\gamma \in \mathbb{R}^C$ and $\beta \in \mathbb{R}^C$. When applied to an input matrix $\mathbf{y} \in \mathbb{R}^{C \times L}$ the normalization could be defined element-wise as:

$$\text{LN}(\mathbf{y})_{i,j} = \frac{\mathbf{y}_{i,j} - \mu_i}{\sigma_i} \gamma_i + \beta_i, \quad \mu_i = \sum_j \mathbf{y}_{i,j}, \quad \sigma_i = \sqrt{\sum_j (\mathbf{y}_{i,j} - \mu_i)^2} \quad (9)$$

Definition 7 $\text{GLN} : \mathbb{R}^{C \times L} \rightarrow \mathbb{R}^{C \times L}$ defines a parametric normalization layer with learnable parameters $\gamma \in \mathbb{R}^C$ and $\beta \in \mathbb{R}^C$. When applied to an input matrix $\mathbf{y} \in \mathbb{R}^{C \times L}$ the normalization could be defined element-wise as:

$$\text{GLN}(\mathbf{y})_{i,j} = \frac{\mathbf{y}_{i,j} - \mu}{\sigma} \gamma_i + \beta_i, \quad \mu = \sum_{i,j} \mathbf{y}_{i,j}, \quad \sigma = \sqrt{\sum_{i,j} (\mathbf{y}_{i,j} - \mu)^2} \quad (10)$$

2.3 Decoder

Our decoder module \mathcal{D} is the final step in order to transform the latent space representation $\hat{\mathbf{v}}_i$ for each source back to the time domain. In our proposed model we follow a similar approach as in [36] where each latent source representation $\hat{\mathbf{v}}_i$ is fed through a different transposed convolution decoder $\text{ConvTr1D}_{C_\varepsilon, K_\varepsilon, \kappa_\varepsilon/2}$. The efficacy of dealing with different types of sources using multiple decoders has also been studied in [2]. Ignoring the permutation problem, for the i th source we have the following reconstruction in time:

$$\hat{\mathbf{s}}_i = \mathcal{D}_i(\hat{\mathbf{v}}_i) = \text{ConvTr1D}_{C_\varepsilon, K_\varepsilon, \kappa_\varepsilon/2}(\hat{\mathbf{v}}_i) \quad (11)$$

2.4 Improved version with no mask estimation SuDoRM-RF++

In the improved version of the proposed architecture, namely, SuDoRM-RF++, the model estimates directly the latent representation for each target signal $\hat{\mathbf{v}}_i \in \mathbb{R}^{C_\varepsilon \times L}$ and uses only one decoder module. Our intuition lies in the aspect that a highly parameterized neural network could potentially estimate those targets without the need of the hard-regularized element-wise multiplication process of the masks on top of the mixture encoded representation $\mathbf{v}_\mathbf{x} \in \mathbb{R}^{C_\varepsilon \times L}$. Essentially, SuDoRM-RF++, which is presented in Figure 1b, can be derived from our initial SuDoRM-RF model by applying the following alternations to the architecture:

- We replace the mask estimation and element-wise multiplication process with a direct estimation of the latent target signals $\hat{\mathbf{v}}_i$ after the final output of the model $\mathbf{y}^{(B)}$. We have validated experimentally that by removing the mask estimation layer leads to similar or slightly improved results.

- We use only one trainable decoder in order to transform the latent representation back to the time domain instead of two separate ones, namely, $\hat{\mathbf{s}}_i = \mathcal{D}(\hat{\mathbf{v}}_i) = \text{ConvTr1D}_{C, K_\varepsilon, K_\varepsilon, K_\varepsilon/2}(\hat{\mathbf{v}}_i)$.
- We replace the layer normalization layers (Equation 9) with global layer normalization (GLN) layers as defined in (Equation 10). This change significantly improves the convergence of our models probably because of the interdependence of the gradient statistics between the channels.
- For each intermediate representation with C channels, we simplify the activation layers and we use PReLU activation layers with only one learnable parameter instead of C as it was initially defined in Equation 8. In this way, we are able to achieve similar results as before with less parameters.

We would like to underline that the structure of the initial SuDoRM-RF models could potentially outperform the improved SuDoRM-RF++ variation in cases where the direct estimation of the latent targets would be more difficult than estimating the masks (e.g. unconstrained optimization in those latent spaces might be worse than estimating a bounded mask with values in the $[0, 1]$ region). Moreover, the alternation of containing two decoders proposed in the initial version might be more useful in cases where one wants to solve an audio source separation problem containing two distinct classes of sounds (e.g. speech enhancement) where each decoder could be fine-tuned towards decoding the class-specific characteristics of each estimated latent representation.

2.5 Group communication variation

We also propose a new variation of our model, namely, SuDoRM-RF++ GC, where we combine group communication (GC) with our improved version of our model SuDoRM-RF++. GC is a novel way to significantly reduce the parameters of an audio processing network which has been recently proposed in [23]. In the proposed architecture, the intermediate representations are being processed in groups of sub-bands of channels. We divide the channels of each 1×1 convolutional block into 16 groups and we process them first independently by sharing the parameters across all groups of sub-bands. At a second step, we apply a self-attention module [32] to combine them. The resulting architecture leads to a significant improvement in the number of trainable parameters which is mainly dominated by the bottleneck dense layers.

