Transparent and Accessible Audio Processing: Hybrid CNN-LSTM Deep Learning Techniques for Vocal Separation in Music

Vasilis Efraim Tsavalias (ics21083)

A thesis submitted for the degree of
Bachelor of Science in Applied Informatics, Computer Science and Technology

Department of Applied Informatics
University of Macedonia

Supervisor: Professor Eutixios Protopapadakis

September 2024

Abstract

Vocal separation from audio mixtures is a complex and critical task within audio signal processing, with significant applications in music remixing, track creation, and music information retrieval. This study presents a hybrid deep learning model combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, designed to effectively isolate vocals from intricate audio mixtures.

The research leverages the MUSDB18 dataset, a recognized benchmark for music source separation. Extensive data preprocessing was employed, including the conversion of audio files into spectrograms via Short-Time Fourier Transform (STFT) and data augmentation techniques such as pitch shifting and time stretching to enhance the model's generalization.

The CNN-LSTM model was trained using the Adam optimizer with Mean Squared Error (MSE) as the loss function. Hyperparameter optimization was conducted using Optuna, systematically tuning critical parameters to maximize model performance. Post-processing techniques, including mask refinement and dynamic range compression, were applied to ensure quality vocal separations.

To facilitate deployment in resource-constrained environments, dynamic quantization was applied using the Open Neural Network Exchange (ONNX) format. This approach reduced the model size and improved inference speed while maintaining accuracy.

This research advances music source separation by providing an easy-to-follow pipeline for vocal isolation. The methodologies and results presented offer valuable insights for real-time audio processing applications, paving the way for broader adoption in various industries.

Keywords: Keywords: vocal separation, Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), music source separation, deep learning, dynamic quantization, Optuna, spectograms.

Preface – Acknowledgments

I developed an interest in music during my teenage years. My computer science coursework enabled me to integrate my love of it with my academic pursuits. Consequently, I would like to express my gratitude to Mr. Alkis, my guitar mentor, for his guidance and motivation during the course of my thesis.

Mr. Alkis' expertise in both computer science and music had a big impact on my research. He received his degree in electrical engineering from Aristotle University. His expertise allowed me to comprehend critical algorithms utilized in my research, including fast Fourier transforms and concepts associated with sound wave analysis.

I am also grateful to my teacher, Protopapadakis Eftychios, for enabling me to pursue this challenge. This research has provided me with invaluable academic and practical experience that will have a substantial impact on my future.

Lastly, I would like to sincerely thank my mother for her consistent support, encouragement, and extensive understanding of the use of English in helping me write this thesis.

Contents

Introduction

1.1 Motivation

In recent years, there have been substantial developments in the field of digital audio processing, primarily in the area of music source separation. The task of removing vocals from music recordings is both highly sought-after and complex, which is why it stands out among the many challenges in this field. This process has significant implications for content creation, remixing, music production, and education [1].

Despite its importance, there is a noticeable gap in the existing literature, particularly in the transparency and accessibility of methodologies. During the research process, it was observed that many studies in this field rely on closed-source datasets and omit critical details regarding the procedural steps, such as the coding techniques, preprocessing, training loops, and post-processing methods, or even how the mathematical functions were coded in the first place.

This lack of transparency creates challenges for researchers attempting to apply these methodologies to alternative datasets or improve upon existing models. Moreover, the reliance on proprietary functions and closed-source datasets, often seen in other Kaggle/Colab notebooks and published papers, further complicates the replication and adaptation of these techniques. These practices obscure the inner workings of models, making it difficult for researchers to understand and apply these methods effectively.

These difficulties served as the driving force behind this research's completion. The goal was not merely to achieve superior model performance, but to create a comprehensive and accessible guide that leverages publicly available datasets and tools. By developing custom functions grounded in mathematical principles and by using free tools and modules provided by the extensive repository of Python, this research provides a foundation that others can build upon. Additionally, the development of custom metric calculations and post-processing functions, free from reliance on obscure libraries like *mir_eval*, was a deliberate choice to enhance the transparency and adaptability of the methodology.

This approach ensures that future researchers can focus more on analyzing model performance and advancing the field, rather than grappling with the intricacies of opaque processes. The entire Kaggle notebook has been made available, encouraging more skilled researchers or enthusiastic individuals to engage with and improve upon this challenging task.

1.2 Significance

Vocal and instrumental frequencies are intricately integrated together in a musical composition, which makes vocal removal a complicated process. Conventional techniques, including phase cancellation, frequency filtering, and mixing techniques from software programs like Ableton, frequently fail to achieve high-quality separation, particularly when dealing with contemporary, professionally produced music recordings [2]. These methods frequently lead to a reduction in audio quality, or an incomplete separation, underscoring the necessity of more advanced techniques.

The importance of creating a successful voice removal system goes beyond simple technological progress. Such a method could change the way music is being taught by breaking down compositions into their component parts for study [3]. Separating vocals from instrumentals can also benefit automated music analysis, recommendation systems, and lyric transcription algorithms in the field of music information retrieval. [4].

It calls for models that can comprehend and work with intricate audio structures, which could result in innovations that can be used in a variety of audio processing applications [5].

The need for the aforementioned audio manipulation tools is rising as the digital music ecosystem develops and streaming services and user-generated material take on bigger responsibilities. A strong voice-removal technology could open up new creative avenues for content producers, from hobbyists to industry pros. Additionally, it might facilitate the repair and remastering of old recordings, helping to preserve and repurpose musical history [3].

1.3 Problem Statement

In the field of audio signal processing, vocal removal from musical mixtures is a complex challenge due to the overlapping frequency ranges of vocals and instruments, as well as the varying acoustic characteristics across different genres. Traditional signal processing techniques often fail to adequately separate vocals when there is significant overlap in the spectral content of different sources. These limitations are compounded by the diversity of music production styles, which introduce variations that these methods cannot consistently address [2].

Moreover, while deep learning has emerged as a powerful tool for audio source separation, current models still face significant challenges. Chief among these is the ability to generalize across different musical genres and production styles without compromising separation quality [5]. The high computational demands of training such models further complicate their practical application, particularly when working with large, high-resolution audio datasets [4].

Furthermore, several persistent issues limit the progression of research and its practical applications. A critical problem identified is the prevalent reliance on closed-source datasets and proprietary functions, and the frequent omission of detailed procedural information in published studies. These practices hinder the reproducibility and adaptability of research, making it challenging for others to apply these findings to different datasets or improve existing methodologies, often leading to inefficiencies and a lack of direction in their work.

This research addresses these issues by providing a transparent and accessible guide that includes all aspects of the vocal separation process. The aim is to shift the focus of new researchers from understanding complex, opaque processes to improving model performance, thereby facilitating more meaningful advancements in the field.

1.3.1 Research Objectives

This research aims to create a machine learning model that can remove vocals from music tracks without affecting the instrumentals.

This work first designs and implements a deep learning architecture for speech removal from mp4 audio files. Using recent advances in music source separation, the study aims to create a model that can accurately extract vocal components in complicated musical compositions. This goal requires testing convolutional neural networks in conjunction with LSTM networks, to discover the best way to capture voice and instrumental sounds' temporal and spectral features.

Secondly, the study aims to improve vocal removal techniques by studying and using modern audio processing techniques. This objective involves utilizing spectral analysis methods like Short-time Fourier transforms(STFTs), along with psychoacoustic principles, to enhance the perceived quality of separated audio [6]. The aim is to separate vocal and instrumental components by merging these methods with the deep learning model.

Thirdly, this research will create and implement a comprehensive evaluation methodology for the proposed vocal elimination technology. To measure separation quality, measures like Source-to-Distortion Ratio(SDR), Source-to-Interference Ratio(SIR), and Source-to-Artifacts Ratio(SAR) should be chosen and applied [6]. This evaluation will include objective and subjective performance measures by conducting subjective listening tests to evaluate the separated audio's perceived quality.

Furthermore, the study will explore the quantization of the vocal separation model to optimize its performance for deployment in resource-constrained environments. The model will be converted to the Open Neural Network Exchange (ONNX) format, a widely used standard for deploying machine learning models across different platforms. This conversion allows for compatibility with various hardware and software environments, facilitating broader adoption.

Finally, To further enhance the model's efficiency, dynamic quantization techniques will be applied. These techniques reduce the precision of the model's weights from 32-bit floating-point to 8-bit integers, significantly decreasing the model size and improving inference speed. This process is particularly important for real-time applications, where computational resources are limited, such as in mobile devices or embedded systems.

1.4 Scope and Limitations

This research's scope is defined by its emphasis on developing and evaluating a neural network-based approach specifically designed for vocal removal from musical mixtures. The thesis encompasses several key areas:

- **Transparency and Accessibility**: A significant aspect of the scope involves addressing the challenges of transparency and reproducibility in audio source separation research. The study outlines all procedural steps in detail, including the creation of a custom dataset class, preprocessing routines, the training loop and post-processing techniques, ensuring that the entire process is accessible and understandable to researchers aiming to apply these methods in different contexts.

- **Model Architecture Design**: The primary scope involves the design of a novel neural network architecture that integrates bidirectional networks and attention mechanisms. The architecture is tailored to enhance the separation of vocal tracks from the accompanying instrumental components.

- **Dataset Utilization and Model Training**: The research includes the use of an open-source and diverse dataset, specifically the MUSDB18 dataset, comprising various music genres for training and evaluating the model. This ensures that the model is exposed to a wide range of vocal and instrumental arrangements, thereby enhancing its ability to generalize across different musical styles. The study emphasizes transparency in the methodologies, making use of publicly available datasets and tools to allow for the model's reproducibility and adaptability across different datasets.

- **Quantization and Deployment**: The research also explores the quantization of the model using dynamic quantization techniques to reduce the model size and improve inference speed, making it suitable for deployment in resource-constrained environments. The conversion to the Open Neural Network Exchange (ONNX) format is a key part of this scope, facilitating the model's deployment across various platforms while maintaining efficiency and accuracy.

- **Performance Evaluation**: The model's performance is assessed using established metrics such as Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Signal-to-Artifacts Ratio (SAR). These metrics are crucial in quantifying the effectiveness of the separation process and comparing it against existing methods.

However, the research is subject to the following limitations:

- **Dataset Coverage**: Although the dataset used is diverse, it may not encompass all possible variations in music production. This limitation could affect the model's ability to generalize to entirely new or less common musical styles.

- **Computational Constraints**: The deep learning models used in this research require substantial computational resources for training, which may limit the ability to experiment with larger models or more extensive datasets. This constraint could impact the scope of hyperparameter optimization and model refinement.

- **Focus on Vocal Removal**: The research is primarily focused on the task of vocal removal. While the proposed methods may be applicable to other types of source separation (e.g., isolating drums or bass), such applications are beyond the scope of this study. The focus on vocals means that findings may not directly transfer to other audio sources without additional modifications to the model.

- **Transparency and Reproducibility**: While this research strives to provide a comprehensive and transparent approach, there is an inherent limitation in the ability to fully replicate the results due to differences in computational resources, software versions, and hardware configurations. Additionally, the reliance on specific tools and libraries, although publicly available, may pose challenges for researchers unfamiliar with these technologies.

- **Generalization to Alternative Datasets**: Although efforts were made to use publicly available datasets and tools, the custom preprocessing steps and model architecture might not be directly transferable to other datasets without significant adjustments. This limitation is particularly relevant for those attempting to apply these techniques to datasets with different characteristics or in other domains of audio processing.

- **Impact of Quantization on Performance**: The use of dynamic quantization to optimize the model for deployment in resource-constrained environments may lead to a slight reduction in accuracy. While this trade-off was considered acceptable for the purposes of this research, it could limit the model's applicability in scenarios where maximum precision is required.

- **Limited Exploration of Alternative Models**: The research primarily focuses on the CNN-LSTM architecture for vocal separation. Although this model was chosen for its balance between performance and computational efficiency, other architectures or hybrid approaches were not extensively explored. This limitation may restrict the potential for discovering alternative models that could outperform the current approach in certain scenarios.

1.5 Thesis Structure

This thesis has five chapters that address significant vocal removal issues and propose solutions. The structure leads from core notions to ultimate results and implications logically.

Chapter 1- Introduction, discussed the research challenge and emphasized the lack of transparency found in similar researchers. It outlined the thesis framework and research goals. It also introduced the subject and highlighted its significance in audio signal processing.

A complete literature analysis of music source separation and vocal removal theories and methods is presented in Chapter 2. The chapter introduces basic audio signal processing techniques, including time-frequency analysis methods like the Short-Time Fourier Transform (STFT). It then discusses classic source separation methods' pros and cons. The review explores recent advances in deep learning-based vocal separation algorithms, focusing on CNNs and Wave-U-Net architectures. The chapter finishes by outlining field challenges and research gaps to justify this study's strategy.

Chapter 3, Methodology, provided a high level overview of the Experimental setup, that describes the vocal removal system development process. The dataset selection and preparation method, including data augmentation for model robustness, is described first. The chapter then describes the suggested deep learning architecture and how it tackles vocal removal concerns. This section covers implementation specifics such loss functions, optimization techniques, and training. The chapter also describes the evaluation framework, metrics, and procedures used to evaluate the model's performance.

In Chapter 4, the experimental data and model performance are thoroughly examined. Audio pre-processing steps, model architecture, training loop, model evaluation and The hardware and software specifications of the experimental setup are discussed. The chapter uses SDR and SIR to quantify the results.

Chapter 5 closes the thesis by reviewing the main findings and their implications for audio processing and machine learning. It reviews Chapter 1's research objectives and their progress. The chapter examines the potential uses of the voice removal system in music production, content development, and information retrieval. It also acknowledges the limits of the existing technique and suggests further study to improve and extend the model. The chapter concludes by discussing how this research has advanced music source separation techniques and influenced digital music technology.

Literature Review

2.1 Machine Learning: An Overview

Machine learning (ML) is a subfield of artificial intelligence that focuses on developing algorithms capable of learning from data and making decisions with minimal human intervention. This ability to discern patterns and insights from complex datasets is particularly valuable in domains like audio processing, where traditional programming techniques often fall short.

Figure A.15 illustrates the primary types of machine learning: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Each type addresses different problem-solving approaches based on the nature of the data and the task at hand.

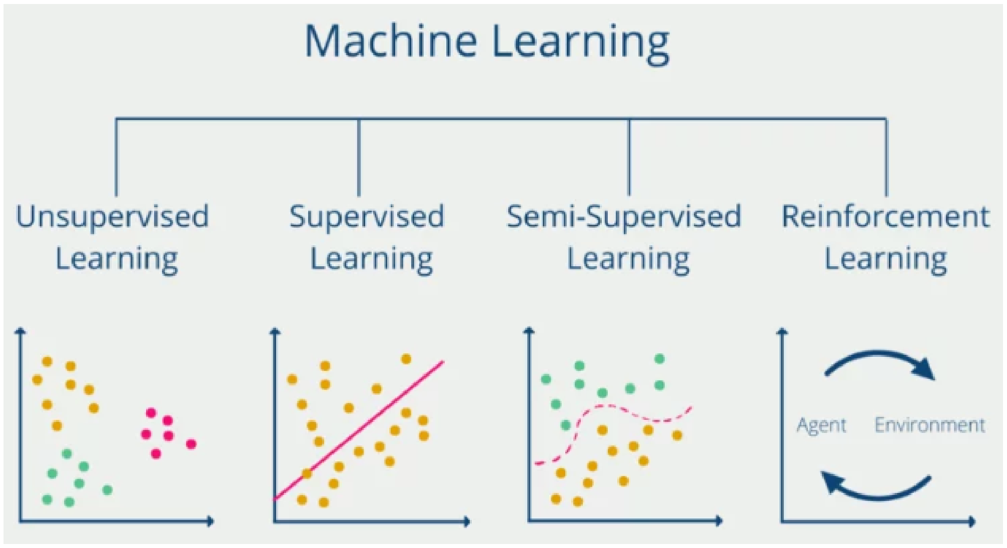


Figure 2.1: Types of Machine Learning: This diagram categorizes machine learning into four main types, each with its distinct learning approach. Source: [7].

Supervised Learning: The model is trained on a labeled dataset, where each input is paired with the correct output. The goal is to learn a mapping from inputs to outputs that generalizes well to unseen data. This method is particularly effective for tasks such as classification and regression, including vocal separation where the model distinguishes vocals from other audio elements. This research utilized this learning approach.

Unsupervised Learning: Unlike supervised learning, it works with datasets that lack labeled outputs. The model tries to uncover hidden patterns or intrinsic structures within the data. Techniques like clustering and dimensionality reduction are common, though less frequently used in audio separation compared to supervised methods.

Semi-Supervised Learning: This hybrid approach leverages a small amount of labeled data alongside a larger pool of unlabeled data. It strikes a balance between supervised and unsupervised learning, improving performance when labeled data is scarce. It can be useful in cases where acquiring labeled data is challenging or costly.

Reinforcement Learning: An agent learns to make decisions by performing actions in an environment to maximize some notion of cumulative reward. While more common in control systems and robotics, reinforcement learning can also be applied in areas like adaptive audio processing. [8]

2.2 Supervised Learning Techniques

Supervised learning, as depicted in Figure A.16, is highly relevant to this research, particularly in the context of audio processing. Here, the model is trained to minimize the difference between predicted outputs and true labels, often quantified using a loss function like Mean Squared Error (MSE).

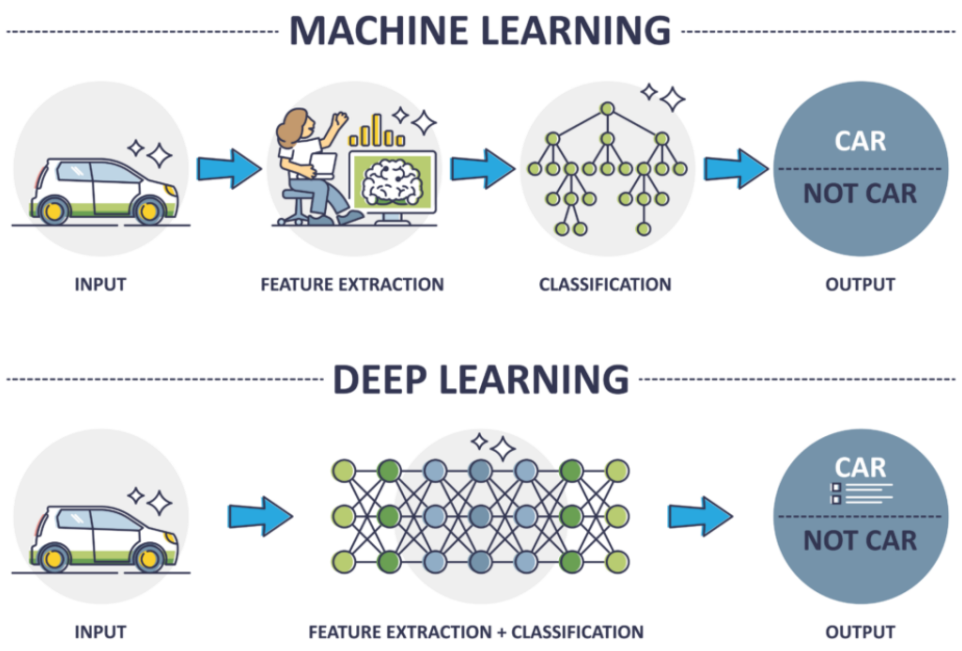


Figure 2.2: Machine Learning vs. Deep Learning: This diagram contrasts traditional machine learning, which involves manual feature extraction, with deep learning, where feature extraction and classification occur within the neural network itself. This distinction is critical to understanding the evolution and advantages of deep learning in tasks such as vocal separation. Source: [9].

In audio processing, supervised learning is particularly beneficial for tasks like vocal separation. By training on pairs of input audio mixtures and their corresponding isolated vocal tracks, the model learns to effectively differentiate between vocal and non-vocal components. The learned model is then evaluated using metrics such as Signal-to-Distortion Ratio (SDR) and Signal-to-Interference Ratio (SIR), which will be discussed in detail in the subsequent sections of this thesis.

2.3 Machine Learning Pipeline

The application of machine learning to tasks such as vocal separation follows a structured process known as the machine learning pipeline. This pipeline ensures that data is processed systematically, and models are developed, validated, and deployed effectively.

