Channel-wise Subband Input for Better Voice and Accompaniment Separation on High Resolution Music

Haohe Liu, Lei Xie*, Jian Wu, Geng Yang

Audio, Speech and Language Processing Group (ASLP@NPU), School of Computer Science, Northwestern Polytechnical University, Xian, China
haoheliu@gmail.com, {lxie, jianwu, gengyang}@nwpu-aslp.org

Abstract
This paper presents a new input format, channel-wise subband input (CWS), for convolutional neural networks (CNN) based music source separation (MSS) models in the frequency domain. We aim to address the major issues in CNN-based high-resolution MSS model: high computational cost and weight sharing between distinctly different bands. Specifically, in this paper, we decompose the input mixture spectra into several bands and concatenate them channel-wise as the model input. The proposed approach enables effective weight sharing in each subband and introduces more flexibility between channels. For comparison purposes, we perform voice and accompaniment separation (VAS) on models with different scales, architectures, and CWS settings. Experiments show that the CWS input is beneficial in many aspects. We evaluate our method on musdb18hq test set, focusing on SDR, SIR and SAR metrics. Among all our experiments, CWS enables models to obtain 6.9% performance gain on the average metrics. With even a smaller number of parameters, less training data, and shorter training time, our MDenseNet with 8-bands CWS input still surpasses the original MMDenseNet with a large margin. Moreover, CWS also reduces computational cost and training time to a large extent.

Index Terms: voice and accompaniment separation, deep learning, subband, music source separation

1. Introduction
Music Source Separation (MSS) has raised much interest in recent years. The goal of the task is blindly separate sources from a mixed track, for example vocal, drums, bass and accompaniment. In this paper, we particularly focus on the voice and accompaniment separation (VAS) from a mixture. As a practical tool, separating these two components allows us to remix, suppress or up-mix the sources [1]. VAS can also facilitate automatic transcription, karaoke track generating as well as music information retrieval [2].

High-resolution music usually sounds better but suffers from high computational cost in the VAS task. For example, 44.1kHz is a commonly used sample rate for music, while many high-quality formats may be up to 48kHz or even higher. However, due to the high-computational cost, many of the current VAS studies perform downsampling in advance. For instance, the approach using M-U-Net [3] downsamples the audio to 10.88kHz before processing and Dense-Unet only works on 16kHz music in [4]. The downsampling process seriously affects the auditory quality to the separated vocal and accompaniment in practical applications.

Convolutional Neural Networks (CNN) has shown tremendous success in multiple fields, especially image-related tasks. The input data for these tasks, such as image classification, usually have problem that the position of a certain object is not fixed. Mechanisms like local receptive fields and shared weights [5] enable CNN to become position invariant, which means once a feature has been detected, its exact location becomes less important [5]. In audio processing, most of the state-of-the-art (SOTA) MSS models are also based on convolutional networks, like Deep-Unet [6], which have shown considerable improvements over the traditional methods. Although CNN-based architecture has demonstrated effectiveness on MSS tasks in frequency domain, it still has apparent limitations. Frequency spectrogram based SOTA models trained on high-resolution audio, e.g. TPC-TIF [7], usually take the whole spectrogram as the input feature. In this case, they assume each frequency band has filter parameters to share and are equally important. However, local patterns are usually different between bands [8], as can be seen in Fig 1. This means different bands do not necessarily need the same set of filters (in CNN) for parameter sharing. Hence, treating different frequency bands differently might better facilitate the separation process.

Some prior efforts have already emphasized the difference between bands. In [9], Taghia et al. first took the subband decomposition, and then they used a hybrid system of empirical mode decomposition [10] and principle component analysis to construct artificial observations from the single mixture. Finally a synthesis process was used to reconstruct full band signal. Takahashi et al. [11] also noticed the problem of global kernel sharing. They pointed out that the global weight sharing works well on natural photos, in which local pattern appears in any
position of the input [8]. But this is not the case for audio. To handle this problem, they designed the dedicated MDenseNets for each frequency bands and a full band MDenseNet for full band rough structure [8]. This model achieved the state-of-the-art performance on SiSEC 2016 competition [12].

