Extracting Vocals from Songs using U-Net

Skyler Kimura
University of Hawaii
UHM, Honolulu
skyler8@hawaii.edu

Abstract

Vocal and instrumental separation from musical tracks is a complex yet essential task in music production, remixing, and audio engineering. This study explores using a U-Net convolutional neural network (CNN) for isolating vocals from mixed audio signals. Audio waveforms are transformed into power spectrograms via short-time Fourier transform (STFT) and used as input to the U-Net model, which predicts the vocal components. The resulting spectrograms are reconstructed into audio using the inverse STFT (ISTFT) and the Griffin-Lim algorithm for phase estimation. The MUSDB18 dataset provides the training data, and several preprocessing and augmentation techniques, including normalized spectrogram scaling and stem mixing, enhance model performance. Evaluation metrics reveal mixed results, highlighting challenges like static noise and phase reconstruction limitations. This work demonstrates the potential of machine learning in audio source separation while identifying areas for future improvements, such as incorporating phase-aware methods and advanced post-processing techniques.

1. Introduction

Separating vocals and instrumentals from a musical track is a challenging problem with significant practical applications in music production, remixing, and audio engineering. As a music producer and computer science major currently enrolled in a computer vision class, I am motivated to develop an offline solution for this task. Existing online services and proprietary software for source separation often involve privacy concerns, recurring costs, or limitations on customization. While open-source solutions are available, I want to build a model and train it for learning purposes.

The task of recognizing vocals from accompaniment, while intuitive for human perception, poses significant challenges for computational methods. The complexity arises from the overlapping spectral characteristics of different audio sources in a mixed signal. To address this, I will convert audio waveforms into power spectrograms—a representation of frequency magnitude over time—using short-time Fourier transform (STFT) 1 as input for a deep learning-based solution. Specifically, I intend to utilize a U-Net convolution neural network (CNN) architecture for image segmentation on the spectrograms to mask the target source. Then, I will convert the spectrogram back into an audio waveform using inverse short-time Fourier transform (iSTFT) 3 with the Griffin-Lim algorithm. This approach involves identifying and isolating the frequency components within the spectrogram associated with vocals, resulting in one isolated vocal spectrogram.

If time permits, the tool will be integrated into a user-friendly web application to enable public access and real-world usability.

By leveraging Pytorch for model development and TorchAudio for audio preprocessing, this project aims to demonstrate the effectiveness of machine learning techniques, particularly U-Net, in addressing a complex audio processing problem. The successful implementation of this tool will not only highlight the interdisciplinary applications of computer vision but in addition, provide a valuable learning experience with audio machine learning.

2. Related Work

Music source separation is a popular audio task that many researchers and companies have solutions for. There are many state-of-the-art machine learning U-Net-based architectures with their respective research paper. Most related work modified the base U-Net architecture which will improve results.

2.1. Open-Unmix - A Reference Implementation for Music Source Separation

Open-Unmix is a popular open-source deep learning neural network reference implementation for music source separation that reached state-of-the-art performance in 2019 [8]. The Open-Unmix architecture uses three bidirectional long short-term memory (BiLSTM) allowing the model to leverage the full temporal context of the input spectrogram at each time step [8]. Each layer of the BiLSTM extracts higher temporal features to improve the recognition of the target source to separate. The model is trained to predict the magnitude spectrogram of a target source from the magnitude spectrogram of the mixed audio source. This work takes a different approach to my approach because I am training a U-Net model to recognize the vocals within the mixed audio source.

2.2. Wave-U-Net: A Multi-scale Neural Network for End-To-End Audio Source Separation

Wave-U-Net is a one-dimensional time-domain adaptation of the U-Net model that does music source separation on audio signals in the time domain which is represented by the raw waveform of the amplitude [7] unlike the frequency-domain represented by the spectrogram. The model uses raw waveforms to include the audio phase information, so the phase does not need to be reconstructed, which magnitude spectrograms lack. Keeping the phase intact will lead to more accurate source separation since the phase information is not lost during the preprocessing stage. My method uses the U-Net to predict the target source, vocals, in the frequency domain on a spectrogram and convert it back into the time domain with iSTFT. The phase of the audio source will be reconstructed using the Griffin-Lim algorithm.