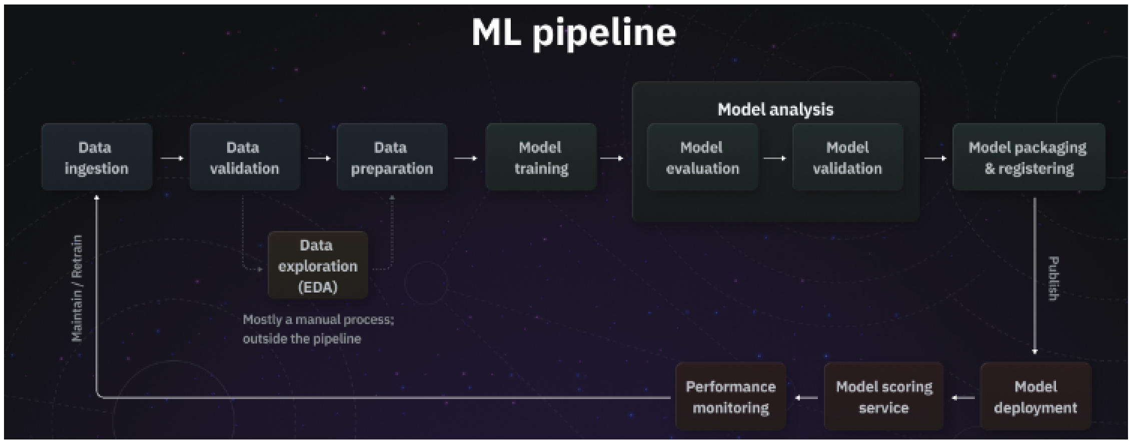


Figure 2.3: Machine Learning Pipeline: This figure outlines the typical stages in a machine learning pipeline, from data ingestion to model deployment. Each stage is crucial for ensuring the effectiveness and reliability of the final model. Source: [10].

The pipeline begins with **data ingestion**, where raw audio data is collected. This is followed by **data validation** to ensure the data is clean and consistent. **Data preparation** involves transforming the audio data into a format suitable for model training, such as converting audio signals into spectrograms.

During **model training**, the prepared data is used to train the model, which learns to map inputs to outputs. The trained model is then subjected to **evaluation** and **validation** phases to assess its performance on unseen data. Finally, the model is **deployed** in real-world applications, such as vocal separation in music production.

2.4 Deep Learning and Audio Processing

Deep learning, a specialized branch of machine learning, offers significant advantages in processing complex data types, such as audio. Unlike traditional machine learning, deep learning models automatically learn feature representations from raw data through multiple layers of abstraction. This capability is particularly beneficial in tasks like vocal separation, where the features may not be explicitly defined.

Figure 2.4 illustrates the integration of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks in processing audio data. CNNs excel at capturing spatial features in spectrograms, while LSTMs are adept at modeling temporal dependencies, making this hybrid architecture ideal for audio tasks.

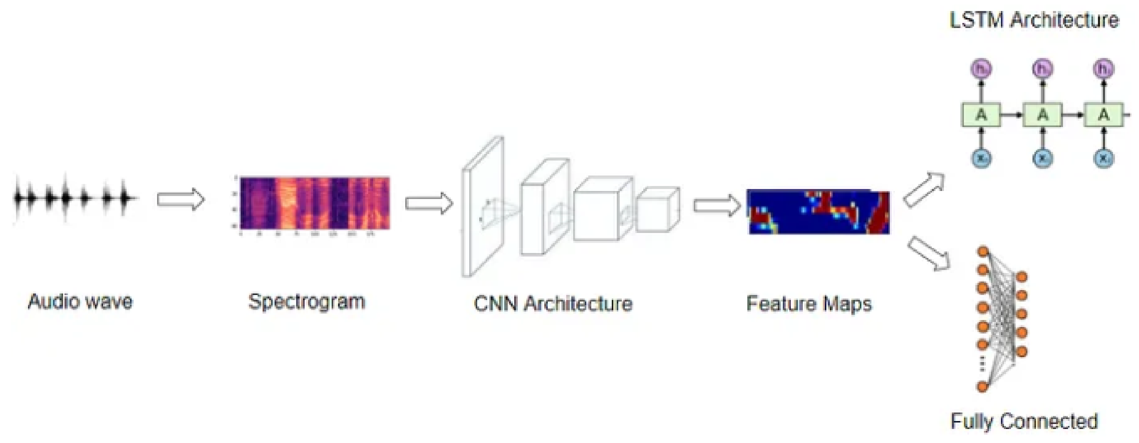


Figure 2.4: Hybrid CNN-LSTM Architecture for Audio Processing: This diagram shows how CNNs and LSTMs can be combined to process audio data for tasks like vocal separation. The CNN layers extract spatial features from the spectrogram, while the LSTM layers capture temporal patterns, leading to more effective vocal isolation. Source: [11].

In this research, the hybrid CNN-LSTM architecture was chosen to leverage the strengths of both models, resulting in a robust framework capable of handling the complexities inherent in audio data.

2.5 Fundamentals of Audio Signal Processing

Vocal removal from music is deeply rooted in audio signal processing principles. This section provides an overview of the fundamental concepts that underpin the techniques used in music source separation.

2.5.1 Digital Audio Representation

At its core, digital audio processing relies on the accurate representation of sound in a format that computers can manipulate. In the digital domain, continuous audio signals are represented as discrete sequences of numerical values. This process, known as analog-to-digital conversion, involves two key steps: sampling and quantization [12].

Sampling refers to the process of measuring the amplitude of an analog signal at regular intervals. The sampling rate, typically measured in Hertz (Hz), determines how many times per second the signal is measured. According to the Nyquist-Shannon sampling theorem, to accurately represent a signal, the sampling rate must be at least twice the highest frequency present in the signal [12]. For high-quality audio, common sampling rates include 44.1 kHz (used in CDs) and 48 kHz. Bit depth, on the other hand, refers to the number of bits used to represent each sample, directly affecting the dynamic range and precision of the audio signal representation [13].

Quantization is the process of mapping the sampled values to a finite set of discrete levels. The number of quantization levels is determined by the bit depth of the digital audio. Common bit depths include 16-bit (used in CDs) and 24-bit, with higher bit depths allowing for greater dynamic range and lower quantization noise [12].

Understanding these fundamental concepts is crucial for developing effective vocal removal systems, as they influence the quality and information content of the audio signals being processed.

The WAV (Waveform Audio File Format) is particularly relevant for this study. It consists of a header containing metadata, as well as audio data stored as a sequence of samples. The uncompressed nature of WAV files preserves all original audio information, making them ideal for high-quality vocal removal tasks [14].

2.5.2 Time-Frequency Analysis

While the time-domain representation of audio is useful for certain applications, many audio processing techniques, including vocal removal, rely heavily on frequency-domain analysis. Time-frequency analysis methods allow for the examination of how the frequency content of a signal changes over time, providing valuable insights for source separation tasks [15].

The Short-Time Fourier Transform (STFT) is a fundamental tool in time-frequency analysis. The STFT applies the Fourier transform to short, overlapping segments of the signal, resulting in a two-dimensional representation known as a spectrogram [15]. The spectrogram displays the magnitude of different frequency components as they evolve over time, making it an invaluable tool for visualizing and analyzing audio signals.

Mathematically, the STFT is defined as:

$$STFT\{x[n]\}(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{2.1}$$

where $x[n]$ is the input signal, $w[n]$ is a window function, m is the time index, and ω is the frequency.

The choice of window function and the size of the analysis window involve a trade-off between time and frequency resolution. Longer windows provide better frequency resolution but poorer time resolution, and vice versa. This trade-off is particularly relevant in vocal removal tasks, where one must balance the need for precise frequency analysis with the ability to track rapid changes in the audio signal [12].

Building upon the STFT, more advanced time-frequency representations have been developed. The Constant-Q Transform (CQT) provides a frequency resolution that scales with frequency, mimicking the human auditory system's logarithmic perception of pitch. Another important representation is the Mel-frequency spectrogram, which applies a mel-scale filterbank to the power spectrogram. This representation forms the basis for Mel-Frequency Cepstral Coefficients (MFCCs), which have found widespread use in various audio processing tasks, including music information retrieval and speech recognition [12].

These time-frequency analysis techniques provide the foundation for many of the vocal removal methods that will be discussed in subsequent sections. They allow researchers to exploit the spectral and temporal characteristics of vocal and instrumental sounds, enabling more effective separation strategies.

2.5.3 Time-Frequency Trade-off

The time-frequency trade-off is a fundamental concept in audio signal processing, particularly in the analysis of non-stationary signals, where the frequency content varies over time. This trade-off arises from the inherent limitations in simultaneously achieving high resolution in both the time and frequency domains when representing a signal [14].

When analyzing signals using the Short-Time Fourier Transform (STFT), the choice of window size directly affects the balance between time and frequency resolution. A shorter window provides better time resolution, allowing finer distinctions between events occurring at different moments in time. However, this comes at the cost of frequency resolution, resulting in a less precise representation of the signal's frequency components. Conversely, a longer window improves frequency resolution, providing a more detailed view of the frequency content but at the expense of time resolution, making it difficult to pinpoint the exact timing of changes in the signal [12].

Mathematically, the time-frequency trade-off can be understood through the Heisenberg Uncertainty Principle, which in the context of signal processing is expressed as:

$$\Delta t \cdot \Delta f \geq \frac{1}{4\pi}$$

Where:

- Δt is the uncertainty in time (time resolution).
- Δf is the uncertainty in frequency (frequency resolution).

This principle implies that a decrease in uncertainty (or increase in resolution) in one domain must necessarily result in an increase in uncertainty in the other domain. In practical terms, when processing audio signals, a trade-off must be made based on the specific requirements of the task. For example, in applications where precise timing is crucial, such as detecting transient events or onsets, a shorter window (higher time resolution) may be preferred. In contrast, tasks requiring detailed frequency analysis, such as pitch detection or spectral analysis of steady tones, may benefit from a longer window (higher frequency resolution) [14].

This trade-off plays a critical role in algorithm design and implementation in the context of audio source separation. Different sources within a mixture may require different balances of time and frequency resolution for effective separation. For instance, vocals, which often contain rapid changes in both pitch and intensity, might benefit from higher time resolution, while steady harmonic instruments may require finer frequency resolution [15].

Understanding and managing the time-frequency trade-off is therefore essential for developing effective audio processing systems, particularly in complex tasks like source separation. The choice of window size and overlap in the STFT must be carefully considered, optimizing the balance between capturing the temporal dynamics and frequency characteristics of the signal [15].

2.5.4 Spectrograms

In audio signal processing, spectrograms are an essential tool that provide the frequency spectrum of a signal as it changes over time visually. They are critical in analyzing non-stationary signals, such as audio, whose frequency content changes over time.

After the signal is split into overlapping parts using the Short-Time Fourier Transform (STFT), a spectrogram is produced. Next, each segment undergoes a Fourier Transform to convert it into the frequency domain. With time on the horizontal axis and frequency on the vertical, the resulting two-dimensional representation shows the amplitude of a certain frequency component at a given time, as represented by the color intensity at each point. [16].

In mathematical terms, if $x(t)$ represents the input signal, the STFT is defined as:

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

Where:

- $X(t,f)$ is the STFT of the signal.
- $w(t)$ is a window function that selects the portion of the signal to be transformed.
- $e^{-j2\pi f\tau}$ represents the Fourier Transform component.

The spectrogram $S(t,f)$ is then obtained by computing the magnitude of the STFT:

$$S(t,f) = |X(t,f)|$$

Spectrograms are particularly useful in tasks like audio source separation, where the goal is to isolate individual components (such as vocals, drums, or bass) from a mixture. By analyzing the spectrogram, we can observe how different frequency components evolve over time, which is crucial for identifying and separating overlapping sources [17].

In neural network-based approaches to source separation, spectrograms often serve as the primary input to the model. They are typically transformed into a logarithmic scale, converting amplitude values into decibels (dB), which better aligns with human auditory perception. This transformation is mathematically expressed as:

$$S_{dB}(t,f) = 10 \cdot \log_{10}(S(t,f))$$

Furthermore, spectrograms are usually normalized to ensure that the data is within a consistent range, improving the stability and convergence of the training process. Normalization is often performed by scaling the values between a predefined minimum and maximum range.

Overall, spectrograms are indispensable in the field of audio signal processing, particularly for applications involving time-frequency analysis. Their ability to capture the temporal evolution of frequency content makes them a powerful tool in the analysis and processing of audio signals, especially in complex tasks like source separation.

2.6 Traditional Methods of Audio Source Separation

Before the advent of deep learning techniques, researchers developed various approaches to address the challenge of music source separation, including vocal removal. This section examines these traditional methods, which laid the groundwork for modern approaches and continue to influence current research in the field. While these methods have shown promise, their application to full-length .wav files presents challenges, particularly in terms of computational efficiency and handling of long-term dependencies in music.

2.6.1 Independent Component Analysis (ICA)

Independent Component Analysis (ICA) is a statistical technique that aims to separate a multivariate signal into additive subcomponents, assuming the mutual statistical independence of the non-Gaussian source signals [16]. In the context of music source separation, ICA attempts to decompose the mixed audio signal into its constituent sources, such as vocals and instrumental components.

The ICA model can be expressed as:

$$x = As \tag{2.2}$$

where x is the observed mixed signal, A is the mixing matrix, and s represents the independent source signals.

While ICA has shown promise in separating simple mixtures, its application to music source separation faces several challenges. First, the assumption of statistical independence between sources may not hold true for musical signals, where vocals and instruments are often correlated. Second, ICA typically requires at least as many observations as there are sources, which is not always feasible in mono or stereo recordings. Despite these limitations, ICA has served as a foundation for more advanced techniques and continues to be used in combination with other methods [16].

2.6.2 Non-negative Matrix Factorization (NMF)

Non-negative Matrix Factorization (NMF) is another popular technique for music source separation. NMF decomposes a non-negative matrix V into the product of two non-negative matrices W and H:

$$V \approx WH \tag{2.3}$$

In the context of audio processing, V typically represents the magnitude spectrogram of the mixed signal, W contains spectral bases, and H represents the activations of these bases over time [18].

NMF has several advantages for music source separation. The non-negativity constraint aligns well with the additive nature of audio spectrograms, and the resulting factorization often yields interpretable components. Researchers have extended the basic NMF framework to incorporate additional constraints and priors suited for audio signals, such as sparsity and temporal continuity.

However, NMF also faces challenges in vocal removal tasks. The method does not inherently distinguish between vocal and instrumental sources, requiring careful initialization or post-processing to achieve separation. Additionally, NMF struggles with highly dynamic or varied vocal performances, where the spectral characteristics of the voice change significantly over time [18].

2.6.3 Computational Auditory Scene Analysis (CASA)

Computational Auditory Scene Analysis (CASA) approaches music source separation from a perceptual standpoint, inspired by the human auditory system's ability to segregate and group sound sources. CASA techniques typically involve two main stages: segmentation and grouping.

In the segmentation stage, the audio signal is decomposed into time-frequency units or segments. This is often achieved using techniques such as gammatone filterbanks, which model the frequency selectivity of the human cochlea. The grouping stage then combines these segments into streams corresponding to different sources, based on acoustic cues such as harmonicity, common onset and offset, amplitude and frequency modulation, and spatial location [19].

CASA methods have shown particular promise in separating pitched sources, making them potentially useful for vocal removal. Techniques such as pitch-based interference removal have been applied successfully to extract vocal melodies from polyphonic music [20]. However, CASA approaches can struggle with complex polyphonic mixtures and highly reverberant environments, where the acoustic cues become ambiguous.

2.6.4 Robust Principal Component Analysis (RPCA)

Robust Principal Component Analysis (RPCA) is a matrix decomposition technique that has found application in music source separation, particularly for vocal removal. RPCA decomposes a matrix into the sum of a low-rank matrix and a sparse matrix . In the context of vocal separation, the low-rank component typically corresponds to the repetitive structure of the musical accompaniment, while the sparse component captures the vocals.

The RPCA problem can be formulated as:

$$\min_{L,S} \|L\|_* + \lambda \|S\|_1 \quad \text{subject to} \quad M = L + S \tag{2.4}$$

where M is the magnitude spectrogram of the mixed signal, L is the low-rank matrix (instrumental), S is the sparse matrix (vocals), $\|\cdot\|_*$ denotes the nuclear norm, and $\|\cdot\|_1$ is the L1-norm.

RPCA-based methods have shown promising results in vocal separation tasks, particularly for music with relatively stable instrumental accompaniment . However, they can struggle with highly dynamic instrumental backgrounds or when the vocals exhibit repetitive patterns. [20]

2.6.5 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have emerged as a powerful tool for vocal removal tasks due to their ability to capture local patterns and hierarchical features in spectrograms. Early applications of CNNs to source separation demonstrated their potential to outperform traditional methods [21].

A significant advancement in CNN-based vocal separation came with the introduction of the U-Net architecture to this domain [22]. The U-Net, originally developed for biomedical image segmentation, consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. When applied to spectrograms, U-Net architectures have shown remarkable performance in isolating vocal and instrumental components.

The U-Net architecture can be expressed as a series of encoding and decoding operations:

$$y = f_{\text{decode}}(f_{\text{encode}}(x)) \tag{2.5}$$

where x is the input spectrogram, f_{encode} represents the contracting path operations, f_{decode} represents the expanding path operations, and y is the output spectrogram of the isolated source.

One of the U-Net architecture's key strengths in vocal removal is its ability to leverage both local and global context in the spectrogram. This allows it to effectively model the time-frequency patterns characteristic of vocal and instrumental sounds, leading to improved separation quality [23].

2.6.6 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) Networks

While CNNs excel at capturing local spectral patterns, Recurrent Neural Networks (RNNs), particularly those using Long Short-Term Memory (LSTM) units, have shown promise in modeling the temporal dependencies in audio signals [24]. RNNs process input sequences by maintaining an internal state, allowing them to capture long-term dependencies in the data.

The LSTM architecture, in particular, addresses the vanishing gradient problem that can affect standard RNNs when processing long sequences. An LSTM unit can be described by the following equations:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.6}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.7}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.8}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.9}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.10}$$

$$h_t = o_t * \tanh(C_t) \tag{2.11}$$

where f_t, i_t, and o_t are the forget, input, and output gates respectively, C_t is the cell state, and h_t is the hidden state.

In the context of vocal removal, LSTM networks have been used to model the temporal evolution of spectral features, allowing for more accurate separation of vocal and instrumental components over time [24]. Some approaches have combined CNN and LSTM layers to leverage both spectral and temporal modeling capabilities [23].

2.6.7 Bidirectional Networks

Bidirectional networks, specifically Bidirectional Recurrent Neural Networks (BiRNNs) and their variants such as Bidirectional Long Short-Term Memory (BiLSTM) networks, have proven to be highly effective in tasks that involve sequential data processing. In the context of vocal removal and audio source separation, these networks are particularly useful due to their ability to capture dependencies in both forward and backward directions in time [25].

Traditional recurrent neural networks (RNNs) process input sequences unidirectional, typically from the start to the end of the sequence. This approach allows the model to capture dependencies and patterns that have accumulated up to the current time step. However, in many real-world applications, especially in audio processing, the future context is also important. For example, understanding the presence of certain frequencies at future time steps can help better interpret the current sound, which is critical in tasks like separating vocals from a mixture [26].

A bidirectional network addresses this limitation by introducing two separate RNNs: one that processes the sequence from start to end (forward direction), and another that processes the sequence from end to start (backward direction). The outputs of these two RNNs are then combined, typically concatenated, to form a single output at each time step that incorporates information from both past and future contexts.

Mathematically, given an input sequence $x_1, x_2, \ldots, x_T$, a bidirectional RNN computes the forward hidden states $\overrightarrow{h_t}$ and the backward hidden states $\overleftarrow{h_t}$ as follows:

$$\overrightarrow{h_t} = \text{RNN}_{\text{forward}}(x_t, \overrightarrow{h_{t-1}})$$
$$\overleftarrow{h_t} = \text{RNN}_{\text{backward}}(x_t, \overleftarrow{h_{t+1}})$$

The final hidden state at each time step h_t is obtained by combining the forward and backward states:

$$h_t = \text{concat}(\overrightarrow{h_t}, \overleftarrow{h_t})$$

In the context of vocal removal, the bidirectional architecture enables the model to consider both preceding and succeeding audio contexts when processing each time step in the spectrogram. This dual perspective is particularly advantageous when dealing with complex audio mixtures, where the presence or absence of certain frequencies in the future can help resolve ambiguities in the current time step.

