2021

Convolutional Audio Source Separation Applied to Drum Signal Separation

Marius Orehovschi
Colby College

Convolutional Audio Source Separation Applied to Drum Signal Separation

Marius Orehovschi
Computer Science Department
Colby College

Abstract
This study examined the task of drum signal separation from full music mixes via both classical methods (Independent Component Analysis) and a combination of Time-Frequency Binary Masking and Convolutional Neural Networks. The results indicate that classical methods relying on predefined computations do not achieve any meaningful results, while convolutional neural networks can achieve imperfect but musically useful results. Furthermore, neural network performance can be improved by data augmentation via transposition – a technique that can only be applied in the context of drum signal separation.

Introduction & Problem Definition

This project explored sound source separation with neural networks. Sound source separation can be used in a variety of contexts, from improving speech recognition, to de-noising, to audio sampling and other kinds of creative music processing. The methods explored in this study can in theory be used for any of these tasks, but the study has focused primarily on music sound separation, specifically drum sound separation, for creative purposes.

Given a mono or stereo mix track, we want to separate one or two individual instruments from the mix. The difficulty of the problem lies in the fact that mixing is a process in which multiple instrument tracks are added together into a single mix track, where all input tracks as well as the output mix track have the same dimension. Separating single instrument tracks from mixes is therefore not a trivial operation.

Classical Method

Until recently, the prevailing method for blind source separation of sound signals has been Independent Component Analysis (ICA) [1]. Independent component analysis is a popular solution to the classic “cocktail party problem”, where there are two or more microphones recording several people speaking simultaneously and we would like to separate the voice of one or more speakers. Given at least n recordings of n simultaneous sources, where every recording is a linear combination of the sources with different coefficients, ICA estimates n independent tracks corresponding to n sources. ICA is a “blind” source separation technique – it does not require any information about the expected output and works correctly for any audio signal as long as the separable sources are unrelated [1].

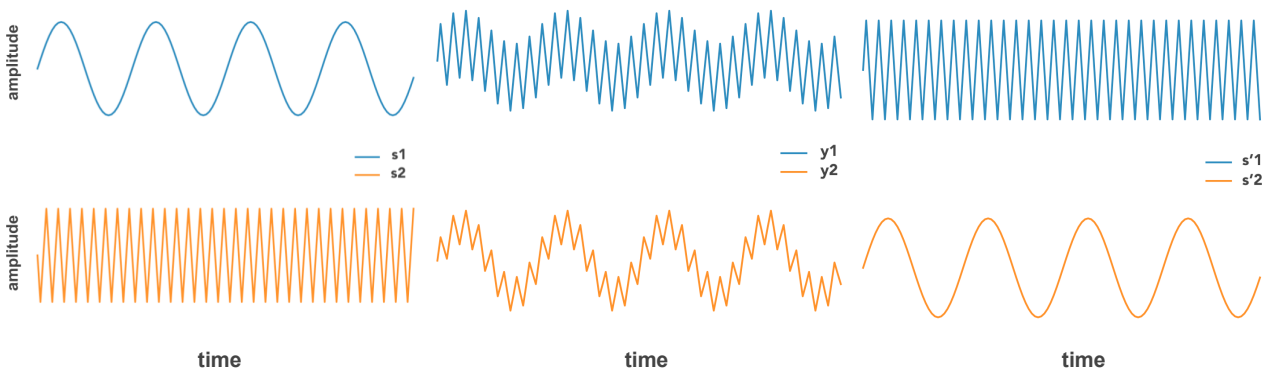

Figure 1. Example of ICA on simple wave inputs. **Left**: sine waves s_1 and sawtooth waves s_2. **Center**: mixes $y_1 = s_1 + \frac{1}{2}s_2$ and $y_2 = \frac{1}{2}s_1 + s_2$. **Right**: estimated sources s'_1 and s'_2. Note that ICA does not necessarily preserve the order ($s_1 -> s'_2$) or the phase ($s'_1 \simeq - s_2$) of the estimated sources.

The formulation of the "cocktail party problem" – separate n independent sources from n mixes – is different from this paper's problem – separate a "drum" track from a full mix track. Fortunately, most digital music is distributed in stereo, meaning there are two tracks – the left track and the right track. Since this study is concerned with separating a "drum" track and a residual "non-drum" track, two input channels are in theory enough to meet the ICA requirements for separating two sources. But another concern with ICA is that it separates completely unrelated signals – and music signals are not necessarily unrelated, as there is a high degree of time synchronization and pitch correlation of different music parts.

To examine whether these two characteristics of music signals are an impediment to separation via ICA, the method was tested first on an example in classic ICA formulation, and then on a "music in the wild" example. In the first test, the drum track s_1 (recording of two drums and a cymbal) and the non-drum track s_2 (recording of bass, piano, and synth) were combined into a drum-dominant mix $y_1 = s_1 + \frac{1}{2}s_2$ and a non-drum-dominant mix $y_2 = \frac{1}{2}s_1 + s_2$ (the source ratios in each mix are the same as in the visual example in Fig. 1). The input mixes y_1 and y_2 are provided in the Accompanying Audio (see page 14) folder as *ica_ex1_mix1.wav* and *ica_ex1_mix2.wav*. Independent Component Analysis was performed on these inputs using the FastICA implementation [2] provided in the scikit-learn library [3]. The sources s'_1 and s'_2 are provided in the Accompanying Audio folder as *ica_ex1_est_src1.wav* and *ica_ex1_est_src2.wav*.

Upon subjective evaluation, the quality of the estimated sources is surprisingly high. The estimated drum track sounds almost identical to the input, while the estimated non-drum track has a faint trace of the drum signal – which only reassures us that the method produces imperfect but nevertheless impressive results.

But these impressive results on an example handcrafted to match ICA specifications do not answer the question of whether the approach can be useful on real-world music examples. Thus it was also applied on a two-channel music example: a 10-second fragment from John Coltrane's

version of My Favorite Things, provided in Accompanying Audio as *favorite_things.wav*. This example is particularly promising for ICA, as its two stereo channels, left and right, are significantly different from each other – the left channel (*favorite_things_left.wav*) has more of the higher pitched sounds, like piano and cymbals, while the right channel (*favorite_things_right.wav*) has most of the bass.

