INVESTIGATING U-NETS WITH VARIOUS INTERMEDIATE BLOCKS FOR SPECTROGRAM-BASED SINGING VOICE SEPARATION

Woosung Choi[1] **Minseok Kim**[1] **Jaehwa Chung**[2]
Daewon Lee[3] **Soonyoung Jung**[1]
[1] Department of Computer Science and Engineering, Korea University, Republic of Korea
[2] Department of Computer Science, Korea National Open University, Republic of Korea
[3] Department of Computer Engineering, Seokyeong University, Republic of Korea
jsy@korea.ac.kr

ABSTRACT

Singing Voice Separation (SVS) tries to separate singing voice from a given mixed musical signal. Recently, many U-Net-based models have been proposed for the SVS task, but there were no existing works that evaluate and compare various types of intermediate blocks that can be used in the U-Net architecture. In this paper, we introduce a variety of intermediate spectrogram transformation blocks. We implement U-nets based on these blocks and train them on complex-valued spectrograms to consider both magnitude and phase. These networks are then compared on the SDR metric. When using a particular block composed of convolutional and fully-connected layers, it achieves state-of-the-art SDR on the MUSDB singing voice separation task by a large margin of 0.9 dB. Our code and models are available online. [1]

1. INTRODUCTION

Singing Voice Separation (SVS), a special case of Music Source Separation (MSS), aims at separating singing voice from a given mixed musical signal. Recently, many machine learning-based methods have been proposed for SVS and MSS tasks. They can be categorized into two groups: waveform-to-waveform models and spectrogram-based models. While the former tries to generate the vocal waveforms directly, the latter estimates spectrograms (usually magnitude) of vocal waveforms.

Typical spectrogram-based models apply Short-Time Fourier Transform (STFT) on a mixture waveform to obtain the input spectrograms. Then, they estimate the vocal spectrograms based on these inputs and finally restore the vocal waveform with inverse STFT (iSTFT). A variety of spectrogram-based models have been proposed in the music information retrieval community and the machine learning community. For example, [1] employed the U-Net [2] architecture, an encoder-decoder structure with symmetric skip connections. These symmetric skip connections allow models to recover fine-grained details of the target object during decoding effectively. Several works [3–6] also used similar architectures.

They have revealed that U-Net-like architectures can provide promising performance for SVS and MSS. Existing works have proposed various types of neural networks for intermediate blocks. While some models [1, 3] used simple Convolutional Neural Networks (CNNs) for intermediate blocks, other advanced models tried more complex intermediate blocks. For instance, MMDenseLSTM [6] used densely connected CNNs followed by Long Short-Term Memory (LSTM) networks to efficiently model long-term structures, where LSTM is a variant of Recurrent Neural Networks (RNNs). However, a thorough search of the relevant literature indicated that there were no existing works that evaluate and directly compare these different types of blocks.

In this paper, we conduct a comparative study of U-Nets on various intermediate blocks. We designed several types of blocks based on different design strategies, which we present in section 3. For each type of block, we implemented at least one SVS model, which are all based on an identical U-Net framework for fair comparisons. In section 4, we summarize the experimental results and discuss the effect of each design choice. We validate hypotheses such as that inserting time-distributed operations (see §3.1) into intermediate blocks can significantly improve performance, which led to state-of-the-art (SOTA) performance on the MUSDB [7] SVS task.

Finally, our U-Net framework directly estimates the target complex-valued spectrogram (viewing real and imaginary as separate channels), when many existing models estimate the target magnitude without phase. In general, considering phase information improves the separation quality, as discussed in [8, 9]. Several phase-aware methods have been proposed for speech enhancement, such as phase reconstruction methods [8,9], or using raw complex-valued STFT outputs [10, 11]. In section 4, we show that the latter method is an efficient way to improve magnitude-only models, only needing a few minor adjustments.

[1] https://github.com/ws-choi/ISMIR2020_U_Nets_SVS

2. U-NET-BASED SVS FRAMEWORK

In this section, we describe a U-Net-based SVS framework, which is shared by several models in §4. We first introduce the 'Complex as Channel framework' (CaC), a spectrogram-based SVS framework, and then define our U-Net architecture for spectrogram estimation in CaC.

