Lead and Accompaniment Separation

Chi Kwan Cyrus Wong Chunshen Ding Frederico Aberle Shuyuan Zhang

***Abstract*—The task of separating the lead vocal from the accompanying instrumentals in a music mixture is a significant challenge in the field of music information retrieval. Over algorithmic methods, deep learning methods have emerged as the dominant approach for addressing this task. In this report, we present two novel attention-based variants of the TFC-TDF-UNet architecture, namely TFC-TDSA-UNet and TFC-TDT-UNet, which aim to further improve the performance of music source separation. The models adopt the Complex as Channel (CaC) framework to estimate the vocal isolated spectrogram. Experiments on MUSDB18-HQ benchmark show the superior training speed of the UNet architecture, and demonstrate the effectiveness of attention mechanism on lead and accompaniment separation[1].**

***Index Terms*—Music source separation, Deep learning**

I. Background

Lead and accompaniment separation is a downstream task for music source separation, aiming to separate the lead vocal from the accompaniment instrumentals given a music mixture [1]. Extracting the leading vocal from a piece of music is useful in multiple applications, including remixing, karaoke generation, and music information retrieval. Music source separation is inherently a challenging task as frequency and temporal components are intertwined between sources. Traditional algorithmic approaches often require genre specific expert knowledge to implement; generally fall short in handling more complex music mixture.

In recent years, deep learning methods like [1] and [4] has been leading the research in lead and accompaniment separation and demonstrated successful outcomes. This class of methods rely on large amount of annotated data where the mixture and its corresponding lead and accompaniment components are provided. By training a model on these data, the model will capture the relationship between the input mixture and target outputs, eventually become capable of isolating the lead and accompaniment from an unseen music mixture.

[1]Python codes for this project is available on the GitHub repository https://github.com/cyruss081115/LeadAccomSeparation

The current state-of-the-art models on the music separation benchmark MUSDB18-HQ without addtional training samples [5] are TFC-TDF-UNet [1] and Bandsplit-RNN [4]. TFC-TDF-UNet is a UNet-based model that estimates the vocal separated spectrogram using a contracting path, an expanding path, and skip connections [1]. The model learns the temporal and spectral features using Convolutional Neural Networks (CNN) and Fully Connected Networks. It adopts a Complex as Channel framework that takes into account both real and complex channels of a spectrogram [1]. The other model, Bandsplit-RNN [4], is a Recurrent Neural Network (RNN) based architecture that learns temporal and spectral characteristics with Bidirectional Long-Short Term Memory blocks [6]. The model estimates a mask to isolate the singing voice from a spectrogram by splitting the frequency dimension into multiple bands for separate consideration.

In this study, we draw inspirations on the work of TFC-TDF-UNet and the achievements of attention mechanism [2] across various deep learning domains. We implement two attention-based variants of the TFC-TDF-UNet, namely TFC-TDSA-UNet using Vanilla Self Attention blocks and TFC-TDT-UNet using Transformer blocks. The performance of our variant was demonstrated by training and comparing them with baseline models on the MUSDB18-HQ benchmark.

II. Methodologies

A. The Complex as Channel Framework

In this section we refer to the ”Complex as Channel framework” (CaC) [1], a spectrogram-based Singing voice separation (SVS) framework. It consists of a Complex-valued Spectrogram Estimation Network, which we will later substitute with the convolutional neural network ”U-Net”. This Complex-valued Spectrogram Estimation Network takes as an input a c-channeled mixture signal, and outputs a c-channeled singing voice signal, where c denotes the number of channels indicating that it is a multi-channel input and output signal. Those three building blocks make up the framework which we will now further discuss:

1) The spectrogram extraction layer is our preprocessing part. It consists of two parts: The first part is the extraction of a spectrogram. This is done by STFT by applying it to the c-channeled mixture signal and getting a complex-valued spectrogram with c-channels as a result. The second part consists of the reshaping of the complex-valued spectrogram. There the imaginary part of the complex-valued spectrogram is treated as another real-valued spectrogram with c-channels that is concatenated with the real part of the complex-valued spectrogram. Effectively, the amount of channels is doubled into a single real-valued spectrogram ordered by channels, time, and frequency. In Matrix notation this looks as follows: The complex-valued spectrogram $M_{complex} \in \mathbb{C}^{c \times T \times F}$ is reshaped to the (2c)-channeled real-valued $M \in \mathbb{R}^{(2c) \times T \times F}$, where T denotes the number of frames and F denotes the number of the frequency segments in the spectrogram. The approach of treating the real and imaginary part of a spectrogram as two separate real-valued spectrograms has the advantages that it can store both the magnitude and phase of a signal in the spectrogram, whereas other models only use the magnitude for the input of their networks after decomposition and thus not fully leverage the input signal's information.
2) The complex-valued spectrogram estimation network layer serves as a black box for now, which consists of a convolutional neural network called U-Net. Using supervised learning we try to make an estimate, let us call it $\hat{T} \in \mathbb{R}^{(2c) \times T \times F}$, that approximates the output spectrogram $T \in \mathbb{R}^{(2c) \times T \times F}$ that we assume is true and is taken as a reference point.
3) The signal reconstruction layer does the exact opposite as the spectrogram extraction layer: It slices a (2c)-channeled real-valued spectrogram into a complex-valued spectrogram. By applying inverse-STFT on the complex-valued spectrogram a signal can be generated once again that should resemble as much as possible the singing voice only of the mixture signal in the beginning.

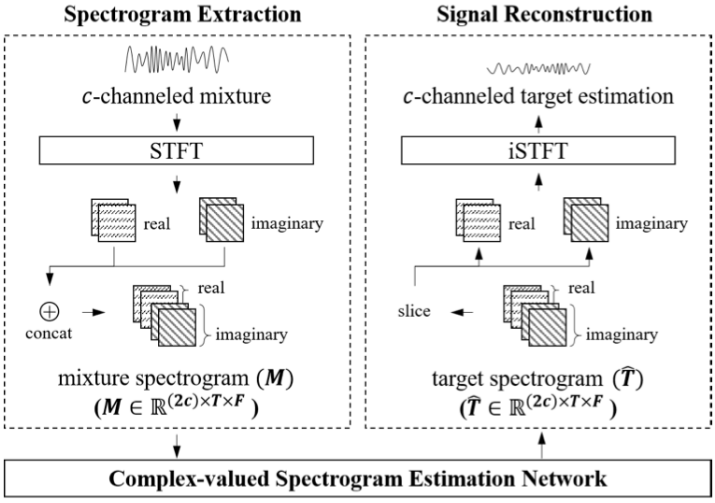

Fig. 1. Pre- and post processing

B. CNN U-Net for Biomedical Image Segmentation

In this section we explain the U-Net, the neural network we will use as the second building block for the CaC framework. But first we investigate where it originates from before we adapt it to our Spectrogram Estimation. The U-Net was first used for biomedical image segmentation such as the segmentation of cells in microscopic images [8]. The network architecture is organized into two parts the contracting path (left side) and the expansive path (right side). The main idea of the U-Net or any other segmentation network is to precisely identify the pixels that belong to a segment while retaining global context, i.e. information about neighboring pixels. The contracting path is used for capturing the context. It performs a series of 3x3 convolutions followed by a rectified linear unit (ReLU) and 2x2 max-pooling operation, where ReLU is an activation function that maps positive input to itself and negative input to zero, and max-pooling is a down-sampling method by partitioning the image into 2x2 blocks and taking the maximum value of each block. Notice how the dimensions of the feature are halved and the number of feature channels are doubled by the max-pooling operation. When repeating convolutions, ReLU and max-pooling, each time the image looses detail. The convolution operation tries to extract the pixels belonging to edges and the max-pooling operation aggregates 2x2 pixels making the edges appear more sharp. Convolution involves applying a kernel, here a 3x3 matrix, in a sliding-window fashion to the image to perform element-wise multiplication of each participating pixel within the window with the values of the kernel. All the multiplications are summed together into a final value that describes how much the pixel belongs to an edge or not. Throughout the learning process, the network adjusts the parameters of the kernel and gets better at estimating the edges. In the second part, the expansive path, the performs up-sampling of the image. Here everything

