MUSIC SOURCE SEPARATION USING DEEP LEARNING

Siddharth Goel, Lejun Huang, Olaoluwa Owoputi, Prasanna Poudyal

University of Pennsylvania

1. INTRODUCTION

In this project work, we study the music source separation (MSS) problem using various classical and deep learning methods. We present separation of individual source stems from a mixture. Music source separation (or Audio Source Separation (ASS)) involves isolating each of the source stems in an audio. A *stem* is an individual unit of a song, typically corresponding to a particular musical instrument (e.g. guitar, bass, vocals) or group of instruments (e.g. accompaniments, percussion). All the stems when combined constitute an audio track or "mix". The decomposition of a mix into its constituent stems serves many purposes in the music and entertainment industry. For instance, stem tracks are widely used to create DJ sets and remixes. Stems are also used by live performers to create new variants of their work. Generating isolated instrumental tracks for karaoke is another one of many real-world applications of MSS.

In this work, we use the musdb18 dataset to solve the challenging MSS problem using one classical method and three deep learning methods. The musdb18 dataset consists of 150 mixed audio tracks along with their four constituent stereo stems: drums, bass, vocals, and accompaniments. These individual stems act as labels in our supervised learning problem. Though this problem has its roots in the field of signal processing, recent research and developments have led to learning algorithms based state-of-the-art technology. Various deep learning-based techniques have been utilized to learn models that extract specific target instruments from the mix. We start by looking at classical methods such as Non-negative Matrix Factorization (NMF) as a baseline and then move on to explore two state-of-the-art solutions: Open-Unmix and Demucs. The rest of the report is organized as follows.

In the next section, we discuss some of the existing related works which have achieved this task. We then explain our main methods along with their underlying approaches and architectures. Following this, results obtained from the experiments conducted using these methods are recorded and compared. Finally, we conclude our discussion and highlight the scope of future work in this domain.

***Index Terms*—** Music Source Separation, Blind Signal Separation, Deep learning, Audio deep learning, musdb18

2. RELATED WORKS

U-Net [1]

This work introduces the well-known U-Net, which is a convolutional autoencoder network with residual "skip" connections between the encoder and decoder. These skip connections prevent diminishing gradients and allow for information preservation between encoder and decoder layers. U-Net was initially proposed as an algorithm for medical image segmentation but has since been adapted to a number of computer-vision domains including audio processing. Moreover, many state-of-the-art audio separation models are based on U-Net to some degree—including our deep-learning benchmark Demucs.

Conv-TasNet [2]

In this paper, a Conv-TasNet (convolutional time-domain audio separation network) is presented as an architecture for separating speech audio into individual voices. Conv-TasNet differs from most other audio separation networks in that it forgoes the Fourier transform and instead represents input audio as a 1-dimensional time series. The justification for this is that convolution over frequency space can result in frequency distortion - to which humans are highly sensitive. The Conv-TasNet model consists of an encoder-decoder convolutional network which outputs a separate mask for each voice; masks are combined with the latent encoding of the original audio in order to produce the isolated voice. However, the model suffers from information loss (which is inherent in autoencoder architectures) and decreased performance on long or noisy audio. This paper was of particular interest to our group due to the model's waveform-to-waveform architecture and usage of time-dimension convolution. This idea has been adapted by Facebook AI researchers to achieve state-of-the-art perfor-

mance on musdb18.

MMDenseNet [3]

This paper introduces a multi-scale, multi-band DenseNet (MMDenseNet). This architecture is based on the original DenseNet proposed in 2016 which is a convolutional network with residual connections from each layer to all successive layers. The authors used DenseNet blocks with a U-Net style architecture such that the high-resolution blocks close to the input layer contain fine spectro-temporal details while the low-resolution blocks in the middle contain broad spectro-temporal context (hence, "multi-scale"). Furthermore, the MMDenseNet processes information in different frequencies independently by learning convolution kernels that are specific to each frequency band (hence, "multi-band"). This strategy was later expanded by the authors to create MMDenseLSTM.

MMDenseLSTM [4]

MMDenseLSTM is an extension of MMDenseNet which combines dense blocks with biLSTM blocks. This modification was intended to give the network the capability to model long temporal contexts, thereby improving its audio separation performance. This model was later used as a benchmark against which the Demucs model was compared [5].

3. METHODS

3.1. Data Exploration

MUSDB18 is the largest freely available dataset for source separation to date. It consists of 150 full-length music tracks (totalling 10 hours) from different genres along with their isolated drums, bass, vocals and accompaniment stems [6]. The training set is comprised of 100 tracks, with the remaining 50 tracks constituting the test set. Each instance in the dataset is a 44.1kHz stereo track composed of the full mix plus four stems. The stems are labeled as follows.

The musdb18 dataset can be accessed through the Python package `musdb`. Each audio track in the dataset is in compressed .mp4 format. There is also the option to use the uncompressed (.wav) format as well if needed. The tracks are 7 seconds in length.

In this section, we look at one of the samples in the musdb18 dataset in detail along with its attribute values. An example of attribute values for one of the sample tracks is described in Table 2. These attributes make data processing simple, straightforward, and easy to use. Another advantage of the musdb18 dataset is that it serves as standard in the MSS field which makes it easier to compare different separation models with each other.

Table 1. musdb18 audio track components

Channel	Description
0	Mixture
1	Drums
2	Bass
3	Accompaniments
4	Vocals

3.2. Data Pre-processing

The `audio` and `stem` attribute of the track contain the time domain signal. These arrays represent the amplitude (loudness) of the signal with respect to time. The sampling rate is the number of samples of amplitude recorded per second.

To represent the audio clips as features, we need to convert the time-domain signal into the frequency domain. Frequency values are required at each time step along with their magnitude. A spectrogram allows us to represent our audio data in this format.