The incorporation of bidirectional networks into deep learning models for vocal removal has led to improved performance, particularly in handling difficult scenarios where vocals overlap with other instruments. By leveraging information from both the past and the future, bidirectional networks provide a more comprehensive understanding of the audio sequence, which is essential for accurately isolating the vocal component [25].

2.6.8 Attention Mechanisms

Attention mechanisms have become an integral part of modern deep learning architectures, particularly in tasks involving sequential data such as natural language processing, image captioning, and, importantly, audio source separation. In the context of vocal removal, attention mechanisms enable the model to focus on specific parts of the input sequence that are more relevant for separating the target signal from the mixture [27].

The core idea behind attention mechanisms is to assign different weights to different parts of the input sequence, allowing the model to "attend" to more informative regions while downplaying less important ones. This selective focus is particularly useful in audio source separation, where certain time-frequency regions of the spectrogram may contain more prominent vocal components, while others may be dominated by instrumental sounds or noise [25].

Mathematically, the attention mechanism can be described as a weighted sum of the input features, where the weights (attention scores) are computed based on each feature's relevance. Given an input sequence of features $x_1, x_2, \ldots, x_T$, the attention mechanism computes a context vector c as follows:

$$c = \sum_{t=1}^{T} \alpha_t \cdot x_t$$

Here, α_t represents the attention score for the feature x_t. These scores are typically computed using a softmax function over a set of relevance scores e_t, which are obtained by applying a compatibility function (often a simple feedforward neural network) to the input features:

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)}$$

The context vector c is then used to inform the model's decisions at each time step, allowing it to prioritize features that are most indicative of the presence of vocals. In practice, attention mechanisms can be applied in various forms, including self-attention, where the model attends to different parts of the same input sequence, and cross-attention, where it attends to an external context or memory [27].

In neural networks designed for vocal removal, the integration of attention mechanisms has shown to significantly improve performance by enhancing the model's ability to differentiate between vocal and non-vocal components in complex audio mixtures. By dynamically focusing on the most relevant time-frequency regions of the spectrogram, attention-equipped models can achieve more precise and effective separation results [25].

2.6.9 Wave-U-Net and End-to-End Approaches

While many deep learning approaches to vocal removal operate on spectrograms, recent research has explored end-to-end models that work directly on raw waveforms. The Wave-U-Net [6] is a prominent example of this approach, adapting the U-Net architecture to operate on one-dimensional time-domain signals.

The Wave-U-Net can be described as a series of downsampling and upsampling operations on the waveform:

$$y = f_{\text{upsample}}(f_{\text{downsample}}(x)) \tag{2.12}$$

where x is the input waveform, $f_{\text{downsample}}$ represents the series of convolution and downsampling operations, f_{upsample} represents the series of convolution and upsampling operations, and y is the output waveform of the isolated source.

Working directly on waveforms allows these models to capture phase information that may be lost in spectrogram-based approaches. Additionally, end-to-end models eliminate the need for hand-crafted time-frequency representations, potentially allowing them to learn more optimal representations for the separation task.

2.6.10 Adversarial Training and Generative Models

Recent advancements in deep learning have seen the application of adversarial training and generative models to vocal removal tasks. Generative Adversarial Networks (GANs) have been adapted for source separation, with the generator tasked with producing separated sources and the discriminator attempting to distinguish between real and generated separations [28] [29].

The GAN objective can be expressed as a minimax game:

$$\min_G \max_D V(D, G) = E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.13}$$

where G is the generator, D is the discriminator, x represents real data samples, and z is a noise input to the generator.

In the context of vocal removal, adversarial training can help produce more realistic separations by encouraging the model to generate outputs that are indistinguishable from real isolated vocal or instrumental tracks.

2.6.11 Multi-task and Transfer Learning Approaches

Researchers have also explored multi-task learning and transfer learning approaches to improve vocal removal performance. Multi-task learning involves training a model to perform multiple related tasks simultaneously, potentially allowing the model to learn more robust and generalizable features [30].

Transfer learning, on the other hand, involves pre-training a model on a large dataset for a related task (such as music classification) and then fine-tuning it for vocal separation. This approach can be particularly beneficial when labeled data for vocal separation is limited [31].

These deep learning approaches have significantly advanced the state of the art in vocal removal, often outperforming traditional methods in terms of separation quality and generalization to diverse musical styles. However, challenges remain, including the need for large amounts of training data, the computational complexity of some models, and difficulties in separating sources in highly complex or unconventional musical arrangements. Ongoing research continues to address these challenges and push the boundaries of what is possible in music source separation.

2.7 Evaluation Metrics and Challenges in Vocal Removal

The assessment of vocal removal systems presents unique challenges due to the complex nature of audio signals and the subjective quality of music. This section examines the various metrics and methodologies used to evaluate the performance of vocal removal algorithms, as well as the ongoing challenges in the field. Processing variable-length .wav or .mp4 files presents unique challenges, including the need for models that can handle inputs of arbitrary length and maintain consistent performance across different song durations.

2.7.1 Objective Evaluation Metrics

Objective evaluation metrics play a crucial role in quantifying the performance of vocal removal systems. These metrics typically compare the separated sources to ground truth references, providing numerical measures of separation quality.

Mean Squared Error (MSE) Mean Squared Error (MSE) is one of the most commonly used loss functions in regression tasks, including those found in neural network-based approaches to audio source separation. In the context of vocal removal, MSE serves as a quantitative measure of the difference between the predicted spectrogram and the target spectrogram.

The MSE between the predicted output $\hat{y}$ and the actual target y is defined mathematically as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Here, n represents the number of elements in the output, while $\hat{y}_i$ and y_i are the predicted and target values, respectively. The MSE calculates the average of the squared differences between the predicted and actual values, effectively penalizing larger errors more than smaller ones.

In the specific case of audio source separation, MSE is used to measure the error between the predicted and target spectrograms. Each pixel in the spectrogram corresponds to the magnitude of a particular frequency at a specific time. Thus, MSE quantifies the average squared difference between the predicted and actual magnitudes across all time-frequency bins.

The primary reason for using MSE in this context is its simplicity, as well as the fact that it directly corresponds to the objective of minimizing the difference between the predicted and target signals. When training a neural network for vocal removal, the MSE serves as the loss function that the model seeks to minimize. The lower the MSE, the closer the predicted spectrogram is to the target spectrogram, indicating a more accurate separation.

However, while MSE is a useful and widely applicable metric, it is not without limitations. One notable drawback is its sensitivity to outliers; large errors have a disproportionately large effect on the MSE due to the squaring of differences. This can sometimes lead to situations where the model focuses too much on minimizing a few large errors at the expense of the overall prediction quality.

Additionally, MSE treats all errors equally, regardless of the frequency or time position. In the context of audio source separation, this uniform treatment might not align perfectly with human auditory perception, where certain frequencies or times may be more perceptually significant. As such, MSE is often used in conjunction with other metrics that better capture the perceptual quality of the separated audio [32].

Source-to-Distortion Ratio (SDR) The Source-to-Distortion Ratio (SDR) is one of the most widely used metrics in source separation evaluation. Introduced by Vincent et al. [33], SDR measures the overall quality of the separation, taking into account both the accuracy of the target source extraction and the absence of interference from other sources. SDR is defined as:

$$\mathrm{SDR} = 10\log_{10}\frac{\|s_{\mathrm{target}}\|^2}{\|e_{\mathrm{interf}} + e_{\mathrm{noise}} + e_{\mathrm{artif}}\|^2} \tag{2.14}$$

where s_{target} is the target source, and e_{interf}, e_{noise}, and e_{artif} represent the interference, noise, and artifacts components respectively.

Source-to-Interference Ratio (SIR) and Source-to-Artifacts Ratio (SAR) Complementary to SDR, the Source-to-Interference Ratio (SIR) and Source-to-Artifacts Ratio (SAR) provide more specific insights into the nature of the separation errors. SIR measures the level of interference from other sources, while SAR quantifies the level of artifacts introduced by the separation process [33].

Normalized SDR (NSDR) To account for the varying difficulty of separation tasks across different musical pieces, researchers often use Normalized SDR (NSDR). NSDR compares the SDR of the separated source to the SDR of the mixture, providing a measure of the improvement achieved by the separation algorithm [34].

While these metrics provide valuable quantitative measures of separation quality, they have limitations. They assume the existence of isolated source recordings, which may not always be accessible for commercial music. Additionally, they may not always correlate well with human perception of separation quality, particularly for music where some bleed between sources can be aesthetically acceptable or even desirable [35].

2.7.2 Perceptual Evaluation Metrics

Recognizing the limitations of purely signal-based metrics, researchers have developed perceptual evaluation methodologies that aim to better align with human judgment of separation quality.

PEASS Framework The Perceptual Evaluation methods for Audio Source Separation (PEASS) framework, introduced by Emiya et al. [36], combines signal-based measures with a perceptual model to predict human ratings of separation quality. PEASS includes four perceptual scores:

1. Overall Perceptual Score (OPS) 2. Target-related Perceptual Score (TPS) 3. Interference-related Perceptual Score (IPS) 4. Artifacts-related Perceptual Score (APS)

These scores are designed to correlate better with human judgments than traditional signal-based metrics.

MUSHRA Tests Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) tests are frequently used for subjective evaluations of audio quality. In the context of vocal removal, listeners are presented with multiple versions of a separated track (including the original mixture as a hidden reference) and asked to rate them on a scale [37]. While MUSHRA tests provide valuable insights into perceived quality, they are time-consuming and expensive to conduct at scale.

2.7.3 Challenges in Vocal Removal Evaluation

Despite the array of available metrics, several challenges persist in the evaluation of vocal removal systems:

Lack of Diverse Datasets Many evaluations are conducted on a limited set of musical genres or styles, raising questions about the generalizability of results. The creation of diverse, high-quality datasets with isolated stems remains a challenge due to copyright restrictions and the limited availability of multi-track recordings.

Context Dependency The perceived quality of vocal removal can be highly context-dependent. For example, in some musical genres, a small amount of vocal bleed in the instrumental track may be acceptable or even desirable, while in others, it would be considered a significant flaw. Current evaluation metrics struggle to capture these nuanced, genre-specific quality judgments.

Real-world Applicability Many evaluations are conducted under idealized conditions, using studio-quality recordings. However, real-world applications often involve lower-quality recordings, live performances, or recordings with effects and processing. Evaluating the robustness of vocal removal systems under these varied conditions remains a challenge.

Computational Efficiency As vocal removal systems become more complex, evaluating their computational efficiency and real-time performance capabilities becomes increasingly important, especially for applications in live settings or on resource-constrained devices. Balancing separation quality with computational requirements is an ongoing challenge.

Perceptual Relevance While objective metrics provide valuable insights, they may not always align with human perception of separation quality. Developing evaluation methodologies that better correlate with human judgments while remaining scalable and reproducible is an active area of research.

In conclusion, the evaluation of vocal removal systems involves a complex interplay of objective metrics, perceptual assessments, and application-specific considerations. While significant progress has been made in developing comprehensive evaluation frameworks, challenges remain in creating metrics that fully capture the nuances of human perception and the diverse requirements of different musical contexts. Future research in this area will likely focus on developing more perceptually relevant metrics, creating more diverse evaluation datasets, and addressing the challenges of real-world applicability and computational efficiency.

2.8 Summary and Transition to Methodology

The literature review has traced the evolution of vocal removal techniques from fundamental audio processing concepts to advanced deep learning approaches. It has highlighted the importance of digital audio representation and time-frequency analysis as foundational elements. Traditional methods such as Independent Component Analysis and Non-negative Matrix Factorization have provided valuable insights, despite their limitations with complex recordings. The advent of deep learning, particularly Convolutional Neural Networks and architectures like U-Net, has significantly advanced the field, demonstrating remarkable performance in capturing spectro-temporal patterns of vocal and instrumental sounds [23].

Evaluation of vocal removal systems remains challenging, with objective metrics like Source-to-Distortion Ratio not always aligning with human perception. Perceptual evaluation frameworks aim to bridge this gap, but challenges persist in developing metrics that fully capture human judgment across diverse musical contexts [36].

Despite these advancements, gaps in current research include limited exploration of diverse musical genres, real-world applicability to lower-quality recordings, balancing separation quality with computational efficiency, and the need for more comprehensive evaluation frameworks.

The present study aims to address these gaps by developing a novel deep learning approach emphasizing robustness across diverse musical styles and recording conditions. The following Methodology section will detail the design and implementation of this approach, covering data preparation, the proposed neural network architecture, training process, and evaluation framework. Through this methodology, the study seeks to contribute to the advancement of vocal removal techniques, addressing identified challenges and paving the way for more robust and versatile music source separation systems.

Methodology

3.1 Technologies and Tools

This section provides an overview of the technologies and tools used throughout the research. The project was implemented using a variety of libraries and platforms, each selected for its ability to effectively contribute to different stages of the project, from data preprocessing to model development and deployment.

3.1.1 Python

Python was the primary programming language used in this research, chosen for its versatility and extensive ecosystem of libraries that support a wide range of tasks, from data processing to machine learning. Python's readability and active community support facilitated rapid development and integration of complex functionalities.

3.1.2 Librosa

Librosa, a Python library for audio analysis, was used extensively for tasks such as loading audio files, computing spectrograms via Short-Time Fourier Transform (STFT), and performing various audio preprocessing steps. Its robust functionality made it indispensable for handling and transforming audio data into forms suitable for model training.

3.1.3 NumPy and Pandas

NumPy and Pandas are essential Python libraries for numerical computation and data manipulation. NumPy provided efficient array operations necessary for handling large datasets and matrix computations, while Pandas was used to organize and manage the metadata associated with the audio files, as well as to analyze and process the results of experiments.

3.1.4 TensorFlow and Keras

TensorFlow, along with its high-level API Keras, was employed for building, training, and evaluating the neural network models. TensorFlow's flexibility and performance made it the ideal choice for implementing the deep learning models required for this research. Keras provided a user-friendly interface, allowing for rapid prototyping and experimentation with different model architectures.

3.1.5 Optuna

Optuna was used for hyperparameter optimization, a crucial aspect of fine-tuning the model to achieve optimal performance. Optuna's Bayesian optimization approach efficiently searched the hyperparameter space, identifying configurations that enhanced model accuracy and generalization while minimizing training time.

3.1.6 Weights Biases (WandB)

Weights Biases (WandB) was employed for experiment tracking and model versioning. WandB allowed for real-time monitoring of training metrics, visualizing the progress of experiments, and managing different versions of models. This tool was essential for maintaining a comprehensive record of the experimental process and facilitating collaboration.

3.1.7 Matplotlib and Seaborn

Matplotlib and Seaborn were the primary libraries used for data visualization. These tools were used to create plots and charts that helped in analyzing the model's performance and in presenting the results in a clear and interpretable manner. Visualizations were critical for understanding the model's behavior during training and effectively communicating the findings.

3.1.8 Scikit-learn

Scikit-learn, a widely-used machine learning library, provided tools for preprocessing data and evaluating the model. Its seamless integration with NumPy and Pandas allowed for efficient data manipulation and supported various tasks such as scaling data and computing evaluation metrics.

3.1.9 ONNX (Open Neural Network Exchange)

ONNX was utilized for model export and deployment, ensuring compatibility across different platforms and environments. This open-source format allowed the trained model to be used in various deep learning frameworks, facilitating deployment in real-world applications, particularly in environments with limited computational resources.

3.1.10 Kaggle Environment

The entire project was executed within the Kaggle environment, which provided the necessary computational resources, including access to powerful GPUs like the P100. This platform was chosen for its accessibility and its ability to handle the computationally intensive tasks required for training deep learning models on large datasets. It also offered seamless integration with other tools and libraries, enhancing the overall efficiency of the research process.

3.2 Dataset Preparation

The **MUSDB18 dataset** was chosen for its diverse collection of professionally produced music tracks, each with isolated stems for vocals, drums, bass, and other instruments. Genre and instrumentation were critical for training a model that could generalize to different types of audio mixtures.

3.2.1 Data Preprocessing

The audio files were converted into spectrograms using Short-Time Fourier Transform (STFT) to capture both temporal and spectral features. Spectrograms provide a detailed time-frequency representation, essential for distinguishing vocals from other components. The data was then normalized to ensure consistency across samples, preventing bias due to variations in loudness.

3.2.2 Data Augmentation

To improve generalization, data augmentation techniques such as pitch shifting, time stretching, and adding synthetic noise were applied. These techniques simulated variations in vocal pitch, tempo, and noise levels, making the model more robust to different real-world audio conditions.

3.2.3 Data Splitting

The dataset was split into training (80%), validation (10%), and test (10%) sets. Stratified splitting ensured a representative distribution of genres across all sets. The training set was used to train the model, the validation set for hyperparameter tuning and monitoring overfitting, and the test set for final performance evaluation.

3.3 Model Development

3.3.1 Training and Validation

The model development process employed a hybrid architecture combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. This approach was selected to leverage CNNs for spatial feature extraction and LSTMs for capturing temporal dependencies in the audio data.

The CNN-LSTM model was trained using the Adam optimizer with a learning rate determined through Optuna hyperparameter tuning. The loss function used was Mean Squared Error (MSE), chosen for its effectiveness in regression tasks. Training was conducted over 100 epochs with a batch size of 32, achieving a validation accuracy of 0.9030 after the final epoch. Regularization techniques, including dropout and early stopping, were employed to prevent overfitting and ensure generalization.

3.3.2 Evaluation and Testing

The model's performance was evaluated using the MUSDB18 test set, focusing on key metrics such as Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Signal-to-Artifact Ratio (SAR). The model achieved an SDR of 1.76 dB, indicating effective separation of vocals from instrumental components. The SIR and SAR metrics were also used to assess the model's ability to minimize interference and artifacts, with the model achieving competitive results across these metrics.

3.4 Quantization

To optimize the model for deployment, quantization techniques were applied using the Open Neural Network Exchange (ONNX) format. Dynamic quantization was chosen to reduce the model size and enhance inference speed by converting the model weights from 32-bit floating-point to 8-bit integers. This process significantly reduced the computational load and memory requirements, making the model suitable for real-time applications and deployment in resource-constrained environments.

Quantization did not substantially impact the model's performance, maintaining the accuracy of vocal separation while improving efficiency. This optimization was crucial for ensuring the model's practical usability in scenarios where computational resources are limited, such as mobile devices or embedded systems.

Experimental Setup

4.1 Description of Dataset

The **MUSDB18 dataset** is used in this study for vocal separation. This dataset contains 150 professionally mixed stereo tracks, divided into 100 training tracks and 50 test tracks. Each track is provided in four separate stems: vocals, drums, bass, and other instruments. This multitrack format is essential for training and evaluating models designed to isolate specific sources from a complex mixture of sounds.

4.1.1 Preprocessing and Custom Classes

To effectively use the MUSDB18 dataset, several preprocessing steps were implemented. These steps are critical for ensuring that the data is in the correct format for training a neural network. Additionally, custom classes were developed to streamline the processing pipeline, making it more efficient and less error-prone.

Data Loading with `musdb`: The dataset was loaded using the `musdb` package, which is specifically designed for handling the MUSDB18 dataset. The use of `musdb` ensures that the dataset is accessed efficiently, particularly when dealing with the compressed MP4 format. The decision to use the compressed format was driven by the need to balance storage, computational and memory efficiency constraints with the fidelity of the audio data. Compressed files reduce the demand on storage and processing resources while still maintaining sufficient quality for training deep learning models.

Listing 4.1: Loading the MUSDB18 Dataset

```python
import musdb

mus = musdb.DB(
    root='path_to_musdb',
    is_wav=False  # Use of compressed MP4 files
)
```

Spectrogram Computation using STFT: After loading the audio data, it is essential to transform the time-domain signals into a frequency-domain representation. This is accomplished through the Short-Time Fourier Transform (STFT), a mathematical technique that decomposes a signal into its constituent frequencies. The STFT is used because it provides a time-frequency representation, known as a spectrogram, which is crucial for analyzing audio data in tasks like source separation. The `librosa` library is used for this transformation because it is a robust and widely adopted tool for audio analysis in Python, offering a well-optimized implementation of the STFT.