remains the same as in the contracting path, except that the max-pooling operation is replaced by a 2x2 up-convolution. In this part, the network tries to reconstruct the image by gaining more and more detail through each up-convolution and the help of contracting path. Namely, after the up-convolution the up-sampled image is concatenated with the image from the contracting path at same level. With the details of the images from the contracting path the pixels of segments can be precisely localized while retaining the global context from the previous result in the contracting path. Notice here how the dimensions are doubled, i.e. the resolution is increased, while the number of feature channels are halved. A final 1x1 convolution maps the result of the remaining feature channels to a final classification value for each pixel. In contrast to regular CNNs that solve biomedical image segmentation U-Net performs much more precise because it leverages both accuracy and context. In the first CNNs, the network was applied to sliding patches, usually 32x32 blocks of pixels, to make us of context. But there we cannot leverage both at same time. Once the patch is too large we loose localization due to more max-pooling operations, and if the patch is too small we loose context. U-Net is also a lot faster than other CNNs because there the networks run for each patch and also the patches overlap creating redundant computations for the overlapping pixels. To remedy the problem of a too large image we can partition an image into tiles (in the figure of size 572x572 pixels) and let the network run on each one of them. In order to solve the limited size of training images, U-Net makes heavy use of data augmentation. It takes current images and distorts them to create more diverse data.

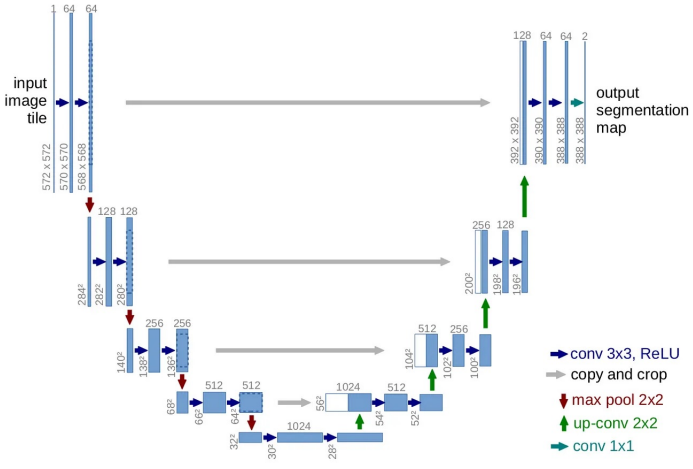

Fig. 2. U-Net for Biomedical Image Segmentation.

C. CNN U-Net for Spectrogram Estimation

The U-Net Architecture for Spectrogram Estimation is also build with an Encoder and Decoder [1]. The Encoder consists of Downsampling layers, where either the time dimension, the frequency dimension or the time-frequency dimension is halved in its size. This corresponds to the max-pooling operation in the segmentation example. The Decoder is responsible for Upsampling. Here the same happens as in the Encoder, just now with doubling all its scales. The result is concatenated with the output of feature maps from the intermediate block, which we will treat as a black box for now again. But the intermediate block represents the two convolution operations followed by a ReLU operation from the segmentation example. It takes as an input a spectrogram-like tensor and outputs a tensor of the same size with a potentially different number of channels. The architecture also consists of two additional layers in the beginning and in the end, which perform a 1x2 convolution. The first layer also perform a ReLU, but not the last layer because there can be negative time-frequency bins in the tensor. The convolution layers are used to increase or decrease the number of channels respectively.