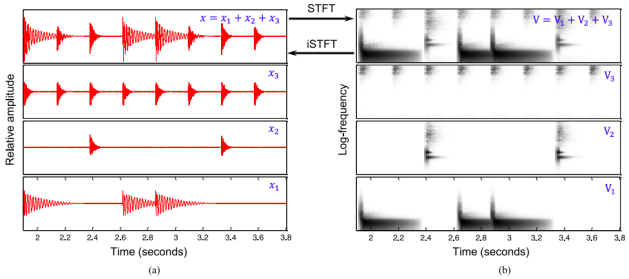

Fig. 1. Signal representation in the time (left) and frequency (right) domain. The arrows show the conversion from one form to the other using STFT and I-STFT transformations.

In Figures 2 and 3, we represent the magnitude spectrogram for our above sample mixture track and the individual source stems using the PyTorch function `torch.STFT()`. In a spectrogram, the x-axis represents time, the y-axis represents frequency and the color represents magnitude (spectral power). In some sense, we have converted the audio data into an image, transforming this from an audio problem to a computer vision problem. The difference in patterns across

Table 2. musdb18 audio format and metadata

Attribute	Description	Example
name	Song title and artist	"Al James - Schoolboy Fascination"
path	Absolute path of the track	".../Al James - Schoolboy Fascination.stem.mp4"
audio	Array containing the stereo mix	(300032 x 2) array
rate	Sample rate (Hz)	44100
sources	Paths to the stems	{'bass': .../Al James - Schoolboy Fascination.stem.mp4, 'drums': .../Al James - Schoolboy Fascination.stem.mp4, 'other': .../Al James - Schoolboy Fascination.stem.mp4, 'vocals': .../Al James - Schoolboy Fascination.stem.mp4}
stems	Array containing the stems and full mix	(5 x 300032 x 2) array
targets	Targets for track separation	OrderedDict([('vocals', vocals), ('drums', drums), ('bass', bass), ('other', other), ('accompaniment', bass+drums+other), ('linear_mixture', vocals+bass+drums+other)])

signals can be noted from their respective spectrograms in Figure 3. This further validates our initial hypothesis that a mixture track can be separated into individual source stems.

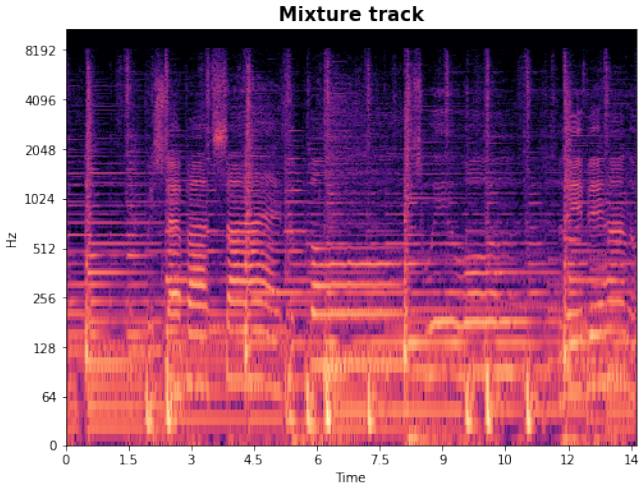

Fig. 2. Spectrogram of mixture track/stem.

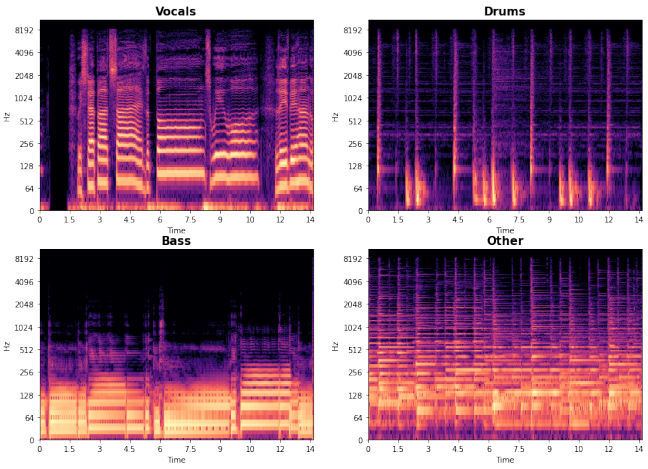

Fig. 3. Spectrograms corresponding to individual sources

3.3. Data Post-processing

3.3.1. Wiener filtering

Wiener filtering is one of the most popular source separation techniques in signal processing. This technique filters the desired source signal from a mixture by learning a filter. The Wiener filter minimizes the mean square error between the estimated source signal and the desired source signal.

One of the Python implementations of a multi-channel Wiener filter is `norbert`. Norbert works under the assumption that we have the desired target audio sources which act as labels. The `norbert` implementation builds and applies the filter that is appropriate for separation, by optimally exploiting multi-channel information. This technique leverages the Expectation Maximization (EM) algorithm, where filtering and re-estimation of the parameters are iterated. To apply Wiener filters on our hidden spectrogram representation of the mixture audio signal, we leverage the `norbert.wiener` method in the `norbert` package along with the target separated signal.

3.3.2. Inverse STFT

Finally, we do an inverse STFT on the filtered spectrograms. This converts the output spectrograms back to the time-domain signal. The `torch.istft()` function in PyTorch is used for this purpose. Figure 1 shows the conversion of the frequency-domain signal back to a waveform.

3.4. Data Hypothesis

Our first hypothesis is that the model should be independent of source instruments. A good model should be able to separate any combination of source instruments present in the dataset labels and not just the labels in musdb18. As mentioned above, the current dataset consists of stems corresponding to drums, bass, vocals, and accompaniments. If, for example, we substitute drums with flute or bass with guitar the model should be able to separate flute and guitar in a similar fashion.

Secondly, we believe that model should be able to filter and separate sources in combinations of two or more depending on the training label. For example, if we train the model using a combination of vocals and guitar stems as the label, the model should be able to output a combination of vocals and guitar from the mixed track and not just single-source stems.