Performing ICA on this real-world example, however, does not seem to result in any meaningful sound source separation. Upon auditory evaluation of the ICA outputs (*favorite_things_out1.wav* and *favorite_things_out2.wav* in Accompanying Audio) of this example, it seems in fact that ICA simply outputs the input channels. Input-output correlation on the first second of audio was used for a quantitative investigation of this subjective impression. Correlation is used in audio engineering for tasks like detecting destructive interference in the left and right channels of an audio mix. In this case, it was simply used to determine if the ICA inputs and outputs are identical. As ICA does not guarantee the order of estimated sources, the “brighter” (i.e. high frequency-dominant) output track was matched to the “brighter” left input channel, while the “darker” (i.e. low frequency-dominant) output track was matched to the “darker” right input channel. The resulting correlation plots are provided below:

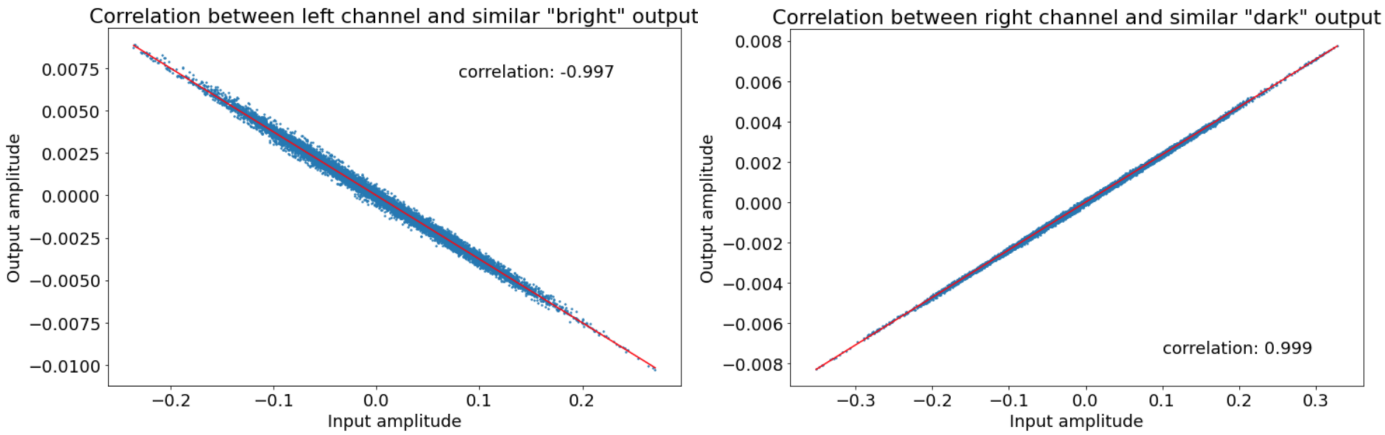

Figure 2. Correlation between input and output channels after performing ICA on the fragment from My Favorite Things by John Coltrane. Note: as illustrated in Fig. 1, ICA can sometimes shift the phase of an estimated source, so the negative sign in the first correlation value does not have any implications – only the absolute value matters in this context.

While a high degree of correlation is expected between the ICA inputs and outputs, the almost perfect correlation in this case means the method simply returns the inputs without any substantial modifications. For context, below are the same correlation plots for the initial, successful example of source separation via ICA:

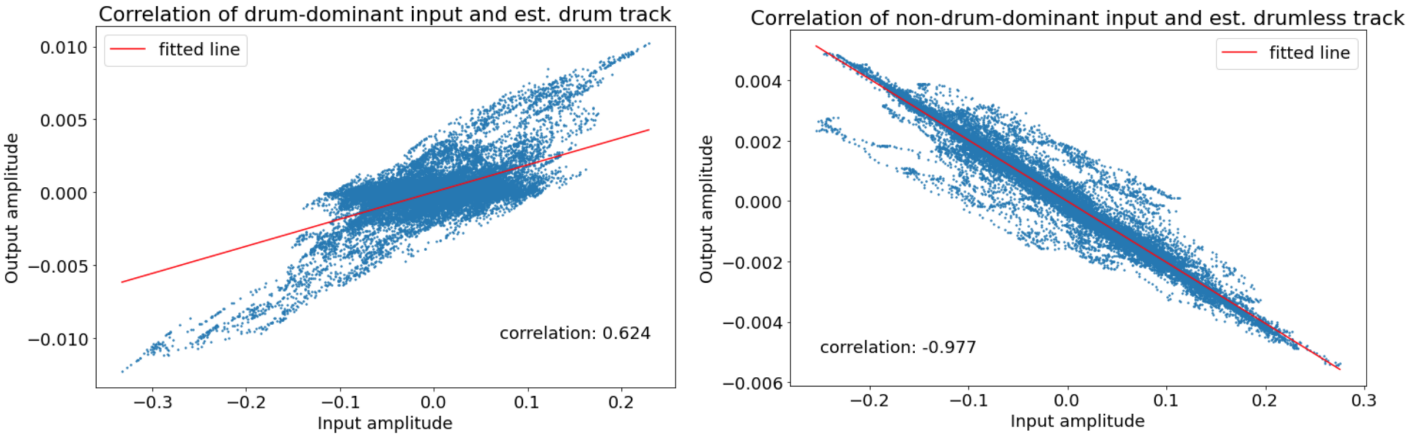

Figure 3. Correlation between input and output channels on the initial, successful example of sound source separation.

It is not entirely surprising that ICA does not perform a successful drum vs. non-drum separation (or any other meaningful instrument track separation) on mixed “music-in-the-wild” examples. From the point of view of an audio mixing engineer, there is no unified “drum track” – individual drum and cymbal tracks are mixed into the left and right mix tracks with variable coefficients. Similarly, there is no unified “non-drum track” – but a variable number of individual harmonic instrument tracks (voice, piano, bass etc.) that get blended together into the left and right mix tracks. In addition, there may be a number of non-linear effects like compression applied to the mixtures after the levels of individual instruments are set. Drum signal separation is thus not the separation of one instrument from a mix, but that of a class of instruments with different pitches and envelopes from mixes containing other varied instruments. Such a task is particularly difficult when relying on predefined, hand-crafted features, therefore a data-driven approach is necessary.