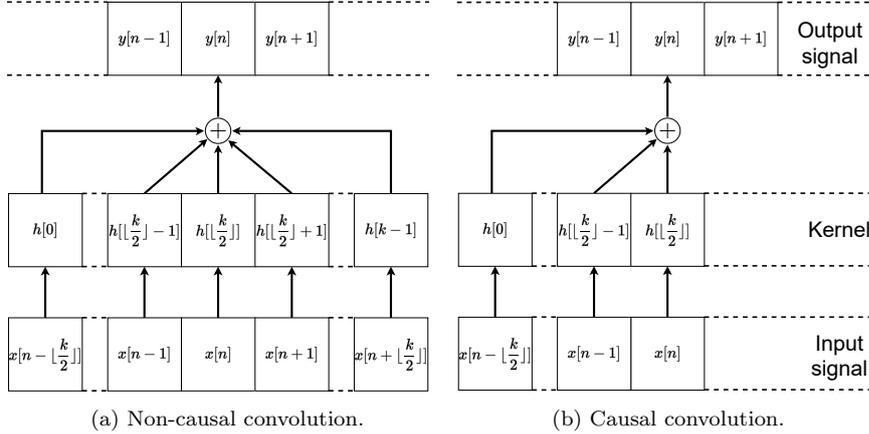
2.6 Causal version C-SuDoRM-RF++

Our latest extension of the proposed model is to be able to run online and enable streamable extensions for real-time applications. To this end, we propose C-SuDoRM-RF++ which is a more shallow and efficient model which has the following differences against the improved model SuDoRM-RF++ .

- We replace all non-causal convolutions with causal counterparts. Now the architecture does not depend on future samples in order to produce the

estimated signal up to the current time-frame. A depiction of those two convolutional modules is presented in Figures 3a and 3b for the non-causal and the causal convolutional layers, respectively.

- In order to simplify the implementation and make those models more efficient in terms of memory footprint and execution time, we also remove all the normalization layers.



3 Experimental Setup

3.1 Audio source separation tasks

Speech separation (2 active speakers): We perform speech separation experiments using the publicly available WSJ0-2mix dataset [10] by following a similar setup with other studies [22, 43, 20]. Speaker mixtures are generated by randomly mixing speech utterances with two active speakers at random signal to noise ratios (SNR)s between -5 and 5 dB from the Wall Street Journal (WSJ0) corpus [29].

Non-speech sound separation (2 active sources): For our non-speech sound separation experiments we follow the exact same setup as in [36] and utilize audio clips from the environmental sound classification (ESC50) data collection [30] which consists of a wide variety of sounds (non-speech human sounds, animal sounds, natural soundscapes, interior sounds and urban noises). For each data sample, two audio sources are mixed with a random SNR between -2.5 and 2.5 dB where each source belongs to a distinct sound category from a total of 50.

Universal sound separation (variable number of sources 1-4): We also evaluate our models under a purely universal sound separation setup where multiple sound classes might be present and also we do not know how many sources are active in each input mixture. To that end, we use the FUSS benchmark dataset presented in [39]. FUSS contains sound clips that might contain at least one and up to four active sources per input mixture. Moreover, the sound clips represent a wide variety of real-world sounds including (speech, engine sounds, music, wind, rain, and many others). Also, the SNR distribution of the input sound mixtures is more realistic by capturing a wide range approximately from -40dB to 40dB .

3.2 Data pre-processing and generation

We follow the same data augmentation process which was firstly introduced in [36] and it has been shown beneficial in other recent studies [43]. We also normalize all processed audio clips by subtracting their mean and divide with their standard deviation.

3.2.1 Fixed number of sources

The process for generating a training mixture is the following: A) random choosing two sound classes (for non-speech sound separation) or speakers (for speech separation) B) random cropping of 4sec segments from two sources audio files C) mixing the source segments with a random SNR (as specified in Section 3.1). For each epoch, 20,000 new training mixtures are generated. Validation and test sets are generated once with each one containing 3,000 mixtures. We also downsample each audio clip to 8kHz.

3.2.2 Variable number of sources

In order to be consistent with the state-of-the-art results on FUSS, we use the same dataset splits as the ones provided in [39]. The augmentation pipeline for each training mixture includes mixing sources from different training samples by sampling them uniformly over the batch. For each epoch, 20,000 new training mixtures are generated. Validation and test sets contain 5,000 and 3,000 mixtures, respectively. Moreover, we also train and test keeping the same length of 10secs at all clips as well as their sampling frequency which is 16kHz.

3.3 Training details

3.3.1 Fixed number of sources

All models are trained for 120 epochs using a batch size equal to 4. As a loss function we use the negative permutation-invariant [41] scale-invariant signal

to distortion ratio (SI-SDR) [19]. The total loss for N sources is computed as the average loss across each source as follows: For the i th source we define the loss between the clean signal \mathbf{s} and the estimates $\hat{\mathbf{s}}$ as:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \text{SI-SDR}(\mathbf{s}_i^*, \hat{\mathbf{s}}_i) = -\frac{1}{N} \sum_{i=1}^N 10 \log_{10} \left(\frac{\|\alpha_i \mathbf{s}_i^*\|^2}{\|\alpha_i \mathbf{s}_i^* - \hat{\mathbf{s}}_i\|^2} \right), \quad (12)$$

where \mathbf{s}^* denotes the permutation of the sources that maximizes SI-SDR and $\alpha_i = \hat{\mathbf{s}}_i^\top \mathbf{s}_i^* / \|\mathbf{s}_i^*\|^2$ is a scalar used for making the loss invariant to the scale of the i th estimated source $\hat{\mathbf{s}}_i$. During training, we use the Adam optimizer [17] with an initial learning rate set to 0.001 and we decrease it by a factor of 5 every 50 epochs. By training the model for more epochs, using the same learning rate scheduler, does not yield any significant gains on the validation set.