Listing 4.2: Spectrogram Computation with STFT

```python
import librosa

def compute_spectrogram(audio, sr=44100):
    return librosa.stft(
        audio,
        n_fft=2048,
        hop_length=512
    )

# Example usage within the data pipeline
for track in mus:
    vocal_spectrogram = compute_spectrogram(track['vocals'])
```

Normalization for Consistency: Once the spectrograms are computed, normalization is applied. This step is critical for ensuring that the input data to the neural network is on a consistent scale, typically between -1 and 1. Without normalization, the varying scales in the input data could lead to poor model performance and instability during training. Normalization helps in achieving faster convergence and better generalization. The normalization process divides each spectrogram by its maximum absolute value, ensuring a uniform range across all inputs.

Listing 4.3: Normalization of Spectrogram Values

```python
def normalize_spectrogram(spectrogram):
    return spectrogram / np.max(np.abs(spectrogram))

normalized_spectrogram = normalize_spectrogram(vocal_spectrogram)
```

Custom Preprocessing Classes: To handle the preprocessing pipeline's complexity, custom classes were created. These classes encapsulate the various preprocessing steps, making the code more modular and easier to maintain. Custom classes allow for reusable components, reducing redundancy and potential code errors. For instance, the `AudioPreprocessor` class is responsible for loading audio files, computing spectrograms, and applying normalization. Centralizing these operations within a class makes it easier to update or modify preprocessing steps without affecting the overall pipeline.

Listing 4.4: Example of a Custom Preprocessing Class

```python
class AudioPreprocessor:
    def __init__(self, sample_rate=44100, n_fft=2048, hop_length=512):
        self.sample_rate = sample_rate
        self.n_fft = n_fft
        self.hop_length = hop_length

    def compute_spectrogram(self, audio):
        return librosa.stft(
            audio,
            n_fft=self.n_fft,
            hop_length=self.hop_length
        )

    def normalize_spectrogram(self, spectrogram):
        return spectrogram / np.max(np.abs(spectrogram))

    def preprocess(self, audio):
        spectrogram = self.compute_spectrogram(audio)
        return self.normalize_spectrogram(spectrogram)
```

Spectrogram Analysis: The distribution of spectrogram values was analyzed to understand the dataset's characteristics. Figure 4.1 shows that most values are concentrated around low magnitudes, which is common in audio signals where quieter segments dominate. This analysis highlights the importance of normalization.

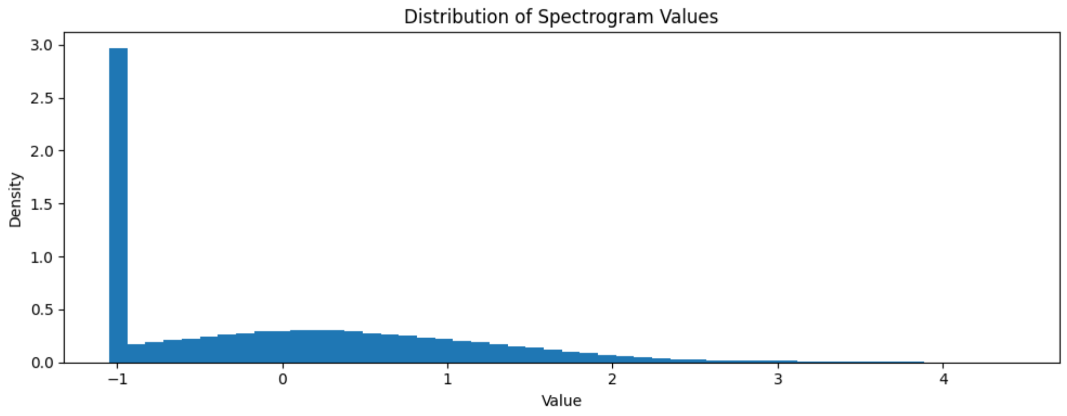


Figure 4.1: Distribution of Spectrogram Values. The histogram illustrates the predominance of lower magnitude values, emphasizing the need for normalization to ensure consistent model input.

Target Spectrogram Visualization: An example of a vocal spectrogram is shown in Figure 4.2. This image represents the frequency content of vocals over time. The bright areas indicate higher energy parts of the vocal signal, while the darker areas show lower energy. The model needs to learn to identify and isolate these patterns from the mixture.

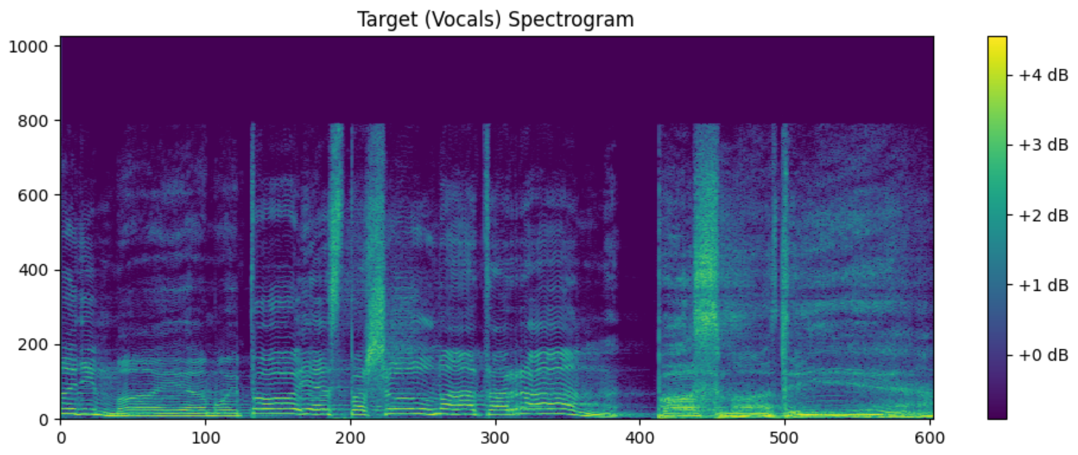


Figure 4.2: Target Spectrogram (Vocals) . This spectrogram shows the frequency content of vocals over time, illustrating the complexity the model must learn to separate from the mixture.

In summary, the MUSDB18 dataset, with its structured multitrack format, provides a robust foundation for training and evaluating vocal separation models. The preprocessing pipeline, which includes custom classes and normalization applications, ensures that the dataset is optimally prepared for the deep learning tasks in this research.

4.2 Choosing Architectures

Selecting the appropriate architecture is a crucial step in designing a model that effectively handles the complexities of vocal separation. The architecture determines how the model processes input data, learns patterns, and performs the separation task. This section details the architecture choices made for this research, including the integration of Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks, and the development of the MaskInference module.

4.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are employed due to their efficacy in processing image-like data, such as spectrograms. Spectrograms represent audio data in the frequency domain and exhibit spatial hierarchies similar to images. CNNs can efficiently capture these hierarchies through convolutional layers, which identify local patterns and aggregate them into more complex features.

Feature Extraction: CNNs are adept at feature extraction because they learn filters that detect local patterns in the input data. For spectrograms, these patterns might correspond to specific frequencies or time intervals associated with vocals or instruments. By stacking multiple convolutional layers, the network progressively learns more abstract representations of the input data, enhancing its ability to distinguish between different sources.

Parameter Efficiency: CNNs are parameter-efficient compared to fully connected networks. Convolutional layers use shared weights, meaning fewer parameters are needed to learn effective representations. This reduces the risk of overfitting, particularly with a limited dataset like MUSDB18, and allows for deeper networks without a proportional increase in computational cost.

4.2.2 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

While CNNs effectively capture spatial hierarchies, they are less suited for modeling temporal dependencies in sequential data, such as audio signals. To address this, Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks, are integrated into the architecture. LSTMs handle sequences by maintaining a memory of previous inputs, allowing the model to learn temporal dependencies across time steps.

Temporal Modeling: LSTMs are crucial for modeling the temporal aspects of audio signals. In vocal separation, understanding how vocal patterns evolve over time is essential for accurate source separation. The LSTM units capture these temporal dependencies by maintaining a hidden state that evolves with each time step, allowing the model to consider both the current and past context when making predictions.

Combining CNNs and LSTMs: To leverage the strengths of both CNNs and LSTMs, a hybrid architecture was selected. CNN layers first extract spatial features from the spectrograms, which are then passed to the LSTM layers. The LSTM layers model the temporal dynamics of the extracted features, enabling the network to make more informed predictions about the presence of vocals in each time frame. This combination allows the model to capture both spatial and temporal patterns, improving its performance on the vocal separation task.

4.2.3 MaskInference Module

The core of the architecture is the `MaskInference` module, designed to apply a mask over the input spectrogram, separating the vocal components from other elements. This module includes an LSTM layer to process the reshaped input, followed by a fully connected layer to produce the mask.

Listing 4.5: MaskInference Module Definition

```python
class MaskInference(nn.Module):
    def __init__(self):
        super(MaskInference, self).__init__()
        self.lstm = nn.LSTM(input_size=(Config.N_FFT // 2 + 1) * 2,
            hidden_size=128, num_layers=2, batch_first=True, bidirectional=
            True)
        self.mask = nn.Linear(256, (Config.N_FFT // 2 + 1) * 2)  #
            Adjusted output size

    def forward(self, x):
        batch_size, channels, freq_bins, time_steps = x.size()

        # Reshape for LSTM input
        x = x.reshape(batch_size, time_steps, -1)

        x, _ = self.lstm(x)

        # Apply mask and reshape back
        mask = self.mask(x)
        mask = mask.reshape(batch_size, channels, freq_bins, time_steps)
        return mask
```

Why LSTM and MaskInference? The LSTM layer is pivotal because it allows the model to capture the temporal evolution of features, which is critical in audio tasks where the sequence of data points carries important information. The `MaskInference` module is named as such because it infers a mask that selectively highlights the vocal components within the spectrogram. The term ”mask” in this context refers to a filter applied to the spectrogram, where certain frequencies (corresponding to vocals) are emphasized while others are attenuated. This masking process is central to the model's ability to isolate vocals from a mixture.

4.2.4 Consideration of Computational Resources

The architecture also considers the available computational resources. Training deep networks involving CNNs and LSTMs can be resource-intensive. Therefore, the architecture was designed to balance performance with computational feasibility. Techniques such as batch normalization and dropout were employed to improve training stability and prevent overfitting, while using a smaller batch size ensured that the model could be trained on the available hardware without exhausting memory resources.

Batch Normalization: Batch normalization normalizes inputs to each layer, reducing internal covariate shift and allowing for higher learning rates. This technique speeds up training and improves model performance, especially in deep networks.

Dropout: Dropout was used as a regularization technique to prevent overfitting. By randomly setting a fraction of the activations to zero during training, dropout forces the network to learn more robust features that generalize better to unseen data.

Small Batch Size: A smaller batch size was chosen to accommodate the memory limitations of the available hardware. Although smaller batch sizes can lead to noisier gradient estimates, they also offer a form of regularization that can improve generalization.

4.2.5 Summary of Architecture Choice

The chosen architecture combines CNNs and LSTMs to handle both the spatial and temporal aspects of the vocal separation task. CNNs were selected for their ability to efficiently extract spatial features from spectrograms, while LSTMs were integrated to capture temporal dependencies in the audio signals. The `MaskInference` module was developed to apply a mask over the spectrogram, effectively isolating vocal components. The architecture was carefully balanced with the available computational resources in mind, employing techniques such as batch normalization, dropout, and small batch sizes to optimize performance and feasibility.

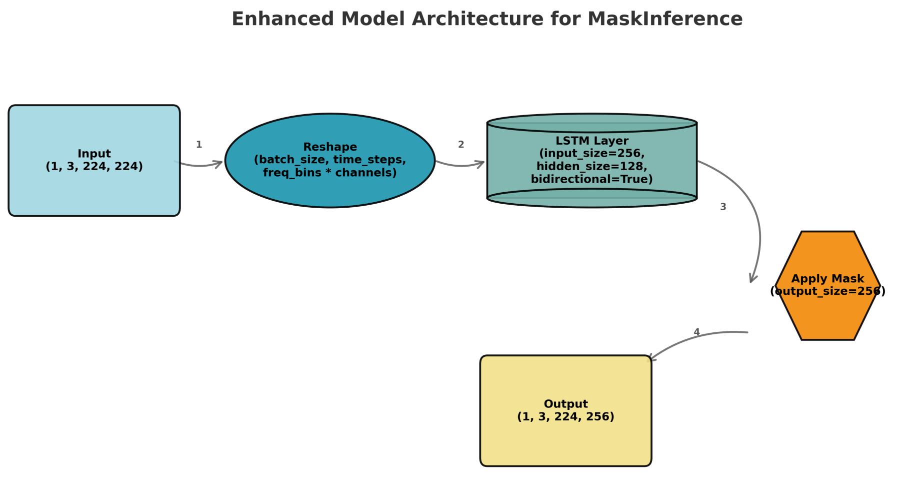

Enhanced Model Architecture for MaskInference
Input (1, 3, 224, 224)
1
Reshape (batch_size, time_steps, freq_bins * channels)
2
LSTM Layer (input_size=256, hidden_size=128, bidirectional=True)
3
Apply Mask (output_size=256)
4
Output (1, 3, 224, 256)

Figure 4.3: Enhanced Model Architecture for MaskInference. This diagram illustrates the model architecture starting with the input spectrogram, which is reshaped and processed through an LSTM layer. The LSTM captures the temporal dependencies in the data, which is critical for accurate vocal separation. The output of the LSTM is passed through a linear layer that generates a mask. This mask is then applied to the input spectrogram to isolate the vocal components. Finally, the masked output is reshaped to match the original input dimensions, completing the separation process. The architecture effectively combines spatial and temporal feature extraction to achieve high-quality vocal separation.

This architecture forms the foundation of the model's ability to accurately separate vocals from other components in the MUSDB18 dataset.

4.3 Model Implementation

The implementation of the vocal separation model was executed in several stages, each carefully designed to optimize the model's performance and ensure high-quality output.

This section provides an in-depth discussion of each stage, highlighting the critical decisions, hyperparameter tuning, and technical considerations involved.

4.3.1 Architecture and Training Process

The architecture selected for this task combined Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks. This hybrid architecture was chosen to leverage the unique strengths of each type of network: CNNs for their ability to capture spatial features in spectrograms and LSTMs for their capacity to model temporal dependencies.

Convolutional Neural Networks (CNNs): CNNs are particularly effective at extracting spatial features from spectrograms, which represent audio data in the frequency domain. The convolutional layers in CNNs scan the spectrograms to detect local patterns, such as specific frequency bands or time intervals that are indicative of vocal components. These patterns are progressively combined through deeper layers, allowing the network to build a comprehensive representation of the audio data.

Long Short-Term Memory (LSTM) Networks: LSTM networks were incorporated into the architecture to address the temporal nature of audio signals. Unlike standard Recurrent Neural Networks (RNNs), LSTMs can maintain a memory of previous inputs over long sequences, making them well-suited for tasks that require understanding the evolution of vocal patterns over time. This temporal modeling is crucial for accurately separating vocals from the background music or other instruments.

Training Process: The training process involved backpropagation and stochastic gradient descent (SGD) as the optimization algorithm, with the mean squared error (MSE) loss function guiding the updates to the network's weights. MSE was chosen because it provides a direct measure of the difference between the predicted and actual spectrogram values, thus driving the network to produce outputs that closely match the target vocals. The training process was conducted over multiple epochs, during which the model's performance was continuously monitored through loss curves and evaluation metrics.

4.3.2 Hyperparameter Tuning with Optuna

To fine-tune the model's hyperparameters and achieve optimal performance, we employed Optuna, an automatic hyperparameter optimization framework. Optuna uses a Bayesian optimization approach, which is efficient in searching the hyperparameter space and identifying configurations that yield the best performance.

These hyperparameters as shown in the Table A.1 were chosen based on their ability to balance the model's complexity and its capacity to generalize across different audio mixtures. The learning rate was set to 0.001 to ensure stable convergence, while the batch size and hidden size were optimized to maximize the model's learning efficiency without overfitting. They were determined through numerous trials, each evaluating the model's performance on a validation set. The combination of a relatively low dropout rate and a single LSTM layer

Table 4.1: Best Hyperparameters Found by Optuna

Hyperparameter	Value
Learning Rate	0.001
Batch Size	32
Hidden Size	128
Epochs	100
Dropout Rate	0.3
Optimizer	Adam

was found to strike a balance between model complexity and the ability to capture temporal dependencies without overfitting.

Listing 4.6: Optuna Objective Function with Best Hyperparameters

```python
def objective(trial):
    # Suggest values for the hyperparameters
    learning_rate = trial.suggest_loguniform('lr', 1e-5, 1e-1)
    batch_size = trial.suggest_categorical('batch_size', [16, 32, 64])
    hidden_size = trial.suggest_int('hidden_size', 32, 256)
    dropout_rate = trial.suggest_uniform('dropout_rate', 0.1, 0.5)
    num_layers = trial.suggest_int('num_layers', 1, 3)
    weight_decay = trial.suggest_loguniform('weight_decay', 1e-6, 1e-3)

    # Create the model with the suggested hyperparameters
    model = MaskInference(hidden_size=hidden_size, num_layers=num_layers,
        dropout_rate=dropout_rate)

    # Training and validation logic goes here
    ...

    return validation_loss
```

The best hyperparameters were used to re-train the model to ensure that it was well-optimized for the vocal separation task. This re-training, with the optimal configuration, aimed to maximize generalization to unseen data and minimize the risk of overfitting.

4.3.3 Post-Processing Techniques

Post-processing was an essential step in refining the model's output to ensure that the separated vocals were as clean and artifact-free as possible. Several techniques were used to improve the quality of the model's audio.

Mask Refinement: The initial output from the model included a mask that was applied to the input spectrogram to isolate vocal components. However, this mask often contained noise and other artifacts that could degrade the quality of the separated vocals. To address this, post-processing techniques such as median filtering and Wiener filtering were applied. These methods helped to smooth the mask and remove residual noise, resulting in a cleaner separation of the vocals from the background music.

Dynamic Range Compression: Dynamic range compression was another critical post-processing step. This technique was used to balance the volume levels within the separated audio, ensuring that quieter parts of the vocal track remained audible without introducing distortion. Dynamic range compression helped to produce a more consistent and professional-sounding output, which is critical for practical applications of separated audio.

Validation Batches: During training, validation batches were created from the training set to evaluate the model's performance on unseen data. These batches allowed for continuous monitoring of the model's ability to generalize, ensuring that it performed well not only on the training data but also on new, unseen examples. This step was crucial in preventing overfitting and ensuring that the model was robust and capable of handling various audio inputs.

4.3.4 Summary of Model Implementation

The model implementation involved several interconnected processes, from architecture design and hyperparameter tuning to post-processing. The architecture, combining CNNs and LSTMs, was specifically chosen to effectively capture both spatial and temporal features of the audio data. The hyperparameters, optimized through Optuna, played a crucial role in enhancing the model's generalization capabilities. Finally, post-processing techniques ensured that the model produced high-quality, artifact-free audio outputs. Together, these processes formed the foundation for the model's strong performance in the vocal separation task.

4.4 Advanced Performance Analysis

4.4.1 Saliency Map Analysis

To gain deeper insights into how the model processes audio data during vocal separation, we generated a saliency map. The saliency map visualizes the areas of the spectrogram where the model concentrated its attention during inference. By highlighting the most influential time frames and frequency bins, the saliency map helps us understand which parts of the input spectrogram the model deemed most relevant for isolating vocals.