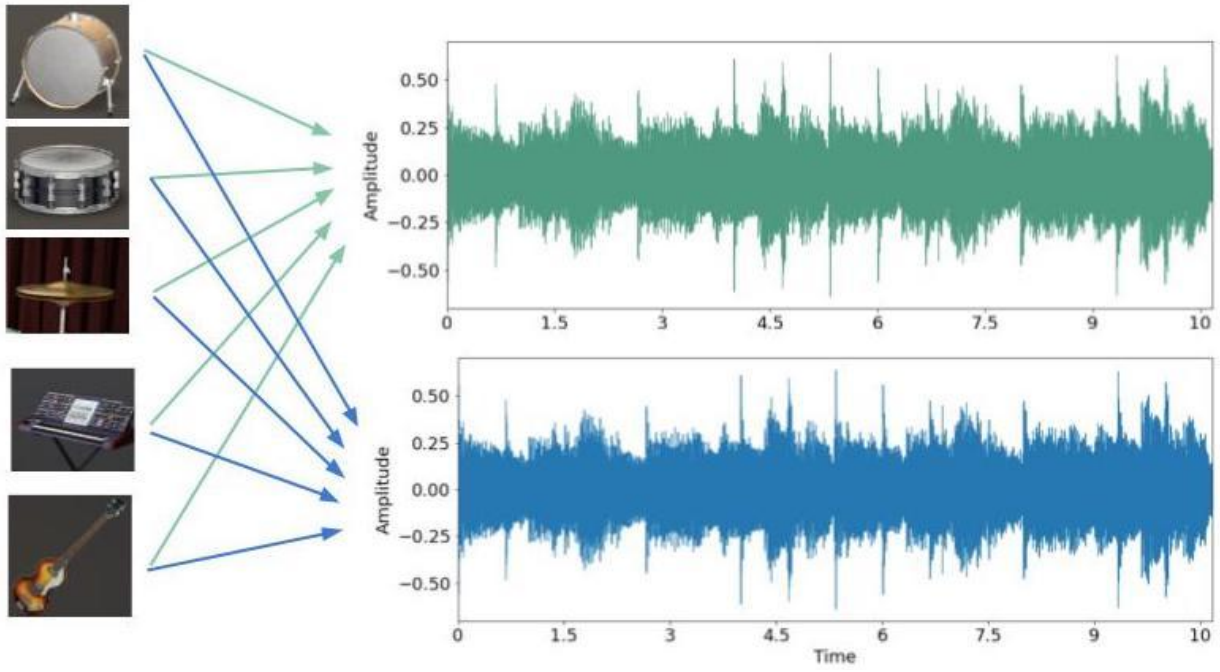

Figure 4. An illustration of the mixing stereo audio. The instruments in the top three images, the kick drum, the snare drum, and the high-hat cymbal are considered drums, while the piano and the bass guitar in the bottom two images are considered non-drum instruments. Instrument image source: Logic Pro X.

Neural Approach

Neural networks are used for a variety of tasks in signal processing, from classification, to de-noising, to generation. They are particularly useful for tasks in which it is impractical to rely on hand-crafted features and is instead necessary to learn a complex combination of simple functions from a large dataset. In the domain of monaural sound source separation, neural networks are the only approach that has produced competitive results in the past 5 years [4].

There are many recent neural network architectures that produce high quality results on sample-by-sample audio generation tasks, such as WaveNet [5] and WaveGAN [6]. But these models can be difficult to implement and may require large amounts of computational power and training time. Instead, this study uses a simple and lightweight architecture for the related task of singing voice separation, proposed by [7] and similar to that proposed in [8]. This model is essentially a Convolutional Neural Network (CNN) for image classification – a different task than the one this study is concerned with. Such a CNN is usually applied to images – 2-dimensional inputs of constant size – while our task relates to audio recordings – 1-dimensional inputs of variable length. In addition, this type of CNN is used for classification – mapping an image to a class label – while our task is concerned with audio source separation – mapping an audio recording to a related audio recording. This mismatch in the data structure is solved through a sequence of data preprocessing steps outlined in [7] – involving the Fourier transform, image slicing, and binary masking.

From Wave to Image

The first step in the preprocessing pipeline is the conversion of audio files, represented as long arrays of audio samples (for example, a10-second audio recording at sample rate of $22,050$samples/second is represented as a an array of size $10 \times 22,050 = 220,500$) into Short-Time Fourier Transform images. The Short-Time Fourier transform is highly useful in sound processing because it can represent sound over short windows of time as a sum of sinusoids of different frequencies. The vertical dimension in a STFT image represents frequency, while the horizontal dimension represents time in steps. With the choice for parameter $hop\ length = 513$,the 10-second recording is converted to an STFT image with $ceil(220,500 \div 513) = 430$timesteps. The height of the image is determined by the number of frequencies represented, which for this study was set to 513,like in [7].

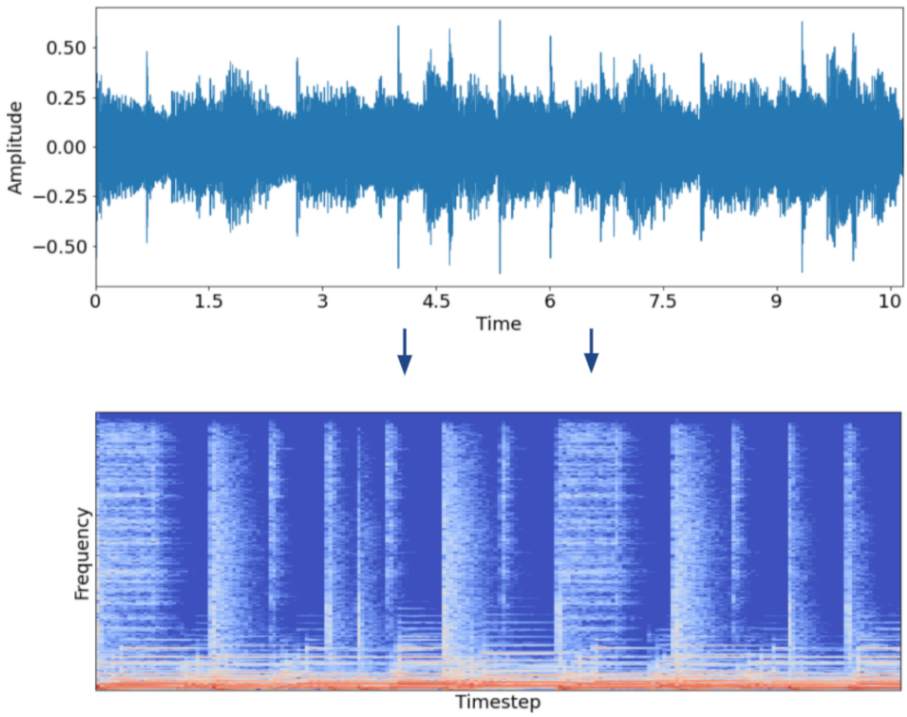

Figure 5. An illustration of the Short-Time Fourier Transform.

