Performance Analysis of Unmix, a Music Source Separation Neural Network

1st Matías E. Charrut
dept. name of organization (of Aff.)
name of organization (of Aff.)
Buenos Aires, Argentina
mcharrut@fi.uba.ar

***Abstract*—In this work, we analyze the performance of Unmix, a neural network that performs music source separation using transfer learning. We test it using different input chunk sizes, comparing quality of results and time costs. Then we analyze it with different stems combinations of the same track and adding others from different ones. This results allow us to learn more about how Unmix works.**

***Index Terms*—unmix, music, source, separation, neural, network, transfer, learning, jukebox, vq-vae**

I. Introduction

Music source separation can be a difficult challenge, especially when we try to achieve it by training the model from a set of examples. There are several neural networks that perform this, such as UMX [1], *Demucs* [2] and Unmix [3]. In this work we analyze the last one.

Unmix is a neural network which allows us to obtain a certain audio channel with an instrument from a complete track. These channels are called stems. This can be used to separate 'vocals', 'bass', 'drums' and 'other' (including keyboards, guitars, violins, etc.) stems. This network uses transfer learning to adapt *Jukebox* [4], which generates real-sounding music using Vector Quantized Variational Autoencoders (VQ-VAE), from its publicly available pre-trained data.

We carried out a performance analysis of Unmix with its already trained values. Firstly, we tested the performance and time cost with different sizes of input chunks, analyzing the outcome using different VRAM capacities. Then we analyzed the performance processing different stems combinations of the same track. Finally, we tested the network inserting a 'vocals' stem in another track, both leaving the original one or not.

All this analysis was done using the test set from the MUSDB18-HQ [5] database as testing data, which is the same one used for training and testing Unmix. Under certain conditions some tracks had processing errors and these were omitted in the final result. We used the latest available checkpoints from Unmix for each stem. As a conclusion of the analysis carried out in Section III, the other tests were done using a chunk size of 147456 samples. The Unmix paper uses a chunk size of 393216, but a smaller one was chosen due to hardware limitations.

As a performance metric, we used signal-to-distortion ratio (SDR), the same one selected in the Unmix paper, which is shown in (1), where $s(n)$ is the original stem (with n the index of the sample), $\hat{s}(n)$ is the estimated one and $\epsilon = 10^{-7}$ is a term to avoid division by zero.

$$SDR_{stem} = 10\log_{10}\frac{\sum_n ||s(n)||^2 + \epsilon}{\sum_n ||s(n) - \hat{s}(n)||^2 + \epsilon} \tag{1}$$

We used for these experiments a computer running Manjaro Linux 22, with an AMD Ryzen 3700x CPU and a NVIDIA RTX2070 Super GPU.

II. Unmix

Unmix uses the default Jukebox's variant of the pretrained VQ-VAE model. The audio signal is mapped by an encoder to a space $e_t = E_1(x_t)$ of 64 dimensions so that it can be mapped to the closest prototype vector of a collection C called *codebook*. These 2048 prototype vector c_{st} are learned in training to obtain a high-quality representation.

The signal, after being mapped to the codebook, is decoded by a D decoder to reconstruct the original signal. In summary, this process through the VQ-VAE is described by (2).

$$y_t = D(\text{argmin}_c(||E_1(x_t) - c||) \text{ for } c \in C \tag{2}$$

A network diagram during training and deployment is shown in Fig. 1. Loss functions are also shown.

The Jukebox pre-trained VQ-VAE network is trained using the different stems, so that it learns good representations for each instrument. In this step the 'Stem Encoder', the codebook and the 'Stem Decoder' are trained. Then, another encoder called 'Mix Encoder' is trained using the complete mixes to map these to the same e_t of the stem trained in the previous step. During the deployment, the network uses the 'Mix Encoder', the codebook and the 'Stem decoder'.

III. Performance variation with chunk size

We analyzed the performance variation using different input chunk sizes (the number of track samples which were processed at the same time). This was done extracting the 'vocals' stem measuring SDR and time used.

The analysis was done using chunk sizes from 10000 to 180000 samples with 10000 steps. There were also added chunk sizes which are multiples of the chunk sizes used by the *Jukebox*'s VQ-VAE (49152). The larger size was 196608 samples due to GPU VRAM capacity limitation.

