Singing Voice Separation and Enhancement

Jayant Manchanda

Introduction

The task of automatic singing voice separation involves estimating what the sung melody would sound like in isolation from a full music mixture. In this project, a two-stage process is employed: first, singing voice is separated from the music mixture, and second, enhancement of the separated audio is performed to remove artifacts and noise. In the first stage, a music file is input and a separated voice track is generated. In the second stage, the separated track is converted into a short-time Fourier transform (STFT) spectrogram, where enhancement operations are applied to improve the quality of the separated audio.

There is extensive literature in the domain of audio source separation, and several state-of-the-art models, such as Demucs, are capable of performing high-quality source separation with good phase preservation. However, for this assignment, a different approach was taken: a mask estimation strategy based on the method proposed by Jansson *et al.*, using a deep U-Net style convolutional neural network for voice separation. The separated spectrogram is then further enhanced by a U-Net style encoder–decoder model with a Temporal Block.

1 Voice Separation

The setup uses the MUSDB18 database for this task, which is a collection of 150 songs spanning a variety of genres and styles [1]. Each song in MUSDB18 is provided in the Native Instruments Stems format, a multitrack format composed of five stereo streams, each encoded in AAC at 256 kbps. For this setup, only the mix (layer 0) and the vocals (layer 1) are used.

All tracks were resampled to 8 kHz. Each stereo track was converted to a mono STFT magnitude spectrogram (using a window length of 1024 and hop size of 768), and then cut into overlapping patches of 128 time frames. These patches were loaded in batches of 16, with no additional data augmentation. Training was conducted for 100 epochs, and F1 loss was used between the masked and ground-truth vocal spectrograms. Optimization was performed using the Adam optimizer.

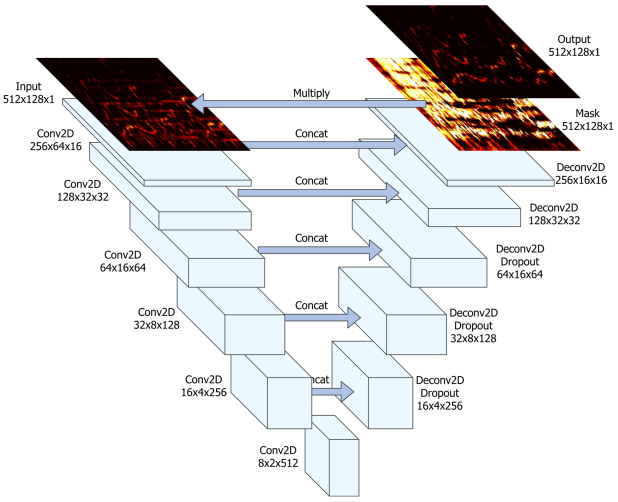

Figure 1: Unet Architecture

The original paper trained two separate models, one for vocals and one for accompaniment. In this setup, only the vocal separation model was trained to produce voice outputs. The implementation follows a four-stage encoder–decoder U-Net architecture for spectrogram-based voice separation, directly following the design from Jansson *et al.* In the encoder, each of the four successive DownBlock modules applies a 5×5 convolution with stride 2 and padding 2, followed by batch normalization and a LeakyReLU activation (slope 0.2). This reduces the time and frequency resolution by half at each step while expanding the channel depth from 1 up to 256. The decoder mirrors this structure with four corresponding UpBlock modules, each using a 5×5 transposed convolution (stride 2, padding 2, output padding 1), followed by batch normalization. To regularize the network, a ReLU activation and a 50% dropout are applied on the first three decoding layers. Skip-connections concatenate the upsampled feature maps with the corresponding encoder outputs.

The final transposed convolution layer reduces the number of channels back to one, and a sigmoid activation produces a soft mask over the STFT magnitude spectrogram. This soft mask, applied element-wise to the input spectrogram, yields an isolated vocal component while preserving both high-level context and low-level fidelity.