These STFT images are still not fixed-size – the horizontal length of an image depends on the length of the input audio file. Thus the next preprocessing step consists of slicing the initial STFT images into overlapping windows of size 25. This process turns a STFT image representing 430 timesteps and 513 frequencies into a list of 430 images of size 513×25, each representing a timestep in the original image.

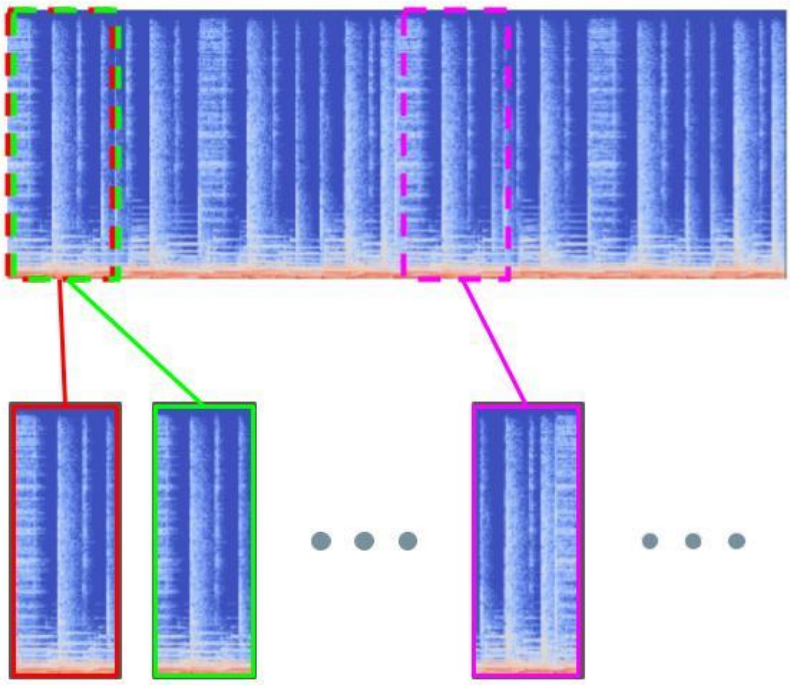

Figure 6. Slicing a $513 \times N$image into 513images that are 513×25.

The resulting list of images represents the input to the network. The labels consist of binary vectors of size 513 – same as the number of represented frequencies in the STFT spectrograms – where '1' means "drum frequency" and '0' means "non-drum frequency." The procedure for determining "drum" and "non-drum" frequencies is described in [8] – compare each STFT pixel in the drum signal spectrogram with the corresponding pixel in the non-drum spectrogram and set the value of the corresponding pixel in the output vector to '1' if the drum signal has a higher amplitude and '0' otherwise (Equation 1):

$$Y_{i,j} = \left\{1 \; if \; D_{i,j} > N_{i,j}; \; 0 \; if \; D_{i,j} \leq N_{i,j}\right\}, \tag{1}$$

where i is the index of the timestep, $j$is the index of the frequency, $D$is the drum signal spectrogram, $N$is the non-drum signal spectrogram, and $Y$is the output list of binary vectors.
Since Y has as many vectors as there are timesteps, and each vector is of the same size as the number of frequencies, Ycan be thought of as a binary mask of the same size as the initial STFT spectrogram. [8] refers to this mask as the Ideal Binary Mask (IBM).

The Ideal Binary Mask makes for a good set of labels to train a classification CNN network on. But estimating the IBM of a music mix spectrogram is still a classification problem – not a source separation problem. The gap between classification and separation is bridged by the Inverse Short-Time Fourier Transform (ISTFT). A mix spectrogram is multiplied by the IBM element-wise, which results in a new spectrogram where the "non-drum" frequencies are set to 0. An isolated drum audio file is then retrieved from this new spectrogram via Inverse Fourier Transform.

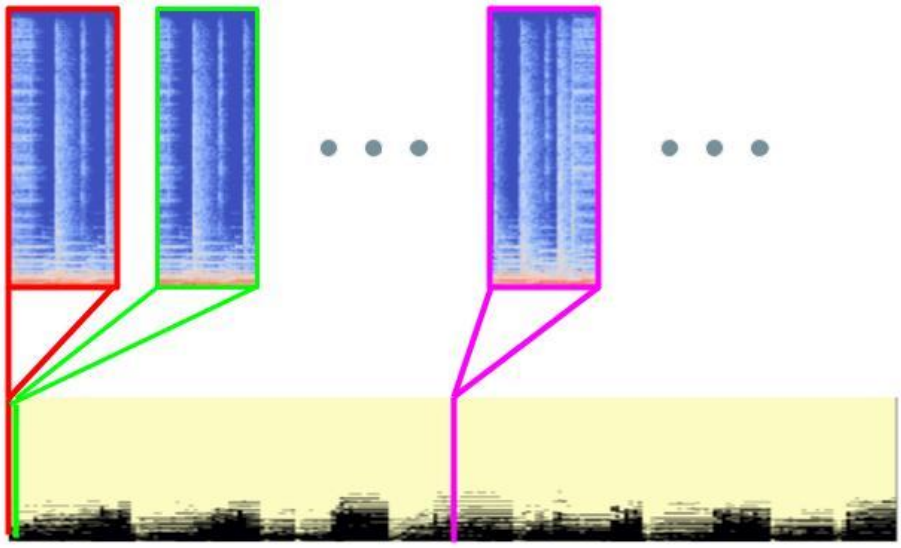

Figure 7. Input spectrogram slices and their corresponding columns in the Ideal Binary Mask.

It is important to note that this process of audio reconstruction is fundamentally lossy. There may be many frequencies that are prevalent in both the drum and the non-drum spectrograms, but any frequencies at a given time that have higher values in the non-drum spectrogram will be suppressed, leading to a deteriorated reconstructed drum sound. This kind of artifact is inevitable in the current approach to drum source separation. Furthermore, ideal binary masking sets the upper limit to the quality of retrieved sound in the ideal-case scenario – in practice the neural

network does not learn to predict the IBM perfectly, thus the quality of the retrieved sound on test set examples is necessarily lower than the already deteriorated IBM-reconstructed audio.