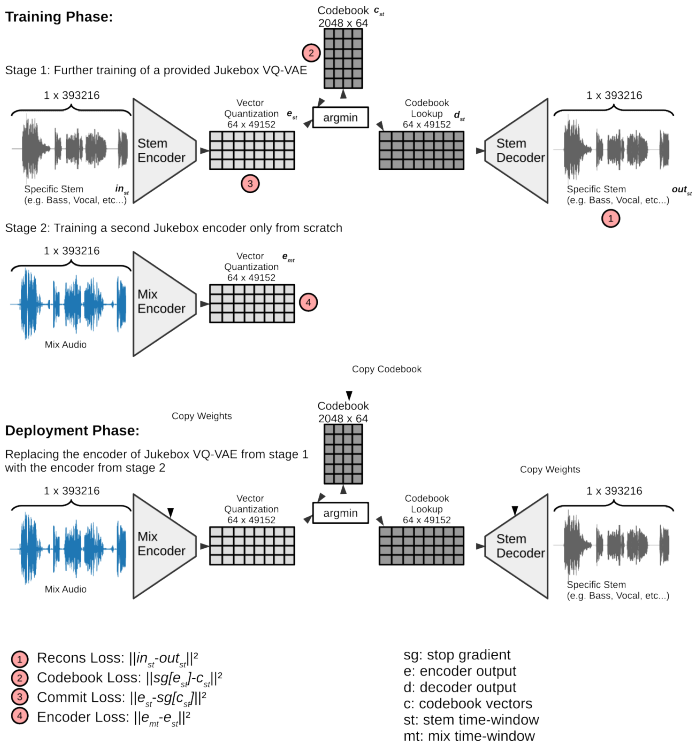

Fig. 1. Diagram of the Unmix neural network.

From these data, we calculated an average result for each chunk size and it is shown in Fig. 2. We can observe that the SDR varies by less than 1 dB along all chunk sizes. In the case of smaller chunk sizes, the performance is worse and takes more time. Once a size of 50000 samples is reached, the SDR varies by less than 0.2 dB. This is expected because the chunk size of the *Jukebox*'s VQ-VAE mentioned above is surpassed.

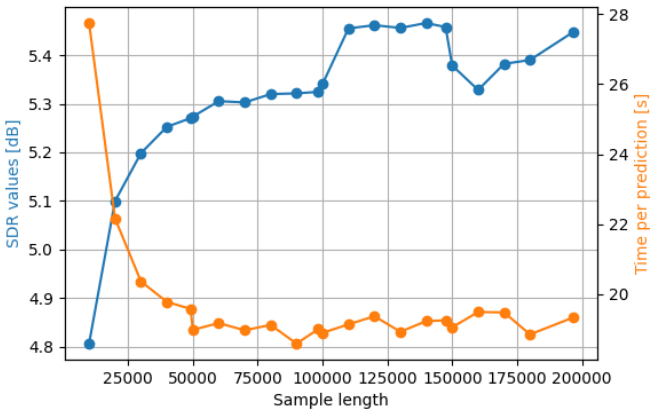

Fig. 2. Performance (SDR and time) in function of input chunk size while extracting the 'vocals' stem.

To conclude, once the threshold of 49152 samples is reached, the performance and time used are stable. We will be using a size of 147456 in the following analysis because it is one of the best performing and is a multiple of 49152.

IV. Performance Analysis of Stem Combinations

We analyzed the Unmix performance extracting all the stems for all possible combinations. For example, in 'vocals' we measured the performance for each track including 'vocals' only, 'vocals' and 'drums', 'vocals' and 'bass', 'vocals', 'bass' and 'drums' and so on until all possible combinations were tested. We built the track from the separate stems of the database. It should be noted that the operation of mixing is an addition, so two combinations of the same stems are the same one, even if they were added in a different order.

The results are shown in Fig. 3, where the SDR values for different combinations are plotted for each extracted stem.