2.3. Demucs - Hybrid Transformers for Music Source Separation

Demucs is a popular open-source state-of-the-art music source separation model. The model architecture is a U-Net architecture that is inspired by Wave-U-Net [1] The model does source separation in both the time and frequency domain, so it incorporates raw waveform and spectrograms when predicting the target source. The architecture has two U-Nets, one in the time domain and one in the frequency domain. The model uses a cross-domain transformer encoder in the innermost layers that does self-attention within the same domain and cross-attention across domains [6]. After the decoder layers in both U-Nets, the frequency domain is transformed to waveform using iSTFT and summed with the time domain decoder output which results in the actual prediction of the target source. Their method differs from mine because I used a U-Net to predict the frequency domain. Also, I am not using transformers across domains, but due to attention-based learning, the model understood how the target source fits within the mix over 10 seconds of context. However, the model requires more training data than the other because of the data-hungry transformer layer.

3. Dataset

3.1. MUSDB Dataset

MUSDB dataset is the standard dataset for music sound separation which is used to train the previous models mentioned in this paper [3]. The dataset has 150 songs in .wav format along with the sound source stems which include mixture, drum, bass, vocals, accompaniment, and others. I will only need the mix (all stems combined) and vocal stems because I will use vocal stems as the ground truth and the mixture as the input to my model.

3.2. Data Preprocessing Naive: Data loss conversion

I thought it would be a good idea to convert the WAV files into a spectrogram and then into an image and then use the mixed spectrogram image as an input, and vocal spectrogram image as the ground truth mask. However, converting spectrograms into an image would cause detrimental data loss due to the float32 to uint8 conversion. Unfortunately, data loss in the spectrograms will result in inaccurate audio reconstruction which would not help solve the problem.

3.3. Data Preprocessing: Normalized 2D Spectrogram

$$X(t,f) = |\int_{-\infty}^{\infty} x(\tau)w(\tau - t)e^{-j2\pi f\tau}\, d\tau, |^2 \qquad (1)$$

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad (2)$$

I used STFT (1) to convert the audio into a spectrogram. Then I min-max normalized the data using (2) so it would stay within the range from 0 to 1. I need to normalize the data because the spectrogram values are not ideal for training a model because they can be large. Since I used the MUSDB Python parser [3], the audio data is prepared uniformly which is ideal for training. Therefore, I do not have to resample the audio data from a different sample rate, and I do not need to do any padding or cutting of the data because all of the data is the same length and sample rate. Since I am using a power spectrogram, I have the magnitudes of the frequency bins, so I do not need to do any further calculations.

4. Methods

4.1. Naive U-Net Approach

Given that U-Net models are commonly used for image segmentation [4], I initially considered applying image segmentation to the mixed spectrograms by using the vocal spectrogram as a mask. However, as discussed in Section 3, accurately reconstructing audio from a spectrogram image is not feasible due to data loss. Additionally, the model

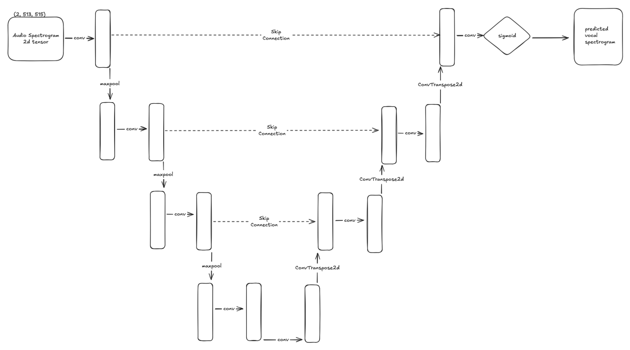


Figure 1. U-Net Architecture

would likely perform poorly in segmenting the mixed spectrogram into its vocal components, as overlapping frequencies in the spectrogram would create ambiguity. It is not ideal for the U-Net model to predict the vocal spectrogram from the pixel values, as the mixed spectrogram image lacks sufficient definition of the vocal components.