2.1 Complex as Channel Framework

CaC is a singing voice separation framework based on complex-valued spectrogram estimation. It takes a c-channeled mixture signal, and outputs c-channeled singing voice signal. As shown in Figure 1, CaC consists of three parts as follows:

1. The *spectrogram extraction layer* extracts a mixture spectrogram by applying STFT to the c-channeled input signal. The output of STFT is a complex-valued spectrogram with c-channels. Considering the imaginary and real parts as separate real-valued channels, we view the mixture spectrogram $M_{complex} \in \mathbb{C}^{c \times T \times F}$ as a $(2c)$-channeled real-valued spectrogram $M \in \mathbb{R}^{(2c) \times T \times F}$, where T denotes the number of frames and F denotes the number of the frequency bins in the spectrogram.

2. The *complex-valued spectrogram estimation network* is a neural network that takes the spectrogram $\mathbf{M}$ of a mixture signal as input and estimates the target spectrogram $\hat{T} \in \mathbb{R}^{(2c) \times T \times F}$, which is used for reconstructing the vocal signal later.

3. The *signal reconstruction layer* reshapes the estimated spectrogram $\hat{T}$ into the complex-valued spectrogram $\hat{T}_{complex} \in \mathbb{C}^{c \times T \times F}$, as shown in Figure 1. It then restores the estimated singing voice signal via inverse-STFT on $\hat{T}_{complex}$.

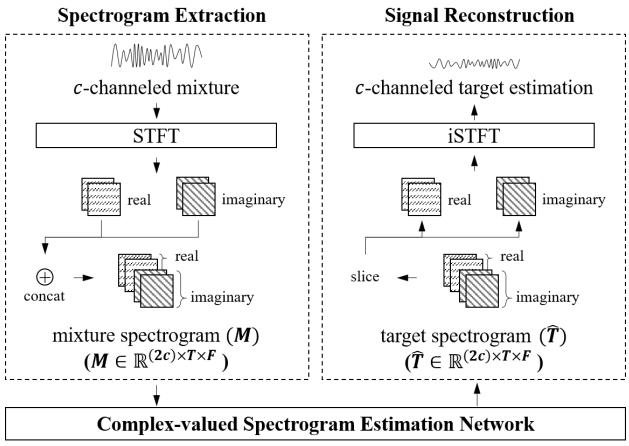

Figure 1. The Complex as Channel Framework

For a given mixture spectrogram $\mathbf{M}$, we train the complex-valued spectrogram estimation network in a supervised fashion to minimize the mean square error between the output $\hat{T}$ and the ground-truth spectrogram $\mathbf{T}$ of the singing voice signal.

It should be noted that the shape of $\mathbf{M}$ and $\hat{T}$ is $(2c) \times T \times F$, considering real and imaginary parts of a spectrogram as separate real-valued channels. This approach allows CaC to fully utilize the information in complex-valued spectrograms for both the input and the output. Meanwhile, current SOTA models (e.g., SA-SHN [4] and DGRU-DGConv [12]) decompose a complex-valued spectrogram into magnitude and phase, and only use the magnitude for the input of their networks. Although SA-SHN and DGRU-DGConv yielded impressive results by introducing novel attention method [4] and by adopting dilated 1-D convolutions [12] with Gated Recurrent Units (GRU) [13] respectively, they do not consider phase information. In §4.5, we compare the Source-to-Distirtion (SDR) [14] performance of models based on the CaC framework and that of models based on the Magnitude-only framework.

2.2 U-Net Architecture for Spectrogram Estimation

For spectrogram estimation in CaC, we use a U-Net-based architecture. It consists of an encoder and a decoder: the encoder transforms $\mathbf{M}$ into a downsized spectrogram-like representation, and the decoder takes it and returns the estimated target spectrogram $\hat{T}$. Before we describe them in detail, we introduce two types of main components in the architecture as follows.