Nevertheless, imperfect but still high quality sound can be reconstructed using Ideal Binary Masking in the frequency domain. Three short examples from the MUSDB18 test set [9] (*ibm_ex1_mix.wav*, *ibm_ex2_mix.wav*, and *ibm_ex1_mix.wav*) as well as their corresponding IBM-reconstructed drum tracks (*ibm_ex1_sep.wav*, *ibm_ex2_sep.wav*, and *ibm_ex1_sep.wav*) are provided in the Accompanying Audio folder to illustrate the quality of such reconstruction.

Dataset

This study has used the MUSDB18 [9] dataset for all training and testing. The MUSDB18 train set and test set contain 100 songs (~6.5 hours of audio) and 50 songs (~3.5 hours of audio) respectively. For each song the following tracks are provided: “drums,” “bass,” “other,” “vocal,” and “mixture” (track combining together all the others). The “bass,” “other,” and “vocal” track were combined into the non-drum track, which was then used against the drum track to determine the IBM. The mixture track was converted to STFT frames and sliced as described in the previous section to create the network input. After preprocessing, there were $985,7762$ frames in the train set and $537,108$in the test set. At training time, only training data was used, with a 0.2 validation split, while all the test data was saved for subsequent perceptual evaluation.

Model

The main model used in the training process is the same as that proposed in [7]. Some modifications were attempted, but changes to the model architecture did not seem to lead to better results. The model is represented in Table 1:

Input
Conv2D, 32 3x3 filters. Activation: Leaky ReLU
Conv2D, 16 3x3 filters. Activation: Leaky ReLU
MaxPool2D, 3x3.
Dropout, rate: 0.25.
Conv2D, 64 3x3 filters. Activation: Leaky ReLU
Conv2D, 16 3x3 filters. Activation: Leaky ReLU
MaxPool2D, 3x3.
Dropout, rate: 0.25.
Dense, 128 units. Activation: Leaky ReLU
Dropout, rate: 0.5.
Dense, 513 units. Activation: Sigmoid

Loss: Binary Cross Entropy
Output
Total trainable parameters: 323,233

Table 1. Initial model architecture.

Training

The initial model was trained on 3 different versions of the preprocessed input data: raw STFT spectrograms, spectrograms normalized by feature based on training set statistics, and spectrograms with amplitude values transformed to decibels. Loss, recall, and precision plots are presented in Figure 8. Precision and recall were chosen as metrics for training because these metrics are more suitable for the multi-class problem the network learned to solve.

Counterintuitively, the choice of data scaling (none, min-max normalized, or converted to decibels) did not seem to have a significant impact on training – all 3 networks maintained roughly the same training values throughout the process.

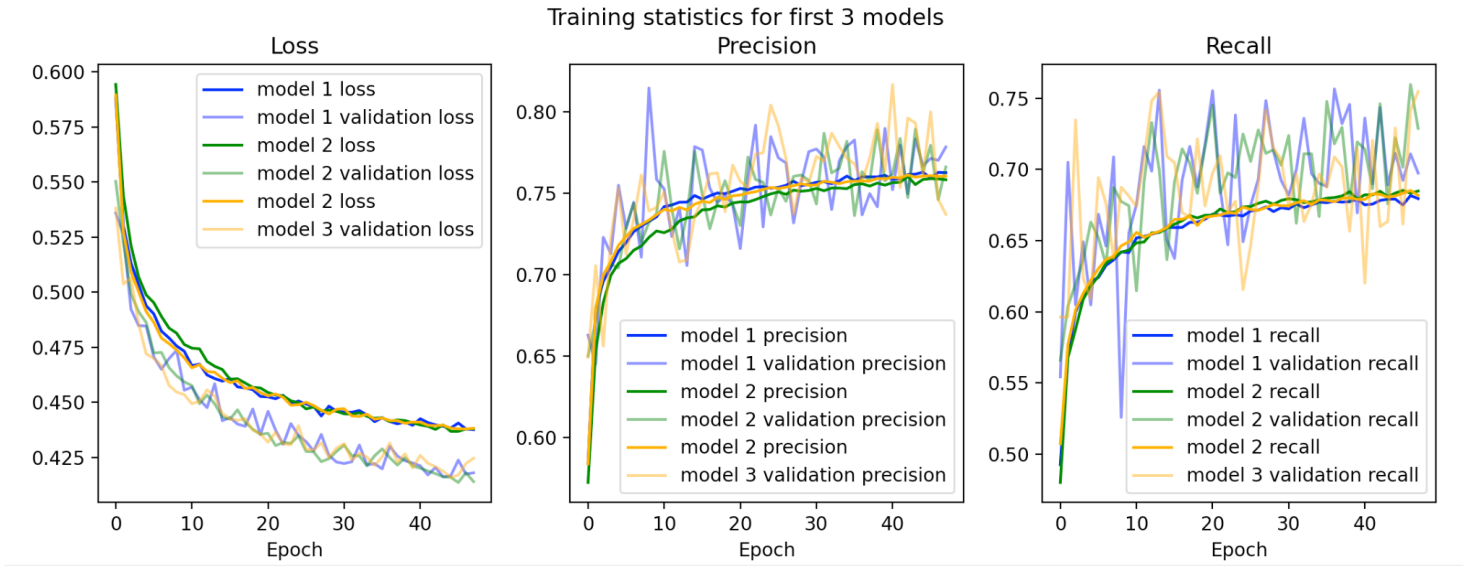

Figure 8. Training statistics for the first 3 models.

