

MUSIC SOURCE SEPARATION BASED ON A LIGHTWEIGHT DEEP LEARNING FRAMEWORK (DTTNET: DUAL-PATH TFC-TDF UNET)

Junyu Chen^{*} Susmitha Vekkot[†] Pancham Shukla^{*}

^{*} Department of Computing, Imperial College London, UK

[†]Department of Electronics & Communication Engineering,
Amrita School of Engineering, Bengaluru, Amrita Vishwa Vidyapeetham, India

ABSTRACT

Music source separation (MSS) aims to extract ‘vocals’, ‘drums’, ‘bass’ and ‘other’ tracks from a piece of mixed music. While deep learning methods have shown impressive results, there is a trend toward larger models. In our paper, we introduce a novel and lightweight architecture called DTTNet¹, which is based on Dual-Path Module and Time-Frequency Convolutions Time-Distributed Fully-connected UNet (TFC-TDF UNet). DTTNet achieves 10.12 dB cSDR on ‘vocals’ compared to 10.01 dB reported for Bandsplit RNN (BSRNN) but with 86.7% fewer parameters. We also assess pattern-specific performance and model generalization for intricate audio patterns.

1. INTRODUCTION

Music source separation (MSS) separates a specific target waveform $s_{target} \in R^{C \times T}$ from a mixture waveform $s_{mix} \in R^{C \times T}$ [1]. This problem is similar to the “cocktail party effect” [2], where humans can focus on a specific speaker/instrument in a noisy environment. In particular, if the target waveform is vocal, then this subtask is named Singing Voice Separation (SVS) [3]. The separated vocals improve the performance of pitch tracking algorithms, which are vital for tasks like pitch correction and speech analysis [4].

In the realm of deep learning, MSS can be reformulated as a regression problem [5]. For the waveform domain models [6, 7], the input and output of the neural network are both audio waveforms in $R^{C \times T}$. For the frequency domain models [8], the model operates on the spectrogram from Short Time Fourier Transform (STFT). To recover the signal, the Inverse Short Time Fourier Transform (ISTFT) is applied to the predicted spectrogram. The input of these models used to be real-value spectrogram [9, 10]. But recent state-of-the-art models [11, 12, 13] focus on complex domain spectrogram. The studies in [14] show that complex spectrograms are well suited for improved Source-to-Distortion Ratio (SDR) over real-valued spectrograms.

The current state-of-the-art models for separating the ‘vocals’ track in MSS problems are Band-split Recurrent Neural Network (BSRNN) [12] and Time-Frequency Convolutions Time-Distributed Fully-connected UNet (TFC-TDF UNet) v3 [13]. BSRNN predicts a complex mask on the spectrogram and uses fully connected layers (FC) as well as multilayer perceptron (MLP) to encode and decode the features. The encoded features are further processed by 12 Dual-Path RNNs to capture the inter and intra dependencies across subbands. However, the FC and MLP layers introduce a large number of redundant parameters and the 12-layer Dual-Path RNNs require increased training time. TFC-TDF UNet v3 uses residual convolution blocks. Moreover, TFC-TDF UNet v3 does not introduce explicit time modeling, and therefore the performance gain is less prominent when the parameters of the models are increased drastically.

In this paper, we introduce a novel and lightweight framework called DTTNet, which is based on Dual-Path Module and TFC-TDF UNet v3. The contributions of this work are as follows:

1. As shown in Fig. 1, by integrating and optimizing the encoder and decoder from TFC-TDF UNet v3 and the latent Dual-Path module from BSRNN, we reduce redundant parameters.
2. As shown in Fig. 2(b), we partition the channels C within the improved Dual-Path module, which reduces the inference time.
3. We optimize hyper-parameters within DTTNet and improve the Signal-to-Distortion Ratio (SDR) which is comparable to BSRNN [12] and TFC-TDF UNet v3 [13] as shown in Table 3.
4. We test DTTNet with intricate audio patterns commonly misclassified by many models that are trained on MUSDB18-HQ dataset [15].