In this paper, we propose a new input format for the MSS model in the frequency domain, namely channel-wise subband (CWS) input. Different from the band-dedicated approach in [8, 11], our method can handle both sub-bands and full-band in a single model, which makes CWS-based model highly efficient, less complex, and easier to use. We extensively evaluate our method on MDenseNet, UNet [6] with different scales, and three kinds of subband settings on musdb 18hq [13]. We also test our approach on a larger internal dataset. Results show that the model with CWS input not only outperforms the model without CWS by a large margin, but also boosts the speed of model training as well as separation.

2. Methodology

Given the raw music signal $y(n)$, our goal is to separate a set of source signals $x_j(n) = \{x_j(t), j \in \{1, \ldots, J\}$. In this paper, we focus on VAS task. Thus $J = 2$ and $x_1(n), x_2(n)$ are vocal and accompaniment track, respectively. In the time domain, the observed mixture signal is modeled as:

$$y(n) = x_1(n) + x_2(n),$$

and in the frequency domain, it equals to

$$Y(t, f) = X_1(t, f) + X_2(t, f),$$

where $t, f$ index time and frequency axis, $Y, X_1, X_2$ are short-time fourier transform (STFT) of the mixture signal $y(n)$ and the source signal $x_1(n), x_2(n)$. Also we should clarify that $Y$ is not a magnitude spectrogram as was used in conventional frequency domain models [14, 6, 4, 15], but the complex-valued STFT matrices. This method is explored by [7] with considerable improvements. In this section, we will briefly introduce the UNet, MDenseNet, the proposed analysis-synthesis scheme and CWS feature format.

2.1. UNet

Fig 2 depicts the structure of UNet [16] we use in this paper. The number of different scales $s = 5$. It takes the mixture spectra $Y(t, f)$ as input and outputs the Time-Frequency Mask [17] $M_j(t, f)$ for source $j$, which has identical size with the input. The in-conv block firstly expands the input channel to 64. After that, we go through a series of convolution blocks, down/upsampling layers with skip connections. Each convolution block consists of two series of 2D convolution layer, batch normalization and rectified linear units [18]. We use $3 \times 3$ kernel size in convolution layers, with the padding value of 1 to make sure that the frequency and time dimension will not be changed by the convolution operations. We use max-pooling and linear interpolation to scale down and up the feature map by a factor of 2. Skip connections are added between down and up path. The input feature of each convolution layer in the up path is concatenated with the same scale output in the down path. To constraint the mask value between $[0, 1]$, the sigmoid function is added on the model’s output. Finally the source estimation is obtained by multiply the mask and the mixture STFT:

$$\hat{X}_j(t, f) = M_j(t, f) \cdot Y(t, f).$$

The final separated music signal $\hat{x}_j(n)$ is obtained through inverse short-time fourier transform (iSTFT) using the source estimation in Eq. (3).

<table>
<thead>
<tr>
<th>Blocks</th>
<th>MDenseNet</th>
<th>MDenseNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv0((1,8))</td>
<td>conv0((1,4,32))</td>
<td>conv0((3,3,32))</td>
</tr>
<tr>
<td>conv14-pool</td>
<td>conv14-pool</td>
<td>conv14-pool</td>
</tr>
<tr>
<td>conv5-conv</td>
<td>conv5-conv</td>
<td>conv5-conv</td>
</tr>
<tr>
<td>conv14-conv</td>
<td>conv14-conv</td>
<td>conv14-conv</td>
</tr>
<tr>
<td>conc/dense1</td>
<td>conc/dense1</td>
<td>conc/dense1</td>
</tr>
<tr>
<td>conc/dense2</td>
<td>conc/dense2</td>
<td>conc/dense2</td>
</tr>
<tr>
<td>out</td>
<td>DenseBlock(32,2,2,n=256,PK)</td>
<td>Conv1x1ch_in=32,ch_out=256,PK</td>
</tr>
</tbody>
</table>

Table 1: Details of MDenseNet and MDenseNet.