The next variation of the model consisted of making the model more powerful – the number of filters was doubled in each convolutional layer and the number of units in the first dense layer was doubled as well. This modification increased the total number of training parameters to $1,158,977$ – almost 4times as many as in the initial example. The training statistics for this model are presented below, along the ones for the first model, for reference:

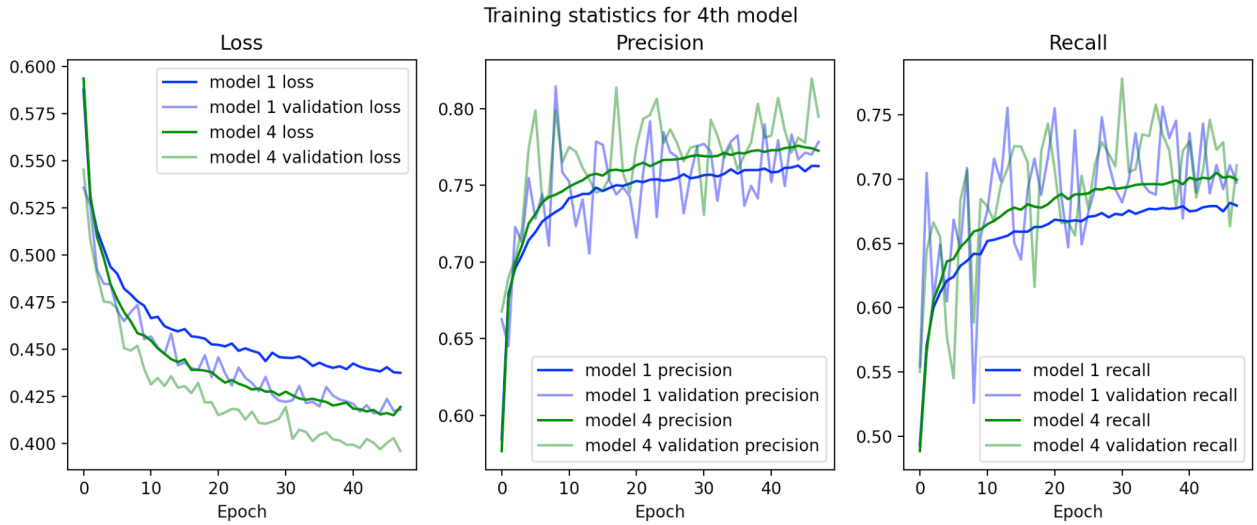

Figure 9. Training statistics for the 4th model and 1st model for reference.

Although this model did take about 3-4 times longer to train, because of the higher number of trainable parameters, its training statistics (lower loss values and higher precision and recall values) seem to suggest this more powerful model to be promising.

Another network architecture variation was attempted for model 5 – another convolution stack (two convolutional layers, a max pooling layer, and a dropout layer) was added before the first dense layer. In addition, the filter size in the max pooling layer was changed from 3×3 to 2×2 drop out rate was reduced from 0.5 to 0.3 in the last dropout layer. The training statistics for the 5th model are presented below:

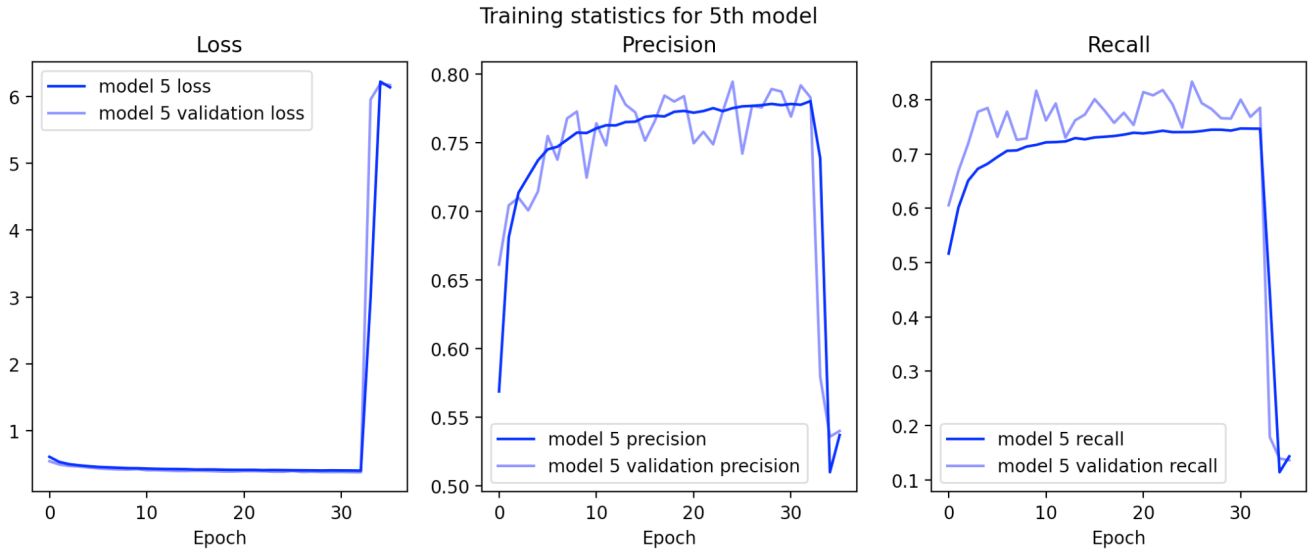

Figure 10. Training statistics for the 5th model

It appears that the model starts to overfit sharply after the 32nd epoch – loss rises rapidly, while precision and recall plummet. This may have been caused by the reduction of the dropout rate in the last dropout layer.

Data Augmentation via Transposition

Deep learning models usually benefit from having more training data available. For this last network variation, the training data was augmented by transposing the harmonic parts (the non-drum parts) using the pitch-shifter-py package [10] up by 4 semitones and down by 4 semitones. This process degrades the quality of the harmonic parts, but that is not considered a problem, since the drum track stays the same. This kind of augmentation only makes sense for the drum separation task – the drum part is the only one that stays the same regardless of a song's key. The architecture used for this network variation is the same as the initial one. The training statistics for the 6th network are presented below, along with those of the 1st one, for reference:

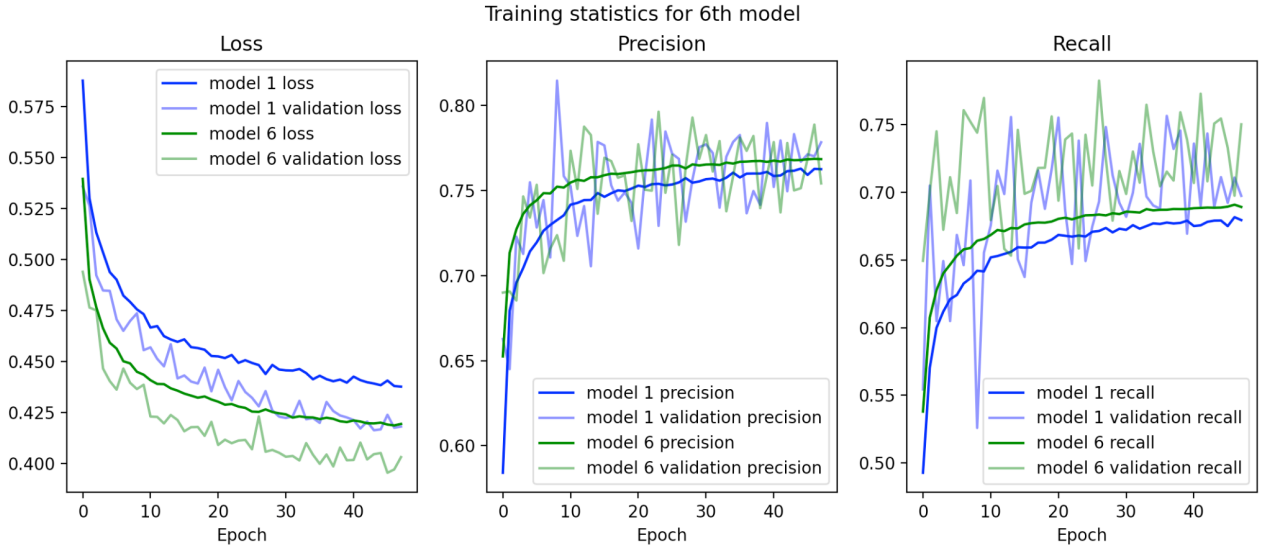

Figure 11. Training statistics for the 4th model

The training statistics for this last model appear promising – lower loss, higher recall and precision. But training statistics only have the purpose of guiding the training process – they do not tell us anything about how well a model performs. A perceptual evaluation was performed using the Sourced to Distortion Ratio metric (SDR).

Perceptual Evaluation

The best evaluation method for a sound source separation approach is to have humans listen to the inputs and outputs and rate the quality of the outputs. But since this type of evaluation is highly costly and time-consuming, automated metrics are often used instead.

The perceptual evaluation was performed with the museval implementation [11] of the Source to Distortion Ratio (SDR). SDR is a popular proxy for human evaluation, and while some of its flaws have been pointed out [12], it was deemed good enough for the purposes of comparing the 6 training approaches in this study.

Before the models were evaluated, SDR values were calculated for Ideal Binary Mask reconstruction of the drum part in mixes. This is an important part of the evaluation process, because, as pointed out previously, IBM reconstruction sets the upper limit for the quality of sound that can be achieved with the current approach. All models were then applied to the test set (50 songs, ~3.5hours of audio, $537,108$ STFT frames) and SDR values were calculated for the drum

sound separation by song. Box-and-whiskers plots are presented below representing the resulting SDR ranges resulting from drum separation by the 6 models, as well as the IBM reconstruction process:

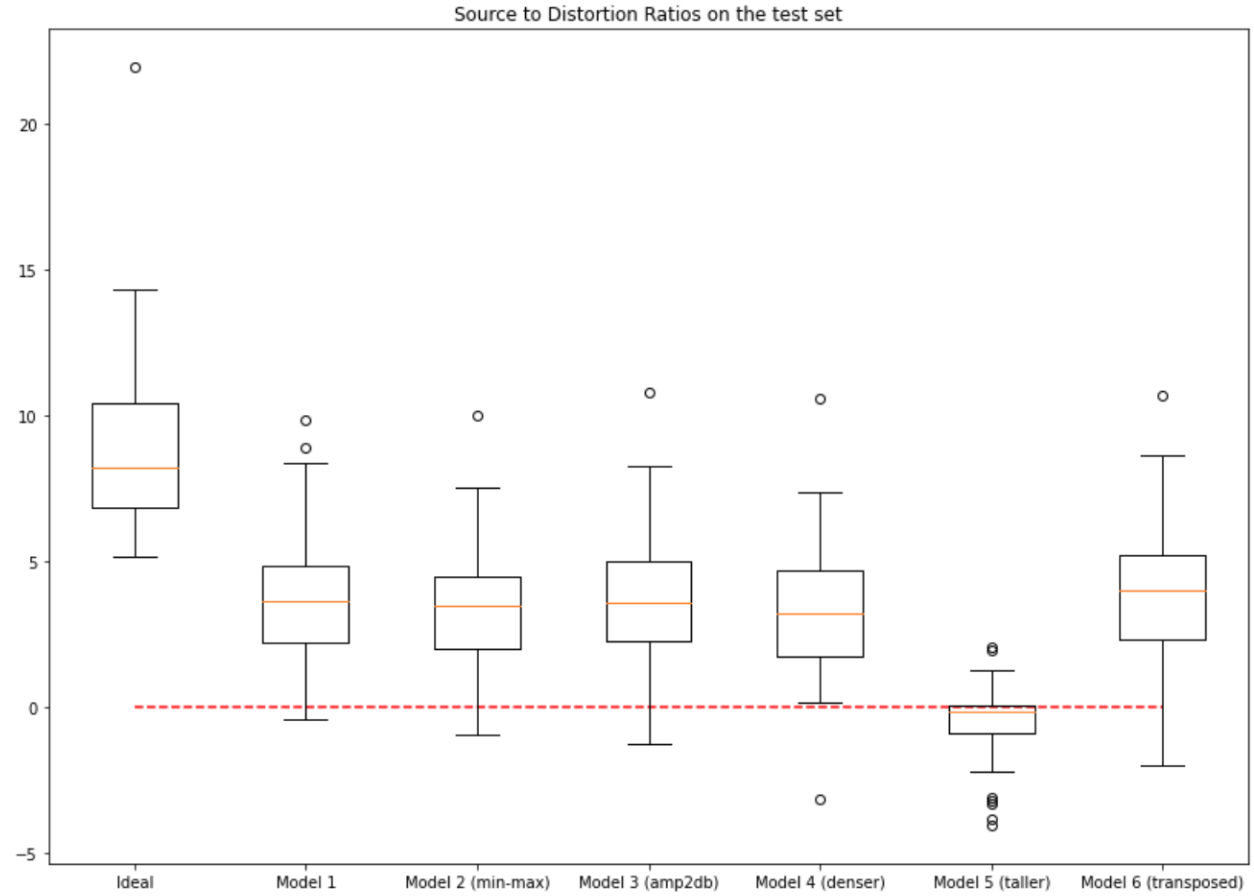

Figure 12. SDR values for IBM separation as well as separation by the 6 models. Note that SDR is represented here in decibel notation: $SDR_{dB} = 10log_{10}(SDR)$. Values greater than 0 represent more desirable signal in the output, while values less than 0 represent more noise.