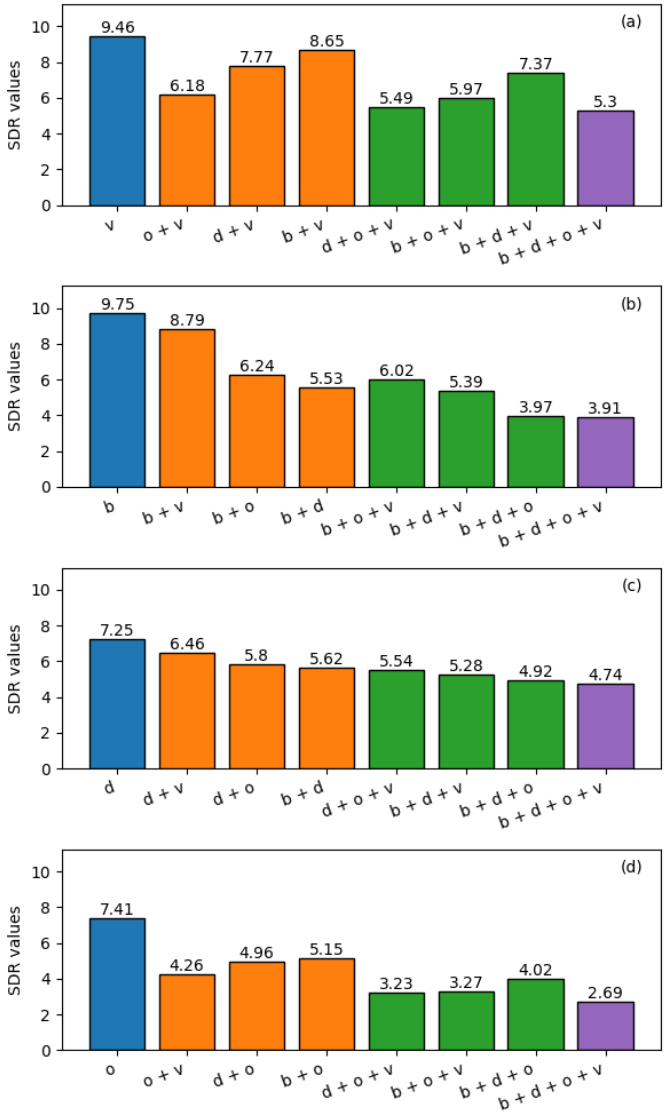

Fig. 3. SDR for different stems combinations extracting every stem ('v' is 'vocals', 'b' is 'bass', 'd' is 'drums' and 'o' is 'others'). Same colors are used to identify amount of stems. (a) 'Vocals'. (b) 'Bass'. (c) 'Drums'. (d) 'Other'.

We can observe that when the extracted stem is the only one present in the track, the highest performances are obtained. Nevertheless, this value is not much higher than the other cases while the reconstruction should be perfect. This is not the case due to the intrinsic nature of the network.

Also, we can note certain conflicts between some stems and their combinations. When extracting the stem 'other', the results were not too good compared with other stems (this result is already clear in the Unmix original paper), but also the presence of this stem usually ruins the performance of the stem 'vocals' and 'bass', the last one only when we extract the 'bass' stem. Another conflicting combination is between 'drums' and 'bass', which had worse results than others.

To complement this analysis, we performed a fast Fourier transform (FFT) on each stem of a track in the set ('Ben Carrigan - We'll Talk About It Tonight'). The FFT is shown in Fig.4.

The result shows that the 'vocals' and 'other' stems occupy a large part of the spectrum, so their conflict could be expected. However, we can observe that the 'vocals' do not occupy the lowest frequencies while the 'other' stem does. This frequencies coincides with the 'bass' ones, so it is logical to observe they present problems. Since the spectrum of the 'bass' is only concentrated in these frequencies, the biggest drop in performance is found when this is extracted.

On the other hand, 'bass' and 'drums' are both concentrated in the lowest frequencies, explaining their conflict.

As a conclusion, there is an influence of the stems frequency distribution when reconstructing each one of them, meaning that stems with similar frequency components interfere in the performance of the network, making it worse.

V. Introducing vocals stem from other tracks

We tested the behaviour of Unmix when the 'vocals' stem of a track, called alternative from now, is introduced into another complete track, from now on original. This was done keeping the 'vocals' stem of the original track and removing it. The alternative stems were chosen randomly (five per original track). If an alternative stem is longer or shorter than the original one, it is trimmed or zero-padded respectively.

In Fig. 5 we showed the results for this test: SDR for the original track with its original 'vocals' stem, SDR for the original track with the alternative 'vocals' stem and three SDR's for the track with two 'vocals': comparing with the original 'vocals' stem, with the alternative one, and with the mixture of both.

We can observe that in all the cases in which the comparison is done with the stem present in the track, the results are similar. It is a bit lower for the alternative stem only but comparable with the extraction of the original stem. However, when the stem extracted from the mixture of 'vocals' stems is compared with one of the two separately the results were inferior. Then we know that the network is extracting the mixture of both 'vocals' stem.