2.2. MMDeenseNet

With only about 0.3M parameters and SOTA performance, MDenseNet [8] is currently one of the most effective models for audio source separation. It utilizes the characteristics in each frequency band to design different MDenseNets dedicated for each band. The frequency axis size of band-dedicated MDenseNet will be reduced to $f/K$ for it only processes data within that specific band. The outputs of these MDenseNets, $\hat{x}_j(t, f/K)$, are concatenated in the frequency axis to recover full band prediction $\hat{X}_{\text{mm}}(t, f)$. To capture the global rough structure, MDenseNet also has a MDenseNet for the full band. Then the output of the full band MDenseNet $\hat{X}_{\text{mm}}(t, f)$, is concatenated with $\hat{X}_{\text{mm}}(t, f)$ and pass through a dense block to recover the final prediction $\hat{X}(t, f)$. Each MDenseNet inside MMDeenseNet can be designed independently according to its function and data complexity. MMDeenseNet is highly parameter efficient due to the use of denseblocks [19] and skip-connection between denseblocks. Inside denseblock, the input of Denselayers is the concatenation of previous layers’ output or the skip connection of the former dense block. This scheme enables model to reuse feature effectively, and thus it is highly parameter efficient. The MMDeenseNet we use in this paper has a scale of 4, with three MDenseNets, as is shown in Table 1. The detail of MMDeenseNet is described in [8].

In this paper, we also conduct experiments on MDenseNet, both with CWS and without CWS. The structure of MMDeenseNet we use is shown in Table 1. For a fair comparison with MMDeenseNet, we add two additional dense blocks to the lower part of the network.
of MMIDenseNet to form the MDenseNet we use. In this way, the scale of our MDenseNet is 5 and the total parameter number is 0.27 M.

2.3. Channel-wise Subband Input

We follow the method in [20] for subband decomposing and signal reconstruction in the analysis and synthesis procedure. Both analysis and synthesis include a group of finite impulse response (FIR) uniform filter banks. We design three sets of analysis filter banks \( H_k(e^{j\omega}) \) and corresponding synthesis filters \( G_k(e^{j\omega}) \), where \( k \in 1, ..., K \) stands for the number of subbands. The design of these filters follows the procedure in [21]. We use \( y_k(n) \) to denote the output of \( H_k(e^{j\omega}) \).

After downsampling, the sample rate of \( y_k(n) \) is \( \frac{1}{K} \) of \( y(n) \).

\[
\begin{align*}
\text{Analysis} & : \quad H_1(e^{j\omega}), H_2(e^{j\omega}), ..., H_K(e^{j\omega}) \quad \text{and} \quad Y(f/K,t) = Y_1(f/K,t) \oplus Y_2(f/K,t) \oplus \cdots \oplus Y_K(f/K,t) \\
\text{Separate} & : \quad y_1(n), y_2(n), \ldots, y_K(n) \\
\text{Synthesis} & : \quad \tilde{Y}_1(f/K,t) \rightarrow \tilde{X}_j(f/K,t) \\
& \vspace{1em}
\end{align*}
\]

Figure 3: Channel-wise subband input

Though the total volume of input feature does not change, the channel-wise concatenation of subbands is a better input format for the frequency domain model. Here we give a simple explanation. We use \( \beta^{(l)}_{\mu} \) to denote the output feature map in \( l \)-th layer, \( \mu \) channel, and \( S^{(l-1)}_{\lambda,\mu} \) to stand for the \( \lambda \)-th convolution filters in layer \( l-1 \), which output the \( \mu \)-th channel in \( l \)-th layer. The 2D convolution layer can be described as

\[
\begin{align}
\beta^{(l)}_{\mu}(i,j) &= \sum_{\lambda} \beta^{(l-1)}_{\lambda}(i,j) \ast S^{(l-1)}_{\lambda,\mu} \\
&= \sum_{\lambda} \sum_{m} \sum_{n} \beta^{(l-1)}_{\lambda}(i-m,j-n) S^{(l-1)}_{\lambda,\mu}(m,n) \\
\beta^{(l)}_{\mu}(i,j) &= \phi \left( \sum_{\lambda} \beta^{(l-1)}_{\lambda}(i,j) \right).
\end{align}
\]