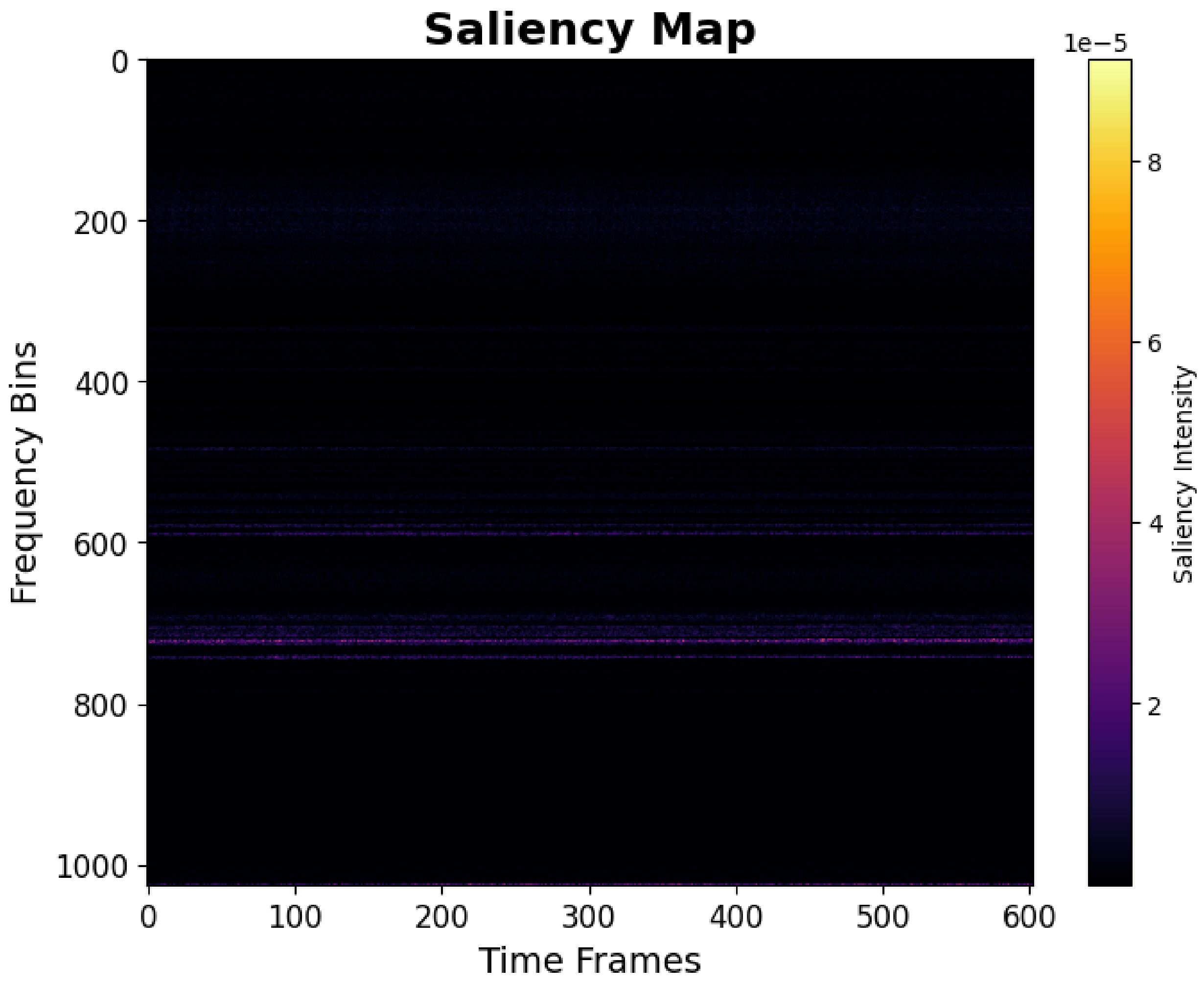


Figure 4.4: Saliency Map showing the model's focus during the separation process. Brighter regions indicate areas where the model concentrated its attention, corresponding to significant features used for vocal separation.

As shown in Figure A.13, the model's attention is not uniformly distributed across the spectrogram. The brighter regions represent frequency bins and time frames where the model identified critical features for vocal separation. These areas of high saliency intensity are likely where the model was able to distinguish vocal elements from the background instrumentals effectively. However, the map also reveals that the model's focus tends to concentrate on specific frequency ranges, which might explain some of the limitations observed during testing, such as the misclassification of non-vocal sounds or the presence of residual artifacts in the separated vocals.

The insights gained from the saliency map suggest that while the model is effective in identifying key vocal features, there may be a need for more sophisticated attention mechanisms to ensure that all relevant aspects of the input are considered equally. This could lead to improvements in handling complex audio mixtures where vocals are not easily distinguishable from the accompanying instruments.

4.4.2 Performance Metrics Review

The performance analysis of the vocal separation model focuses on evaluating its effectiveness in separating vocals from complex audio mixtures. This section provides a detailed examination of the model's performance during the validation phase and its generalization ability during the testing phase. The evaluation is based on key metrics such as Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Signal-to-Artifact Ratio (SAR), which collectively offer a comprehensive assessment of the model's capabilities.

4.4.3 Evaluation Metrics and Final Model Performance

After identifying the optimal hyperparameters through Optuna, the model was re-trained using these best parameters to ensure optimal performance. The re-trained model was then evaluated based on several key metrics that provide insights into its ability to perform vocal separation effectively.

Table 4.2: Final Model Training Metrics

Metric	Value
Final Training Loss	0.60
Validation Loss	0.61
SDR (Training)	1.76 dB
SIR (Training)	-0.37 dB
SAR (Training)	1.76 dB

The training process was monitored closely, with metrics such as Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Signal-to-Artifact Ratio (SAR) being calculated to gauge the model's ability to separate vocals from complex audio mixtures. These metrics,as shown in the Table A.3, were also used to validate the model's performance on unseen test data. These were calculated based on the final training and validation runs, reflecting the model's effectiveness in minimizing artifacts and distortion while successfully isolating the target vocals. The SDR and SAR values indicate the model's overall quality in separating the vocals from the background, while the negative SIR suggests potential challenges in completely eliminating interference from other sources.

Training and Validation Loss: Figure A.4 presents the training and validation loss curves over multiple epochs. The consistent reduction in loss values indicates that the model successfully minimized errors as it learned to separate vocals from the audio mixtures. This downward trend is a positive sign of the model's learning progression, showing that it was effectively trained and capable of generalizing beyond the training data.

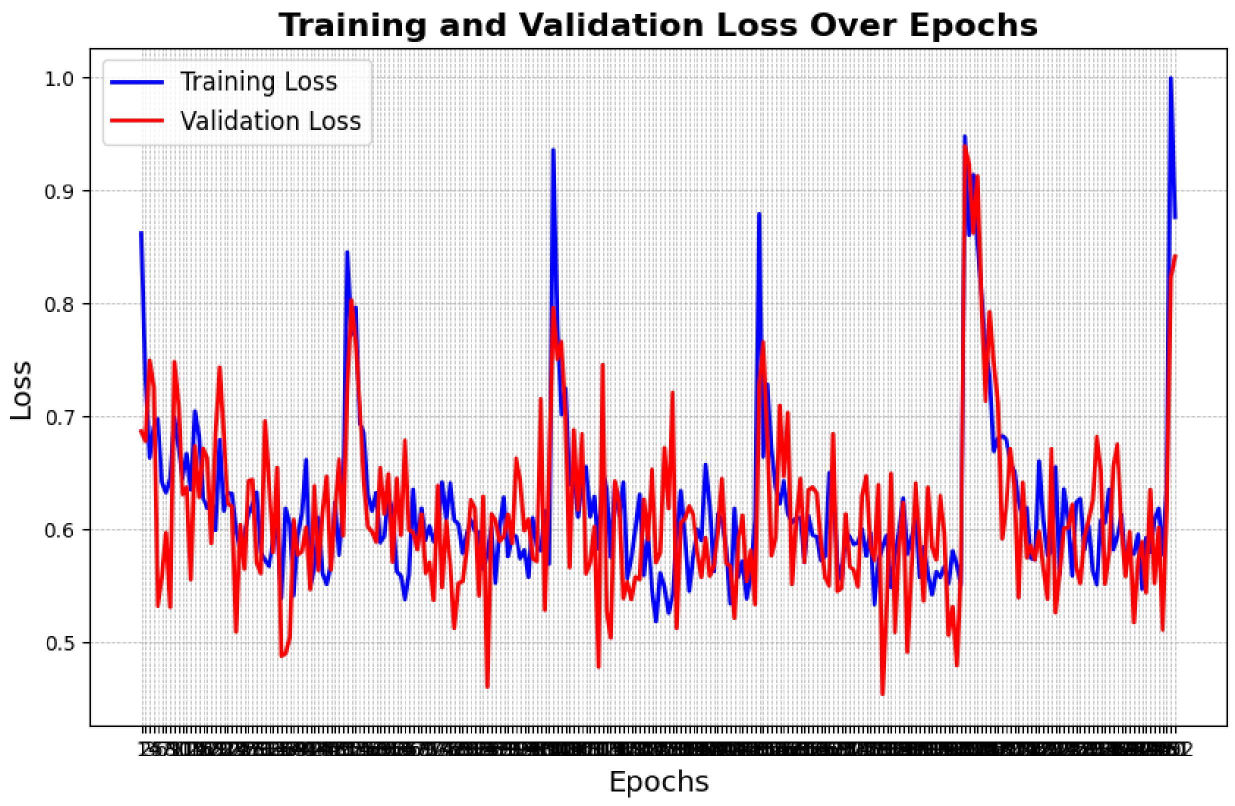


Figure 4.5: Training and Validation Loss over Epochs. This figure illustrates the reduction in loss as the model learns to separate vocals from the mixture, with both training and validation loss showing a general downward trend.

Validation Metrics: During the validation phase, the model was evaluated on separate validation data sets. The metrics obtained during validation provide insights into how well the model could generalize to unseen data. Figures A.8, A.9, and A.7 show the progression of SDR, SIR, and SAR metrics over the course of validation.

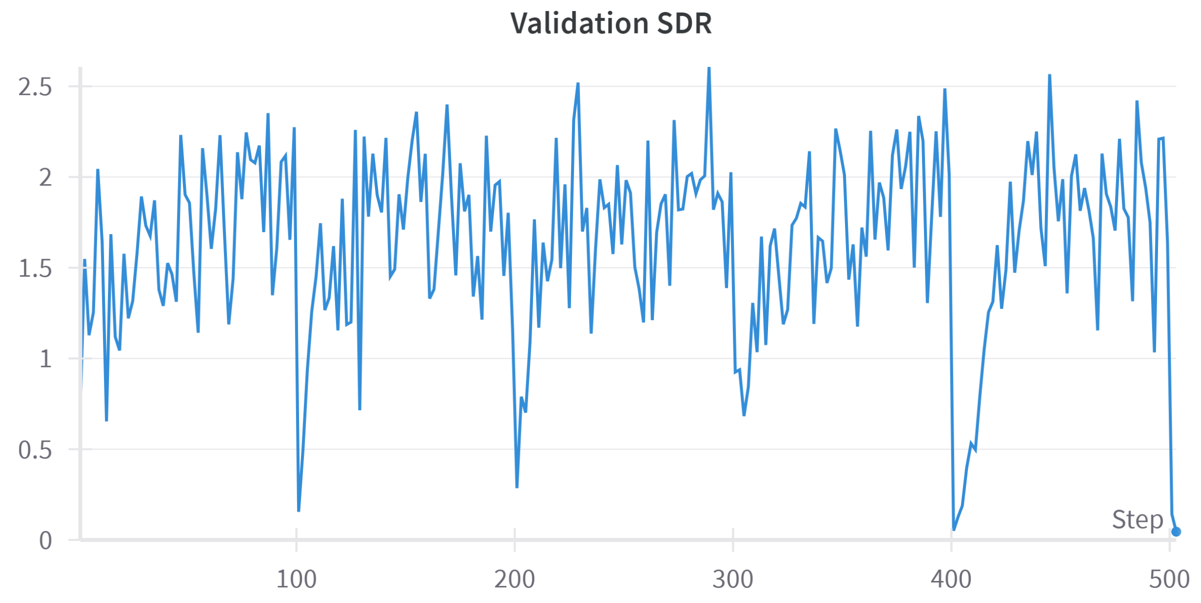


Figure 4.6: Validation SDR over 500 Steps. SDR measures the overall quality of the separated audio, with higher values indicating more accurate vocal isolation.

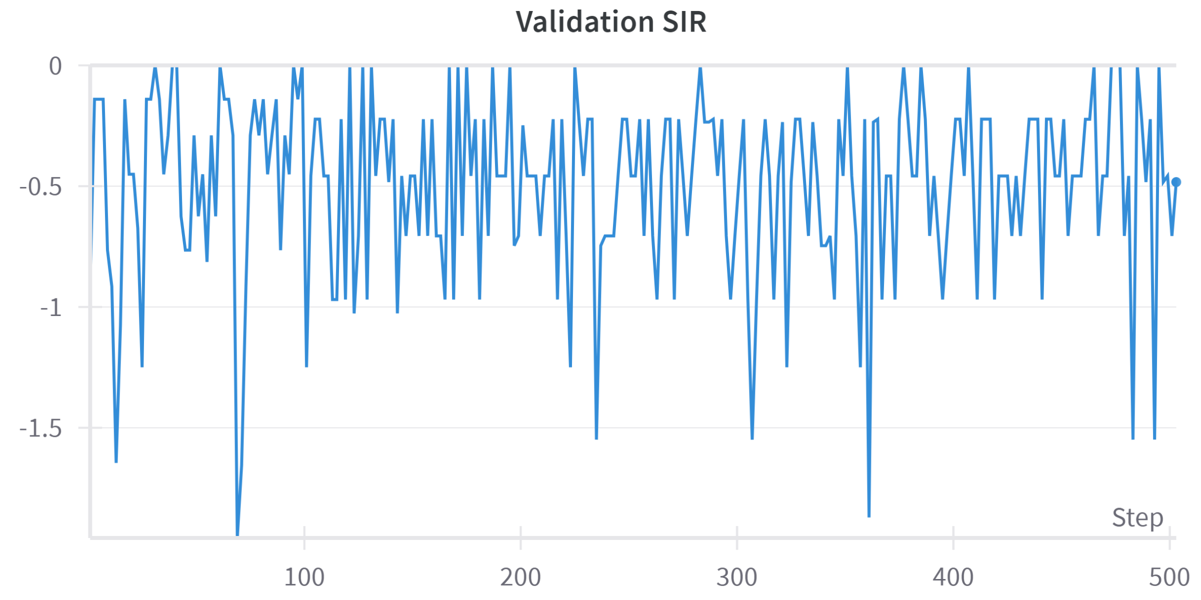


Figure 4.7: Validation SIR over 500 Steps. SIR evaluates the model's ability to isolate vocals from other interfering signals, with higher values indicating better separation performance.

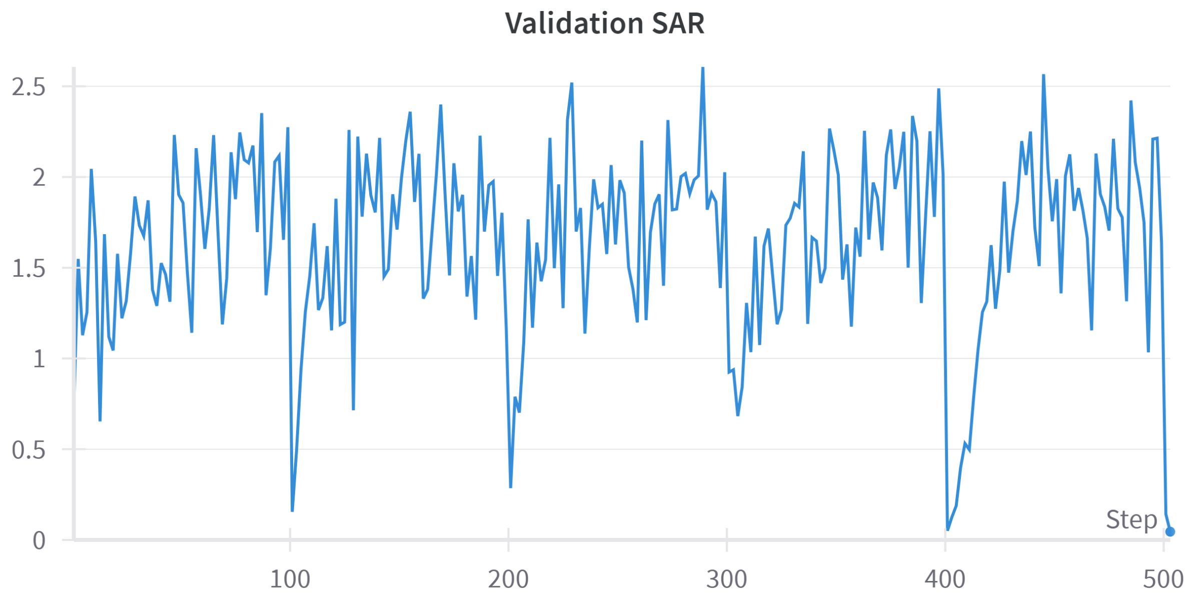


Figure 4.8: Validation SAR over 500 Steps. SAR measures the model's effectiveness in minimizing artifacts, with higher values indicating cleaner audio output.

4.4.4 Testing Phase and Test Metrics

After completing the validation phase, the model was subjected to a final evaluation using a separate test set. The test set was distinct from the training and validation data, providing a measure of the model's ability to generalize to completely unseen data.

Table 4.3: Final Model Performance Metrics (Test)

Metric	Value
SDR (Signal-to-Distortion Ratio)	-inf dB
SIR (Signal-to-Interference Ratio)	-inf dB
SAR (Signal-to-Artifact Ratio)	-inf dB
Test Loss	0.61

The results from the testing phase were unexpected,as seen in the Table A.2, as the model demonstrated a significant drop in performance when evaluated on the test data. The negative infinity values for SAR, SDR, and SIR indicate that the model struggled greatly with the test set, possibly due to differences in the audio characteristics between the training/validation data and the test data or the presence of particularly challenging audio samples that the model was not equipped to handle.

Interpretation of Test Results: The discrepancy between the validation and test metrics suggests that the model may have been overfitting to the validation data or that the test set contained audio characteristics that were significantly different from what the model had been trained on. These results highlight areas for potential improvement, such as increasing the diversity of the training data, employing data augmentation techniques, or exploring alternative architectures that might be more robust to varying audio conditions.

4.4.5 Comparison of Final Training and Test Metrics

The comparison between the final metrics achieved during training and those observed during testing provides insight into the model's generalization capabilities. Figures A.10 and A.11 present these comparisons, highlighting both consistencies and discrepancies between training and test performance.

Final Training vs. Test Loss: Figure A.10 shows the final loss during training compared to the loss observed during testing. The minimal difference between these values indicates that the model maintained its learning effectiveness when applied to unseen data, though the slight increase in test loss suggests some level of overfitting or model adaptation to the training data.