4.2. Better U-Net Approach

A more effective approach involved inputting the normalized mixed spectrogram data into the U-Net model, with the vocal spectrogram data serving as the ground truth. The model was trained to predict the vocal spectrogram values from the mixed spectrogram. This method is advantageous because it avoids data loss and retains more detailed information in the spectrogram values. Additionally, this approach leads to a more accurate reconstruction of the vocal component.

4.3. U-Net Architecture

I implemented a standard **U-Net Convolutional Neural Network (CNN)** architecture designed for image-to-image predictions. Instead of processing traditional images, I used 2D arrays representing spectrogram values as inputs. The network outputs another 2D array, corresponding to the predicted spectrogram values.

The architecture consists of the following components:

1. **Feature Layers:** I used feature layers with dimensions 32, 64, 128, and 512. These layers progressively extract higher-level features as the network encodes the input spectrogram.

2. **Encoding Steps: (Downsampling)**

   - Each encoding step applies two convolutional layers, both with a kernel size of 3 and a stride of 1.
   - The output of these convolutions is passed through a *LeakyReLU* activation function, which helps stabilize the network and prevent vanishing gradients.
   - A max-pooling layer with a size of 2 and a stride of 2 is then applied to reduce the spatial dimensions (halving the array size) while retaining the most important features.

3. **Bottleneck:** At the bottleneck, I used another two convolutional layers with similar kernel and stride settings as in the encoding steps. This step represents the deepest and most compressed representation of the input spectrogram.

4. **Decoding (Upsampling) Steps:**

   - I upsample the compressed array using transposed convolutions, which effectively increase the spatial dimensions of the array.
   - The decoder reconstructs the array step by step, reintroducing details and combining low- and high-level features through skip connections from the corresponding encoding layers.

5. **Output Layer:**

   - The final output layer applies a *sigmoid* activation function to ensure that the output values are normalized between 0 and 1, suitable for the target spectrogram format.

This architecture effectively leverages the U-Net's ability to capture local and global features [5] in spectrogram data, making it well-suited for tasks like vocal separation or audio source enhancement.

4.4. Reconstructing Audio From Power Spectrogram

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{k=0}^{N-1} X(m,k) W(t-mH) e^{j\frac{2\pi k}{N}(t-mH)} \quad (3)$$

To reconstruct the audio from the estimated spectrogram, it is necessary to denormalize the data to restore the original magnitude values. Following this, the ISTFT 3 is applied to convert the spectrogram into an audio signal. However, directly using ISTFT 3 on the estimated spectrogram data results in missing phase information, which is essential for accurate audio reconstruction. Without the phase information, the reconstructed audio may contain elements of the vocals but will lack fidelity and accuracy.

To address this, I employed the Griffin-Lim algorithm [2], which is effective for estimating the phase of an audio signal from its magnitude spectrogram. Since the model outputs only the magnitude spectrogram, the Griffin-Lim algorithm was used to reconstruct the phase, enabling a more accurate approximation of the vocal audio signal.

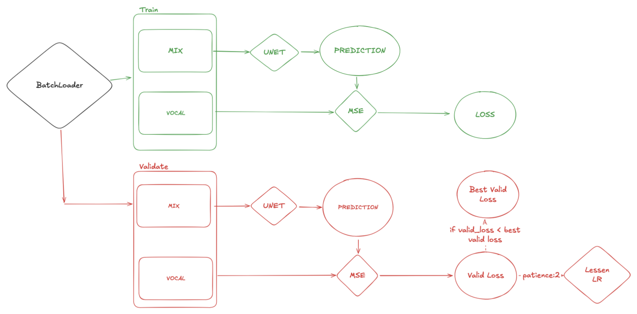


Figure 2. Training Pipeline

5. Data Augmentation

To enhance the diversity and size of the training dataset, several data augmentation techniques were applied.

5.1. Stem Source Mixing

The target vocal stem was saved, and a random accompaniment was generated by combining stems from other songs. The target vocal was then placed on top of the randomly selected accompaniment. This process effectively increases the training set by introducing varied background accompaniments while maintaining the same vocal stem.

5.2. Intensity (Volume) and Stereo Channel Flipping

To further augment the data, the intensity (volume) of the stem, excluding the vocals, was randomly adjusted within a specified range. Additionally, during training, there was a 50% chance of flipping the stereo channels when sampling the data. This introduced variability in the stereo field and volume levels, ensuring the model learns to generalize across different sound conditions.

