Deep Denoising Autoencoder for Music Source Separation

Yazhou Li[1]

Queen Mary University of London, London, the United Kingdom

Abstract. Wave-U-Net has shown success in music source separation, but only clean music signals can generate good separation results. In this work, I present a deep denoising autoencoder framework for noisy and reverberant music source separation. I use pretrained Wave-U-Net model to evaluate the source separation performance using denoised audio and the proposed approach shows improvement on the source separation under subjective listening and some objective evaluation metrics.

1 Introduction

Music source Separation is the task to separate different tracks and instruments from the mixed music. The application includes upmixing and remixing for music production, karaoke, music transcription and so on. Related work about source separation are focused on deep learning in recent years. There are methods in frequency domain and in time domain. Wave-U-Net is a time domain U-Net model, which shows near state-of-the-art performance[1]. In this work, we use Wave-U-Net as the source separation model.

Current source separation methods have been successful on the task and are able to separate audible tracks. But when there is noise or reverberation added to the clean music signal, the different sources seem to merge together and it is difficult for source separation algorithms to work. Therefore, it is important that the noisy and reverberant signals are preprocessed first to generate clean signals as the input of source separation model. The denoising autoencoder in [2] uses Mel-frequency cepstral coefficients (MFCC) as input. I propose a deep Autoencoder framework using spectrogram as input to denoise and dereverberate the noisy music audio.

I validate the effectiveness of the proposed approach on MUSDB18 dataset, which is a 150 full-track songs dataset for music source separation.

The rest of the paper is organized as follows. Section 2 introduces the two framework. Section 3 talks about the details about the denoising environment. Section 4 analyses the results using the audio example and proposes future improvements.

2 Model

2.1 Wave-U-Net

Wave-U-Net is the U-Net model in wave domain [1]. A diagram of the Wave-U-Net architecture is shown in 1. It is composed of downsampling blocks and upsampling blocks. The downsampling block is composed of a convolution layer and a downsampling layer, the upsampling block is composed of an upsampling layer and a convolution layer. There are totally L downsampling and upsampling blocks. The downsampling block is cropped and concatenated with the upsampling block.

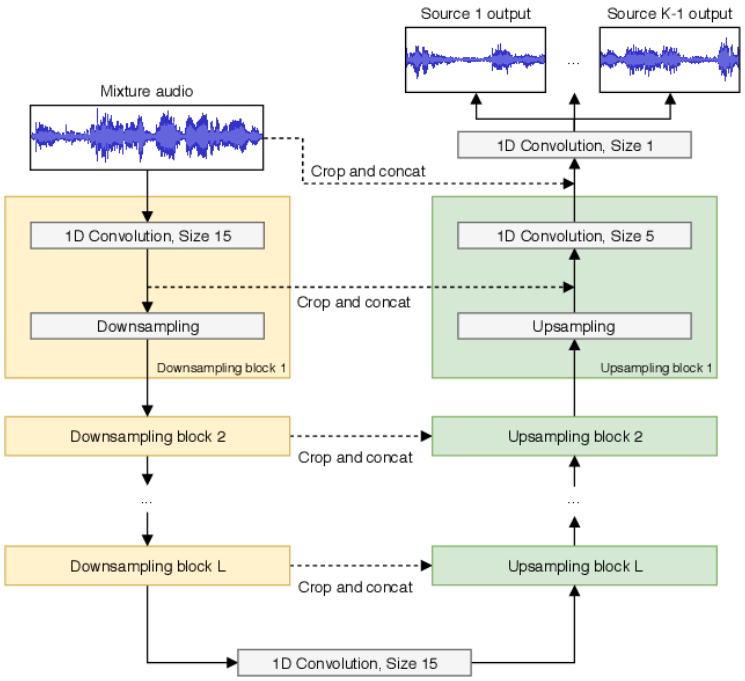


Fig. 1: Wave-U-Net architecture[1]

2.2 Autoencoder

A diagram of the proposed autoencoder architecture is shown in 2. It composes of a downsampling block, an upsampling block and two two convolution blocks. The convolution block extracts the features of the spectrogram, which contains a convolution layer, followed by a ReLU (rectified linear unit) activation layer and a batch normalization layer. The convolution layer weights are

initialized using kaiming initialization. I use max pooling as the downsampling block and use deconvolution or bilinear interpolation as the upsampling block. The spectrogram is first put into a convolution block, then downsampled to half size, upsampled to the original size, and put into a convolution layer to extract the feature again.