Lastly, if we combine the source-separated outputs of the model, we should be able to reconstruct the original audio track with negligible loss. Otherwise, this would indicate that the model is adding noise by itself to one or more of the output stems.

3.5. Non-Deep Learning Baseline: Non-negative Matrix Factorization (NMF)

Non-negative matrix factorization (NMF) is one of the classical methods often used to decompose the magnitude of time-frequency distributions in audio processing. Apart from source separation, NMF has been used for tasks such as transcription and structure analysis in music processing. Application of NMF to MSS has been discussed in several works [7, 8, 9]. Here, we present a high-level overview of the NMF technique. The fundamental process of NMF is factorizing the matrix V containing the audio data spectrogram into two separate matrices referred to as bases (W) and activations (H) respectively. All three matrices V, W, H are non-negative.

$$\mathrm{V}_{m\times n} = W_{m\times r} H_{r\times n} \tag{1}$$

where $r = min(m, n)$.

The parameter r represents the number of sources or stems that we want to isolate from the mix.

In other words, each column of V can be written as a linear combination of the columns of W. Having said that, NMF is an NP-hard problem and has multiple solutions. Therefore, Eqn. 2 serves as an approximation.

$$\mathrm{U} \approx WH \tag{2}$$

NMF starts with a suitable initialization of W and H depending on the task and availability of prior information. All the values in both W and H are non-negative. After initialization, both W and H are iteratively updated to approximate V with respect to a cost function. Different implementations involve different cost functions, but the standard cost function used is the Kullback-Leibler Divergence (KLD).

$$D_{KL}(\mathrm{V} \parallel U) = \sum \left(V \odot \log \frac{V}{U} - V + U \right) \tag{3}$$

The symbol $\odot$ denotes element-wise multiplication. The logarithm and division are to be performed element-wise as well. The sum is to be computed over all $m \times n$ elements of V.

The above loss function is convex in either W or H but not both. Therefore, we need to use iterative algorithms to minimize the above loss function. The multiplicative updates for W and H are as follows.

$$W \leftarrow W \odot \frac{\frac{V}{U} \times \mathrm{H}^T}{J \times \mathrm{H}^T} \tag{4}$$

$$H \leftarrow H \odot \frac{\mathrm{W}^T \times \frac{V}{U}}{\mathrm{W}^T \times J} \tag{5}$$

where J is a matrix of ones with dimension $r \times n$. These update rules are typically applied for a limited number of iterations.

In this implementation, a variant of NMF called non-negative matrix factor deconvolution (NMFD) specified in [9] is applied to magnitude spectrogram of the audio track. The details of the algorithm can be referred to in [7]. The parameter r is set to 4 to match the number of sources in the mixed track. V is set to the input spectrogram and W is initialized to the data-driven statistical mean of many source sounds of that type while H is initialized to all ones. The multiplicative updates to approximate W and H are done for 30 iterations. After NMF estimates are obtained, spectrogram estimates for the components are computed using Wiener filtering. The final separated audio components are obtained after applying an inverse STFT.

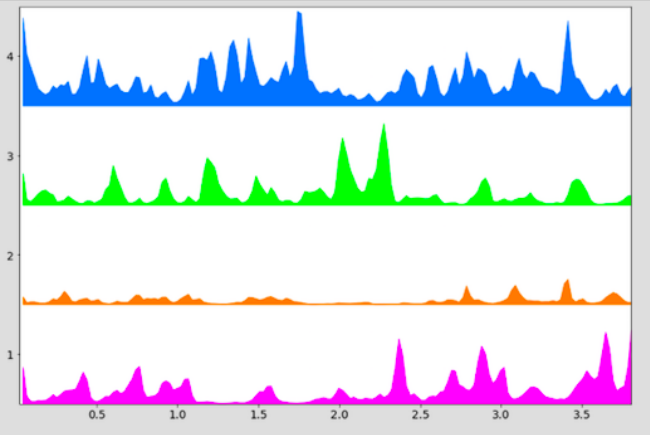

Fig. 4. Spectrogram of a mixed audio track.

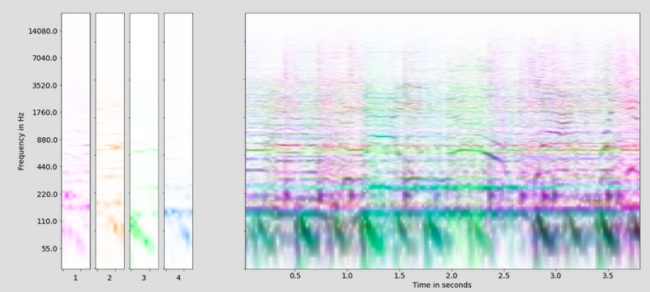

Fig. 5. Spectrograms of individual audio sources.

We leveraged the Python package of the NMF toolbox mentioned in [9]. The tracks from the musdb18 dataset were downloaded and converted into WAV format. The audio signal was converted from stereo to mono. The STFT transform of the mono channel time domain signal was passed to the NMFD method to get the source separated outputs. The hyper-parameters for this model are specified in Table 3. The spectrograms of the source separated outputs were visualized and compared with the spectrogram of the mixture audio track. Figures 4 and 5 contain the spectrograms of the respective audio signals before and after separations. Also, the separated source tracks were played for human evaluation using the interactive Python library display method. The energy ratio results calculated using BSS Eval metrics are compared along with the results of other methods in the next section.

Table 3. NMF hyperparameters

Parameter	Value
No. of components	4
No. of iterations	30
No. of template frames	8
W Initialization Strategy	random
H Initialization Strategy	uniform
STFT fft size	4096
STFT window size	4096
STFT hop size	1024
no. of channels	1

3.6. Deep Learning method: Open-Unmix

3.6.1. Open-Unmix Overview

Open-Unmix is one of the state-of-the-art deep neural networks for music source separation. The model learns to predict the magnitude spectrogram of a target, from the magnitude spectrogram of a mixture input.