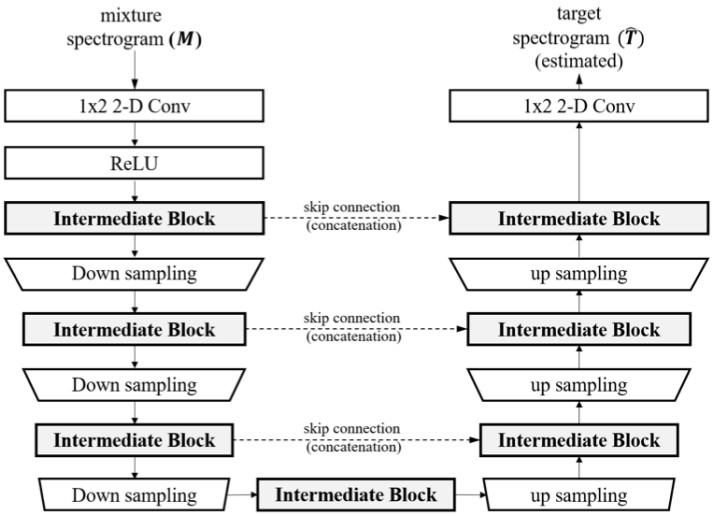

Fig. 3. U-Net for Spectrogram Estimation.

D. Basic Blocks

In this section, we discuss the details of basic blocks that is used to build an intermediate block.

1) Time Frequency Convolution Block: The Time Frequency Convolution Block (TFC) [1] take into account both time and frequency domain. It has a DenseNet-like structure. Each dense layer has a Batch Normalization

layer, a 2-D convolution layer, and a ReLU activation layer. Every layer is densely connected such that the input of a layer is the concatenation in the channel dimension of the output of all previous layers. The block is applied to a spectrogram-like input in the time-frequency domain to jointly model temporal and spectral features.

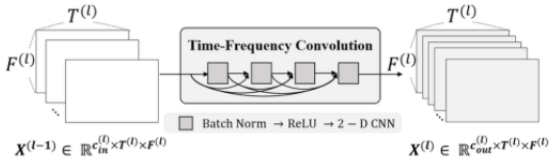

Fig. 4. Time-Frequency Convolution Block, adopted from [1]

2) Time Distributed Fully Connected Block: The Time Distributed Fully Connected Block (TDF) [1] belongs to the time-distributed block family that specialize at capturing long-range temporal features where time-frequency blocks are often unable to. Given an input $X \in \mathbb{R}^{c \times T \times F}$, where c denotes the channel dimension, T denotes the temporal dimension, and F denotes the frequency dimension, TDF is time-distributed such that the network is applied separately and identically to each frequency frame, i.e. $X[i, j, :]$. A layer in TDF consist of a Batch Normalization, a Fully Connected Network, and a ReLU activation layer. On multiple-layered settings, a bottleneck factor is introduced to determine the dimensions of the hidden space.

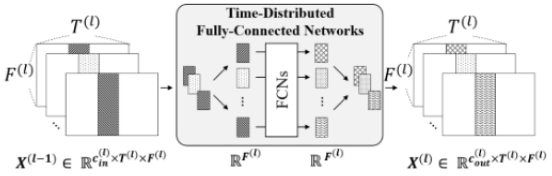

Fig. 5. Time-Frequency Fully Connected Block, adopted from [1]

3) Time Distributed Self Attention Block: The Time Distributed Self Attention Block (TDSA) is a variant of the TDF block where the fully connected layers are replaced by vanilla self attention layers [2]. It is a naive adoption of the attention mechanism into a Time Distributed block. Given an input $X \in \mathbb{R}^{1 \times 1 \times F}$, self attention maps the input to query $Q \in \mathbb{R}^{1 \times 1 \times F}$, key $K \in \mathbb{R}^{1 \times 1 \times F}$, and value $V \in \mathbb{R}^{1 \times 1 \times F}$ using trainable weight matrices W^Q, W^K, and W^K respectively. Scaled attention scores are given by $\frac{Q \times K^T}{\sqrt{F}}$. Attention weights are defined as the softmax values of scaled attention scores, yielding the final output as:

$$\text{Self-Attention} = \text{Softmax}(\frac{Q \times K^T}{\sqrt{F}})V \tag{1}$$

4) Time Distributed Transformer Block: Since vanilla self attention in TDSA treats each input position indifferently, the Time Distributed Transformer Block (TDT) adds positional encoding prior to the input of network and attaches an additional feed forward layer after self attention. The positional encoding introduces non-linear variations to the input according to their positions in the input sequence. We use the absolute positional encoding presented in [2]:

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2}$$

We also modified the vanilla self attention such that the network is applied on both the channel and frequency dimensions.