2. DUAL PATH TFC-TDF UNET

As depicted in Fig. 1, our framework consists of 3 parts: encoder, decoder and latent part. The encoder and decoder are connected through skip connections which are identified as

¹The code can be accessed from: <https://github.com/junyuchen-cjy/DTTNet-Pytorch>.

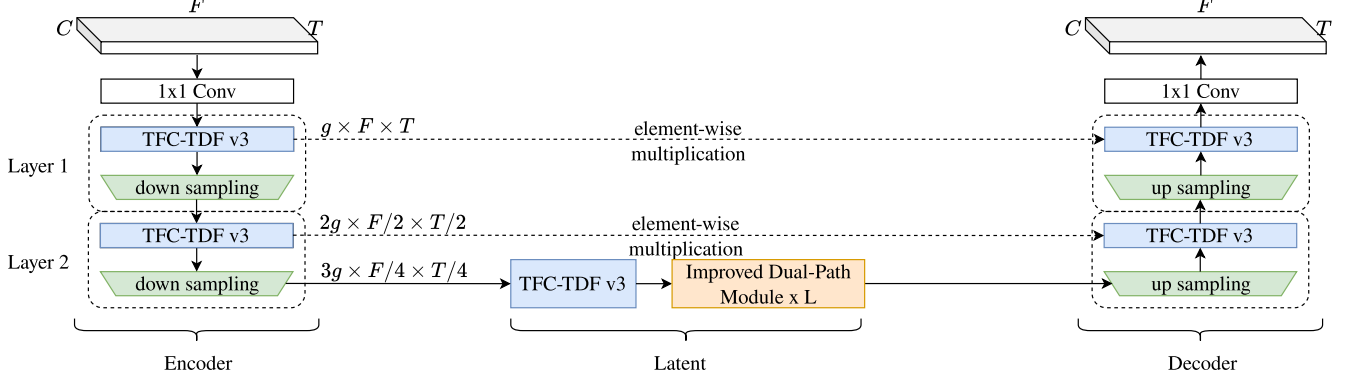


Fig. 1: A framework of Dual Path TFC-TDF UNet when layer depth $D = 2$, where L is the number of repeats of Improved Dual-Path Module (IDPM); C as the number of channels of input spectrogram with g being the channel incremental factor; T and F are the time and frequency axes that 2D convolution operates on.

element-wise multiplications on the dotted arrows. For each layer inside the encoder and decoder, we use the residual convolution block (TFC-TDF v3) [13]. The latent part is composed of a TFC-TDF v3 block and L layers of the Improved Dual-Path Module (IDPM).

2.1. Encoder

The encoder initially uses 1×1 convolution block to increase the number of channels from C to g . It is then, followed by D layers, wherein each layer contains a TFC-TDF v3 block followed by a down-sampling block which consists of 3×3 convolution layer to reduce the feature map by half and increase the number of input channels by g . The inner structure of TFC-TDF v3 in each layer is shown in Fig. 2(a). It has a Time-Frequency Convolutions (TFC) block, which contains 3 convolution blocks, followed by a residual Time-Distributed Fully-connected layer (TDF) where the frequency axis is first reduced by bottleneck factor bf and then recovered to the original input dimension. This is followed by another TFC block. Finally, a residual connection from a single convolution layer is added to the output.

2.2. Improved Dual-Path Module

The Improved Dual-Path Module (IDPM) has a similar structure as the Band and Sequence Module in BSRNN [12]. This module is repeated L times.

Fig. 2(b) shows the structure of a single IDPM. To reduce the inference time while maintaining a high input dimension size C , we first split the input channel C into H heads. Then, H heads are first processed along the time axis in the first RNN block and then along the frequency axis in the second RNN block. At the end of the IDPM, the reverse process is applied to merge the H heads into C channels.

In each RNN block, the split channels C' are first normalized by a group normalization layer (Group Norm) [16],

followed by a Bidirectional Long Short-Term Memory (BLSTM) [17] which has input size C' and hidden size $2C'$. The output size of BLSTM is $4C'$ and is reduced to C' by a fully connected layer (FC). Finally, the residual is added to the output.