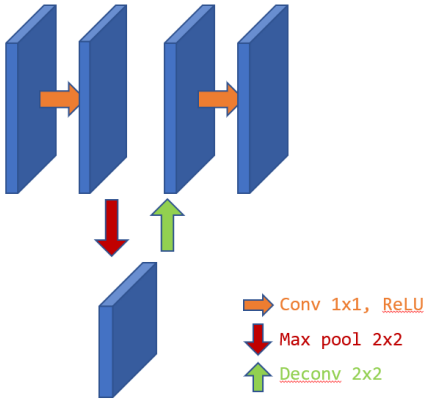


Fig. 2: autoencoder architecture

3 Experiment

3.1 Dataset Introduction

MUSDB18 is a dataset for music source separation, consisting of a total of 150 full-track songs of different styles and includes both the stereo mixtures and the original sources. 100 tracks are from The 'Mixing Secrets' Free Multitrack Download Library, 46 tracks are from MedleyDB and 2 tracks are from Native Instruments.

All files from the musdb18 dataset are encoded in the Native Instruments stems format (.mp4). It is a multitrack format composed of 5 stereo streams. These signals correspond to:

0 - The mixture,
1 - The drums,
2 - The bass,
3 - The rest of the accompaniment,
4 - The vocals.

For each file, the mixture correspond to the sum of all the signals. All signals are stereophonic and encoded at 44.1kHz[3].

Two kinds of data are provided: MUSDB18 and MUSDB18HQ. In MUSDB18, audio data is stored in STEM structure and each file contains five tracks. In MUSDB18HQ, all separate tracks are stored in a wav file.

As the MUSDB18 is encoded as STEMS, it relies on ffmpeg to read the multi-stream files. A python package called musdb is used to parse and read the dataset. With this tool, we can easily handle STEM data in the database, for example, use track.targets['vocals'].audio to get the vocal of an audio and use track.audio to get the mix audio.

3.2 Dataset Noising

To reduce training time, I just use a subset of the first 50 tracks of all audio files. For each song, I crop it into 1000000 time sampling points, starting from step 1000000 to step 2000000 (about 24s), because the first 1000000 are most instruments without vocals, which is not suitable for source separation task evaluation.

The noisy music dataset is made by adding noise and reverberation to the MUSDB18 manually. I convolve the impulse response and the original audio to produce the reverberant audio, then I add the noise audio and reverberant audio together to produce the noisy audio. I also divide the audio by 10 to avoid peak cutting when writing audio files. The noise file I use is downloaded from freesound.org. The reverberation impulse response I use is block_inside.wav from IMreverb.

I first try to do the convolution using MATLAB, but my local MATLAB version can only read the vocal track of the STEM data. Although the online MATLAB is able to read the mix track of STEM data, it is not convenient to upload the dataset to MATLAB drive. So at last I use python to do the work, but python seems to be much more slower than MATLAB when doing signal convolution.

3.3 Experiment Details

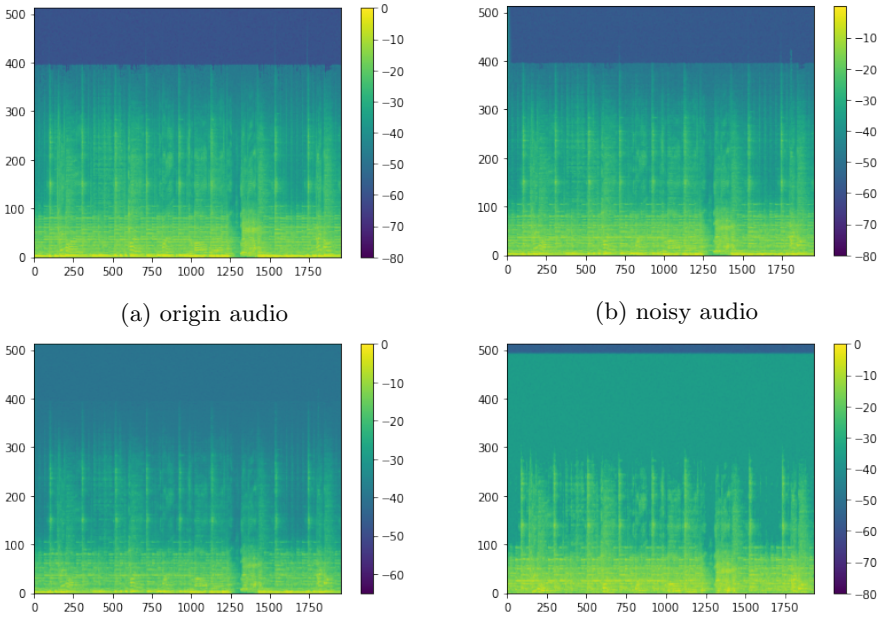
The audio files are first converted to spectrogram using STFT (short time fourier transform), the window type is hanning window, fft number is 1024, window length is 1024, hot length is 512 and sample rate is 44100.