These augmentation techniques help create a more diverse training dataset, improving the model's robustness and ability to generalize to unseen data.

6. Training

The training pipeline implements a dual-path architecture for processing audio spectrograms. The process begins with a BatchLoader that feeds data into two parallel streams:

1. The training step (denoted in green) processes:

   A normalized [0,1] random mix with vocal 2D spectrogram A normalized [0,1] vocal 2D spectrogram

   The mix spectrogram is inputted into the U-NET model, which generates a predicted vocal spectrogram (also normalized to [0,1]). This prediction, along with the original vocal spectrogram, feeds into a Mean Squared Error (MSE) loss function to compute the training loss.

2. The validation step (marked in red) has the same steps as the training step but operates independently on the validation dataset. It processes:

   A normalized [0,1] mix 2D spectrogram A normalized [0,1] vocal 2D spectrogram

   The validation process uses the same U-NET model to generate predictions, followed by MSE loss calculation to produce a validation loss metric. The system implements a patience-based learning rate reduction scheduler. If the validation loss falls below the previous best valid loss, it updates the best valid loss metric with the lower loss. If the best validation loss does not lower after 2 training epochs, the learning rate is divided by 10. I wanted to decrease the learning rate after a plateau in loss because I noticed the loss would hover for multiple epochs and never converge. Lowering the learning rate will make coarse adjustments to the model weights which should solve the hovering issue and help the values converge.

Average Vocal Separation Evaluation Scores	
Metric	Score
ISR	-1.741754
SAR	-10.584725
SDR	-8.669975
SIR	2.534616

Figure 3. Evaluation Score Table

Training This architecture ensures robust model training while maintaining careful validation checks and adaptive learning rate optimization, which are crucial for achieving optimal vocal separation performance.

7. Results

The evaluation of the vocal separation model was conducted using four key metrics: ISR, SAR, SDR, and SIR. The average scores for these metrics, presented in Table 3, provide insights into the model's performance and areas for improvement.

1. Image-to-Spatial Ratio (ISR): -1.741754

   ISR evaluates the spatial distortion introduced during the separation process [9]. The negative ISR score indicates that the spatial characteristics of the separated vocals were poorly preserved. This suggests that the model struggles to maintain the spatial fidelity of the vocal source, which is critical for high-quality audio separation. This is expected because I used a power spectrogram which does not hold the phase information of the audio.

2. Signal-to-Artifacts Ratio (SAR): -10.584725

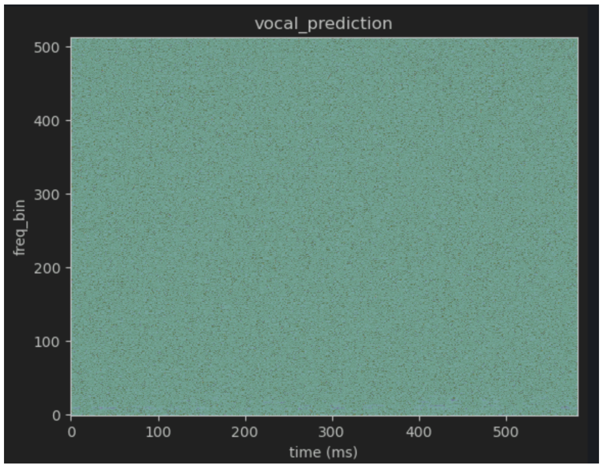
   SAR measures the artifacts introduced during the separation [9]. The highly negative SAR score suggests a significant presence of artifacts in the separated vocals. These artifacts likely degrade the perceptual quality of the isolated audio, highlighting the need for improved artifact suppression mechanisms in the model. After listening to the predicted vocal audio, there seems to be overbearing static noise which is most likely introduced in the multiple convolution process. There is a possibility that the spectrogram originally had an artifact, so the model increased the magnitude of the artifact with the sigmoid function. Possibly removing the sigmoid function at the end will improve the results.