2.3. Decoder

Similar to the structure of the encoder, each layer in the decoder block contains an up-sampling block, which is a single 3×3 transposed convolution block that up-samples the feature map by a factor of two and decreases the number of input channels by g . At each layer, the up-sampled output is multiplied by the feature map from the encoder and the multiplied feature map is refined by a TFC-TDF v3 block without changing any of the shapes in the feature map. Finally, a 1×1 convolution block is applied to reduce the channels from g to C .

3. GENERALIZATION TO OTHER AUDIO PATTERNS

In this section, we explore the generalization ability of DTTNet trained on the 'vocals' of MUSDB18-HQ [15] dataset for 5 intricate patterns, namely: *Wah Guitar* (26 min), *Horns* (1h 23 min), *Sirens* (2h 24 min), *Up-filters* (37 min), and *Vocal Chops* (42 min) taken from a bespoke dataset. The segments in each pattern are further divided into training (b-train), validation (b-valid), and test (b-test) sets in the ratios of 5:1:4. The sets consist of 4 to 8 second segments. The *Vocal Chops* is included as the prediction target, whereas the other 4 patterns are not considered as targets for prediction².

For each training mixture chunk $m_t \in R^{C \times T}$ in MUSDB18-HQ, we randomly sample a segment $seg \in R^{C \times T'}$ from

²MUSDB18-HQ classifies *Vocal Chops* as positive samples (e.g. 'PR - Happy Daze' in the test set). In the real-world scenario, they should be considered as accompaniment since pitch-tracking applications are only interested in human vocals.

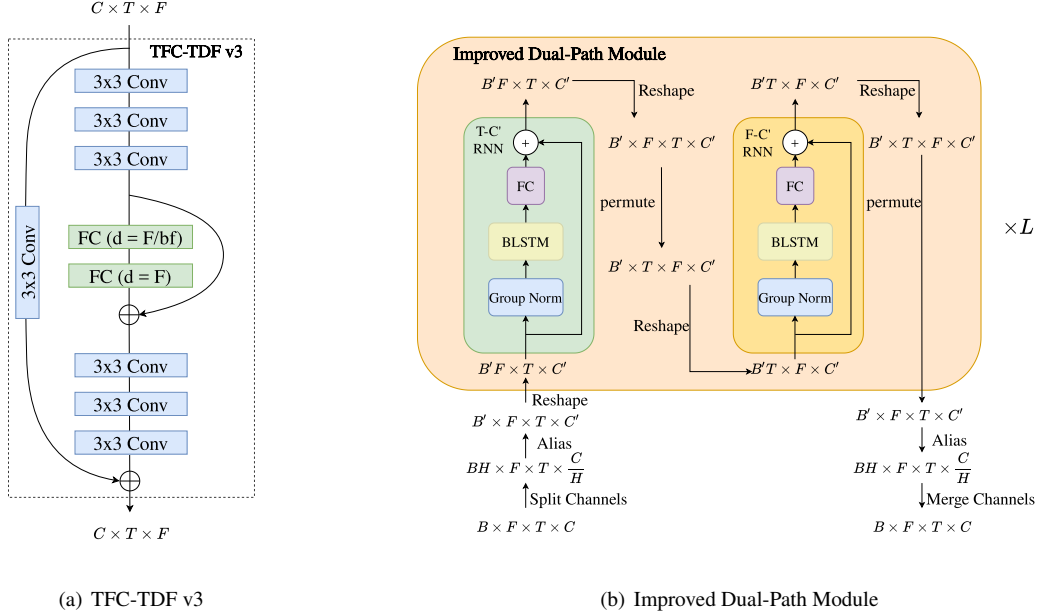


Fig. 2: Sub-blocks of DTTNet, where bf is the bottleneck factor of Time Distributed Fully-connected layer (TDF); B is the batch size; F is the number of features on the frequency axis; T is the number of features on the time axis; C is the number of channels generated by the convolution layer; L is the number of repeats of IDPM.

the b-train and the new mixture is defined as $m'_t = m_t + \text{pad_or_truncate}(\text{seg})$.