3.4 Variable number of sources

All models are trained for 40 epochs using a batch size equal to 4. In this case we assume that we know the maximum number of sources in each input mixture \mathbf{x} , e.g. N , but in reality the mixture might be comprised of only $N' < N$ active sources and $N - N'$ inactive sources. We follow a similar training procedure with the one presented in [39] and we force the network to produce zero outputs for the inactive slots after inferring the permutation that maximizes the total SNR. The loss for the active sources and inactive sources is defined in a permutation invariant sense as follows:

$$\mathcal{L} = \min_{\pi \in \Pi} \left[-\frac{1}{N'} \sum_{i=1}^{N'} 10 \log_{10} \left(\frac{\|\mathbf{s}_i\|^2 + \epsilon}{\|\mathbf{s}_i - \hat{\mathbf{s}}_i^{(\pi)}\|^2 + \epsilon} \right) + \frac{1}{N - N'} \sum_{i=N'+1}^N 10 \log_{10} \left(\|\hat{\mathbf{s}}_i^{(\pi)}\|^2 + \tau \|\mathbf{x}\|^2 + \epsilon \right) \right], \quad (13)$$

where Π is the set of all possible permutations of the estimated sources $\hat{\mathbf{s}}$ and we assume that the first N' target signals represent the active sources. The first part of the loss function forces the model to maximize the reconstruction fidelity of the N' active sources while the second part forces it to produce close to zero energy estimates for the last $N - N'$, assuming that the best permutation is already in place. The constants $\epsilon = 10^{-9}$ and $\tau = 10^{-3}$ solve numerical stability issues created by zero target signals \mathbf{s} . During training, we use the Adam optimizer [17] with an initial learning rate set to 0.001 and we decrease it by a factor of 2 every 10 epochs. Using the aforementioned scheduler we are able to obtain good performance with only 40 epochs.

3.5 Evaluation details

In order to evaluate the performance of our models we use a stable version of the permutation invariant SI-SDR improvement (SI-SDRi) as proposed in [39]. The SI-SDRi metric has also been chosen in several studies in the source separation literature [37, 36, 22, 43, 20, 39]. There are several other SNR-based metrics which also reflect the fidelity of the reconstructed sources such as signal to distortion ratio (SDR), signal to interference ratio (SIR) and signal to artifacts ratio (SAR) [38]. SDR incorporates the errors from both artifacts and interference but is an overly optimistic performance measure compared to SI-SDR [19]. The improvement is defined as the gain that we get on the SI-SDR measure using the estimated signal instead of the mixture signal \mathbf{x} , as shown next:

$$\text{SI-SDRi}(\hat{\mathbf{s}}, \mathbf{s}) = \max_{\pi \in \mathcal{H}} \left[\frac{1}{N'} \sum_{i=1}^{N'} 10 \log_{10} \left(\frac{\|\alpha_i \mathbf{s}_i\|^2 + \epsilon}{\|\alpha_i \mathbf{s}_i - \hat{\mathbf{s}}_i^{(\pi)}\|^2 + \epsilon} \right) \right] - \frac{1}{N'} \sum_{i=1}^{N'} 10 \log_{10} \left(\frac{\|\alpha_i \mathbf{s}_i\|^2 + \epsilon}{\|\alpha_i \mathbf{s}_i - \mathbf{x}\|^2 + \epsilon} \right), \quad (14)$$

where $\alpha_i = \mathbf{s}_i^\top \hat{\mathbf{s}}_i^{(\pi^*)} / \|\mathbf{s}_i\|^2$, $\epsilon = 10^{-9}$ and π^* is the permutation that maximizes the average SI-SDRi over the active sources. For the case where non-active sources exist, thus, $N' < N$, we omit to compute the aforementioned metric as it provides infinity values. However, we follow a stricter evaluation metric than the one proposed in [39] in the sense that we do not exclude pairs of estimates-targets with low energy estimated sources, as we believe that the evaluation metric should also reflect a penalty for the cases where the model under-separates the input mixture, leading to $M < N'$ non-zero estimated sources. For the variable number of sources case and specifically for the single-source mixtures we simply report the maximum absolute SI-SDR obtained by each one of the estimated sources.

3.6 SuDoRM-RF configurations

We describe the default values for all proposed architectures SuDoRM-RF, SuDoRM-RF++ and, C-SuDoRM-RF++. In the following experimental sections, all those values are going to be described as such, unless otherwise specified. For the encoder \mathcal{E} and decoder modules \mathcal{D} we use a kernel size $K_{\mathcal{E}} = 21$ for input mixtures sampled at 8kHz and $K_{\mathcal{E}} = 41$ for 16kHz. Also, the number of basis is equal to $C_{\mathcal{E}} = 512$. For the configuration of each U-ConvBlock we set the input number of channels equal to $C = 128$, the number of successive resampling operations equal to $Q = 4$ and, the expanded number of channels equal to $C_U = 512$. In each subsampling operation we reduce the temporal dimension by a factor of 2 and all depth-wise separable convolutions have a kernel length of $K_U = 5$ and a stride of $S_U = 2$. Only for the causal variation

C-SuDoRM-RF++, we increase the number of input channels to $C = 256$ and the default kernel length to $K_U = 11$ in order to increase the receptive field in shallower and more efficient architectures needed for real-time applications. For simplicity we use the following naming convention based on the number B of U-ConvBlocks inside the separator module \mathcal{S} . Namely, SuDoRM-RF 2.0x , SuDoRM-RF 1.0x , SuDoRM-RF 0.5x , SuDoRM-RF 0.25x consist of 32, 16, 8 and 4 blocks, respectively. The same applies to the improved version SuDoRM-RF++ and its causal variation C-SuDoRM-RF++.