3. Signal-to-Distortion Ratio (SDR): -8.669975

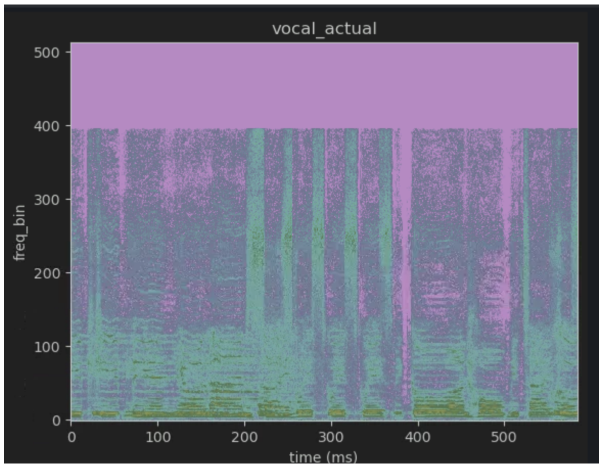
   SDR provides a comprehensive measure of separation quality by assessing distortions caused by noise, interference, and artifacts [9]. The negative SDR score reflects poor overall performance, indicating that the separated vocals suffer from a combination of distortions and contamination from background sources.

4. Signal-to-Interference Ratio (SIR): 2.534616

   SIR quantifies the model's ability to suppress interference from non-vocal sources. While the positive SIR score suggests some level of success in isolating vocals, the relatively low value indicates that interference from other sources remains a significant challenge.



(a) Vocal Prediction



(b) Vocal Actual

Figure 4. Comparison of Vocal Prediction and Actual Spectrograms

7.1. Compare Spectrogram

Figure 4 compares the predicted spectrogram (Figure 4a) with the actual spectrogram (Figure 4b). The predicted spectrogram exhibits a significant amount of noise, with no discernible structure resembling the patterns present in the actual spectrogram. This discrepancy highlights the challenges in accurately reconstructing vocal elements, as the predicted spectrogram fails to capture the harmonic and temporal features of the actual audio. The excessive noise in the prediction suggests that the model struggles to generalize effectively, potentially due to limitations in the training dataset, model architecture, or the absence of robust phase estimation techniques. Further refinement of the model or post-processing methods, such as denoising algorithms, may be necessary to improve the quality of the predictions.

7.2. Conclusion

The custom U-Net-based spectrogram model developed for this project demonstrated mixed performance in vocal separation tasks. While the model isolated vocals reasonably well for certain songs in the test dataset, it struggled with others, as reflected in the evaluation metrics. A consistent issue across all estimations was the significant static noise, which greatly impacted the Signal-to-Artifacts Ratio (SAR) scores. This suggests that the model introduces considerable artifacts during the separation process.

The results highlight limitations in this approach, particularly its inability to fully remove the accompaniment. This appears to stem from the overlap between vocal and accompaniment frequencies in the spectrogram. Modifying only the magnitude spectrogram seems insufficient for completely isolating the vocal component, as phase information and spatial cues are not considered. Additionally, static noise and residual accompaniment suggest that further post-processing is necessary to enhance the separation quality.

In future work, incorporating advanced filtering techniques, such as a Wiener filter, could help reduce artifacts and static noise, as seen in approaches like Open-Unmix [8]. To address limitations in spatial fidelity, transitioning from power spectrograms to complex spectrograms—which include both magnitude and phase information—may significantly improve the Image-to-Spatial Ratio (ISR). By leveraging phase information, the model could better account for the spatial characteristics of the audio, leading to more accurate separation results.

Overall, while this implementation represents a naive approach to source separation, it provides a solid foundation for future refinements and advancements in vocal isolation methods.

References

[1] Alexandre Défossez. Hybrid spectrogram and waveform source separation. In *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021. 2

[2] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. In *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 804–807, 1983. 3

[3] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, Dec. 2017. 2

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 2

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 3

[6] Simon Rouard, Francisco Massa, and Alexandre Défossez. Hybrid transformers for music source separation. In *ICASSP 23*, 2023. 2

[7] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation, 2018. 2

[8] F.-R. St
"oter, S. Uhlich, A. Liutkus, and Y. Mitsufuji. Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*, 2019. 1, 2, 6

[9] O. Yilmaz and S. Rickard. Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing*, 52(7):1830–1847, 2004. 4, 5