The architecture is built using pytorch framework and the model is run on whitby server. Batch size is set to 16, initial learning rate is 0.001, optimizer is Adam optimizer, loss function is MSE (mean square error) of the output spectrogram and target spectrogram.

3.4 Parameters Tuning

After trying different network structure and listening to the estimate audio file, I find zero-padding will introduce high frequency buzz artefacts. Therefore, I center-crop the target audio spectrogram to make its dimension the same as the network output to avoid artefacts for loss calculation.

For the upsampling method, I tried two methods: transposed convolution (also called deconvolution) and bilinear interpolation. The two upsampling methods yield similar results and none of them introduce artefacts.



(a) origin audio

(b) noisy audio

(c) 1 channel estimate audio

(d) 128 channel estimate audio

Fig. 3: Power spectrograms (dB) of different audios

Spectrograms of audio files are shown in Fig.3. As we can see, high frequencies of the original audio are attenuated, maybe due to the segment cropping of input audio files. Therefore, all outputs of the network are small in high frequencies. When there are 128 channels and more parameters, the network seems to be leaning this attenuation and weakens high frequencies even more. Therefore, the audio file sounds like lower tuned and the low frequency is very obvious. The more parameters there are, the more severe the problem is. So I just use one channel output and use kernel size 1, in which case the convolution layer degrades into a dense layer.

4 Case Study

4.1 Evaluation Metrics

The results for the models are evaluated using the BSS evaluation criteria [4] including source to distortion (SDR), source to artifacts (SAR), source to interference (SIR) ratios, plus the image to spatial ratio (ISR).

4.2 Results and Discussion

The case study audio I use is the noisy and reverberant song A Classic Education by NightOwl, which is the first file in MUSDB18 dataset. I use the pretrained Wave-U-Net model to predict the vocal of the original clean audio, the noisy and reverberant audio and the estimate denoised audio respectively. The results are
'A Classic Education - NightOwl_vocals.wav',
'A Classic Education - NightOwl_noise_vocals.wav'
'A Classic Education - NightOwl_output_vocals.wav'.

The source separation model can not work well on noisy audio and denoised audio, the metrics are mostly negative as we can see in Table 1. But SIR is better for the denoised audio than the noisy audio. When we hear to the separation results, we can hear the vocals are separated better in the denoised audio, which is almost total silence when we use noisy audio as the input, which means the proposed approach is able to reduce the reverberation in some extent and improve the singularity of each track.

Table 1: Music Source Separation Performance

audio	SDR	ISR	SIR	SAR
origin	6.73	10.93	12.49	5.73
noise	-1.16	-0.09	**1.03**	-4.82
denoised	-1.33	-0.22	**5.33**	-6.58

For future work, I plan to use data augmentation methods to enlarge datasets, for exmaple, the current dataset is created using only one kind of noise and reverberation, we should use different kinds of noises and reverberations. Also, only a fragment of each audio is used in this work. A larger dataset can be produced by using different fragments of audio files.

Furthermore, there may be some bugs with the high frequency problem. I plan find out what causes the attenuation of high frequency, and train the model again using convolutional layers to see if the performance will be improved.

References

1. D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *arXiv preprint arXiv:1806.03185*, 2018.
2. X. Feng, Y. Zhang, and J. Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014, pp. 1759–1763.
3. Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

4. 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.