3.7 Literature models configurations

We compare against the best configurations of some of the latest state-of-the-art approaches for speech [24, 22], universal [36] and music [5] source separation. For a fair comparison with the aforementioned models we use the authors' original code, the best performing configurations for the proposed models as well as the suggested training process. For Demucs [5], 80 channels are used instead of 100 in order to be able to train it on a single graphical processing unit (GPU). For the universal sound separation experiments with a variable number of sources, we compare against the reported numbers in [39], where an enhanced variation of the ConvTasNet is used, namely, TDCN++.

3.8 Measuring computational resources

One of the main goals of this study is to propose models for audio source separation which could be trained using limited computational resources and deployed easily on a mobile or edge-computing device [18]. Specifically, we consider the following aspects which might cause a computational bottleneck during inference or training:

1. Number of executed floating point operations (FLOPs).
2. Number of trainable parameters.
3. Memory allocation required on the device for a single pass.
4. Time for completing each process.

We are using various sampling profilers in Python using Pytorch [28] (version 1.7.1.) for tracing all the requirements of the non-causal models on a server with an Intel(R) Core(TM) i7-3820 @ 3.60GHz CPU and a GeForce GTX TITAN X GPU. For the causal models, we focus on the computational requirements on a much more resource-constrained hardware in order to show the applicability of our models for real-time source separation on devices used by typical users. Thus, we evaluate our causal models on a laptop with an Intel(R) Core(TM) i7-8750H @ 2.20GHz CPU. We measure the inference pass as a simple forward pass always on CPU while we consider that a backward pass is comprised of a forward pass in order to compute the estimated signals and the full back-propagation of the gradient and we measure its computational requirements on GPU.

Model	SI-SDRi (dB)		GFLOPs	Mem. (GB)	Time (sec)
	Speech	Non-speech			
ConvTasNet [24]	15.3*	7.7	5.16	0.61	0.15
Demucs [5]	12.1	7.2	3.42	1.95	0.58
DPRNN [22]	18.8*	7.2	48.81	2.27	2.18
Two-Step TDCN [36]	16.1*	8.2	7.11	1.03	0.24
SuDoRM-RF 1.0x	17.0	8.4	2.45	0.79	0.17
SuDoRM-RF 0.5x	15.4	8.1	1.51	0.49	0.13
SuDoRM-RF 0.25x	13.4	7.9	1.04	0.30	0.09
SuDoRM-RF++ 1.0x	17.0	8.6	2.11	0.79	0.17
SuDoRM-RF++ 1.0x GC	12.5	8.5	0.69	0.87	0.20
C-SuDoRM-RF++ 0.5x	10.6	4.6	2.14	0.39	0.08
C-SuDoRM-RF++ 0.25x	9.6	4.5	1.25	0.30	0.05

Table 1: SI-SDRi separation performance for the proposed models and models in the literature on both separation tasks (speech and non-speech) alongside their computational requirements for performing inference on an Intel(R) Core(TM) i7-3820 @ 3.60GHz CPU for one second of input audio or equivalently 8000 samples. * We assign the maximum SI-SDRi performance obtained by our runs and the reported number on the corresponding paper.

4 Results & Discussion

In Tables 1, and 2 we show the separation performance under the speech and non-speech separation tasks for some of the most recent state-of-the-art models in the literature and the proposed SuDoRM-RF configurations alongside computational requirements. Specifically, in Table 1, we focus on the computational aspects required during a forwards pass of those models on a CPU while in Table 2, the same computational resource requirements are shown for a backward pass on GPU as well as the number of trainable parameters. It is easy to see that the proposed models can match and even outperform the separation performance of other several state-of-the-art models using orders of magnitude less computational requirements across the board.

In Sections 4.1, 4.1.1, 4.3, we focus on specific computational aspects for all the models presented in this study. Moreover, a better visualization for understanding the Pareto efficiency of the proposed architectures is displayed in Figure 4. Specifically, we show for each model, its performance on non-speech sound separation vs a specific computational requirement. In Section 4.4, we conduct a small ablation study to shed light on the most important aspects of the proposed SuDoRM-RF models and how the performance is affected by changing the corresponding hyperparameters. Experiments for universal sound separation with a variable number of sources are presented in Section 4.5. Finally, we present the causal setup of our model and evaluate it for different configurations under a two-speaker separation task in 4.6.

Model	SI-SDRi (dB)		Params (1e6)	GFLOPs	Mem. (GB)	Time (sec)
	Speech	Non-sp.				
ConvTasNet [24]	15.3*	7.7	5.05	21.7	0.68	0.14
Demucs [5]	12.1	7.2	415.09	17.65	8.77	0.24
DPRNN [22]	18.8*	7.2	2.63	135.88	3.28	0.32
Two-Step TDCN [36]	16.1*	8.2	8.63	28.36	1.17	0.22
SuDoRM-RF 1.0x	17.0	8.4	2.72	17.19	0.99	0.27
SuDoRM-RF 0.5x	15.4	8.1	1.42	9.47	0.33	0.11
SuDoRM-RF 0.25x	13.4	7.9	0.79	5.62	0.28	0.08
SuDoRM-RF++ 1.0x	17.0	8.6	2.72	16.23	0.99	0.27
SuDoRM-RF++ 1.0x GC	12.5	8.5	0.30	2.72	1.21	0.35
C-SuDoRM-RF++ 0.5x	10.6	4.6	2.81	11.43	0.29	0.09
C-SuDoRM-RF++ 0.25x	9.6	4.5	1.63	6.27	0.17	0.05

Table 2: SI-SDRi separation performance for the proposed models and models in the literature on both separation tasks (speech and non-speech) alongside their computational requirements for performing a backward pass on a GeForce GTX TITAN X GPU for one second of input audio or equivalently 8000 samples. * We assign the maximum SI-SDRi performance obtained by our runs and the reported number on the corresponding paper.