The Open-Unmix model is a combination of a feed-forward neural network and a recurrent LSTM neural network. The raw outputs of the feed-forward neural net go into the LSTM network as input and are concatenated with the raw output of the LSTM network using skip connections. This concatenated output is passed through another feed-forward net before performing a multi-channel Wiener filter post processing. [10]

The three recurrent LSTM network layers take into account the context information from neighbouring mixture frames, capturing the temporal structure of the song.

3.6.2. Model Architecture

Open-Unmix is comprised of a three-layer bidirectional LSTM network along with dropout in combination with fully connected layers, batch-normalization layers, and ReLU nonlinearity. The architecture is shown in Figure 6.

3.6.3. Model Pipeline

The samples from the dataloader are converted into the spectrogram representation. The input magnitude spectrogram is standardized using the global mean and standard deviation for every frequency bin across all frames. The standardized magnitude spectrogram of the training samples are passed to the network. The forward pass of the model passes the input through the layers in order as described above. The output of the last ReLU layer is multiplied with the input magnitude spectrogram so that the model is able to learn a mask. This is achieved using norbert which is described in the previous section.

The model returns spectrograms corresponding to each

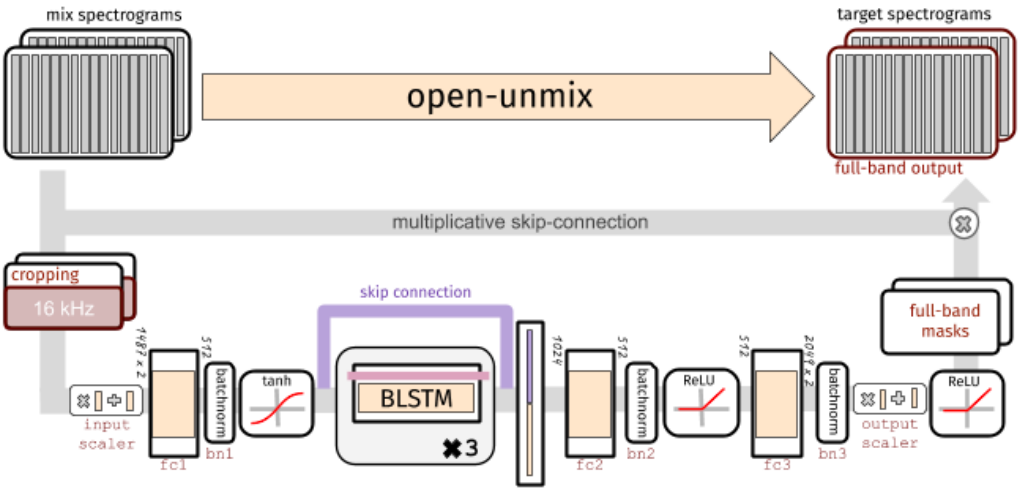

Fig. 6. Open-Unmix model architecture. Note the 3-layer biLSTM surrounded by skip connections.

Table 4. Open-Unmix hyperparameters (pre-trained)

Parameter	Value
Epoch	1000
Batch Size	16
Learning Rate	1e-3
Weight Decay	0.00001
STFT fft size	4096
STFT window size	4096
STFT hop size	1024
hidden size of dense bottleneck layers	512
maximum model bandwidth (Hz)	16000
no. of channels	2

source. Mean squared error loss is calculated between the output of the model and the target label spectrograms. The weights are adjusted using the Adam optimizer with a learning rate of 10^{-3}. The pre-trained model has been trained for 1000 epochs with a training batch size of 16. Refer to Table 4 for the full list of parameters.

Internally, the prediction is obtained by applying a mask on the input. The model is then optimized in the magnitude domain using mean squared error and the actual separation is done in a post-processing step involving a multi-channel Wiener filter. To perform separation into multiple sources, separate models are trained for each target. While this makes the training more complex, it allows for the flexibility to customize the training data for each target source.

Apart from transfer learning, we also trained the open-unmix model using our own set of hyper-parameters. A different model was trained for each target using the target stem. All the four models were trained for 50 epochs and the details of training along with hyper-parameters are mentioned in section 4. Finally, the model was inferenced using a test set sample and the separations were evaluated using the standard evaluation metrics.

3.6.4. Open Unmix Extension: AlexNet style

We also tried an extension to the existing architecture of Open Unmix. Inspired from AlexNet, we trained 4 Open Unmix model at the same time with connections within the models for feedback. While the original has only one skip connection from Tanh to fc2 layer as shown in Figure 6, our extension has 3 more signals coming from the other 3 models being trained. Similar to how AlexNet [11] specializes by having two major pipelines which are laterally connected by convolutional layers, our extension also allows for signals within the models for efficient training. This approach also saves training time as a lot of overhead for training 4 target models is eliminated. Results for our approach are discussed in more detail in Section 4.2

3.7. Deep Learning method: Demucs

3.7.1. Overview

Similarly to Open-Unmix, Demucs is one of the state-of-the-art model for music source separation. However, Demucs is a waveform-to-waveform model, unlike Open-Unmix which makes use of the Fourier transform. The Demucs model is

primarily an encoder-decoder model. The encoder and decoder are convolutional networks which are connected by a BiLSTM as well as U-net skip connections. [5]

3.7.2. *Model Architecture*

Demucs is comprised of a 6 layer encoder where the i^{th} layer has a kernel of size 8, stride of 4, C_{i-1} input channels and C_i output channels and a ReLU activation function. This configuration is then followed by a convolutional layer of kernel size 1, $2C_i$ output channels and a GLU activation function. Finally, $C_0 = C = 2$, $C_1 = 100$ and all other $C_i = 2C_{i-1}$.

The encoder is followed by a bidirectional LSTM with 2 layers and a hidden size of $C_6 = 3200$. The output of this Bi-LSTM is of size $2C_6$ channel per time step which is sent through a Linear layer to reduce it back to C_L.

Similarly, the decoder is the inverse of the encoder. with the indices in reverse order of the encoder. Figure 7 provides more detail.