- An *intermediate block* transforms an input spectrogram-like tensor into an equally-sized tensor (possibly with a different number of channels).
- A *down/up sampling layer* halves/doubles the scale of an input tensor either in the time, frequency, or Time-Frequency domain.

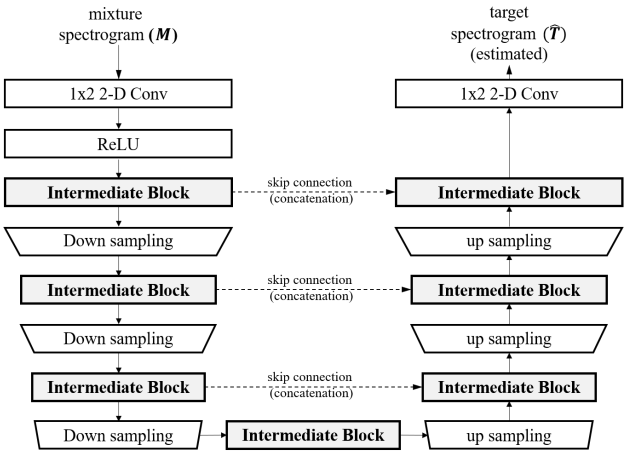

Figure 2. U-Net Architecture for Spectrogram Estimation

As shown in Figure 2, the number of down-sampling layers and the number of up-sampling layers are the same. Also, it uses the same number of intermediate blocks in the encoding and the decoding phase. It has an additional block in between its encoder and decoder. Thus, the total number of blocks should be an odd integer. It has skip connections that concatenate output feature maps of the same scale between the encoder and the decoder.

Besides basic components, our architecture has two additional convolution layers, as illustrated in Figure 2. We use them to increase or restore the number of channels. Before describing them, let us introduce some notations. We denote the input of the l-th intermediate block by $X^{(l-1)}$, and the output by $X^{(l)}$. The size of $X^{(l-1)}$ is denoted by $c_{in}^{(l)} \times T^{(l)} \times F^{(l)}$, where $c_{in}^{(l)}$ represents the number of channels and and $T^{(l)} \times F^{(l)}$ represents the size of the spectrogram-like tensor. Also, we denote the size of $X^{(l)}$ by $c_{out}^{(l)} \times T^{(l)} \times F^{(l)}$, where $c_{out}^{(l)}$ is the number of channels. Using these notations, we denote the input of the first block by $X^{(0)}$, and its size by $c_{in}^{(1)} \times T^{(1)} \times F^{(1)}$. To increase the number of channels, it applies a 1×2 convolution with $c_{in}^{(1)}$ output channels followed by ReLU [15] activation to the given input $\mathbf{M}$. To adjust the number of channels, it also applies a final 1×2 convolution with $(2c)$ output channels to the output of the final block. Note that the last layer is not followed by any activation function since target TF bins can be negative. We empirically set the parameter $c_{in}^{(1)}$ to be 24 in our experiments. Models with smaller $c_{in}^{(1)}$ (e.g., 12) are trained faster, but usually perform inferior than models with larger size of $c_{in}^{(1)}$.

We can implement various SVS models based on this architecture in the CaC framework because multiple options are available for intermediate blocks. In section 3, we present several neural networks which can be used as intermediate blocks in this paper.

3. INTERMEDIATE BLOCKS

We present several types of intermediate blocks based on different design strategies. We first present time-distributed blocks and then present time-frequency blocks.

3.1 Time-Distributed Blocks

Some existing models use CNNs (e.g., [16]) for intermediate blocks to extract timbre features of the target source. However, the authors of [8] reported that conventional CNN kernels are limited for this task. They found that long-range correlations exist along the frequency axis in the spectrogram of voice signals, which Fully-connected Neural Networks (FCNs) can efficiently capture. They proposed a model named Phasen for speech enhancement, which uses the Frequency Transformation Block (FTB) that has a single-layered FCN without bias. This FCN is applied to each frame of the internal representation in a *time-distributed* manner.