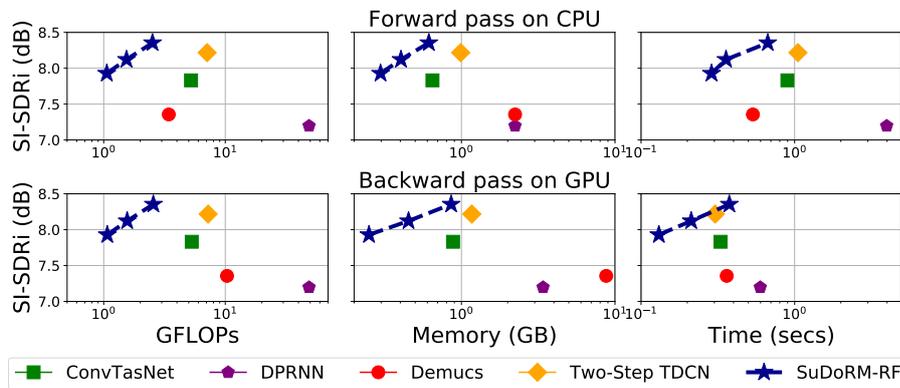


Fig. 4: SI-SDRi non-speech sound separation performance on ESC50 vs computational resources with an input audio recording of 8000 samples for all models. (Top row) computational requirements for a single forward pass on CPU (Bottom) for a backward pass on GPU. All x-axis are shown in log-scale while the 3 connected blue stars correspond to the three SuDoRM-RF configurations from Section 3.6.

4.1 Floating point operations (FLOPs)

Different devices (CPU, GPU, mobiles, etc.) have certain limitations on their FLOPs throughput capacity. In the case of an edge device, the computational resource one might be interested in is the number of FLOPs required during inference. On the other hand, training on cloud machines might be costly if a

huge number of FLOPs is needed in order to achieve high separation performance. As a result, it is extremely important to be able to train and deploy models which require a low number of computations [11]. We see from the first column of Figure 4 that SuDoRM-RF models scale well as we increase the number of U-ConvBlocks B from $4 \rightarrow 8 \rightarrow 16$. Furthermore, we see from Tables 1 and 2 that for both forward and backward passes, correspondingly, the family of the proposed SuDoRM-RF models appear more Pareto efficient in terms of SI-SDRi performance vs Giga-FLOPs (GFLOPs) and time required compared to the other state-of-the-art models which we take into account. Specifically, the DPRNN model [22] which performs sequential matrix multiplications (even with a low number of parameters) requires at least 45 times more FLOPs for a single pass compared to SuDoRM-RF 0.25x while performing worse when trained for the same number of epochs under the non-speech separation task.

Moreover, we see from both Tables 1 and 2 that the improved version SuDoRM-RF++ achieves similar or even better results than the original version of the proposed model SuDoRM-RF with a lower number of FLOPs both in forward and backward for a similar number of parameters and execution time. A significant drop in the absolute number of FLOPs is also obtained by combining the group communication mechanism proposed in [23] with SuDoRM-RF++ . However, that does not automatically entail a lower execution time. The causal variations C-SuDoRM-RF++ perform competitively with the same number of FLOPs but they are still performing much worse than all the other non-causal models showing that there is much room for improvement.

4.1.1 Cost-efficient training

Usually one of the most detrimental factors for training deep learning models is the requirement of allocating multiple GPU devices for several days or weeks until an adequate performance is obtained on the validation set. This huge power consumption could lead to huge cloud services rental costs and carbon dioxide emissions [3]. In Figure 5, we show the validation SI-SDRi performance for the speech separation task which is obtained by each model versus the total amount of FLOPs performed. For each training epoch, all models perform updates while iterating over 20,000 audio mixtures. Notably, even the original SuDoRM-RF models outperform all other models in terms of cost-efficient training as they obtain better separation performance while requiring significantly fewer training FLOPs. For instance, SuDoRM-RF 1.0x obtains ≈ 16 dB in terms of SI-SDRi compared to ≈ 10 dB of DPRNN [22] which manages to complete only 3 epochs given the same number of training FLOPs.