From Eq. (4), we can observe that internal variable \( \beta^{(l)}_{\mu} \) is the linear product and sum of \( \beta^{(l-1)}_{\lambda} \) and \( S^{(l-1)}_{\lambda,\mu} \). Thus we can not only view the convolutional kernel as feature extractor, but also weight between different filters. For example, if some filters in \( S^{(l-1)}_{\lambda,\mu} \) are set to all zero, then the corresponding channels in \( \beta^{(l-1)}_{\lambda} \) will not be able to pass their value to \( \beta^{(l)}_{\mu} \). In this way, the \( \mu \)-th channel in feature map \( \beta^{(l)}_{\mu} \) can select the exact channels to be used in the previous layer. The channel-wise concatenation enables model to assign different capability on channel dimension, which is helpful to make the model highly efficient.

After the analysis process, we perform STFT for each \( y_k(n) \) and the result is denoted as \( Y_k(f/K,t) \). Here the sample rate of \( y_k(n) \) reduce by a factor of \( K, K \in \{2, 4, 8\} \). So the size of frequency axis will also be reduced \( K \) fold. Then we concatenate \( Y_k(f/K,t) \) along the subband dimension

\[
Y = \begin{bmatrix}
Y_1^T, Y_2^T, \ldots, Y_K^T
\end{bmatrix}
\]

(6)

to form the input feature of the network \( Y(f/K,t) \). Since the data we use in this paper are all stereo, \( C \) equals to 2 here. The subscript \( k \) indexes subband and we omit the \( (f/K,t) \) part in this equation for simplicity. We treat different bands as different channels so that model can both learn different channels independently and incorporate bands’ features in a deeper layer.

The synthesis procedure is a reverse version of the analysis. We split the network output \( \tilde{X}_j(f/K,t) \) channel-wise as the prediction of each subband. After iSTFT, we pass the result through a set of synthesis filters to reconstruct source signal \( \tilde{x}_j(n) \).

2.4. Model Training

The synthesis procedure is not performed during training and the loss function is defined as the sum of two components

\[
\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_c,
\]

(7)

where \( \mathcal{L}_1 \) is the \( \mathcal{L}_1 \) norm and \( \mathcal{L}_c \) denotes conservation loss. Conservation loss could help when two dedicated models are trained jointly because it follows the basic model in Eq. (2) and unites two independent dedicated-models. Each loss function measures the mean absolute error between network output \( \tilde{X}_j(f/K,t) \) and the corresponding reference magnitude:

\[
\mathcal{L}_1 = \sum_{j=1}^{2} \sum_{t=0}^{2} \left| \tilde{X}_j(f/K,t) - X_j(f/K,t) \right| \\
\mathcal{L}_c = \sum_{j=1}^{2} \sum_{t=0}^{2} \left| \tilde{X}_j(f/K,t) - Y_j(f/K,t) \right|.
\]

(8)

We perform validation with every two hours of the training data and stop the training progress if no validation improvement exists in 20 consecutive epochs. All the models are trained using Adam optimizer [22] with an initial learning rate of 0.001 and a dropout rate of 0.1. The learning rate decays every thirty hours of training data with a decay rate of 0.87. The STFT matrices with a FFT size of 32 ms and a hop size of 8 ms are used as the model input. The actual frame length and shift size (in number of the samples) are automatically calculated with the sample rate of the input audio.

3. Experiments

In this section, we will first describe the dataset and evaluation metrics used in this paper. The experimental comparison and analysis of the advantage of CWS will then be discussed.