Inspired by TFB, we introduce *time-distributed* blocks, which are applied to a single frame of a spectrogram-like feature map. These blocks try to extract time-independent features that help singing voice separation without using inter-frame operations. We first introduce an FCN-based block and then propose an alternative time-distributed block based on 1-D CNNs.

3.1.1 Time-Distributed Fully-connected networks

We present an FCN-based intermediate block, called Time-Distributed Fully-connected network (TDF). As illustrated in Figure 3, a TDF block is applied to each channel of each frame separately and identically.

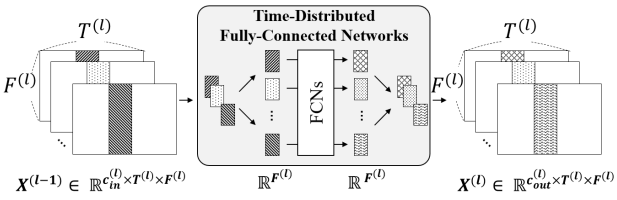

Figure 3. Time-Distributed Fully-connected networks

Suppose that the l-th intermediate block in our U-Net structure takes input $X^{(l-1)}$ into an output $X^{(l)}$. As shown in Figure 3, a fully-connected network is applied separately and identically to each frame (i.e., $X^{(l-1)}[i, j, :]$) in order to transform an input tensor in a time-distributed fashion. While an FTB of Phasen [8] is single-layered, a TDF block can be either single- or multi-layered. Each layer is defined as consecutive operations: a fully-connected layer, Batch Norm (BN) [17], and ReLU [15]. If it is multi-layered, then each internal layer maps an input to the hidden feature space, and its final layer maps the internal vector to $\mathbb{R}^{F^{(l)}}$. The number of hidden units is $\lfloor F^{(l)}/bn \rfloor$, where we denote the bottleneck factor by bf. We can reduce parameters if we use two-layered TDFs of $bf > 2$. We investigate the effect of adding additional layers in §4.2.

3.1.2 Time-Distributed Convolutions

We propose an alternative time-distributed block named Time-Distributed Convolutions (TDC), which is applied separately and identically to each multi-channeled frame. It is a series of 1-D convolution layers. Inspired by [5,6], it takes form of a *dense block* [18] structure. A dense block consists of densely connected composite layers, where each composite layer is defined as three consecutive operations: 1-D convolution, BN, and ReLU. As discussed in [5, 6, 18] the densely connected structure enables each layer to propagate the gradient directly to all preceding layers, making a deep CNN training more efficient.

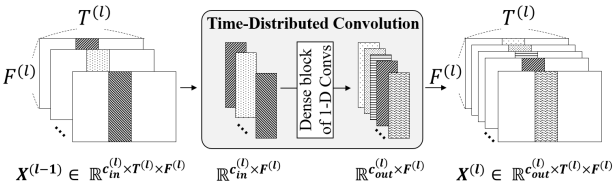

Figure 4. Time-Distributed Convolutions

3.2 Time-Frequency Blocks

The performances of U-Nets with time-distributed blocks were above our expectation (see §4.2), but were still inferior considerably to those of current SOTA methods. The reason is that features observed in musical sources include sequential patterns (e.g., vibrato, tremolo, and crescendo) or musical patterns (e.g., rhythm, repetitive structure), which cannot be modeled by time-distributed blocks.

While time-distributed blocks cannot model the temporal context, time-frequency blocks try to extract features considering both the time and the frequency dimensions. We introduce the Time-Frequency Convolutions (TFC) block, which is used in [5]. We also propose two novel blocks that combine two different transformations.

3.2.1 Time-Frequency Convolutions

The Time-Frequency Convolutions (TFC) is a dense block of 2-D CNNs, as shown in Figure 5. The dense block consists of densely connected composite layers, where each layer is defined as three consecutive operations: 2-D convolution, BN, and ReLU. It is applied to the spectrogram-like input representation in the time-frequency domain. Every convolution layer in a dense block has kernels of size (k_F, k_T). Its 2-D filters are trained to jointly capture features along both frequency and temporal axes.