Therefore, we can conclude that the quality of the extracted stem does not have any correlation with the content of the other

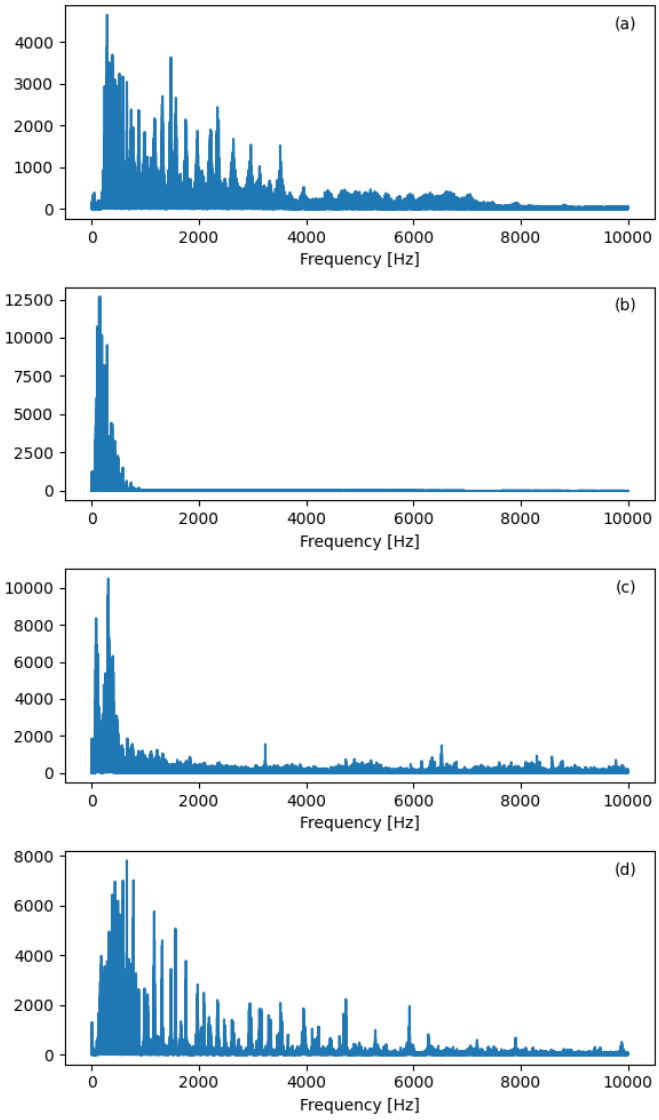

Fig. 4. FFT of each stem of the track 'Ben Carrigan - We'll Talk About It Tonight'. (a) 'Vocals'. (b) 'Bass'. (c) 'Drums'. (d) Other.

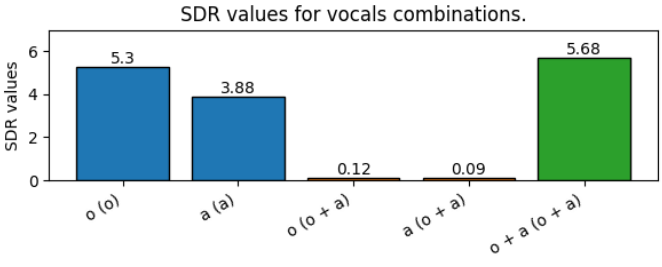

Fig. 5. SDR performance for tracks with different combinations of original and alternative 'vocals' stems. Outside parenthesis is shown with which track is made the comparison and inside parenthesis how is composed the track. ('o' is original and 'a' is alternative).

existing stems in the track, so whether that stem originally belongs to that track or not it will not change the performance of the network.

VI. Conclusions

We tested and analyzed the behavior and performance of Unmix under different conditions. We concluded that the number of samples in the input chunk does not have a significant impact on the performance, as long as a threshold value of 49152 is reached.

On the other hand, the frequency components of the different stems do influence the quality of the separation since stems that share similar frequency characteristics tend to conflict with each other and it is more difficult to separate them.

Finally, we verified that network's performance is not influenced by the context of the track. For example, the network will work the same if the 'vocals' stem has no correlation with the rest of the track.

References

[1] F.R. Stöter, S. Uhlich, A. Liutkus, Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," Journal of Open Source Software 4(41), 1667 (2019), https://doi.org/10.21105/joss.01667

[2] A. Défossez, N. Usunier, L. Bottou, F. Bach, "Demucs: Deep extractor for music sources with extra unlabeled data remixed," CoRR abs/1909.01174 (2019), https://doi.org/10.48550/arXiv.1909.01174

[3] W. Zai El Amri, O. Tautz, H. Ritter, A. Melnik, "Transfer Learning with Jukebox for Music Source Separation," Conference paper AIAI 2022 (2022), https://doi.org/10.48550/arXiv.2111.14200

[4] P. Dhariwal, H. Jun, C. Payne, J.W. Kim, A. Radford, I. Sutskever, "Jukebox: A generative model for music," (2020), https://doi.org/10.48550/arXiv.2005.00341

[5] Z. Rafii, A. Liutkus, F.R. Stöter, S.I. Mimilakis, R. Bittner, "MUSDB18-HQ - an uncompressed version of MUSDB18," (2019), https://doi.org/10.5281/zenodo.3338373