3.7.3. *Model Pipeline*

For the training procedure, the model takes as input the exact waveform of the audio file. The forward pass of the model passes the input through the layers as described above.
The model returns 4 waveforms that corresponds to the sources as specified (by default: vocals, drums, bass, other). For the loss calculation, the model uses either the average mean squared error or average absolute error between waveforms of the estimates by the model and the ground truths from training set. The optimizer used is Adam with learning rate from $[0.0003, 0.0005]$. The pre-trained model has been trained for 120 epochs. Refer to Table 5 for extensive list of hyper-parameters.

In contrast to the Open-Unmix model, the model is optimized in the time domain without any pre-processing or post-processing step. However, it's worth noting that the model by itself is not time-equivariant (meaning that shifting the input by X samples should shift the output Y by the same amount). While Open-Unmix is naturally time invariant (proven by experimentation), Demucs needs a workaround called *Randomized Equivariant Stabalization*. In this approach, the model samples S random shifts of the input mixtures and averages the predictions.

Besides transfer learning, we also trained Demucs model using our own set of hyper-parameters. We trained a model with 1 unidirectional-LSTM instead of 2 Bi-LSTMs, 3 layer deep encoders and decoders and a batch size of 4 instead of 64. Details of the training are mentioned in Section 4.3.

Table 5. Demucs hyperparameters (pre-trained)

Parameter	Value
Epoch	120
Batch Size	64
Learning Rate	$3 \cdot 10^{-4}$
Channels	64
Loss	MSELoss
No. of LSTM layers	2
Depth of encoder and decoder	6
kernal size	8
stride	4
Sample rate (Hz)	44100
no. of channels	2
no. of samples	441000
Data stride (samples)	44100

4. EVALUATION

A good source separating algorithm is one which is able to separate the source signals as distortion-free as possible from the mixed signal. Technically, there can be three types of "distortion" in the reconstructed/separated signal: *interference*, *artifacts*, and *noise*. The distortion due to mis-separation is called interference or crosstalk. It arises due to the presence of outside sources in a separated source. Artifacts originate from the reconstruction algorithm itself—for example, glitches due to the STFT phase estimation process. The third type of distortion mentioned in the literature is sensor noise.

Broadly, there are two ways to evaluate the results for this task. The first method is to manually listen and subjectively rate the quality of separation through human listeners. The separated tracks can be evaluated for naturalness and absence of contamination from other sources.

The second method is to do an objective evaluation by using one or more than one of the following energy ratios expressed in decibels (dB) as the evaluation metrics.

Signal to Interference Ratio (SIR):

$$SIR = 10 \log_{10} \frac{\| s_{target} \|^2}{\| e_{interf} \|^2} \tag{6}$$

Signal to Noise Ratio (SNR):

$$SNR = 10 \log_{10} \frac{\| s_{target} + e_{interf} \|^2}{\| e_{noise} \|^2} \tag{7}$$

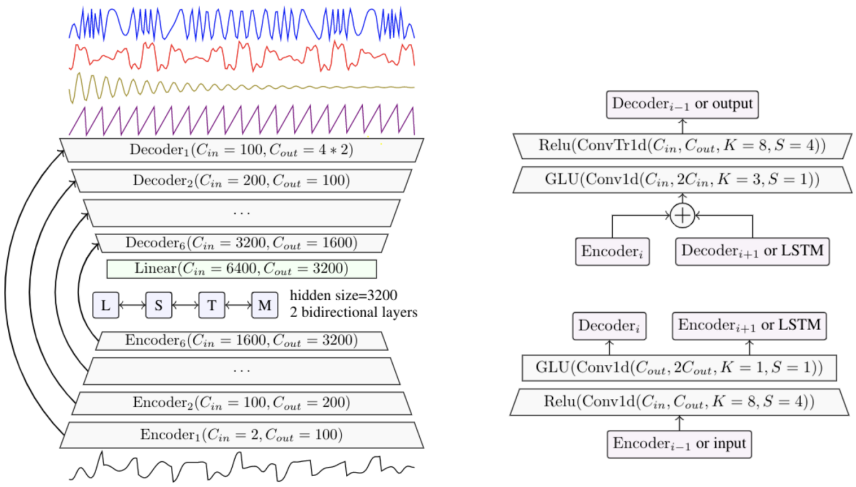

Fig. 7. Demucs model architecture (left) with a zoomed-in view (right).

Signal to Artifact Ratio (SAR):

$$SAR = 10\log_{10}\frac{\| s_{target} + e_{interf} + e_{noise} \|^2}{\| e_{artif} \|^2} \quad (8)$$

Signal to Distortion Ratio (SDR):

$$SDR = 10\log_{10}\frac{\| s_{target} \|^2}{\| e_{interf} + e_{noise} + e_{artif} \|^2} \quad (9)$$

where $e_{interf}, e_{noise}, e_{artif}$ are respectively the interference, noise and artifacts error terms and s_{target} is the true source signal.

BSS Eval is a MATLAB toolbox for performance measurement with respect to the above metrics for the source separation task developed by Vincent et al [12]. The biggest advantage of BSS Eval is that it can be used to evaluate outputs of all types of separating algorithms and is used as the standard metric for this task. `museval` [13] is the Python implementation of BSS Eval designed specifically for the musdb18 dataset.

In Table 6, the results for all the implementations done as a part of this project are consolidated for a qualitative and quantitative comparison. As expected, the pre-trained Open-Unmix and Demucs models performed the best given their sophisticated architectures and large training times. Detailed evaluations of each model are given in the following sections.

4.1. NMF results

The inductive bias of the algorithm in taking the number of source components to be separated as an input parameter allows us to separate the mixture track into as many sources as required. This makes the model independent of the musical instrument or type of source. The model identifies the sources of the mixture track by itself. This validates the first hypothesis, mentioned in section 3.4, according to which the algorithm can be generalized to all kinds of sources. Next, in NMF the mixture track is the linear combination of the separated sources. Therefore, we can combine one or more sources to get a combination of two or more sources. This property of NMF confirms the second and third hypothesis. If we combine all the separated sources, we will be able to reconstruct the matrix V or on combining two or more separations model will be able to separate sources in combinations of two or more targets.