For each song $s_v \in R^{C \times T}$ in MUSDB18-HQ validation set, we randomly select a subset of segments $l_s \subset \{R^{C \times T'}\}$ from b-valid and create $s'_v = s_v + \text{concat_and_pad}(l_s)$ such that the 55% of $\text{concat_and_pad}(l_s)$ are intermediately padded with zeros. A similar process is followed for the test set.

Since *Vocal Chops* are pitched and distorted, the untuned DTTNet (DTT) is fine-tuned when sampling from b-train in two ways. The first one takes the pattern *Vocal Chops* into account (DTT + VC). The second one does not take *Vocal Chops* into account (DTT + NVC).

4. EXPERIMENT

4.1. Dataset

The MUSDB18-HQ dataset [15] consists of 150 songs, each sampled at 44100 Hz with 2 channels. Each song contains 4 independent tracks: 'vocals', 'drums', 'bass', and 'other'. For our experiment, the dataset is split into a training set with 86 songs, a test set with 50 songs, and the remaining 14 songs are left for hyperparameter tuning. For data augmentation, we use pitch shift semitone in $\{-2, -1, 0, 1, 2\}$ and time stretch with the percentage in $\{-20, -10, 0, 10, 20\}$.

4.2. Experimental Setup

We use AdamW optimizer with learning rate 2×10^{-4} for our model. The model is trained on 6 second chunks with L1 loss rather than L2 on waveforms, since some of the training data is unclean³. We use two A40 GPUs with batch sizes of 8 each. For MUSDB18-HQ training, the epoch size is set to 3240 chunks and the model is trained for 4082 epochs. For fine-tuning, the epoch size is set to 324 chunks and the model is trained for 300 epochs. In both scenarios, we picked the model that has the best Utterance-level SDR (uSDR) [5] on the validation set.

For STFT, we use a window size of 6144 and hop length 1024. For 'vocals', 'drums' and 'other', we cut the frequency bins to 2048 [11]. We set the bottleneck factor $bf = 8$ in TDF of Fig. 2(a). For 'bass', we cut the frequency bins to 864 and we set $bf = 2$ in TDF. For IDPM, we set $L = 4$ and ensure that each Group Norm has 16 channels [16]. We set $g = 32$ for the up/down-sampling layers as shown in Fig. 1.

4.3. Evaluation Metrics

We use Source-to-Distortion Ratio (SDR) for performance measurement.

Chunk-level SDR (cSDR) [1] calculates SDR on 1 second chunks and reports the median value.

Utterance-level SDR (uSDR) [5] calculates the SDR for each song and reports its mean value.

³E.g. 'The So So Glos - Emergency' and 'The Districts - Vermont' at 1:44.

5. RESULTS AND DISCUSSION

5.1. Hyper-parameters and Performance

In this section, we experiment with different combinations of hyper-parameters and study the impact on the uSDR and inference time. The results are presented in Table 1.

TFC-TDF Block We noticed that TFC-TDF v3 residual block is more effective than TFC-TDF v2, significantly improving uSDR with minimal impact on training time.

Heads and Channels Increasing the number of channels g from 32 to 48 offers slightly better performance but results in 4x more inference time. It was observed that inference time is appreciably reduced by setting $H = 2$ compared to $H = 1$ provided $g = 32$. However, it takes double the epochs to attain the same level of uSDR while maintaining similar training times.

Layers of IDPM Increasing the repeats L of IDPM significantly increases the inference time, however, the SDRs tend to decrease. Hence, setting $L = 4$ is sufficient to achieve good SDRs and inference time as compared to $L = 10$.