3.1. Dataset

We mainly conduct experiments on the publicly available musdb18q dataset [13]. It has a training set with 100 songs and a test set of 50 songs. We choose 14 songs from the training set as the validation set, the same as the definition in python package musdb. To explore the limitation of the data, we also trained our model on an internal training set aslp. It has additional 617 songs of pure vocal and 1496 songs of pure instrument, which are collected from the internet. Although some of

https://github.com/sigsep/sigsep-mus-db
them may not be absolutely clean, experiments show that using additional data improves the separation performance. We follow the steps in [15] for data augmentation. During the training stage, we randomly select, chunk, and mix vocal and instruments and multiply two streams with a scaling factor randomly sampled between 0.6 and 1.0. All the songs in musdb18hq and aslp are stereo and the sample rate is 44.1 kHz.

3.2. Evaluation Metrics

We use museval [23] toolkit to compute SDR, SIR, and SAR [24] metrics for evaluation. In details, we calculate the metrics for all the segments of the song in the test set with a window size of 1s and hop length of 1s, as commonly used in SISEC 2018 [23]. We aggregate both the average SDR, SIR, and SAR by frames as the final score of a song, and pick the median value from each song as the final score of test set. All our experiments are performed on a single GTX 1080Ti GPU. For fair comparison, we report some other metrics, e.g., parameter number and training time, as shown in Table 2. We also use Giga Floating Point Operations (GFLOPs) to weight the computational cost. The floating operation here is measured by a three-second long stereo input.

3.3. Result Comparison

The result is shown in Table 2. Here we name MMD as abbreviation of MMDenseNet and MDenseNet. UNET-N denotes scale N Unet and the prefix BD means the model is trained with extra internal aslp dataset. A stands for accompaniment and V stands for vocal. Value K stands for total subband number in CWS, as shown in Fig 3.

In general, the result shows that the CWS input can considerably improve the performance. All the models with CWS surpass the models without CWS on the average SAR, SDR, and SIR by a large margin. Since the computational cost drops drastically with the increase of K, the model with a higher K value will converge more quickly. This is beneficial when the dataset is huge. Besides, a higher K will lead to a smaller feature map. This can save a lot of memory during training and evaluation, making the model and training process more flexible and easier to deploy.

As can be seen from Table 2, splitting 4 bands usually has the best average score on all the evaluation metrics. The average performances of MDN, UNET-5, UNET-6 and BD-UNET-6 increase by 5.7%, 10.1%, 10.2%, and 7.8%, when using the CWS_K=4 input. Although CWS_K=4 outperforms CWS_K=8 by 1.5%, it takes more time, i.e., 38.5% for model training. Comparing with the model without CWS, CWS_K=8 increases the performance by 6.8% and costs only 31.8% of the original training time. Moreover, UNET-5/6 and MDN with CWS_K=8 achieve the best average SDR, which is valid as a global performance measurement [24]. Thus in practice, CWS_K=8 may be the most effective one because it can yield comparable results in a shorter training time. CWS_K=2 scenario might be the least preferred setting but still it has contributions to the final score.

It’s also worth to mention that MDN_K=8 surpasses the performance of MMDenseNet in [11] even with fewer parameters, far less training data and shorter training time, as shown in Table 3. The training set only has 84 songs, but MDN_K=8 is still able to exceed the performance of the model trained with a larger dataset. Moreover, our training time might be much shorter. In [8], a single MDDenseNet trained on DS100 [12] dataset, which is comprised of 100 songs, will take 37 hours for each instrument to train. MMDenseNet trained with extra data will cost more than that time. By contrast, our model only takes 9.7 hours to train. All the evidence strongly demonstrates the advantage of using CWS as model input. Audio samples and codes are available online: https://haohelui.github.io/Channel-wise-Subband-Input/.

4. Conclusions

We present an alternative structure of input feature, namely channel-wise subband (CWS) for VAS model in frequency domain, in order to handle the high computational cost and limitation of the conventional CNNs in high-resolution MSS tasks. It overcomes the limitation of the widely used full-band approach and enables the model to learn weight independently in each subband. Experimental results show that the proposed CWS improve the separation performance and reduce the computational cost significantly. On the public musdb18hq dataset, the MDDenseNet with 8-bands CWS input exceeds original MDDenseNet by 1.67 dB on average SDR of the voice and accompaniment.
5. References