In the NMF source separation the best scores on the energy ratios were recorded for target “other”. The high SIR score for “other” is almost like an outlier as compared to the other scores. On the other hand, the scores for rest of the targets are quite low. The SDR score is negative for all three while SIR has negative values for vocals and bass. These results indicate that this implementation of NMF technique is not able to separate the sources from the mixture in the desirable way. One possible explanation of this can be that the NMF toolbox takes mono signals as input and the output is a mono signal again. Therefore, to calculate the metrics we had to convert the target/gold label tracks to mono format and then calculate the metrics. Converting the tracks from stereo to mono leads to a large information loss and therefore not very great results.

Table 6. Comparison of evaluation metrics across algorithms. Features were averaged across all four generated audio sources.

Model	SDR	SIR	SAR
NMF	1.99	1.55	4.00
Open-Unmix (pre-trained)	4.991	8.186	**6.482**
Open-Unmix (real-time)	0.77	5.048	2.089
Open-Unmix Extension (our model) (real-time)	1.018	4.672	2.782
Demucs	**5.6**	**10.39**	6.08

Table 7. Evaluation scores for source separation of track "Al James - Schoolboy Fascination" in the musdb18 test set using NMF

Target Source	SDR	SIR	SAR
vocals	−8.019	−5.498	2.118
drums	−1.745	0.724	4.545
bass	−10.929	−9.012	3.069
others	2.724	12.885	3.382
Overall	−1.992	1.554	4.006

Table 8. Evaluation scores for source separation of track "Al James - Schoolboy Fascination" in the musdb18 test set using Open-Unmix pre-trained model

Target Source	SDR	SIR	SAR
vocals	7.127	12.742	9.335
drums	4.022	3.751	3.363
bass	6.187	12.988	8.331
others	2.629	3.264	4.901
Overall	4.991	8.186	6.482

4.2. Open-Unmix results

In its most basic form, the open-unmix model trains a dedicated model for each target. Therefore, the model is independent of the target source. A different model is trained for each musical instrument or stem without knowing the type of source. This inductive bias of the model confirms the first hypothesis and the mdel cann be trained for any musical instrument or source depending upon the target. Any two or more separations produced by the model can be combined to get a combination of sources. Further, if we train a model on the combination of sources we will be able to separate two stems together. This is demonstrated in separating the target "other" which is a combination of all the musical instruments. Thereby, validating the hypothesis that the model can perform separations in groups of two or more. Lastly, the additive inductive bias of the spectrogram makes us confirm the last hypothesis that if the spectrograms of the separated sources are added together we get the original track back.

We conducted three experiments using Open-Unmix. First, we separated the test samples using the pre-trained model. The hyper-parameters and the details of the pre-trained model are mentioned in Section 3.6. The loss curve for the pre-trained model which has been trained for 1000 epochs are shown in Figure 8. The performance of the model is very close to state-of-the-art. All the separations are quite clean and distortion-free to the human ear.

Second, we trained the Open-Unmix model using our set of hyper-parameter choices. The hyperparameters are specified in Table 9.

Table 9. Open-Unmix hyperparameters (real-time trained and Extended)

Parameter	Value
Epoch	50
Batch Size	8
Learning Rate	1e-3
Weight Decay	0.00001
STFT fft size	2048
STFT window size	2048
STFT hop size	1024
Hidden size of dense bottleneck layers	256 - 512
Maximum model bandwidth (Hz)	16000
No. of channels	1
Optimizer	RMSProp

As per human evaluation, the quality of the model trained in real-time is not as good as that of the pre-trained model. The separated tracks are not clean and have noise in form of signal from other target sources. This likely occurred since we halved the window size, batch size and other parameters given our constraints on time and computational resources. For the full Open-Unmix model it takes approximately 20 minutes to train for 1 epoch on Google Colab GPU. This forced us to reduce the number of epochs for this experiment from 1000 to 50. The intention of conducting this experiment was to implement the model and observe the effects of adjusting the bells and whistles of the model on our own. However, it shall be noted that the size and architecture of the model was unchanged.

The training loss curves corresponding to the model for each target along with the cumulative loss curve are shown in Fig-

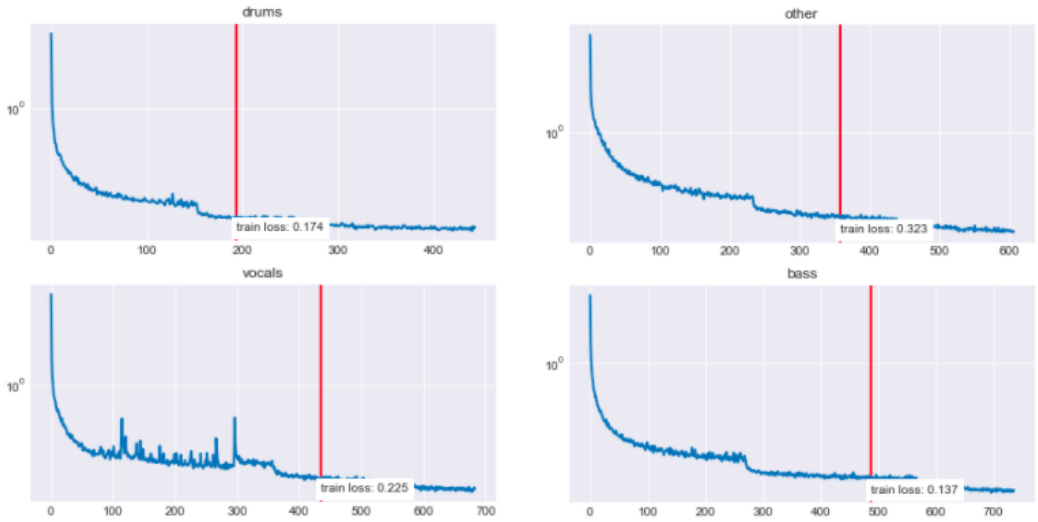

Fig. 8. Open-Unmix pre-trained model training loss curves

ure 9. All the curves have similar trends and almost overlap each other. The black curve which represents the mean of all the four curves is quite close to all the four lines. Taking a closer look at the graph, we observe that the loss is highest for vocals. By the end of 50 epochs, the training losses for drums, bass, and accompaniment converge while the loss for vocals is slightly higher. This indicates that the model has has some difficulty in separating the source signals which have frequencies similar to those of vocals.