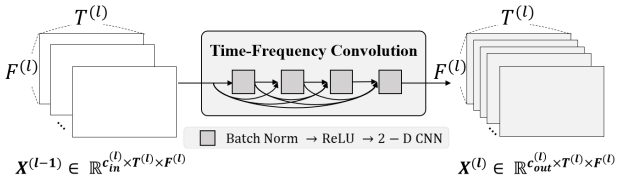

Figure 5. Time-Frequency Convolutions

3.2.2 Time-Frequency Convolutions with TDF

We propose the Time-Frequency Convolutions with Time-Distributed Fully-connected networks (TFC-TDF) block. It utilizes two different blocks inside: a TFC block and a TDF block. Figure 6 describes a TFC-TDF block. It first maps the input $X^{(l-1)}$ to a same-sized representation with $c_{out}^{(l)}$ channels by applying the TFC block. Then the TDF block is applied to the dense block output. A residual connection is also added for efficient gradient flow.

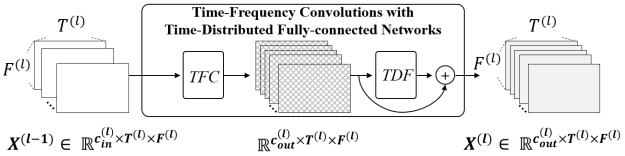

Figure 6. Time-Frequency Convolutions with TDF

Phasen [8] has shown that inserting time-distributed operations into intermediate blocks can improve speech enhancement performance. We validate whether it also works for SVS or not in §4.3.

3.2.3 Time-Distributed Convolutions with RNNs

We propose an alternative way to consider both the time and frequency dimensions. A Time-Distributed Convolutions with Recurrent Neural Networks (TDC-RNN) block uses two different blocks: a TDC block for extracting timbre features and RNNs for capturing temporal patterns. It extracts timbre features and temporal features *separately*, unlike a TFC block. We validate whether this approach can outperform the 2-D CNN approach by comparing TDC-RNNs with TFCs in §4.3.

The structure of a TDC-RNN block is similar to that of a TFC-TDF block. It applies the TDC block to an input $X^{(l-1)}$, and obtains a same sized hidden representation with $c_{out}^{(l)}$ channels. The RNNs compute the hidden representation and output an equally sized tensor. A residual connection is added, as is a TFC-TDF block.

4. EXPERIMENT

We evaluate U-Nets with different types of blocks introduced in §3. We compare the performance of models in §4.2 and §4.3. Also, we compare our models with SOTA models in §4.4. We compare the spectrogram estimation framework in §4.5. We discuss reusable insights in §4.6.

4.1 Setup

4.1.1 Dataset

Train and test data were obtained from the MUSDB dataset [7]. The train and test sets of MUSDB have 100 and 50 musical tracks each, all stereo and sampled at 44100 Hz. Each track file consists of the mixture and its four source audios: 'vocals,' 'drums,' 'bass' and 'other.' Since we are evaluating on singing voice separation, we only use the 'vocals' source audio as the separation target for each mixture track.

4.1.2 Model Configurations

We implemented U-Nets with different blocks (§3). Each model is based on the U-Net architecture (§2.2) on the CaC framework (§2.1). We set $c_{in}^{(1)}$, the number of internal channels to be 24, as mentioned in §2.2. Each model uses a single type of block for its intermediate blocks. We usually used an FFT window size of 2048 and a hop size of 1024 for STFT. However, we used a larger window size in some models for a fair comparison with SOTA methods.

4.1.3 Training and Evaluation

Weights of each model were optimized with RMSprop [19] with learning rate $lr \in [0.0005, 0.001]$ depending on model depth. Each model is trained to minimize the mean square error between $\hat{T}$ and $\mathbf{T}$ as mentioned in §2.1. We use the default validation set (14 tracks) as defined in the *MUSDB* package, and use the Mean Squared Error (MSE) between target and estimated signal (waveform) as the validation metric for validation. Data augmentation [20] was done on the fly to obtain fixed-length mixture audio clips comprised of the source audio clips from different tracks.