Table 1: Validation Set uSDR for ‘vocals’. Experiments are carried out on mono audio (mid-channel) with 1200 epochs. The inference time is measured on a single RTX 3090 (24 GB) with a 3 minute input audio. Batch size is set to 4.

g	H	L	block	uSDR	inference time
32	2	4	v2	10.14	1.77 s
48	2	4	v2	10.47	15.80 s
32	1	4	v2	10.26	12.96 s
32	1	10	v2	10.16	30.65 s
32	2	4	v3	10.51	2.39 s

5.2. Study on Generalization Ability

As depicted in Table 2, for untuned DTTNet, the uSDRs of the intricate test patterns in the bespoke dataset are lower compared to MUSDB18-HQ test set. The uSDR for *All* patterns (MUSDB18-HQ + intricate patterns) is 5.64 dB lower than the MUSDB18-HQ for untuned DTTNet.

For each pattern, DTT + VC outperforms DTT + NVC. On the MUSDB18-HQ test set, we observed 0.07 dB uSDR improvement for DTT + VC compared to DTT. When considering all the patterns, the uSDR for DTT + VC is 5.67 dB higher than the uSDR of DTT possibly because of the diversity of the patterns during training. Additionally, for *Vocal Chops*, DTT + NVC shows lower uSDR possibly because this pattern does not appear in the training data leading to overfitting.

Furthermore, although we obtained 0.07 dB uSDR improvement on the MUSDB18-HQ test set using DTT + VC, cSDR drops by 0.12 dB compared to DTT. This suggests that our bespoke dataset is slightly smaller, posing a risk of overfitting.

Table 2: uSDR results on various test sets, with the best performance highlighted in boldface.

Test Set	DTT	DTT + NVC	DTT + VC
MUSDB18-HQ	10.15	10.07	10.22
+ Wah Guitar	7.97	9.73	10.04
+ Horns	8.23	9.95	10.14
+ Sirens	6.07	9.71	9.91
+ Up-filters	8.56	9.72	9.97
+ Vocal Chops	9.90	9.10	10.87
+ All	4.51	8.38	10.18

5.3. Comparison with the State-of-the-art (SOTA)

As indicated in Table 3, our DTTNet achieves higher cSDR on the ‘vocals’ track against BSRNN (SOTA) with only 13.3% of its parameter size. Moreover, we also achieved a higher cSDR on the ‘other’ track compared to TFC-TDF UNet v3 (SOTA) with only 28.6% of its parameter size.

Table 3: cSDR in dB on MUSDB18-HQ Test Set. For parameter size (Params), we measure the single source model times the number of sources. * uses bag of 4 models. The parameter of † is measured based on the available code.⁴

Model	vocals	drums	bass	other	Params
ResUNetDecouple+ [14]	8.98	6.62	6.04	5.29	102.0 M × 4
CWS-PResUNet [18]	8.9	6.38	5.93	5.84	202.6 M × 4
Hybrid Demucs (v3)* [19]	8.13	8.24	8.76	5.59	83.6 M × 4
HT Demucs [20]	7.93	7.94	8.48	5.72	42.0 M
BSRNN† [12]	10.01	9.01	7.22	6.70	37.6 M × 4
KUIELab-MDX-Net [11]	8.97	7.20	7.83	5.90	7.4 M × 4
TFC-TDF UNet v3 [13]	9.59	8.44	8.45	6.86	70.0 M
DTTNet (Proposed)	10.12	7.74	7.45	6.92	5.0 M × 4

6. CONCLUSION

We introduce DTTNet, a novel and lightweight framework with higher cSDR but with reduced parameter size compared to both BSRNN and TFC-TDF UNet v3 for the ‘vocals’ and ‘other’ track music source separation respectively. Furthermore, we created a bespoke dataset of intricate patterns such as *Vocal Chops* and tested the generalization ability of DTTNet.

In our future work, we plan to improve our framework to enhance the SDR for the ‘drums’ and ‘bass’ tracks. Additionally, we plan to integrate zero-shot systems [21] as a post-processing module to improve the generalization ability.

7. ACKNOWLEDGEMENT

We acknowledge Dr. Lorenzo Picinali for his feedback on the initial background work, and Dr. Aidan O. T. Hogg and Shiliang Chen for writing and submission tips for ICASSP.