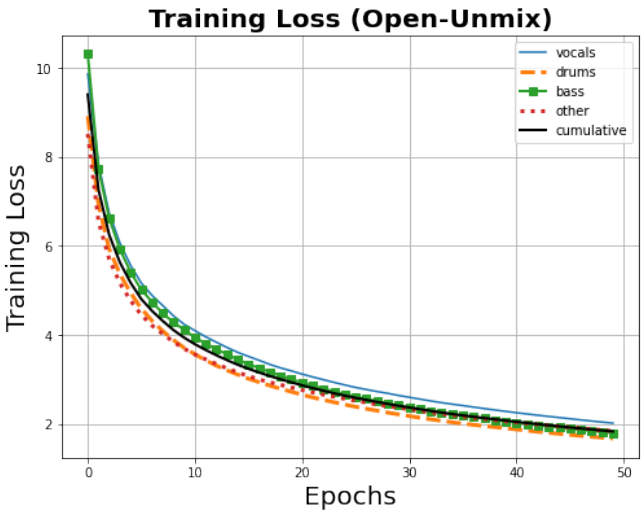

Fig. 9. Open-Unmix training loss curves

The scores mentioned in Table 10 indicate that the "vocals" separation has the highest scores across different ratios as compared to other separations. On the other hand, the "other" separation has the lowest scores apart from SIR,

Table 10. Evaluation scores for source separation of track "Al James - Schoolboy Fascination" in the musdb18 test set using Open-Unmix model trained in real-time

Target Source	SDR	SIR	SAR
vocals	4.170	6.341	5.835
drums	−1.122	2.601	0.358
bass	2.164	7.517	3.385
others	−2.103	3.733	−1.228
Overall	0.77	5.048	2.089

Table 11. Evaluation scores for source separation of track "Al James - Schoolboy Fascination" using our AlexNet-inspired modification of the Open-Unmix model, trained in real-time

Target Source	SDR	SIR	SAR
vocals	4.254	6.929	5.930
drums	−1.557	1.863	0.530
bass	2.490	7.136	4.069
others	−1.112	2.760	0.598
Overall	1.019	4.672	2.782

where "drums" separation performs the worst. Clearly, it is easier to make out vocals from the instrument since the frequency range of human voice is quite different from those of musical instruments. Therefore, the separation of vocals is relatively has less distortion, interference, and artifacts. The next best performance across scores is of the "bass" separation which also has the highest SIR score.

Finally, we trained the Extended Open Unmix model that trains all 4 target models at once (see 3.6). The training loss curve is show in Figure 10. We were able to beat the normal Open Unmix model trained in real time under similar experimental settings. This suggests that our model could outperform the pre-trained if trained for 1000 epoch. We trained the extended unmix model for 50 epochs. Metrics of the model are show in Table 11. Comparing our design's with the original Unmix model's performance, realtime-trained models' performances show that our model has an increase of 0.25 points for SDR and a increase of 0.69 points SAR. Thus, our approach was able to successfully remove some artifacts that the original models' estimates contains. However, Signal to inference ratio for our model is less by 0.376. This could be explained by the fact that there was intentional cross talk between target models to allow for communication.

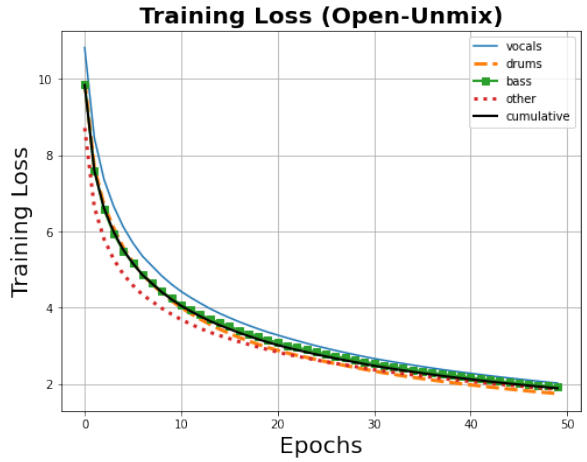

Fig. 10. Open-Unmix Extended training loss curves

As previously mentioned, this implementation was done only for experimental purposes to understand the Open-Unmix model architecture and implementation. The results do not reflect the true capability of the model. The pre-trained model results are the correct representation of the model capability.

It is also observed that SDR values are lowest among all the ratios because of three error terms present in the denominator while SIR and SAR depend only on the error due to interference and artifacts, respectively. This is evident from the overall SDR, SIR, and SAR values mentioned in the last row of the tables.

4.3. Demucs results

At its heart, Demucs is an encoder-decoder model. Unlike Open-Unmix, Demucs trains all target sources at once. But similar to Open-Unmix, Demucs is also general enough to train any number of target sources. This inductive bias of the model confirms the first hypothesis that the model can be trained for any musical instrument or source. However, Demucs is trained in the context of other target sources, which causes the model to produce some artifacts that are generally not very prevalent in Open-Unmix. Nonetheless, it is able to successfully separate multiple sources simultaneously, confirming our second hypothesis. Finally, we see that the original track can be retrieved with negligible loss by simply adding up the separated sources, thus confirming our third hypothesis.