We use the official evaluation tool [2] provided by the organizers of the SiSEC2018 [21] to measure Source-to-Distortion Ratio (SDR) [14]. We use the median SDR value over all the test set tracks to obtain the overall SDR performance for each run, as done in the SiSEC2018. We report the average of 'median SDR values' over three runs for each model.

[2] https://github.com/sigsep/sigsep-mus-eval

block type	# blocks	# params	SDR
TDC (w/ sampling)	17	0.54M	**4.86**
TDC (w/o sampling)	17	0.52M	3.78
TDC (w/o sampling)	3	0.09M	3.56
TDF (w/o hidden layer)	17	2.83M	4.75
TDF (w/ hidden layer)	17	1.44M	4.05
TDF (w/ hidden layer)	3	1.19M	4.01

Table 1. Evaluation results of Time-Distributed Blocks.

model	sampling	# blocks	# params	SDR
TFC	O	17	1.56M	6.89
TFC	X	17	1.56M	6.75
TDC-RNN	O	17	2.08M	6.69
TFC-TDF	O	7	0.99M	7.07
TFC-TDF	O	17	1.93M	**7.12**

Table 2. Evaluation results of Time-Frequency Blocks.

4.2 U-Nets with Time-Distributed Blocks

We implemented and trained U-nets with TDC and TDF blocks. We also implemented models with TDC blocks that do not use down/up-sampling to investigate the effect of down/up-sampling in the frequency axis. The other models use 1-D convolution/transposed-convolution layers with stride 2 for down/up-sampling. Every TDC block is a dense block with 5 composite layers with the growth rate 24 (used in dense blocks [18]). The kernel size of each convolution layer in a dense block is 3. Each TDF block is either single-layered or two-layered. The bottleneck factor bf of each TDF block is set to be 4. All models have 17 intermediate blocks except for two shallow models.

We summarize evaluation results in Table 1. The TDC block-based U-Net with sampling achieves an SDR of 4.86, the highest among the three models. Results show that the use of down/up-sampling in TDC-based U-Nets was significant, although the model without sampling can exploit higher resolution of internal representations. It may indicate that enlarging receptive fields via sampling may help the model to capture long-term dependencies better, and long-term dependencies are preferred over local features when distinguishing unique time-independent frequency patterns. (at least for these configurations).

Although FCNs can capture long-ranged patterns along the frequency domain, as mentioned in [8], TDF-based U-Nets did not perform well enough compared to the TDC-based models in a deep architecture. Among TDF-based models, the U-Net equipping single-layered TDFs (the fourth row of Table 1) outperforms the other models. However, it is notable that we can reduce parameters when we use two-layered TDFs. Also, we found that the TDF blocks can outperform TDC blocks in a shallow architecture (the third and sixth row of Table 1). The reason is that the U-Nets with few TDC blocks has a small receptive field, while a single TDF block has a full receptive field in the frequency dimension, which has led us to inject it in a time-frequency block instead of TDC (see §3.2.2).

4.3 U-Nets with Time-Frequency Blocks

We implemented U-Nets with time-frequency blocks. All models are trained on 3 seconds (128 STFT frames) of music. Since the number of frequency bins is much larger than the number of frames, models with more than 7 neural transforms use both 2×2 or 2×1 sized down/up-sampling layers to scale the frequency axis more than 3 times while maintaining the number of scales in the temporal axis to 3. Exceptionally, we use different down/up-sampling layers for one model to investigate the effect of down/up-sampling in the temporal axis.