2 Voice (Spectrogram) Enhancement

To prepare the dataset for this task, each track in the MUSDB18 dataset was processed using a pre-trained UNetVocalSeparator. The original stereo stems were summed to mono and resampled, preserving the original phase, following

the approach described in Jansson *et al.* Both the network inputs (magnitude spectrograms paired with original phase) and the ground-truth vocal targets (magnitude spectrograms) were saved for downstream enhancement model training.

his task initially began with the use of a Variational Autoencoder (VAE). However, the VAE struggled to produce even satisfactory results and proved cumbersome to handle in the context of spectrogram enhancement. Since VAE's generate outputs from a sampled latent space, their generalization capability for this task appeared limited, often leading to blurry results.

I also experimented with expanding the temporal receptive field, inspired by the use of state-space models in *CleanMel* [3]. Instead of implementing a full state-space module, I used a simpler Temporal Context Block using dilated convolutions to similarly capture long-range temporal and frequency patterns. Additionally, I attempted adversarial training based on *pix2pix* [4], incorporating a generator and discriminator structure, although these extentions did not work out exactly and had to removed.

Consequently, the approach shifted toward a simpler variation of an encoder-decoder model, which allowed for more direct reconstruction of spectrogram features and provided a clearer and more controllable framework for enhancement.

The Spectrogram Enhancer is a U-Net style encoder–decoder model with a Temporal Block as its bottleneck. The encoder consists of three sequential convolutional layers. The first encoder layer applies a 4×4 convolution with stride 2 and padding 1 to transform the 1-channel input spectrogram into 32 channels, followed by instance normalization and ReLU activation. The second encoder layer further downsamples the output by applying another 4×4 convolution to increase the channels from 32 to 64, again followed by normalization and ReLU. The third encoder layer applies a similar 4×4 convolution to produce 128 channels, further compressing the feature map to one-eighth of its original resolution. At the bottleneck, the model uses a Temporal Context Block, consisting of three 2D convolutional layers with increasing dilation rates of 2, 4, and 8 respectively, each followed by ReLU activation to capture long-range temporal and frequency patterns without downsampling further. A residual connection adds the block's input back to its output.

The decoder mirrors the encoder's structure. The first decoder layer upsamples from 128 to 64 channels, followed by instance normalization and ReLU activation, and concatenates its output with the corresponding encoder feature map. The second decoder layer upsamples from 128 to 32 channels, again applying normalization and ReLU. Finally, the last decoder layer upsamples from 64 to 1 channel. A ReLU activation is applied at the output to ensure non-negative spectrogram values. Skip connections between the encoder and decoder are maintained.

Each input–target pair is first scaled and cropped to 512 frequency bins,

undergoes logarithmic dynamic–range compression, and is zero–padded to a fixed width of 4096 frames. During training, both predictions and targets are center–cropped to their minimum overlapping height and width (to ensure exact spatial alignment) before any loss is computed. The SpectrogramEnhancer model, initialized with Kaiming normal weights on all Conv2d and ConvTranspose2d layers, is optimized on a weighted sum of an absolute reconstruction loss and a perceptual feature loss. The reconstruction term is simply the L_1 distance between the center–cropped spectrograms, scaled by a factor of 600 to prioritize accurate waveform recovery. The perceptual feature loss is computed by passing both cropped predictions and targets through a frozen FeatureExtractor network (three Conv2d–InstanceNorm2d–ReLU blocks that downsample by a factor of four), and then taking the L_1 distance between their feature maps. Training uses the Adam optimizer with a ReduceLROnPlateau scheduler over 300 epochs, batch size 8, on an 80%/20% train/validation split.

The model outputs an STFT spectrogram, which can be converted back to the time domain using either the original phase, the Griffin-Lim algorithm for phase estimation, or by converting to a mel-spectrogram and passing it to a vocoder such as HiFi-GAN.

3 Combined Architecture

Both models can be joined together in a sequence-to-sequence pipeline. This takes a full music mixture as input, converts it to a mono STFT magnitude spectrogram, and passes it through a U-Net-based voice separation model to generate a separated vocal spectrogram while preserving the original phase. This separated spectrogram becomes the input to the second stage, where a spectrogram enhancement model refines it to reduce artifacts and recover lost spectral details. The enhanced spectrogram, along with the original phase or an estimated phase, is then used to reconstruct the final enhanced time-domain vocal waveform as the output. You see a demonstration of this pipeline during inference for the Enhancer model. It imports a file that takes in a file or a test track from MUSDB and runs them in a sequence. The results show similar outputs. At this moment its using an in-efficient code pipeline which can be further improved upon. As mentioned before the output of this can involve a Neural Vocoder or Griffin lim for perhaps better audio results.