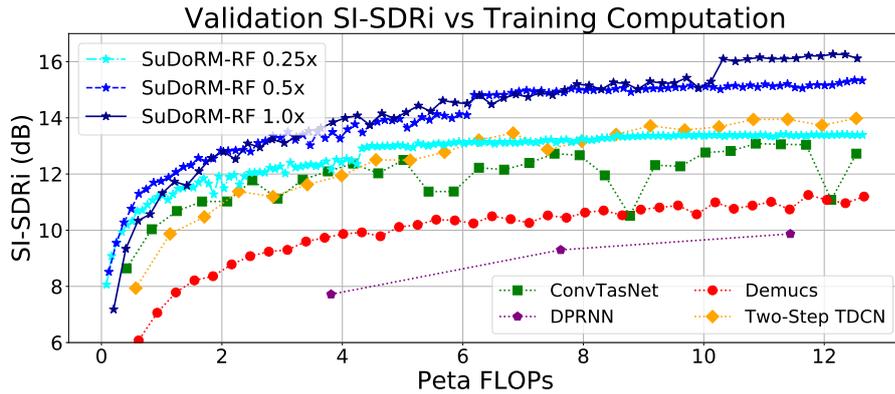


Fig. 5: Validation SI-SDRi separation performance for speech-separation vs the number of FLOPs executed during training. All models are trained using batches of 4 mixtures with 32,000 time-samples each. Each point corresponds to a completed training epoch.

4.2 Trainable parameters

From Table 2 it is easy to see that SuDoRM-RF architectures are using orders of magnitude fewer parameters compared to the U-net architectures like Demucs [5] where each temporal downsampling is followed by a proportional increase to the number of channels. Moreover, the upsampling procedure inside each U-ConvBlock does not require any additional parameters. The SuDoRM-RF models seem to increase their effective receptive field with significantly fewer parameters compared to dilated convolutional architectures [24, 36]. Notably, our largest model SuDoRM-RF 1.0x matches the relatively low number of parameters of the DPRNN [22] model which is based on stacked RNN layers. Group communication combined with SuDoRM-RF++ is one of the most effective ways to reduce the number of trainable parameters caused by the bottleneck dense layers between the U-ConvBlocks. Essentially, the SuDoRM-RF++ 1.0x GC model with $B = 16$ number of U-ConvBlocks has less than half of the parameters of a shallow original SuDoRM-RF 0.25x model with only $B = 4$ U-ConvBlocks.

4.3 Memory requirements

In most of the studies where efficient architectures are introduced [11, 4, 26, 42] authors are mainly concerned with the total number of trainable parameters of the network. The same applies to efficient architectures for source separation [24, 22, 25]. However, the trainable parameters comprise only a small portion of the total amount of memory required for a single forward or backward pass. The space complexity could easily be dominated by the storage of intermediate representations and not the actual memory footprint of the model itself. The

latter could become even worse when multiple skip connections are present, gradients from multiple layers have to be stored or implementations require augmented matrices (dilated, transposed convolutions, etc.).

From Tables 1 and 2, we see that SuDoRM-RF++ and the initial SuDoRM-RF models have almost the same memory footprint. However, when combining the SuDoRM-RF++ with the group communication (GC) mechanism we see that even if the number of trainable parameters is significantly reduced then still the actual memory requirement increases which positively validates our hypothesis that the actual memory requirement in GBs is a very important metric to show when proposing new efficient models. The causal variation of our models C-SuDoRM-RF++ is the more light-weight model that we propose which also secures a very competitive inference time.

In Figure 4, we see that even the original SuDoRM-RF models are more Pareto efficient in terms of the memory required compared to the dilated convolutional architectures of ConvTasNet [24] and Two-Step TDCN [36] where they require an increased network depth in order to increase their receptive field. Although SuDoRM-RF models do not perform downsampling in every feature extraction step as Demucs [5] does, we see that the proposed models require orders of magnitude less memory especially during a backward update step as the number of parameters in Demucs is significantly higher. Finally, SuDoRM-RF models have a smaller memory footprint because the encoder \mathcal{E} performs temporal downsampling by a factor of $\text{div}(K_{\mathcal{E}}, 2) = 10$ compared to DPRNN [22] which does not reduce the temporal resolution at all.

4.4 Ablation study on WSJ0-2mix

We perform an ablation study in order to show how different parameter choices in SuDoRM-RF++ models affect the separation performance. In order to be directly comparable with the numbers reported by several other studies [24, 22, 43, 20], we train our models for 200 epochs and test them using the given data splits from WSJ0-2mix dataset [10]. The results are shown in Table 3.

We see that a significant aspect is the stride of the encoder and decoder which is always defined as $K_{\mathcal{E}}/2$. By decreasing the size of the stride we force the model to perform more computations and also estimate the signal in a more fine-grained scale closer to the time-domain resolution leading to better results which is also consistent with other studies [15]. Moreover, we see that the GLN significantly helps our model to reach a better solution compared to the simple layer norm, presumably acting as a better regularizer in between the intermediate activations. Furthermore, when keeping all the other parameters the same except for the number of U-ConvBlocks B and the number of resampling procedures Q we see one of the most important aspects of the SuDoRM-RF++ model which is the benefit in the receptive field that we are getting by analyzing the signal at multiple scales. Specifically, we see that a model with $B = 18$ and $Q = 7$ outperforms a deeper model in terms of U-ConvBlocks $B = 20$ which only processes the signal at $Q = 2$ more

$K_{\mathcal{E}}$	C	B	Q	Normalization	SI-SDRi
17	128	16	4	LN	15.9
17	128	16	4	GLN	16.8
21	256	20	4	GLN	17.7
41	256	32	4	GLN	17.1
41	256	20	4	GLN	16.8
21	512	18	7	GLN	18.0
21	512	20	2	GLN	17.4
21	512	34	4	GLN	18.9

Table 3: SI-SDRi separation performance on WSJ0-2mix for various parameter configurations of SuDoRM-RF++ models. GLN corresponds to the global layer normalization as described in Equation 10 and LN corresponds to the classic layer normalization layer proposed in [1] and explained in Equation 9. All the other parameters have the same values as described in Section 3.6.

scales. We need to underline that increasing the parameter Q does not lead to significant computational requirements mainly because of the downsampling operation which is relatively a cheap way to increase the receptive field of the model without carrying its whole information end-to-end.