E. Model architecture

The proposed models TFC-TDSA-UNet and TFC-TDT-UNet have the identical architecture as its predecessor TFC-TDF-UNet [1] with the only difference that the intermediate blocks are replaced with TFC-TDSA blocks and TFC-TDT blocks respectively. Figure 6 illustrates an intermediate block. In TFC-TDSA blocks, the output of a Time Frequency Convolution block is followed by a Time Distributed Self Attention Block while that is followed by a Time Distributed Transformer Block in TFC-TDT.

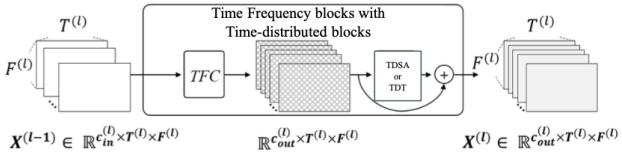

Fig. 6. Structure of Intermediate blocks, adopted from [1]

III. Experiments

A. Dataset

Training and testing data are obtained from the MUSDB18-HQ dataset [3]. The dataset contains 100 training tracks and 50 testing tracks. All tracks are stereo and sampled at 44100 Hz. We use the vocal and mixture audio from each track.

TABLE I
EVALUATION RESULTS OF ALL MODELS

Model	Number of parameters	MSE Loss ↓
OpenUnmix	9.46M	0.946
TFC-TDF-UNet	1.80M	0.680
TFC-TDSA-UNet	5.29M	0.773
TFC-TDT-UNet	0.08M	**0.574**

B. Data processing

Due to memory restrictions, we chunk the audio into 3 seconds and randomly sample a chunk from a randomly selected track in the dataset during training. On average, 64 samples per track are used in 1 epoch. To further exploit the dataset, we augment the data by applying a random gain between 0.25 and 1.25 and randomly swap the channels.

C. Model Configurations

We implement OpenUnmix [7], TFC-TDF-UNet, TFC-TDSA-UNet and TFC-TDT-UNet in Pytorch. We set the STFT window size to 2048 and the hop size to 1024. For OpenUnmix, the hidden state dimension is 512. For the UNet models, they have a depth of 3 and the basic blocks in the intermediate blocks are single layered. The internal channels are set to 24 and a kernel size of (3, 3) is used.

D. Training and Evaluation

The models are optimised on the mean squared error between the target spectrogram and the expected spectrogram using the Adam optimiser with a weight decay of 0.000001. The learning rate is scheduled using the ReduceLROnPlateau scheduler in Pytorch with a factor of 0.3 and an initial rate of 0.001. All models are trained on RTX3090 with a batch size of 8 for 13 epochs. The average validation loss values across all epochs are reported in table I.

IV. RESULTS

A. OpenUnmix and UNet models

We use OpenUnmix [7], a RNN based architecture as the baseline of our methods. Figure 7 plots the training loss and validation loss of OpenUnmix with the average of TFC-TDF-UNet, TFC-TDSA-UNet and TFC-TDT-UNet. The results demonstrated the rapid convergence of UNet models and their superior performance in terms of validation loss. We attribute this performance to the skip connections and dense connections within the UNet models, which encourage strong gradient flow and facilitate faster convergence.

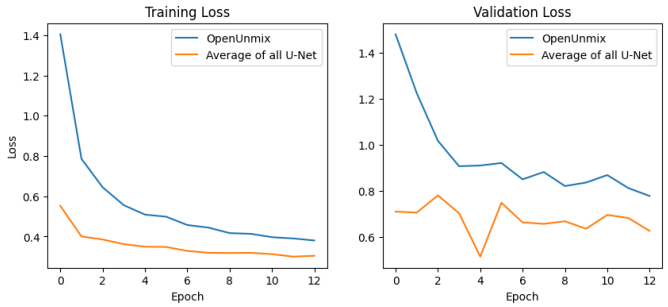

Fig. 7. Plotted losses of OpenUnmix and average of other UNet models.