Table 1: Voice Separation Results

Metric	Value (dB)
SDR	1.83
SAR	1.83
NSDR	8.48
SIR	Infinite

4 Results

4.1 Voice Separation

For evaluation, the `mir_eval` library was used with metrics including Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), Signal-to-Artifacts Ratio (SAR), and Normalized SDR (NSDR). Evaluation on the MUSDB18 test set showed that the U-Net model achieved a mean SDR of approximately 1.83 dB, SAR of 1.83 dB, and NSDR of about 8.48 dB, with SIR values reported as infinite. These results suggest modest success in vocal separation but with noticeable artifacts. The NSDR of +8.48 dB indicates meaningful improvement over the original mixture. Compared to Jansson *et al.*, who reported significantly higher performance on larger datasets, our setup with a smaller corpus and less augmentation likely limited final results.

4.1.1 Case Study

Using "Rivers and Leafs" by Tiptoe from Jamendo, subjective evaluation revealed that the model effectively suppressed non-singing regions but allowed some bleed during vocal passages. The model performed well in low and mid frequencies, while higher frequencies appeared sparser and dimmer. Temporal structure aligned with singing activity as shown in spectrogram visualizations. The overall subjective results are not completely off either, the singing parts can be heard clearly with minimal bleed and the silent parts stay silent

4.2 Voice (Spectrogram) Enhancement

The model was evaluated on the MUSDB18 test set using Mean Squared Error (MSE), Mean Absolute Error (MAE), Spectral Convergence, and Log-Spectral Distance (LSD) after 400 epochs of training. A reconstruction loss weight (λ_{recon}) of 300 and a GAN loss weight (λ_{gan}) of 5.0 were used. The evaluation produced an MSE of 28,905.39, an MAE of 23.44, a Spectral Convergence of 0.9788, and an LSD of 3.93.

Overall, the results indicated that the model did not significantly improve or enhance the spectrogram, exhibiting notable reconstruction errors, particularly at finer levels of spectral detail. The output results look very

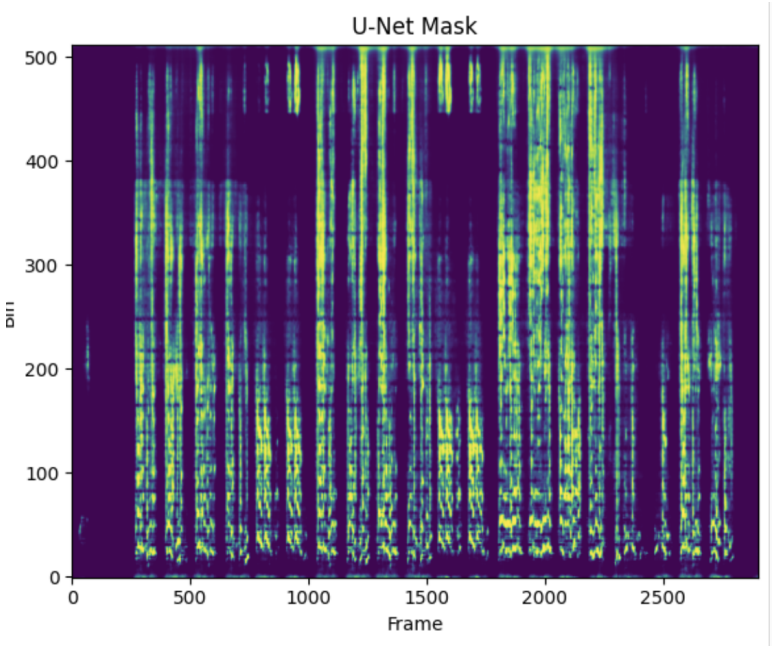

Figure 2: Mask prediction over "Rivers and Leafs"