We set every TFC block to have 5 convolution layers with kernel size 3×3. We set the growth rate to be 24, the same growth rate of §4.2. By using this TFC block configuration, we implemented a TFC-based U-Net (the first row of Table 2). We set the model in the second row to use different down/up-sampling layers to investigate the effect of down/up-sampling in the temporal axis. Every kernel size used in each down/up-sampling layer of this model is 2×1 to preserve the temporal resolution while scaling frequency resolution. The first two rows of Table 2 summarize the experiment results of two TFC-based models. The model that preserves the temporal resolution was slightly inferior to the other model. It is also notable that our U-Nets with TFC blocks achieve comparable results with state-of-the-art methods 4.4, even using lower frequency resolution. Compared to the frequency axis where the TDC-based U-Net with down/up-sampling outperforms the counterpart model, no significant SDR was gained by enlarging the receptive field by down/up-sampling.

We reused the configuration of TDC in of §4.2, for TDCs in TDC-RNN blocks. The RNN layers were implemented with bidirectional GRUs with a single hidden layer, which has $f/16$ hidden units, where f is the number of input frequency bins. Although having more parameters and a better potential for capturing long temporal dependencies than the two fully convolutional models, TDC-RNN performs lower than them. Increasing the number of hidden units or hidden layers could have increased SDR since many other state-of-the-art recurrent models use a hidden size that is at least 512. Increasing the number of STFT frames, thus training on longer clips of music, might have also worked. Although it performs the worst among the time-frequency blocks, it is superior to all the time-distributed blocks. It indicates that inter-frame operations are necessary for higher quality separation.

The fourth and fifth rows of the Table 2 shows promising results regarding the U-Nets with TFC-TDF blocks. We reused the same TFC setting above, and we set bf to be 16 for each TDF. The 7-blocked U-Net with TFC-TDFs outperforms the other 17-blocked models. These results show that inserting FCNs into intermediate blocks can be useful for MSS as well as for Speech Enhancement [8]. Also, results show that it is also achievable with fewer parameters by using FCNs with a bottleneck layer.

model	# parameters	SDR (vocals)
DGRU-DGConv	more than 1.9M	6.99
TAK1	1.22M	6.60
UMX	8.89M	6.32
TFC-TDF (small)	0.99M	**7.07** $\pm.08$
TFC-TDF (large)	2.24M	**7.98** $\pm.07$

Table 3. Comparison: SDR median value on test set.

esimation	n_fft	# blocks	# params	SDR
CaC	2048	7	0.99M	**7.07**
Mag	2048	7	0.99M	6.43
CaC	4096	9	2.24M	**7.98**
Mag	4096	9	2.24M	7.24

Table 4. Comparison of TFC-TDFs: CaC vs Mag

4.4 Comparison with SOTA models

We compare our models with other spectrogram-based models on the MUSDB benchmark. The first three rows of Table 3 shows the SDR performance of SOTA models, namely DGRU-DGConv [12], TAK1 [6], and UMX [22]. Their SDRs can be found in [12], SiSEC2018 repository [3], and UMX repository [4]. We estimated the lower bound of the number of parameters of DGRU-DGConv with 1-D CNN parameters without considering its GRUs.

Comparing with Table 2, we can see that our models perform comparably to or even outperform existing models even with less frequency resolution and fewer parameters. On top of that, our TFC extensions do not use recurrent layers, which is a key factor in the other previous models. It may lead to shorter forward/backward propagation time. Also, it is worth noting that previous models adopt Multichannel Wiener Filtering as a post-processing method to further enhance SDR. Ours directly use the signal reconstruction output without such post-processing.

For a fair comparison with SOTA models, we trained an additional U-Net with 9 TFC-TDF blocks (notated as 'large' in Table 3) with the same frequency resolution as the other SOTA models (FFT window size = 4096) and achieved outstanding results with a 0.9 dB gain over DGRU-DGConv.

4.5 Spectrogram Estimation: Complex vs Magnitude

For our final experiment, we see how much SDR was gained by extending a magnitude-only model into a CaC model. Our TFC-TDF-based U-Nets in Table 4 are compared to their magnitude-only form (referred to as 'Mag'). They use the same hyperparameter set except for $c_{in}^{(0)}$, the input/output number of channels. Mag also has an additional ReLU after the final 1×1 convolution to obtain non-negative-valued output spectrograms. Results show that training with raw STFT outputs instead of magnitudes significantly boosts SDR performance. It is also notable that the Mag model with n_fft of 4096 still outperforms all previous state-of-the-art models in Table 3.