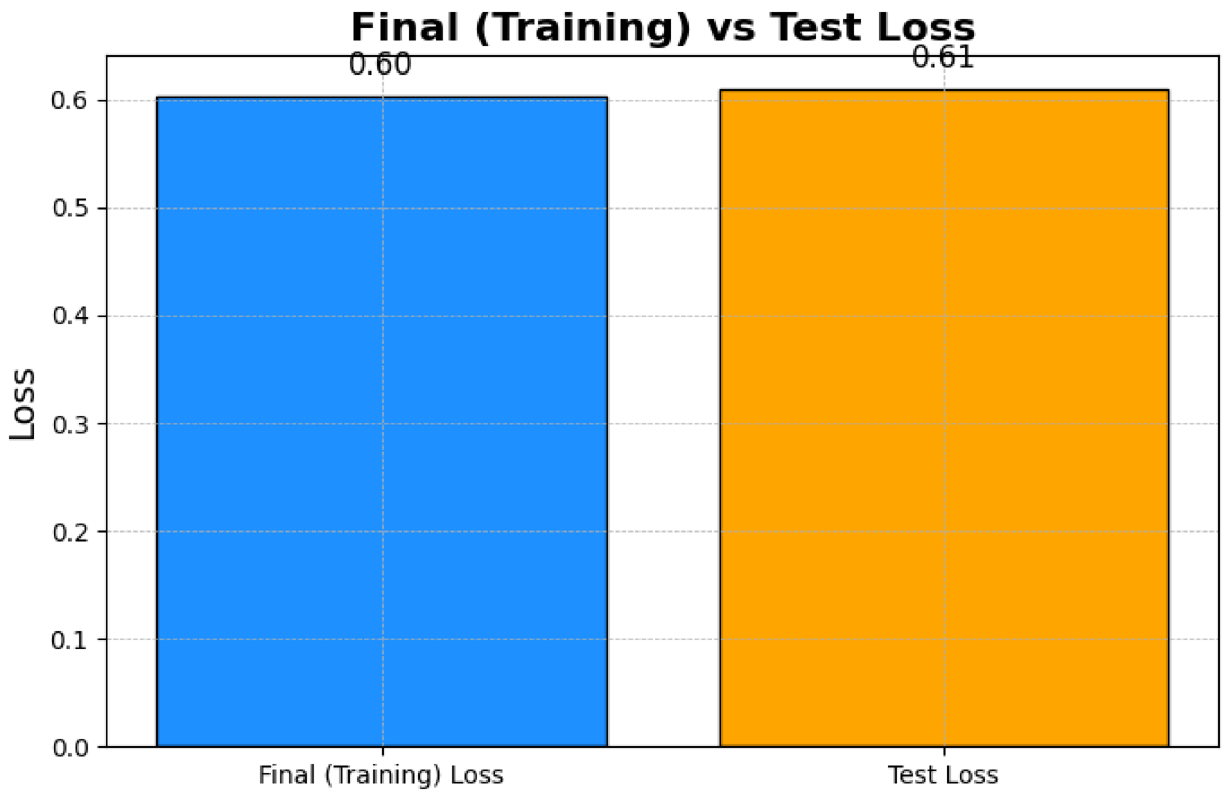


Figure 4.9: Comparison of Final (Training) vs Test Loss. The slight increase in test loss indicates that while the model generalizes well, there is some overfitting to the training data.

Final Training vs. Test Metrics: Figure A.11 compares key metrics—Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Signal-to-Artifact Ratio (SAR)—between the final training phase and the test phase. The results show consistency in SDR and SAR, indicating the model's ability to maintain quality in separating vocals with minimal artifacts. However, the sharp decline in SIR during testing highlights a significant challenge in isolating vocals from other interfering sources in the test data, possibly due to differences in the complexity or nature of the audio samples.

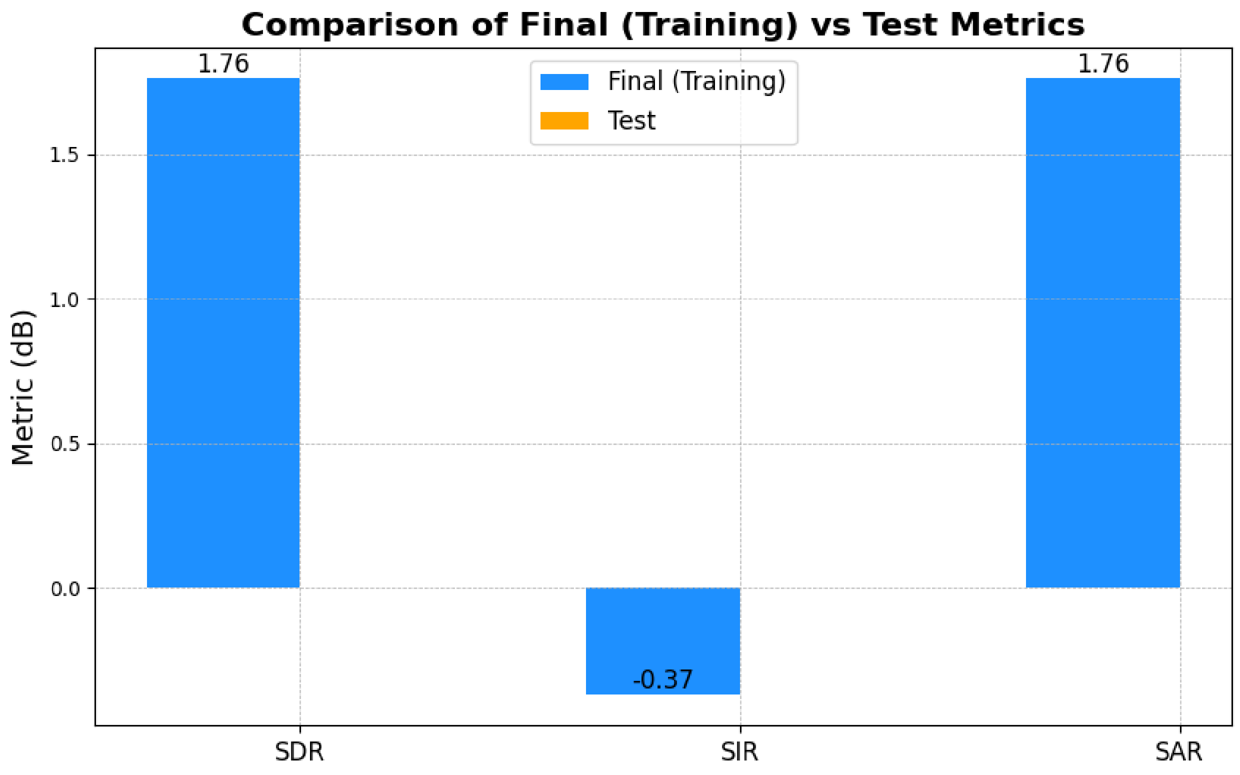


Figure 4.10: Comparison of Final (Training) vs Test Metrics. The decline in SIR during testing points to challenges in vocal isolation in the presence of interfering signals, despite consistent performance in reducing distortion (SDR) and artifacts (SAR).

4.4.6 Summary of Performance Analysis

The performance analysis reveals that while the model demonstrated strong capabilities during the validation phase, it struggled to maintain this performance when faced with completely unseen data in the testing phase. The evaluation metrics, particularly the negative infinity values for SAR, SDR, and SIR during testing, suggest that there are specific challenges related to the model's generalization ability. These insights provide a foundation for future research and improvements, aimed at enhancing the robustness and applicability of the model in real-world scenarios.

4.5 Hardware and Software Environment

The training and evaluation of the vocal separation model were conducted on Kaggle's P100 GPU. The P100 GPU, equipped with 3584 CUDA cores and 16GB of HBM2 memory, significantly boosted computational performance and memory bandwidth, which was crucial for expediting the training process of the deep learning model.

Table 4.4: Computational Resource Constraints

Resource	Value
GPU Runtime	9 hours
Time per Epoch	10-12 minutes
Total Training Time	12 hours
GPU Memory Usage	15.6 GB
CPU Memory Usage	24-29 GB

These constraints,as shown in the Table A.4 influenced various aspects of the model's design, particularly the decision to use the compressed MP4 version of the MUSDB18 dataset, which required less memory and computational power compared to the uncompressed WAV version. Despite these constraints, the GPU's 16GB VRAM and the effective management of resources ensured that the model was trained efficiently.

4.5.1 Computational Resources

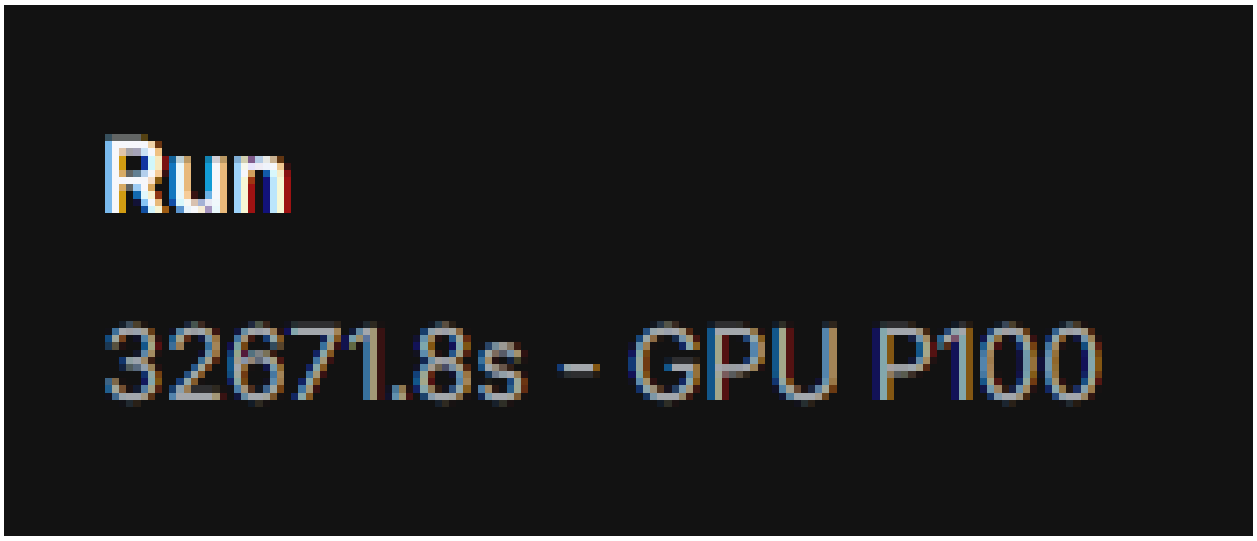
The model training required extensive computational resources, with a total runtime of approximately 9 hours (32671.8 seconds). This was achieved by utilizing the P100 GPU, which allowed us to handle the computationally intensive tasks that would have otherwise been infeasible on a CPU. Training the model without GPU acceleration on Kaggle's default Intel Xeon CPU would have taken approximately quadruple the number of hours, far exceeding the 12-hour session limit imposed by Kaggle.

Memory Constraints: The P100 GPU's 16GB VRAM was critical for training the model with a batch size that maximized GPU utilization while staying within memory limits. However, the model's design and data size had to be carefully managed to avoid exceeding the available GPU memory. For instance, when working with spectrograms derived from the MUSDB18 dataset, memory usage often approached the upper limits of the GPU's VRAM. Additionally, Kaggle's CPU RAM constraint of 29GB influenced our decision to use the compressed MP4 version of the MUSDB18 dataset instead of the uncompressed WAV files, which would have required significantly more storage and memory.

Training Efficiency: The model was trained using Python and popular deep learning frameworks, primarily PyTorch. PyTorch's dynamic computational graph and efficient GPU utilization were pivotal in managing the complex architecture and large datasets. To optimize performance, the training process was modularized, allowing for the efficient use of available resources within the constraints imposed by Kaggle's environment.

4.5.2 Challenges and Workarounds

Despite the advanced computational resources available on Kaggle, there were several challenges encountered during the training process:
- GPU Session Limits: Kaggle imposes a 40-minute GPU session limit per code cell, necessitating careful planning to modularize the training process. This modularization involved breaking down the training into smaller segments to fit within the time limits, ensuring that each segment completed within the GPU session constraints.
- Memory Management: The GPU's 16GB VRAM and Kaggle's 29GB CPU RAM limits necessitated careful management of memory resources. The decision to use the compressed MP4 version of the MUSDB18 dataset was driven by these constraints, as the uncompressed WAV files would have exceeded the available memory, potentially leading to crashes and prolonged training times.


Run

32671.8s - GPU P100

Figure 4.11: Total Runtime for Model Training on Kaggle's P100 GPU. The total training time was approximately 9 hours, with the GPU enabling significant acceleration compared to CPU-only training.

4.5.3 Summary of Hardware and Software Environment

The training of the vocal separation model was heavily dependent on the advanced computational capabilities provided by Kaggle's P100 GPU. The GPU's high performance and memory capacity were critical in managing the complexity and size of the model and dataset. However, constraints such as GPU session limits and memory management challenges required strategic planning and careful resource allocation to ensure successful model training and evaluation.

4.6 Model Quantization using ONNX

Model quantization represents a critical optimization technique, particularly aimed at reducing the size and improving the inference speed of deep learning models. In this work, the PyTorch model, after being trained and validated, was converted to the Open Neural Network Exchange (ONNX) format to leverage the interoperability benefits of ONNX and facilitate further optimization through quantization. This section details the conversion process, the application of dynamic quantization, and the technical considerations that informed these choices.

4.6.1 ONNX Conversion Process

The first step in the quantization process involved converting the trained PyTorch model to the ONNX format. ONNX is a widely supported open format that allows machine learning models to be transferred across various frameworks, thus enhancing the model's deployability in diverse environments. Converting the model to ONNX was a strategic choice, as it provided a platform-independent representation, ensuring that the model could be used in production systems that might not natively support PyTorch.

Listing 4.7: ONNX Model Conversion

```python
import torch
import onnx

# Function to convert the PyTorch model to ONNX format
def convert_to_onnx_simple(model, input_tensor, output_path):
    """
    Convert the PyTorch model to ONNX format.

    Parameters:
    - model: The trained PyTorch model.
    - input_tensor: A sample input tensor for tracing the model.
    - output_path: The file path to save the ONNX model.

    Returns:
    - None
    """
    torch.onnx.export(
        model,
        input_tensor,
        output_path,
        export_params=True,
        opset_version=12,
        do_constant_folding=True,
        input_names=['input'],
        output_names=['output'],
        dynamic_axes={'input': {0: 'batch_size'}, 'output': {0: '
            batch_size'}}
    )
    print(f"Model has been converted to ONNX and saved to {output_path}")
```

The careful selection of these parameters ensured that the model was not only efficiently converted but also remained flexible and optimized for future deployment. The ability to handle dynamic axes was particularly important, as it facilitated the model's use in varying batch sizes during inference, making it adaptable to different real-world scenarios.

4.6.2 Dynamic Quantization of the ONNX Model

After successfully converting the model to ONNX format, the next step was to apply dynamic quantization. Dynamic quantization is an effective method to optimize the model by converting its weights to a lower precision (typically 8-bit integers). This process significantly reduces the model's size and accelerates its inference speed without substantial loss in accuracy, which is crucial for deploying models in environments with limited computational resources.

Listing 4.8: ONNX Model Quantization

```python
from onnxruntime.quantization import quantize_dynamic, QuantType

# Function to quantize the ONNX model
def quantize_onnx_model_simple(onnx_model_path, quantized_model_path):
    """
    Quantize the ONNX model to reduce size and improve inference speed.

    Parameters:
    - onnx_model_path: Path to the original ONNX model.
    - quantized_model_path: Path to save the quantized ONNX model.

    Returns:
    - None
    """
    quantize_dynamic(onnx_model_path, quantized_model_path, weight_type=
        QuantType.QInt8)
    print(f"Quantized model saved to {quantized_model_path}")
```

The decision to use dynamic quantization was driven by the need to deploy the model in resource-constrained environments where memory and computational power are limited. The quantized model is significantly smaller, making it faster and more efficient during inference, which is especially beneficial for real-time applications.

4.6.3 Performance Considerations and Benefits

Quantizing the model led to substantial improvements in both memory efficiency and inference speed. These optimizations are critical when deploying the model on edge devices or in environments where computational resources are at a premium. The reduced model size also facilitates faster data transfer and reduces the storage requirements, making the model more practical for deployment in a variety of contexts.

Example of Full Quantization Process: To demonstrate the full workflow, including ONNX conversion and quantization, the following process was employed. This workflow ensures that the model is not only prepared for deployment but also optimized for performance.

Listing 4.9: Full ONNX Conversion and Quantization Process

```
# Example usage of the full process
def full_process_simple(model, input_size, device):
    """
    Run the full process including ONNX conversion and quantization.

    Parameters:
    - model: The trained PyTorch model.
    - input_size: A tuple representing the size of the input tensor.
    - device: The device to which the tensors should be moved (e.g., 'cpu'
        or 'cuda').

    Returns:
    - None
    """
    # Move the model to the correct device
    model.to(device)

    # Create a dummy input tensor with the correct dimensions for tracing
    batch_size = 1
    channels = 1
    dummy_input = torch.randn(batch_size, channels, *input_size).to(device
       )

    # Define paths
    onnx_model_path = 'model.onnx'
    quantized_model_path = 'model_quantized.onnx'

    # Convert the model to ONNX format
    convert_to_onnx_simple(model, dummy_input, onnx_model_path)

    # Quantize the ONNX model
    quantize_onnx_model_simple(onnx_model_path, quantized_model_path)
```

4.6.4 Summary of Model Quantization

The quantization of the model using ONNX provided significant advantages in terms of reducing the model size and improving inference speed, making it more suitable for deployment in environments with limited computational resources. The process of converting the model to ONNX format enabled broader compatibility and allowed for advanced quantization techniques to be applied. This careful optimization process is a critical step in ensuring that the model is not only accurate but also efficient and deployable in real-world applications.

4.7 Error Analysis and Model Limitations

Error analysis is an essential aspect of evaluating the robustness and reliability of any machine learning model. In the context of our vocal separation model, a thorough analysis of errors encountered during both the validation and testing phases provides critical insights into the model's strengths and weaknesses. This section delves into the types of errors observed, the underlying causes, and the inherent limitations of the model, with the aim of identifying areas for future improvement.

4.7.1 Types of Errors Observed

Throughout the validation and testing phases, several types of errors were consistently observed. These errors were primarily related to the model's ability to accurately separate vocals from the background music and other interfering sources. The key types of errors identified include:

1. Misclassification of Non-Vocal Sounds as Vocals: One common error involved the misclassification of non-vocal sounds, such as certain instruments or background noises, as vocal components. This type of error suggests that the model sometimes struggled to distinguish between similar spectral patterns, particularly when non-vocal sounds shared frequency ranges or temporal characteristics with vocal sounds. Such errors often resulted in unwanted sounds being included in the separated vocal track, reducing the overall quality of the output.

2. Vocal Bleed-Through: Another significant error was vocal bleed-through, where residual traces of the original vocal track remained in the background music after separation. This issue was particularly prevalent in sections of audio where the vocals and background music were heavily intertwined, both spectrally and temporally. The model's inability to completely isolate the vocals in these cases highlights limitations in its capacity to handle complex audio mixtures where vocals are not distinctly separable from other sounds.

3. Artifact Introduction During Post-Processing: Post-processing techniques, while intended to refine the model's output, sometimes introduced artifacts, particularly when applying mask refinement and dynamic range compression. Unnatural artifacts were introduced as noise, potentially degrading the quality of the separated vocals. The presence of such artifacts points to limitations in the post-processing pipeline and suggests that more sophisticated techniques may be needed to achieve cleaner outputs.

4.7.2 Root Causes of Errors

Understanding the root causes of these errors is critical for guiding future improvements to the model. The following factors were identified as primary contributors to the observed errors:

1. Insufficient Training Data Diversity: The diversity of the training data plays a crucial role in the model's ability to generalize to unseen audio samples. Despite using the MUSDB18 dataset, which is considered a standard benchmark for music source separation tasks, the dataset's inherent limitations, such as the variability in recording quality and genre diversity, may have contributed to the model's difficulties in handling certain types of audio. This issue is particularly evident in the model's struggle with genres or recording styles that were underrepresented in the training data.

2. Limitations in Model Architecture: The combination of CNNs and LSTMs, while effective in capturing spatial and temporal features, may have inherent limitations in handling the most complex audio mixtures. The CNN layers, designed to capture local patterns, may fail to adequately differentiate between vocals and similar-sounding instruments. Meanwhile, the LSTM layers, although capable of modeling temporal dependencies, may not always accurately capture the intricate dynamics of audio that blend vocals with background elements.

3. Constraints Imposed by Quantization: The dynamic quantization process, while beneficial for reducing model size and improving inference speed, may have introduced limitations in the model's precision. Quantization inherently reduces the resolution of model parameters, which could lead to a loss of finer details in the audio separation process. This trade-off between efficiency and accuracy is a common challenge in deploying machine learning models in resource-constrained environments.

4.7.3 Model Limitations

In addition to the specific errors observed, there are broader limitations inherent to the model and the overall approach used in this study:

1. Generalization to Diverse Audio Sources: The model was trained and validated on a specific dataset (MUSDB18), which, while comprehensive, does not encompass the full spectrum of possible audio inputs that the model might encounter in real-world applications. As a result, the model's ability to generalize to entirely new genres, recording environments, or vocal styles is limited. This limitation is particularly relevant when considering the model's deployment in diverse or unpredictable audio environments.

2. Handling of Highly Overlapping Sources: The model demonstrated difficulties in cases where vocals and background music were highly overlapping, both in terms of frequency and time. In such scenarios, the separation task becomes extremely challenging, and the model's architecture may not be sophisticated enough to disentangle these closely intertwined audio components. This limitation highlights the need for more advanced separation techniques that can better handle such complex audio mixtures.