We conducted two experiments with the Demucs model. Firstly, we separated test samples using the pre-trained model. The details for hyper-parameters are given in Section 3.7. The evaluation metrics for the performance of pre-trained Demucs is shown in Table 12 and the comparison against different models is shown in Table 6. The performance is to state-of-the-art and all the separations are clean.

Table 12. Evaluation scores for source separation using Demucs pre-trained model.

Target Source	SDR	SIR	SAR
vocals	6.29	13.31	6.54
drums	6.08	11.81	6.18
bass	5.83	10.55	6.41
others	4.12	5.90	5.18
Overall	5.6	10.39	6.08

Secondly, we trained Demucs for the set of hyper-parameters specified in Table 13. The loss curve obtained after training is shown in Figure 11.

We trained a light version of Demucs with almost half the depth of the original model and smaller batches. The experimentation was primarily exploratory. Clearly, the results do not reflect the strength of the model. The pre-trained models are more representative of the strength and the weakness of the model. It is clear however that the training of Demucs is expensive for lighter versions of the model.

5. DISCUSSION

As we have seen, music source separation is a complex problem and any model designed for this task must be able to in-

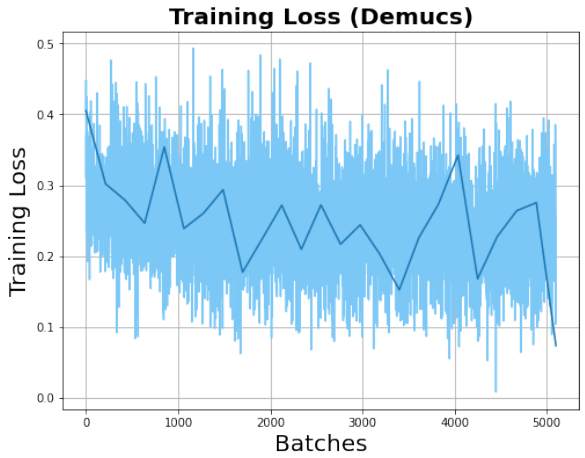

Fig. 11. Demucs training loss curve

Table 13. Demucs hyperparameters (real-time trained)

Parameter	Value
Epoch	2
Batch Size	4
Learning Rate	3e-4
Channels	64
Loss	MSELoss
No. of LSTM layers	1
Depth of encoder and decoder	3
kernal size	8
stride	4
Sample rate	44100
no. of audio channels	2
no. of samples	20000
Data stride	44100

terpret the spectral and temporal nuances present in the mixed track. Our simplest baseline model using NMF was generalizable to any combination of source instruments; however, it was unable to cleanly isolate sources as evidenced by the low SDR, SIR, and SAR (7). This motivated the usage of more sophisticated models. From our main results table (6), we see that our two deep-learning baselines, Open-Unmix and Demucs, vastly outperformed the NMF.

In addition, we also tried an extension of Open-Unmix where we trained all 4 target models simultaneously, allowing skip connections within the models similarly to AlexNet. We were able to beat the original real-time trained Open-Unmix model on all metrics except for SIR which deals with interference. This makes sense because we are allowing for cross talk between models. So for a small loss in interference, we gain substantial points for the other two metrics (SDR, SAR). We predict that our extended model would outperform the pre-trained Open-Unmix model given enough training epochs with the original hyper-parameters.

Finally, one of the other approaches we explored was a Demucs extension. We were looking to train 4 separate decoders—one per target source. More specifically, we kept the encoder and the LSTM layers as-is but had 4 decode pipelines instead of 1 (with reduced depth). The U-Net skip connection went from encoder at i to 4 decoders at i. However, this approach turned out to be very expensive to train and we could get results because of compute constraints.

6. CONCLUSION

In this project, we have showcased the audio source separation task on the musdb18 dataset using various techniques. We began by exploring the dataset and stated our hypothesis. Next, we implemented a non-deep learning baseline in NMF. The results from NMF showed a lot of opportunities for improvement. This led us to implement a deep learning approach. At first, we trained a neural network architecture called Open-Unmix which uses LSTM and later we trained the Demucs model originally proposed by Facebook AI Research. All the results from each of the model were calculated using the BSS Eval metric framework and comprised of overall SDR and other energy ratios.

Future Work

In the future, research can be focused on building a model which can be generalized in ways such that it can be fine tuned on any dataset irrespective of the file format, genre of music, etc. Next, better data pre-processing and post-processing techniques can be devised since a lot of information is lost in converting between waveforms and spectrograms. Finally, there is the potential to prepare larger, more comprehensive datasets than musdb18 or to extend the musdb18 dataset with tracks from other datasets. This will enable the design of deeper and wider architectures which can potentially achieve source separation performance that exceeds the current state-of-the-art.

Acknowledgement: We would like to thank our TA mentor Dewang Sultania for his mentorship and constant feedback throughout the duration of this project. Apart from him, we express our gratitude to the entire staff of CIS 522 for giving us this platform and enabling us to pursue the field of deep learning in our respective future careers.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, Eds., Cham, 2015, pp. 234–241, Springer International Publishing.

[2] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[3] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 21–25.

[4] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 106–110.

[5] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach, "Music source separation in the waveform domain," *arXiv preprint arXiv:1911.13254*, 2019.

[6] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[7] Paris Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2004, pp. 494–499.

[8] Jeongsoo Park, Jaeyoung Shin, and Kyogu Lee, "Separation of instrument sounds using non-negative matrix factorization with spectral envelope constraints," *CoRR*, vol. abs/1801.04081, 2018.

[9] Patricio López-Serrano, Christian Dittmar, Yiğitcan Özer, and Meinard Müller, "Nmf toolbox: Music processing applications of nonnegative matrix factorization," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Birmingham, UK, September 2019.

[10] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, vol. 4, no. 41, pp. 1667, 2019.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] Cédric Févotte Emmanuel Vincent, Rémi Gribonval, "Performance measurement in blind audio source separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[13] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito, "The 2018 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Surrey, UK*, 2018, pp. 293–305.