[3] https://github.com/sigsep/sigsep-mus-2018

[4] https://github.com/sigsep/open-unmix-pytorch

4.6 Discussion: Developing Reusable Insights

Our work provides a practical guideline for choosing fundamental building blocks to develop an SVS or MSS model based on the U-Net architecture as follows.

- TDC-based models are sensitive to the number of blocks, compared to TDF-based models.
- Using down/up-sampling is important for CNN-based blocks, especially in the frequency dimension.
- Stacking 2-D CNNs is a simple but effective way to capture T and F features, compared to TDC-RNNs.
- Injecting a time-distributed block to a time-frequency block can improve SDR.
- A simple extension from a magnitude-only U-Net to a CaC U-Net can improve SDR.

Our work is not limited to the U-Net-architecture nor MSS. Blocks can be used as core components in more complex architectures as well. We can use different types of blocks for a single model, meaning that a lot of space remains for improvement. Also, our observations can be exploited in other MIR tasks such as Automatic Music Transcription (AMT) or Music Generation: for example, we expect that injecting TDFs to intermediate blocks for f_0 estimation model can improve performance since fully-connected layer can efficiently model long-range correlations such as harmonics.

5. CONCLUSION AND FUTURE WORKS

In this paper, we designed several types of blocks based on different design strategies. We implemented U-Net models with these blocks for SVS and evaluated their performance. Our experiments provide abundant material for future works by comparing several U-Nets with different types of blocks. Also, one of our models outperforms SOTA methods. For future work, we would like to extend this model to utilize attention networks for modeling long-term dependencies observed in both the frequency and the temporal axis.

6. ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2019R1F1A1062719, NRF-2020R1A2C1012624).

7. REFERENCES

[1] J. Andreas, H. Eric, M. Nicola, B. Rachel, K. Aparna, and W. Tillman, "Singing voice separation with deep u-net convolutional networks," in *18th International*

Society for Music Information Retrieval Conference, 2017, pp. 23–27.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[3] S. Park, T. Kim, K. Lee, and N. Kwak, "Music source separation using stacked hourglass networks," *arXiv preprint arXiv:1805.08559*, 2018.

[4] W. Yuan, S. Wang, X. Li, M. Unoki, and W. Wang, "A skip attention mechanism for monaural singing voice separation," *IEEE Signal Processing Letters*, vol. 26, no. 10, pp. 1481–1485, Oct 2019.

[5] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2017, pp. 21–25.

[6] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 106–110.

[7] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18 - a corpus for music separation," Dec. 2017, mUSDB18: a corpus for music source separation. [Online]. Available: https://hal.inria.fr/hal-02190845

[8] D. Yin, C. Luo, Z. Xiong, and W. Zeng, "Phasen: A phase-and-harmonics-aware speech enhancement network," *arXiv preprint arXiv:1911.04697*, 2019.

[9] N. Takahashi, P. Agrawal, N. Goswami, and Y. Mitsufuji, "Phasenet: Discretized phase modeling with deep neural networks for audio source separation." in *Interspeech*, 2018, pp. 2713–2717.

[10] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," in *International Conference on Learning Representations*, 2018.

[11] S.-W. Fu, T.-y. Hu, Y. Tsao, and X. Lu, "Complex spectrogram enhancement by convolutional neural network with multi-metrics learning," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2017, pp. 1–6.

[12] J.-Y. Liu and Y.-H. Yang, "Dilated convolution with dilated gru for music source separation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4718–4724. [Online]. Available: https://doi.org/10.24963/ijcai.2019/655

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[14] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[16] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *International conference on latent variable analysis and signal separation*. Springer, 2017, pp. 258–266.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[19] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, p. 8, 2012.

[20] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 261–265.

[21] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 293–305.

[22] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - A Reference Implementation for Music Source Separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, Sep. 2019. [Online]. Available: https://hal.inria.fr/hal-02293689