⁴<https://github.com/amanteur/BandSplitRNN-Pytorch>

8. REFERENCES

- [1] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito, “The 2018 signal separation evaluation campaign,” in *Latent Variable Analysis and Signal Separation: 14th Int. Conf.*, 2018, pp. 293–305.
- [2] E. Colin Cherry, “Some experiments on the recognition of speech, with one and with two ears,” *The Journal of the Acoustic Society of America*, p. 975–979, 1953.
- [3] Woosung Choi, Minseok Kim, Jaehwa Chung, Daewon Lee, and Soonyoung Jung, “Investigating u-nets with various intermediate blocks for spectrogram-based singing voice separation,” in *21th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2020.
- [4] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello, “Crepe: A convolutional representation for pitch estimation,” in *IEEE Int. Conf. Acoustics, Speech Signal Process (ICASSP)*. IEEE, 2018, pp. 161–165.
- [5] Yuki Mitsufuji, Giorgio Fabbro, Stefan Uhlich, Fabian-Robert Stöter, Alexandre Défossez, Minseok Kim, Woosung Choi, Chin-Yun Yu, and Kin-Wai Cheuk, “Music demixing challenge 2021,” *Frontiers in Signal Processing*, vol. 1, 2022.
- [6] Daniel Stoller, Sebastian Ewert, and Simon Dixon, “Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation,” in *Proc. of the 19th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2018, pp. 334–340.
- [7] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis R. Bach, “Music source separation in the waveform domain,” *arXiv preprint arXiv:1911.13254*, 2019.
- [8] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, “Open-unmix - a reference implementation for music source separation,” *Journal of Open Source Software*, 2019.
- [9] Romain Hennequin, Anis Khelif, Felix Voituret, and Manuel Moussallam, “Spleeter: a fast and efficient music source separation tool with pre-trained models,” *Journal of Open Source Software*, vol. 5, no. 50, pp. 2154, 2020.
- [10] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji, “Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation,” in *2018 16th Int. Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 106–110.
- [11] Minseok Kim, Woosung Choi, Jaehwa Chung, Daewon Lee, and Soonyoung Jung, “KUIELab-MDX-Net: A Two-Stream Neural Network for Music Demixing,” in *Proc. of the MDX Workshop at ISMIR*, 2021.
- [12] Yi Luo and Jianwei Yu, “Music source separation with band-split rnn,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1893–1901, 2023.
- [13] Minseok Kim, Jun Hyung Lee, and Soonyoung Jung, “Sound Demixing Challenge 2023 Music Demixing Track Technical Report: TFC-TDF-UNet v3,” *arXiv preprint: arXiv:2306.09382*, 2023.
- [14] Qiuqiang Kong, Yin Cao, Haohe Liu, Keunwoo Choi, and Yuxuan Wang, “Decoupling magnitude and phase estimation with deep resunet for music source separation,” in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2021.
- [15] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner, “The MUSDB18 corpus for music separation,” 2017.
- [16] Yuxin Wu and Kaiming He, “Group normalization,” *Int. Journal of Computer Vision (IJCV)*, vol. 128, no. 3, pp. 742–755, 2020.
- [17] M. Schuster and K.K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [18] Haohe Liu, Qiuqiang Kong, and Jiafeng Liu, “CWS-PResUNet: Music Source Separation with Channel-wise Subband Phase-aware ResUNet,” in *Proc. of the MDX Workshop at ISMIR*, 2021.
- [19] Alexandre Défossez, “Hybrid spectrogram and waveform source separation,” in *Proc. of the MDX Workshop at ISMIR*, 2021.
- [20] Simon Rouard, Francisco Massa, and Alexandre Défossez, “Hybrid transformers for music source separation,” in *IEEE Int. Conf. Acoustics, Speech Signal Process (ICASSP)*. IEEE, 2023, pp. 1–5.
- [21] Ke Chen*, Xingjian Du*, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, “Zero-shot audio source separation through query-based learning from weakly-labeled data,” in *AAAI*, 2022, pp. 4441–4449.