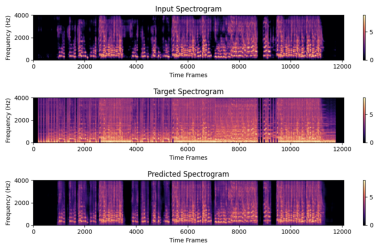

Figure 3: Spectrogram at 300 epochs (Test track from MUSDB)

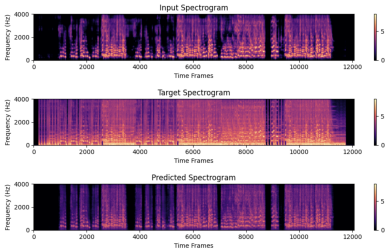

Figure 4: Spectrogram at 100 epochs (Test track from MUSDB)

close to the input (Fig 3) and not close to the ground truth. Although they do show reconstruction in higher bands

However, despite these shortcomings, the model showed improvements with the increase in the number of training epochs and with a higher reconstruction loss weight (increasing λ_{recon} from 300 to 600), resulting in clearer spectrogram outputs. It can be concluded that further optimization or employing deeper model architectures would likely be necessary to achieve high-quality spectrogram reconstruction.

Metric	100 Epochs	300 Epochs
Mean Squared Error (MSE)	34,871.95	28,905.39
Mean Absolute Error (MAE)	25.08	23.44
Spectral Convergence	1.1407	0.9788
Log-Spectral Distance (LSD)	4.14	3.93

Table 2: Comparison of the model performance at 100 and 300 epochs on the MUSDB18 test set.

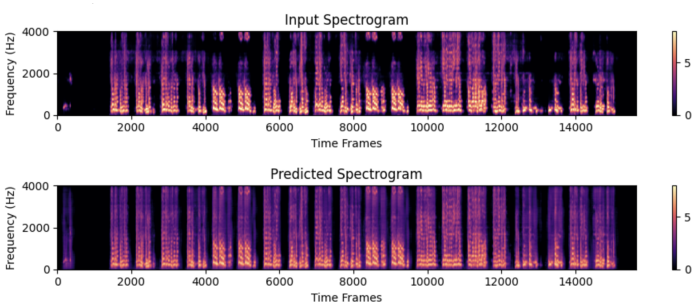

Figure 5: Enhancement over "Rivers and Leafs"

4.2.1 Case Study

Similar results were seen on the on the case study audio file. As seen in the generated output spectrograms, the predicted spectrogram remained similar to the input spectrogram, again they show reconstruction in higher bands and have very clear structure and onsets. Subjective evaluation further reveals that the audio did not really enhance and still had a lot of artifacts and it sounds still degraded, the voice can clearly be heard with precise onsets. Although this could also be an outcome of the right phase, and this output running through a vocoder model for phase re-consturction could significantly improve its quality

5 Conclusion

Overall, the results were not fully satisfactory, and both models would benefit from further improvements. The first model, responsible for source separation, achieved reasonable outputs but was likely limited by the relatively small dataset used for training. With a larger and more diverse corpus of data, the separation performance is expected to improve significantly, especially in capturing more complex temporal and spectral patterns.

The second model, for spectrogram enhancement, showed unsatisfactory results and promise can be further strengthened by architectural upgrades. A deeper network with stronger multi-scale connections, along with more

sophisticated feature extraction and loss strategies, could enable better enhancement of both local fine details and broader temporal structures

References

[1] Zafar Rafii et al. The MUSDB18 corpus for music separation. Dec. 2017 *LVA/ICA*, 2018.

[2] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, T. Weyde, "Singing Voice Separation with Deep U-Nets," *ISMIR*, 2017.

[3] Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X. and Zhang, C., 2021, April. Cleanml: A study for evaluating the impact of data cleaning on ml classification tasks. In 2021 IEEE 37th International Conference on Data Engineering (ICDE) (pp. 13-24). IEEE.

[4] Henry, J., Natalie, T. and Madsen, D., 2021. Pix2pix gan for image-to-image translation. Research Gate Publication, pp.1-5