4.5 Variable number of sources

In Table 4 we report the performance of the proposed SuDoRM-RF models under a universal sound separation task with a varying number of sources in each mixture. We always assume that the maximum number of active sources in each mixture is $N = 4$ and we measure the performance on the same dataset splits where mixtures with $N' \in \{1, 2, 3, 4\}$ active sources. We see that by increasing our SuDoRM-RF and SuDoRM-RF++ model sizes to match the size of TDCNN++ [39], we can match its performance which is also the current state-of-the-art performance in universal sound separation with a variable number of sources. For the single source mixtures we see that our models perform worse than the TDCNN++, however, above 25dB it is really difficult even for a human being to understand the nuance artifacts which are barely audible. Our SuDoRM-RF++ 2.0x outperforms the state-of-the-art for the most difficult case where $N' = 4$ sources are active. Moreover, by using the SuDoRM-RF++ 1.0x GC, we match the performance obtained by the state-of-the-art with a significantly fewer parameters. We also mention that our models perform worse than the state-of-the-art for the cases where $N' = 2$ and $N' = 3$ because we also penalize the performance of our models in the cases of under-separation which has been reported as 25% in [39]. Namely, for the pairs of corresponding estimates-targets with low-energy estimates, the state-of-the-art numbers only consider the pairs with estimated sources which have power higher than 20 dB above the power of the quietest non-zero reference source. However, we always include all pairs of estimates-targets if a target source is non-zero as we believe that this is a more appropriate metric

Model	SI-SDR (dB)	SI-SDRi (dB)			Avg. (2-4)
	$N' = 1$	$N' = 2$	$N' = 3$	$N' = 4$	
TDCN++ [39]	35.5	11.2*	11.6*	7.4	10.1
SuDoRM-RF 2.0x	22.2	9.9	9.6	7.0	8.8
SuDoRM-RF++ 2.0x	25.9	10.9	10.6	7.8	9.8
SuDoRM-RF++ 1.0x GC	22.9	9.9	9.9	7.3	9.0
SuDoRM-RF++ 0.5x	20.0	8.4	8.3	6.0	7.6

Table 4: SI-SDR and SI-SDRi separation performance on the FUSS dataset for a variable number of sources. Specifically, we report the absolute SI-SDR ($N' = 1$) for the reconstruction of single-source mixtures. For mixtures with multiple active sources $N' > 1$ we measure the SI-SDRi performance of the reconstructed sources wrt the input mixture. *Metrics do not consider pairs of corresponding estimates-target sources with low energy estimates, where a 25% of under-separation is reported in the cases of $N' \in \{2, 3, 4\}$.

B	K_U	SI-SDRi	Time (ms)
4	3	8.4	50.8
	5	9.1	50.4
	11	9.5	88.7
8	3	9.6	90.6
	5	10.1	88.2
	11	10.3	165.9

Table 5: SI-SDRi separation performance on WSJ0-2mix for various parameter configurations of causal C-SuDoRM-RF++ models alongside their inference time for 8000 input samples on a laptop with an Intel(R) Core(TM) i7-8750H @ 2.20GHz CPU. For reference, one of the most current state-of-the-art speech denoisers, namely, real-time Demucs [6] runs on the same setup at 92.3ms.

to use. We also like to note that we did not notice any significant performance improvement when we use mixture consistency [40] at the output estimates of our models where the separated sources are forced to sum up to the input mixture using a projection layer.

4.6 Causal setup

In Table 5 we report the performance of the proposed causal version of SuDoRM-RF models under a separation setup where two speakers are active using the WSJ0-2mix dataset [10] alongside the inference time on a laptop for 1 second of input audio sampled at 8kHz. We see that we are able to obtain competitive separation performance for all configurations while remaining ≈ 10 to 20 times faster than real time.

5 Conclusions

In this study, we have introduced the SuDoRM-RF network, a novel architecture for efficient universal sound source separation. Moreover, we have presented several improvements on the original model including SuDoRM-RF++ which directly estimates the of the latent representations of the models, a variation which shares parameters across sub-bands as well as C-SuDoRM-RF++ which is causal and enables real-time inference. The proposed model is based on the U-ConvBlock which is capable of extracting multi-resolution temporal features through successive depth-wise convolutional downsampling of intermediate representations and aggregates them using a non-parametric interpolation scheme. In this way, SuDoRM-RF models are able to significantly reduce the required number of layers in order to effectively capture long-term temporal dependencies. We show that these models can perform similarly or even better than recent state-of-the-art models while requiring significantly less computational resources in FLOPs, memory and, time for experiments with a fixed and a variable number of sources.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Brunner, G., Naas, N., Palsson, S., Richter, O., Wattenhofer, R.: Monaural music source separation using a resnet latent separator network. In: Proc. ICTAI, pp. 1124–1131 (2019)
3. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once for all: Train one network and specialize it for efficient deployment. In: Proc. ICLR (2020)
4. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proc. CVPR, pp. 1251–1258 (2017)
5. Défossez, A., Usunier, N., Bottou, L., Bach, F.: Music source separation in the waveform domain. arXiv preprint arXiv:1911.13254 (2019)
6. Défossez, A., Synnaeve, G., Adi, Y.: Real Time Speech Enhancement in the Waveform Domain. In: Proc. Interspeech, pp. 3291–3295 (2020)
7. Haris, M., Shakhnarovich, G., Ukita, N.: Deep back-projection networks for super-resolution. In: Proc. CVPR, pp. 1664–1673 (2018)
8. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proc. CVPR, pp. 1026–1034 (2015)
9. Hennequin, R., Khlif, A., Voituret, F., Moussallam, M.: Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software* **5**(50), 2154 (2020)
10. Hershey, J.R., Chen, Z., Le Roux, J., Watanabe, S.: Deep clustering: Discriminative embeddings for segmentation and separation. In: Proc. ICASSP, pp. 31–35 (2016)
11. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
12. Huang, P.S., Kim, M., Hasegawa-Johnson, M., Smaragdis, P.: Deep learning for monaural speech separation. In: Proc. ICASSP, pp. 1562–1566 (2014)
13. Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A., Dieleman, S., Kavukcuoglu, K.: Efficient neural audio synthesis. In: International Conference on Machine Learning, pp. 2410–2419 (2018)
14. Kaspersen, E.T., Kounalakis, T., Erkut, C.: Hydranet: A real-time waveform separation network. In: Proc. ICASSP, pp. 4327–4331 (2020)