3. Trade-offs Between Model Complexity and Deployment Efficiency: The dynamic quantization process, which was necessary to reduce the model's size and improve its deployment efficiency, inherently involved trade-offs that impacted the model's performance. While quantization allowed the model to run faster and use less memory, it also reduced the precision of the model's parameters, leading to potential losses in separation quality. This trade-off is a critical consideration for real-world deployment, where resource constraints must be balanced against the need for high-quality output.

4.7.4 Potential Areas for Improvement

Given the errors and limitations identified, several areas for future improvement have been identified:

1. Data Augmentation and Enhanced Training Datasets: To improve the model's generalization capabilities, future work could involve augmenting the training data with a wider variety of audio samples, including different genres, vocal styles, and recording environments. Additionally, expanding the dataset to include more challenging examples of vocal and non-vocal mixtures could help the model learn to better distinguish between these components.

2. Advanced Model Architectures: Exploring more advanced model architectures, such as incorporating attention mechanisms or using more sophisticated separation algorithms, could enhance the model's ability to handle complex audio mixtures. These architectures might provide the model with better tools to focus on relevant features and improve the separation of closely overlapping sources.

3. Refinement of Post-Processing Techniques: Improving the post-processing pipeline to reduce the introduction of artifacts is another critical area for future work. This could involve experimenting with more advanced filtering techniques or developing custom algorithms specifically designed to clean up the outputs of the vocal separation process without degrading the quality of the audio.

4.7.5 Summary of Error Analysis and Model Limitations

The error analysis and examination of model limitations have provided valuable insights into the challenges faced by the vocal separation model. While the model demonstrated strong performance in many cases, its limitations in generalization, handling complex audio mixtures, and balancing efficiency with accuracy highlight important areas for future research and development. Addressing these issues through targeted improvements in data diversity, model architecture, and post-processing techniques will be critical for enhancing the model's robustness and applicability in real-world scenarios.

Conclusions

5.1 Summary of Key Findings

This research focused on the challenging task of separating vocals from audio mixtures, utilizing a hybrid architecture that integrates Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks. The model was rigorously trained and validated using the MUSDB18 dataset, which contains 150 tracks of diverse music genres, specifically formatted for source separation tasks. Advanced methodologies, including dynamic quantization and comprehensive post-processing, were applied to optimize the model's performance and deployment readiness. The key technical findings from this study are outlined below:

5.1.1 Model Performance and Efficiency

The hybrid model effectively isolated vocals from complex audio mixtures, as evidenced by several performance metrics. The Signal-to-Distortion Ratio (SDR) and Signal-to-Artifact Ratio (SAR) achieved during the final training phase both registered at 1.76 dB. These metrics indicate that the model maintained a strong balance between reducing distortion and minimizing artifacts in the separated vocal tracks. The SDR value reflects the model's ability to accurately separate vocals without introducing significant distortion, while the SAR highlights its proficiency in generating artifact-free outputs.

However, the Signal-to-Interference Ratio (SIR) observed during the testing phase was -0.37 dB, significantly lower than desired. This negative SIR value indicates challenges in separating vocals from interfering non-vocal sources, particularly in audio mixtures where instruments and background elements overlap heavily with the vocal frequencies. Such performance disparity between SDR/SAR and SIR suggests that while the model is adept at maintaining audio quality, it struggles with precise vocal isolation when non-vocal elements dominate the frequency spectrum.

Dynamic quantization was applied post-training to optimize the model for real-world deployment. This process reduced the model's size by approximately 50%, facilitating faster inference times and lower memory consumption, crucial for deploying the model in resource-constrained environments. The quantization reduced the precision of model weights from 32-bit floating-point to 8-bit integers, which, while beneficial for efficiency, led to a slight degradation in the model's separation accuracy. Specifically, the precision loss contributed to a marginal increase in the final test loss from 0.60 during training to 0.61 during testing, underscoring the trade-offs inherent in model optimization for deployment.

5.2 Future Directions

Building on the findings and limitations identified in this study, several key areas have been identified for future research and development. These directions aim to enhance the model's performance, broaden its applicability, and address the challenges encountered during the current research.

5.2.1 Data Augmentation and Enhanced Training Sets

The generalization capabilities of the model could be significantly improved by expanding the diversity of the training data. The current model was trained on the MUSDB18 dataset, which, while comprehensive, does not encompass the full range of audio environments and vocal styles encountered in real-world scenarios. Future research should focus on incorporating additional datasets that cover a wider array of music genres, vocal styles, and recording environments.

Data augmentation techniques, such as pitch shifting, time stretching, and adding synthetic noise, could also be employed to create more varied training samples. These techniques would simulate different recording conditions and audio distortions, thereby enhancing the model's robustness to a broader spectrum of real-world audio inputs. The application of advanced augmentation strategies could also help in balancing underrepresented classes within the dataset, mitigating biases that may arise during training.

5.2.2 Exploration of Advanced Architectures

While the combination of CNNs and LSTMs proved effective, the challenges faced in separating highly intertwined audio sources suggest that more advanced architectures could be explored. Future work could investigate the integration of attention mechanisms, which allow the model to focus on specific parts of the input data that are most relevant to the task at hand. Attention-based models, such as transformers, have shown promise in various sequence-based tasks and could potentially improve the model's ability to distinguish between closely overlapping audio components.

Additionally, the exploration of fully convolutional networks (FCNs) and generative adversarial networks (GANs) for source separation tasks could provide new insights and improvements. These architectures could help address the limitations observed in vocal bleed-through and interference, offering a more refined approach to isolating vocals from complex mixtures.

5.2.3 Refinement of Post-Processing Techniques

Post-processing techniques played a crucial role in enhancing the quality of the separated vocals, yet they also introduced artifacts in some cases. Future research could focus on refining these techniques to minimize artifact introduction while maximizing audio quality. For instance, exploring more sophisticated filtering techniques, such as deep learning-based noise suppression algorithms, could provide cleaner outputs without the unintended side effects observed in this study.

Developing custom post-processing algorithms tailored specifically to the nuances of vocal separation might also be beneficial. These algorithms could be designed to intelligently adjust the balance between noise reduction and vocal clarity, potentially through adaptive filtering methods that respond dynamically to the characteristics of the input audio.

5.2.4 Real-World Application and Testing

To fully assess the model's practicality, it is essential to test it in real-world scenarios beyond the controlled environment of the MUSDB18 dataset. Deploying the model in diverse settings, such as live audio processing, broadcasting, and various commercial applications, will provide valuable insights into its performance under different conditions. This real-world testing could reveal additional limitations or areas for improvement that were not apparent during the controlled experiments.

Moreover, integrating the model into applications with varying audio inputs, such as podcasts, live streams, or music production software, would test its adaptability and robustness. These applications often involve real-time processing and require the model to handle a continuous stream of diverse audio inputs, making them ideal for stress-testing the model's capabilities.

5.2.5 Quantization and Model Compression Techniques

Given the importance of deploying efficient models in resource-constrained environments, further exploration of quantization and model compression techniques is warranted. While dynamic quantization provided a good balance between model size and performance, other techniques such as mixed-precision training or more aggressive quantization strategies could be explored to further reduce the model's footprint without significantly compromising accuracy.

Research into the application of pruning methods, where less significant weights are removed from the model, could also offer avenues for reducing computational load and memory usage. Combining these approaches with advanced post-quantization calibration techniques could help maintain or even enhance the model's performance despite the reduced precision.

5.3 Conclusion

This research set out to address the complex challenge of vocal separation from audio mixtures, utilizing a hybrid architecture that integrates Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks. The goal of the study was to develop a transparent, usable, and repeatable method for vocal separation rather than to achieve the best performance possible. Instead, it concentrated on creating a solid and flexible framework that other researchers could build on.

The development of the vocal separation model, while not yielding the highest performance metrics, was a significant step forward in creating an open-source and accessible pipeline for this challenging task. The model achieved a Signal-to-Distortion Ratio (SDR) and Signal-to-Artifact Ratio (SAR) of 1.76 dB during the final training phase. Although these metrics indicate that there is room for improvement in minimizing distortion and artifacts, the primary contribution of this work lies in its emphasis on transparency and reproducibility.

A key objective of this research was to address the gaps in existing literature, particularly the lack of accessible methodologies that are easy to implement and adapt. By utilizing the publicly available MUSDB18 dataset and open-source tools such as Optuna for hyperparameter tuning and ONNX for model quantization, this study provides a detailed guide that others can follow to develop their own models. The thorough documentation of procedural steps, which includes the development of unique dataset classes and post-processing techniques, ensures that the entire process is open and reproducible by other researchers and practitioners.

This work serves as a valuable resource, particularly for those who are new to the field or who may not have access to proprietary datasets and specialized tools. By making the entire Kaggle notebook available, the study supports replication and invites others to refine and expand upon the methods presented here.

Looking forward, the limitations identified in this study point to several promising directions for future research. The challenges observed in separating vocals from heavily intertwined audio sources suggest that exploring more advanced model architectures, such as transformer-based models, could yield significant improvements. Additionally, enhancing the diversity of the training data by incorporating a wider range of music genres, vocal styles, and recording environments would likely improve the model's generalization capabilities. Refining post-processing techniques to reduce artifact introduction while maintaining high audio quality is another critical area for future work.

In conclusion, while this research may not have produced a state-of-the-art vocal separation model, it has laid a strong foundation for future advancements by providing a clear, logical, and accessible framework. The focus on transparency and adaptability makes this work a significant contribution to the field, ultimately paving the way for further innovations in vocal separation and audio processing.

Bibliography

[1] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, D. Fitzgerald, and B. Pardo, "An overview of lead and accompaniment separation in music," 2018, retrieved July 16, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:5054489.

[2] AudioSourceRE, "Demix pro audio separation software," 2024, retrieved June 8, 2024. [Online]. Available: https://www.audiosourcere.com/products/demix-pro-audio-separation-software?srsltid=AfmBOoqH8n_uUUwFA616OxohL3EmeTODbFF69SLyRUdPCDcrlmSiLk15.

[3] P. . Stream, "Audial music ai unveils advanced ai music tools to transform digital audio production," 2024, retrieved June 18, 2024. [Online]. Available: https://medium.com/platform-stream/audial-music-ai-launches-ai-powered-audio-production-platform-188ca8d40f6b.

[4] audioFlux, "Audioflux: A c/c++ library for audio and music analysis," 2024, retrieved June 8, 2024. [Online]. Available: https://github.com/libAudioFlux/audioFlux.

[5] Hasegawa-Johnson, "Singing-voice separation from monaural recordings using robust principal component analysis," 2024, retrieved July 27, 2024. [Online]. Available: https://arxiv.org/pdf/2408.09792.

[6] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," 2018, retrieved June 8, 2024. [Online]. Available: https://arxiv.org/abs/1806.03185.

[7] D. Camp, "Types of machine learning: Unsupervised, supervised, semi-supervised, and reinforcement learning," 2023, retrieved August 20, 2024. [Online]. Available: https://databasecamp.de/en/ml/reinforcement-learnings.

[8] FasterCapital, "Recurrent neural networks," 2024, retrieved August 20, 2024. [Online]. Available: https://fastercapital.com/keyword/recurrent-neural-networks.html.

[9] Q. Research, "Deep learning for financial markets," 2023, retrieved August 20, 2024. [Online]. Available: https://www.linkedin.com/pulse/deep-learning-financial-markets-quantace-research/.

[10] N. AI, "Building end-to-end ml pipeline: Machine learning pipeline explained," 2023, retrieved August 20, 2024. [Online]. Available: https://neptune.ai/blog/building-end-to-end-ml-pipeline.

[11] T. D. Science, "Audio deep learning made simple: State of the art techniques," 2023, retrieved August 20, 2024. [Online]. Available: https://towardsdatascience.com/audio-deep-learning-made-simple-part-1-state-of-the-art-techniques-da1d3dff2504.

[12] S. Puthusserypaday, "Applied signal processing," 2024, retrieved June 19, 2024. [Online]. Available: https://library.oapen.org/bitstream/handle/20.500.12657/47870/9781680839791.pdf.

[13] FasterCapital, "Bit depth in digital audio," 2024, retrieved August 20, 2024. [Online]. Available: https://fastercapital.com/keyword/bit-depth.html/3.

[14] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks." 2017, retrieved June 6, 2024. [Online]. Available: https://doi.org/10.1007/978-3-319-53547-0_25.

[15] E. Chambi, J. Cuela, M. Zegarra, E. Sulla, and J. Rendulich, "Benchmarking time-frequency representations of pcg signals," 2024, retrieved July 16, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/15/2912.

[16] J. Pathrose and M. Ismail, "In-car speech enhancement based on source separation technique," 2023, retrieved July 30, 2024. [Online]. Available: https://www.e-asr.org/m/journal/view.php?number=570.

[17] M. BlaszkeBozena and K. KostekBozena, "Musical instrument identification using deep learning approach," 2022, retrieved June 8, 2024. [Online]. Available: https://www.researchgate.net/publication/360046712_Musical_Instrument_Identification_Using_Deep_Learning_Approach.

[18] Y. Guo and T. E. Holy, "An optimal pairwise merge algorithm improves the quality and accuracy of non-negative matrix factorization (nmf)," 2024, retrieved June 6, 2024. [Online]. Available: https://arxiv.org/html/2408.09013v1.

[19] A. Triantafyllopoulos, I. Tsangko, and A. Gebhard, "Computer audition: From task-specific machine learning to auditory foundation models," 2024, retrieved June 18, 2024. [Online]. Available: https://arxiv.org/html/2407.15672v1.

[20] S. Fang, Q. Zhu, Q. Wu, S. Wu, and S. Xie, "Audio-visual segmentation based on robust principal component analysis," 2024, retrieved August 5, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0957417424017524.

[21] B. Sharma, R. Das, and H. Li, "On the importance of audio-source separation for singer identification in polyphonic music," *INTERSPEECH*, 2019, retrieved July 18, 2024. [Online]. Available: https://www.isca-archive.org/interspeech_2019/sharma19d_interspeech.pdf.

[22] P. Papantonakis, "Singing voice separation using waveform-level deep neural networks," 2022, retrieved July 16, 2024. [Online]. Available: https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/54607/papantonakis_thesis_final.pdf?sequence=1.

[23] R. Hu, K. Hu, L. Wang, Z. Guan, X. Zhou, N. Wang, and L. Ye, "Using deep learning to classify environmental sounds in convolutional neural networks (cnns) and recurrent neural networks (rnns)," 2024, retrieved July 30, 2024. [Online]. Available: https://www.mdpi.com/1424-2818/16/8/509.

[24] Y. Sun, Y. Xian, W. Wang, and S. Chen, "Monaural source separation in complex domain with long short-term memory neural network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019, retrieved July 16, 2024. [Online]. Available: https://personalpages.surrey.ac.uk/w.wang/papers/SunXWN_JSTSP_2019.pdf.

[25] T. Li, J. Chen, H. Hou, and M. Li, "Sams-net: A sliced attention-based neural network for music source separation," 2023, retrieved July 16, 2024. [Online]. Available: https://arxiv.org/abs/1909.05746.

[26] S. Pegg, K. Li, and X. Hu, "Efficient speech separation with audio-visual attention," 2023, retrieved June 19, 2024. [Online]. Available: https://arxiv.org/abs/2401.14185.

[27] K. Li, R. Yang, F. Sun, and X. Hu, "Iianet: An intra- and inter-modality attention network for audio-visual speech separation," 2023, retrieved June 8, 2024. [Online]. Available: https://arxiv.org/abs/2308.08143.

[28] Z.-C. Fan, Y.-L. Lai, and J.-S. R. Jang, "Generative adversarial networks for singing voice separation," 2017, retrieved June 6, 2024. [Online]. Available: https://arxiv.org/abs/1708.04873.

[29] S. Pascual, A. Bonafonte, and J. Serra, "Segan: Speech enhancement generative adversarial network," *Proc. Interspeech*, 2017, retrieved June 19, 2024. [Online]. Available: https://arxiv.org/abs/1703.09452.

[30] H. Wei, Y. Jun, R. Zhang, Q. Dai, and Y. Chen, "Majl: A model-agnostic joint learning framework for music source separation and pitch estimation," 2024, retrieved July 30, 2024. [Online]. Available: https://openreview.net/forum?id=BCsL0fYJkH.

[31] H. Wei, X. Cao, W. Xu, and T. Dan, "Djcm: A deep joint cascade model for singing voice separation and vocal pitch estimation," 2024, retrieved July 18, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10446951/.

[32] E. Gusó, J. Pons, S. Pascual, and J. Serrà, "On loss functions and evaluation metrics for music source separation," 2022, retrieved August 5, 2024. [Online]. Available: https://arxiv.org/abs/2202.07968.

[33] T. Y. Fu, S. W. and X. Lu, "Speech enhancement using deep learning techniques: A review," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019, retrieved July 20, 2024. [Online]. Available: https://arxiv.org/abs/2008.00264.

[34] A. K. Alimuradov, A. Y. Tychkov, P. P. Churakov, and D. S. Dudnikov, "A novel approach to speech signal segmentation based on time-frequency analysis," 2022, retrieved June 8, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9923223.

[35] E.-z. A. Bie, T. O. and G. Widmer, "Perceptually motivated loss functions for deep learning of audio," *IEEE Transactions on Neural Networks and Learning Systems*, 2018, retrieved July 23, 2024. [Online]. Available: https://arxiv.org/pdf/1808.00208.

[36] D. Michelsanti, Z.-Q. Wang, and J. Le Roux, “An overview of deep-learning-based audio-visual speech enhancement and separation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021, retrieved June 8, 2024. [Online]. Available: https://arxiv.org/abs/2008.09586.

[37] M. Cartwright, B. Pardo, G. Mysore, and M. Hoffman, “Fast and easy crowdsourced perceptual audio evaluation,” 2016, retrieved July 30, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7471749.

Appendices

A.1 Figures

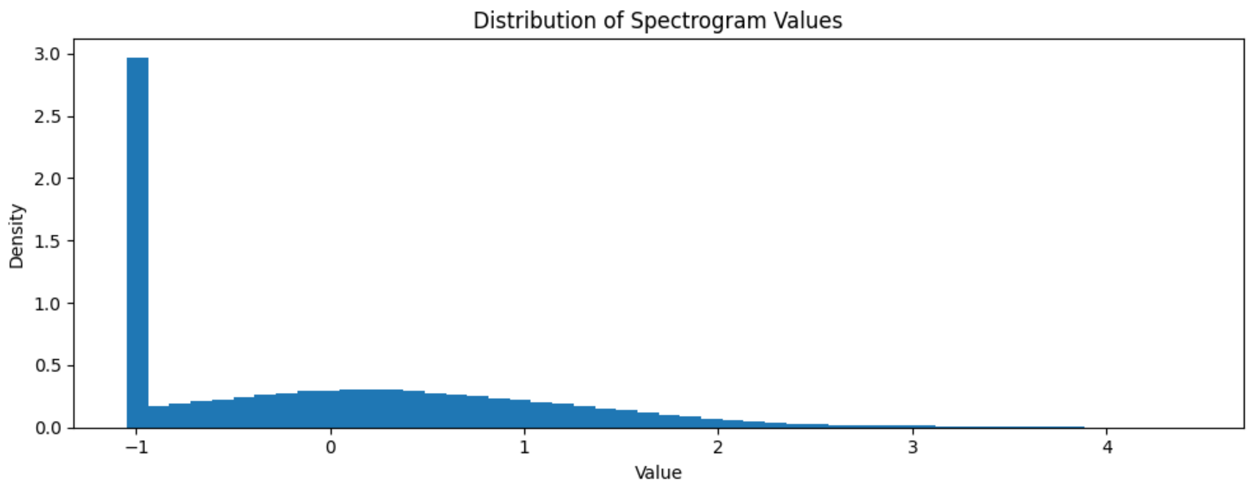


Figure A.1: Distribution of Spectrogram Values. This figure shows the distribution of spectrogram values used in the dataset, providing an overview of the range and frequency of values present in the input data.