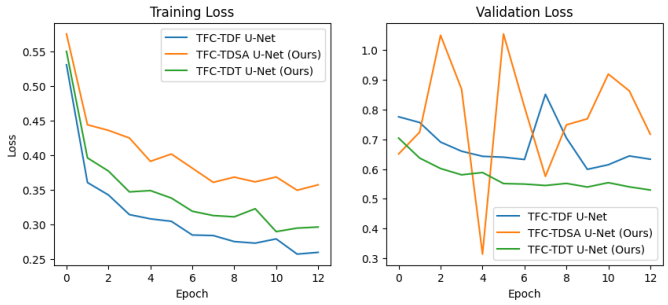

Fig. 8. Plotted losses of TFC-TDF-UNet, TFC-TDSA-UNet and TFC-TDT-UNet.

B. UNet with different Intermediate blocks

We implement our proposed attention-based intermediate block variations, namely TFC-TDSA and TFC-TDT, as well as the original TFC-TDF block. By inserting the corresponding intermediate blocks into the UNet framework as shown in figure 3, we construct TFC-TDSA-UNet, TFC-TDT-UNet, and TFC-TDF-UNet for comparison. Figure 8 illustrates the training and validation losses of the UNet models. Regarding training loss, it is observed that the original TFC-TDF-UNet has the fastest reduction rate, while TFC-TDSA-UNet has the slowest. As for validation loss, TFC-TDT-UNet performs the best, with the lowest average validation loss and the highest level of stability. TFC-TDF-UNet shows a less stable validation loss trace and higher average loss. Unfortunately, TFC-TDSA-UNet demonstrated the worst performance with significant loss fluctuations and the highest loss.

Overall, it is evident that the TFC-TDT-UNet model can generalise the task of vocal separation more effectively than other models due to its efficient attention mechanism implementation. Conversely, the TFC-TDSA-UNet model, which has a naive implementation, performs worse than a simple fully connected network.

Despite the inferior performance of TFC-TDSA-UNet among UNet models, it is worth noting that it outperformed OpenUnmix.

V. Conclusion

In this project, we explored the results of different implementation of attention-mechanism to the task of lead and accompaniment separation. Our experiments show that UNet architecture has faster model convergence, and with an effective implementation, attention-mechanism is able to further elevate the model performance.

VI. Labour distribution

Chi Kwan Cyrus Wong: Implement models in python, responsible for Basic Blocks, Model architecture, Experiments and Results of this report
Chunshen Ding: Experiments on GPU, Slides making, presentation script and demo video
Frederico Aberle: The Complex as Channel Framework, U-Net for Biomedical Image Segmentation, U-Net for Spectrogram Estimation of this report
Shuyuan Zhang: ...

References

[1] W. Choi, M. Kim, J. Chung, D. Lee, and S. Jung, "Investigating U-Nets with various Intermediate Blocks for Spectrogram-based Singing Voice Separation," arXiv.org, Oct. 08, 2020.

[2] A. Vaswani et al., "Attention Is All You Need," arXiv.org, Jun. 12, 2017. https://arxiv.org/abs/1706.03762

[3] Z. Rafii, A. Liutkus, Fabian-Robert Stöter, S. I. Mimilakisand R. Bittner, 'MUSDB18 - a corpus for music separation'. Zenodo, Dec. 17, 2017. doi: 10.5281/zenodo.1117372.

[4] Y. Luo and J. Yu, "Music Source Separation with Band-split RNN," arXiv.org, Sep. 29, 2022.

[5] "Papers with Code - MUSDB18-HQ Benchmark (Music Source Separation)," paperswithcode.com. https://paperswithcode.com/sota/music-source-separation-on-musdb18-hq (accessed Dec. 15, 2023).

[6] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," arXiv:1508.01991 [cs], Aug. 2015.

[7] 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, doi: https://doi.org/10.21105/joss.01667.

[8] 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.