15. Kavalerov, I., Wisdom, S., Erdogan, H., Patton, B., Wilson, K., Roux, J.L., Hershey, J.R.: Universal sound separation. In: Proc. WASPAA, pp. 175–179 (2019)
16. Kim, M., Smaragdis, P.: Efficient source separation using bitwise neural networks. In: Audio Source Separation, pp. 187–206. Springer (2018)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Lane, N.D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Jiao, L., Qendro, L., Kawsar, F.: Deepx: A software accelerator for low-power deep learning inference on mobile devices. In: Proc. IPSN, pp. 1–12 (2016)
19. Le Roux, J., Wisdom, S., Erdogan, H., Hershey, J.R.: Sdr–half-baked or well done? In: Proc. ICASSP, pp. 626–630 (2019)
20. Liu, Y., Wang, D.: Divide and conquer: A deep casa approach to talker-independent monaural speaker separation. arXiv preprint arXiv:1904.11148 (2019)
21. Luo, W., Li, Y., Urtasun, R., Zemel, R.: Understanding the effective receptive field in deep convolutional neural networks. In: Advances in neural information processing systems, pp. 4898–4906 (2016)
22. Luo, Y., Chen, Z., Yoshioka, T.: Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation. In: Proc. ICASSP (2020)
23. Luo, Y., Han, C., Mesgarani, N.: Ultra-lightweight speech separation via group communication. arXiv preprint arXiv:2011.08397 (2020)
24. Luo, Y., Mesgarani, N.: 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)
25. Maldonado, A., Rascon, C., Velez, I.: Lightweight online separation of the sound source of interest through blstm-based binary masking. arXiv preprint arXiv:2002.11241 (2020)
26. Mehta, S., Rastegari, M., Shapiro, L., Hajishirzi, H.: Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In: Proc. CVPR, pp. 9190–9200 (2019)
27. Pandey, A., Wang, D.: Tcnn: Temporal convolutional neural network for real-time speech enhancement in the time domain. In: Proc. ICASSP, pp. 6875–6879 (2019)
28. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (2019)
29. Paul, D.B., Baker, J.M.: The design for the wall street journal-based CSR corpus. In: Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23–26, 1992 (1992)
30. Piczak, K.J.: Esc: Dataset for environmental sound classification. In: Proc. ACM International Conference on Multimedia, pp. 1015–1018 (2015)
31. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, pp. 234–241. Springer (2015)
32. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-attention with relative position representations. In: Proc. ACL, pp. 464–468 (2018)
33. Sifre, L., Mallat, S.: Rigid-motion scattering for image classification. Ph. D. thesis (2014)
34. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Workshop Proc. ICLR (2014)
35. Subakan, C., Ravanelli, M., Cornell, S., Bronzi, M., Zhong, J.: Attention is all you need in speech separation. In: Proc. ICASSP (2021 (To appear))
36. Tzinis, E., Venkataramani, S., Wang, Z., Subakan, C., Smaragdis, P.: Two-step sound source separation: Training on learned latent targets. In: Proc. ICASSP (2020)
37. Tzinis, E., Wisdom, S., Hershey, J.R., Jansen, A., Ellis, D.P.: Improving universal sound separation using sound classification. In: Proc. ICASSP (2020)
38. Vincent, E., Gibbonval, R., Févotte, C.: Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing* **14**(4), 1462–1469 (2006)

-
39. Wisdom, S., Erdogan, H., Ellis, D., Serizel, R., Turpault, N., Fonseca, E., Salamon, J., Seetharaman, P., Hershey, J.: What's all the fuss about free universal sound separation data? arXiv preprint arXiv:2011.00803 (2020)
 40. Wisdom, S., Hershey, J.R., Wilson, K., Thorpe, J., Chinen, M., Patton, B., Saurous, R.A.: Differentiable consistency constraints for improved deep speech enhancement. In: Proc. ICASSP, pp. 900–904 (2019)
 41. Yu, D., Kolbæk, M., Tan, Z.H., Jensen, J.: Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In: Proc. ICASSP, pp. 241–245 (2017)
 42. Yu, J., Yang, L., Xu, N., Yang, J., Huang, T.: Slimmable neural networks. In: Proc. ICLR (2019)
 43. Zeghidour, N., Grangier, D.: Wavesplit: End-to-end speech separation by speaker clustering. arXiv preprint arXiv:2002.08933 (2020)