Statistic\Model	Ideal	Model 1 (initial)	Model 2 (min-max)	Model 3 (amp2db)	Model 4 (denser)	Model 5 (taller)	Model 6 (transposed)
Q3	10.4	4.87	4.52	5.02	4.72	0.06	5.23
Median	9.24	3.67	3.50	3.62	3.22	-0.13	4.05
Q1	6.86	2.24	2.03	2.27	1.75	-0.88	2.32

Table 2. First quartile, median, and third quartile for the SDR values obtained on the test set.

Discussion

Some of these results are not surprising. The first 3 networks had almost identical training statistics, so it was expected that they would perform similarly on perceptual evaluation. Model 5, the one that steeply overfit and reached low values for precision and recall, was expected to not

reach good SDR values (subjective evaluation confirmed – the outputs of this network were completely meaningless).

More surprising are the results of Model 4. This model had 4 times more trainable parameters than 1,2,3, and 6, took 3-4 times longer to train, and showed significantly better training statistics. Its performance on the SDR evaluation however appears all but identical to that of the first 3, lighter models. This may be an indication that in the context of this task, after a certain point a more powerful network can reach lower loss and higher precision and recall, but these improvements in training statistics do not necessarily translate into meaningful improvements on actual sound separation.

The most encouraging are the results of Model 6. This model has the same exact architecture as the first 3, but it was instead trained on augmented data. Even though the SDR values for Model 6 are only slightly higher than those of the first 3 networks, it is also worth noting that the evaluation was performed on $537,108$ STFT frames – so even small numerical improvements may be an indication of consistently higher performance.

This result is particularly important in the context of drum signal separation, which is the only task where such data augmentation by transposition can be deployed without affecting the quality of the sound that needs to be separated from the mix. Furthermore, in future exploration, the data could be augmented even more by transposing each song to the remaining 9 keys (there are 12 keys in Western music notation), thus effectively quadrupling the size of the dataset at virtually no cost (other than the storage space required).

Three examples (*neural_ex1_mix.wav*, *neural_ex2_mix.wav*, and *neural_ex3_mix.wav*) are provided in the Accompanying Audio Folder – the first one from the training set, the second from the test set, and the third one from outside the dataset. The corresponding outputs of the drum separation network (Model 6) are provided as *neural_ex1_sep.wav*, *neural_ex2_sep.wav*, and *neural_ex3_sep.wav*. The model seems to produce similar results on all three examples – an encouraging result suggesting that the model really learns the task and not just features of the dataset.

Conclusion

This study examined the task of drum signal separation from full mixes. It illustrated how classical methods such as ICA are insufficient to produce meaningful results, and instead data-driven methods such as neural networks are required. Different approaches were tested for improving performance of a model proposed in works. While different choices of data scaling did not seem to have any effect, increasing the number of trainable parameters seems to have either no effect or negative effect (in the case of Model 5, where the dropout rate was also reduced), and data augmentation seems to have a positive effect on model performance on a perceptual evaluation metric (SDR). Even higher quality of drum separation could potentially be achieved by simply increasing the amount of data augmentation (transposing the harmonic parts to more keys).

Acknowledgements
This research was supervised by Prof. Oliver Layton and supported by the Colby College Computer Science Department as part of the Honors Program.

Accompanying Audio
The accompanying audio examples referenced throughout the text are available on Google Drive as well as on this project's github repository:
https://drive.google.com/drive/folders/19-N-gTAoqt4eqm8XonXne_pz8SVDmhZV?usp=sharing
https://github.com/morehovschi/drumsep

References:

[1] James V. Stone (2002). Independent component analysis: an introduction. *Trends in Cognitive Sciences, 6(2), 59-64.* https://doi.org/10.1016/S1364-6613(00)01813-1

[2] A. Hyvärinen. Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *IEEE Transactions on Neural Networks* 10(3):626-634, 1999.

[3] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12, 2825–2830.*

[4] Stöter F.R., Ulich S (2020, September 28-29). *Current Trends in Audio Source Separation* [Virtual Conference Presentation]. AES Virtual Symposium: Applications of Machine Learning in Audio. Accessed May 18, 2021.
https://www.youtube.com/watch?v=AB-F2JmI9U4&t=253s&ab_channel=sigsep

[5] Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016, September 19). WaveNet: A Generative Model for Raw Audio. arXiv.org. https://arxiv.org/abs//1609.03499.

[6] Donahue, C., McAuley, J., & Puckette, M. (2019, February 9). Adversarial Audio Synthesis. arXiv.org. https://arxiv.org/abs/1802.04208v3.

[7] Koretzky A. (2019). Audio AI: isolating vocals from stereo music using Convolutional Neural Networks. *Towards Data Science*. Accessed May 18, 2021.
https://towardsdatascience.com/audio-ai-isolating-vocals-from-stereo-music-using-convolutional-neural-networks-210532383785

[8] Lin K. W. E., Balamurali B.T., Koh E., Lui S., Herremans D. (2018). Singing Voice Separation Using a Deep Convolutional Neural Network Trained by Ideal Binary Mask and Cross Entropy. *Neural Computing and Applications, Springer*. https://arxiv.org/abs/1812.01278

[9] Rafii, Zafar, Liutkus, Antoine, Stöter, F.-R., Mimilakis, Stylianos Ioannis, & Bittner, Rachel. (2017). MUSDB18 — A corpus for music separation [Data set]. *Zenodo*. https://doi.org/10.5281/zenodo.1117372

[10] Woodall, Christopher (2021). pitch-shifter-py. *github*. Accessed May 17, 2021. https://github.com/cwoodall/pitch-shifter-py

[11] Stoter, F.R., Liutkus, A., & Ito, N. (2018). The 2018 Signal Separation Evaluation Campaign. In Latent Variable Analysis and Signal Separation: 14th International Conference, *LVA/ICA 2018*, Surrey, UK (pp. 293–305). Accessed May 18, 2021. https://github.com/sigsep/sigsep-mus-eval

[12] Le Roux, J., Wisdom S., Erdogan, H. , Hershey, J.R. (2018). SDR – Half Baked or Well Done? *IEEE International Conference on Acoustics, Speech and Signal Processing*. https://arxiv.org/pdf/1811.02508.pdf