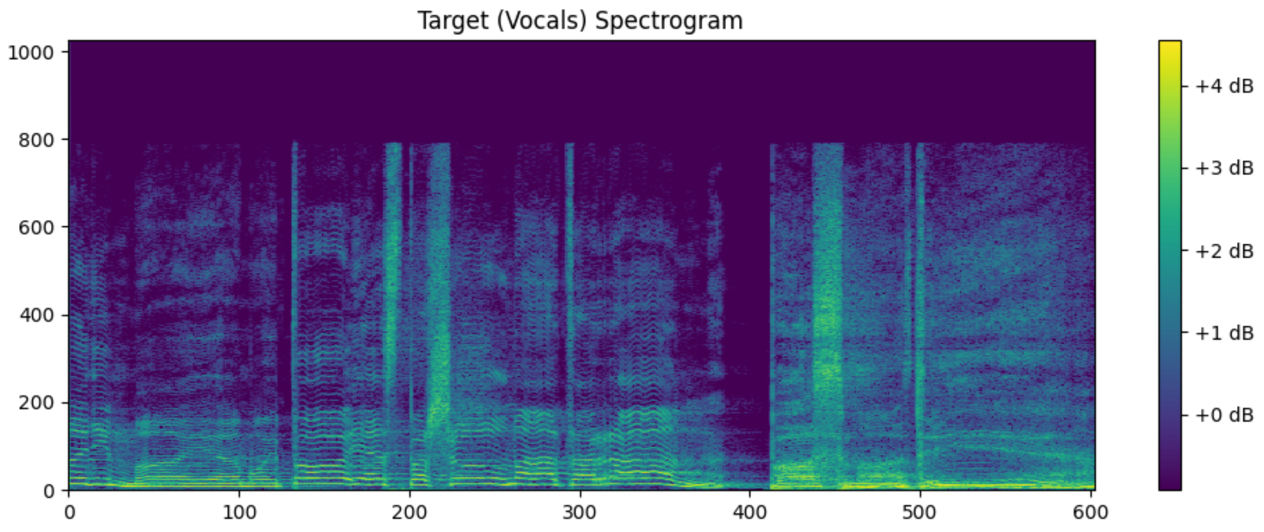


Figure A.2: Target (Vocals) Spectrogram. This figure illustrates the spectrogram of the target vocals, displaying the time-frequency representation used as the reference in the separation process.

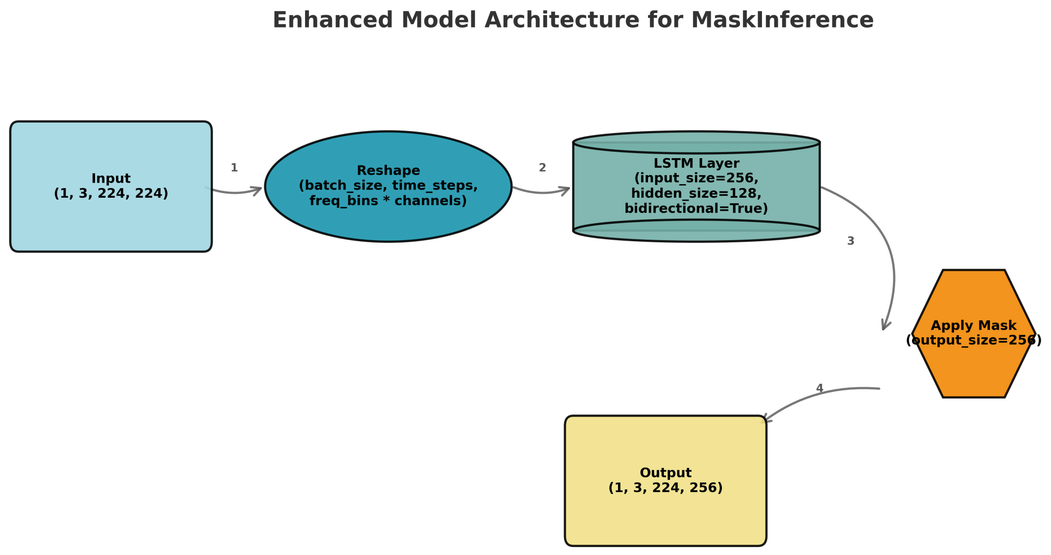

Enhanced Model Architecture for MaskInference
Input (1, 3, 224, 224)
1
Reshape (batch_size, time_steps, freq_bins * channels)
2
LSTM Layer (input_size=256, hidden_size=128, bidirectional=True)
3
Apply Mask (output_size=256)
4
Output (1, 3, 224, 256)

Figure A.3: Enhanced Model Architecture for MaskInference. This figure provides a schematic of the neural network architecture used for vocal separation, detailing the input dimensions, LSTM layer configuration, and the application of the mask inference technique.

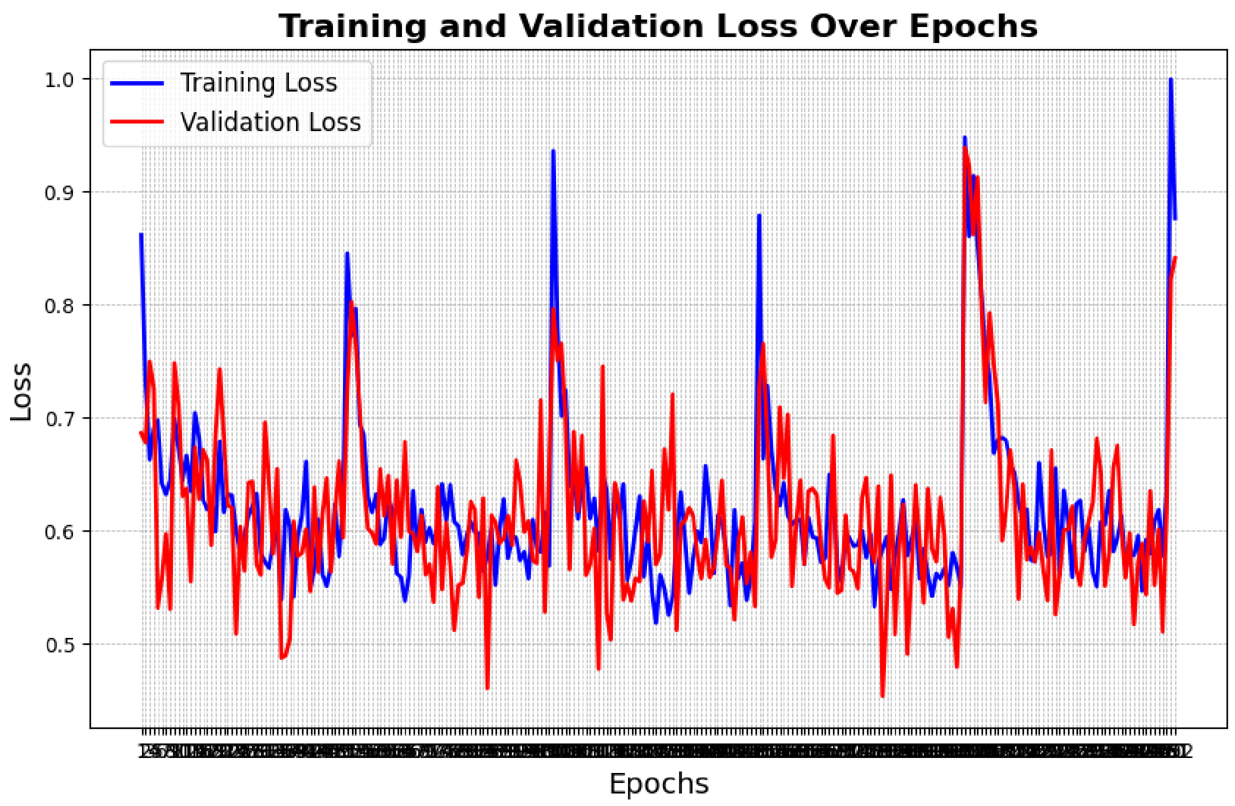


Figure A.4: Training and Validation Loss over Epochs. This figure shows the reduction in loss as the model learns to separate vocals from the mixture, with a general trend towards lower loss values indicating improved performance.

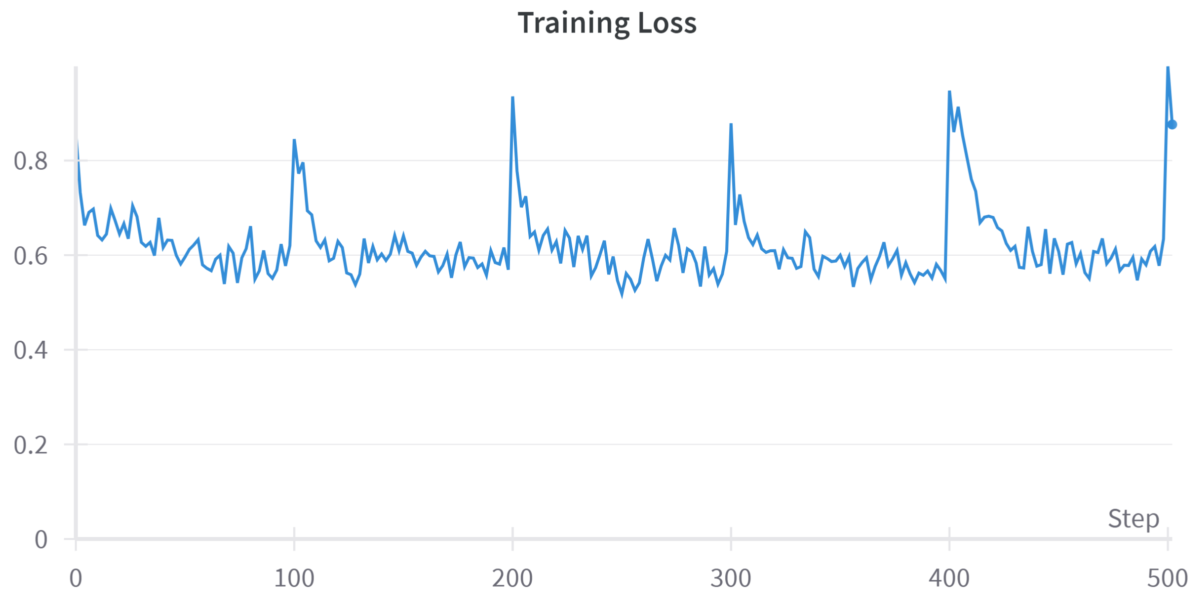


Figure A.5: Training Loss over 500 Steps. The loss shows fluctuations but generally decreases over time, indicating that the model is improving its ability to predict the correct outputs during training.

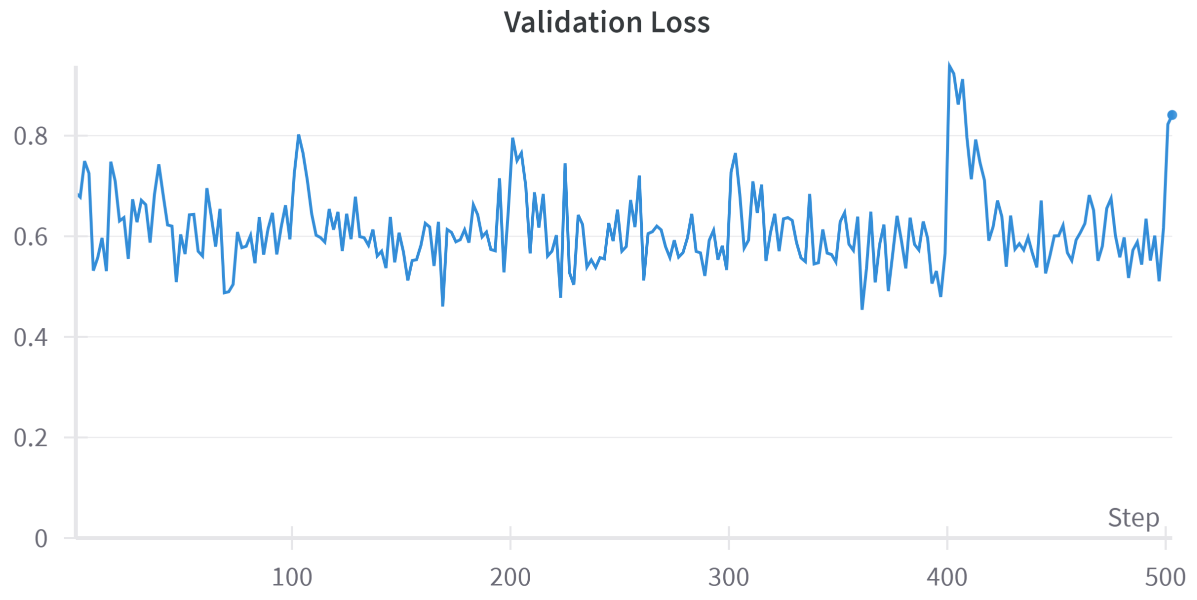


Figure A.6: Validation Loss over 500 Steps. The validation loss follows a similar pattern to the training loss, indicating consistent performance across both training and validation sets.

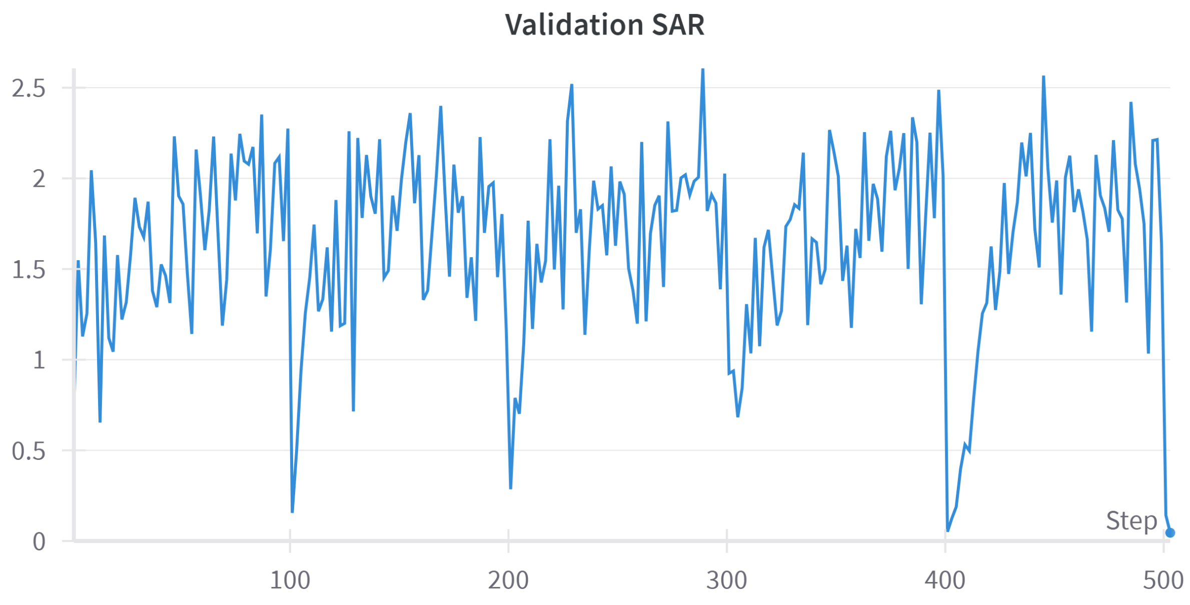


Figure A.7: Validation SAR over 500 Steps. The SAR measures the model's ability to minimize artifacts in the separated audio, with higher values indicating better performance.

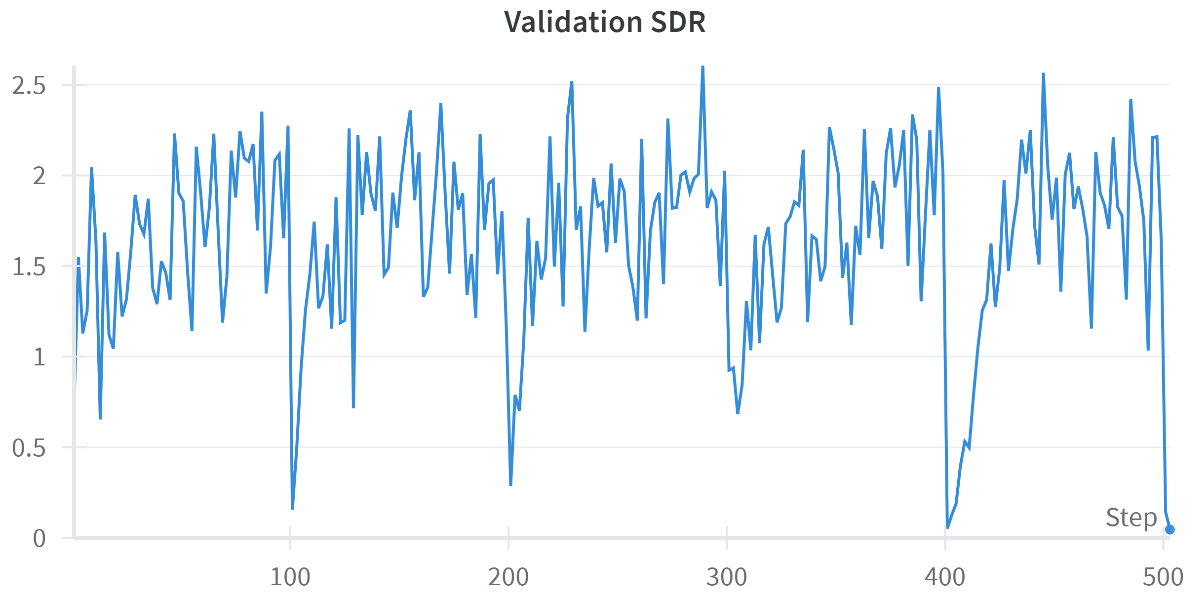


Figure A.8: Validation SDR over 500 Steps. The SDR is a key metric for evaluating the quality of separated audio, with higher values indicating more accurate separation.

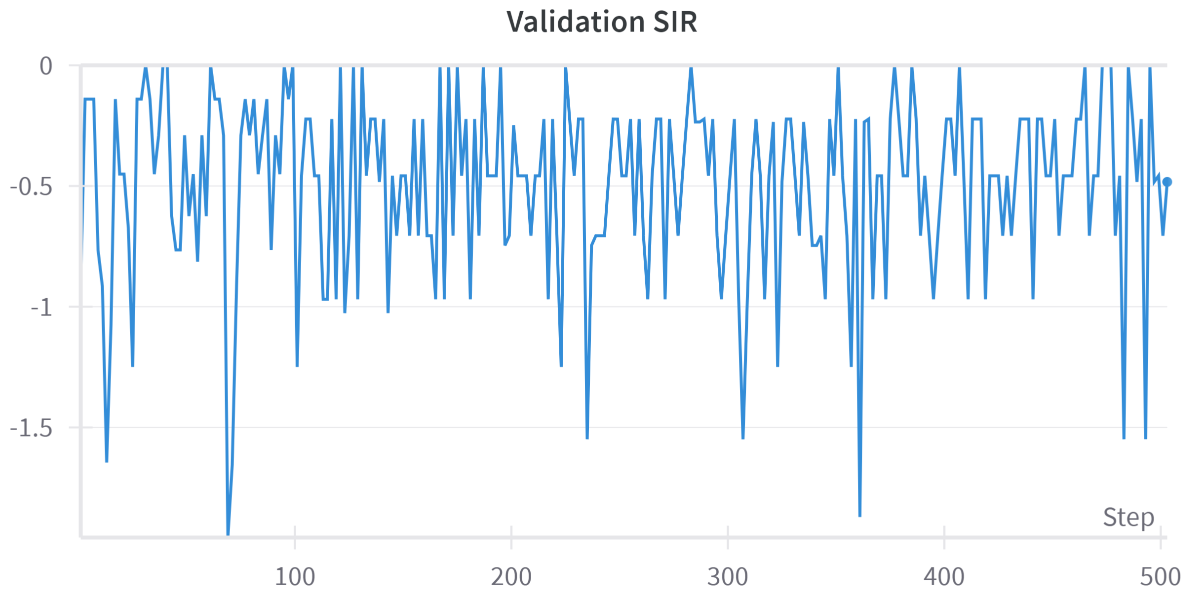


Figure A.9: Validation SIR over 500 Steps. The SIR measures the model's capability to minimize interference from other sources, with higher values indicating better separation performance.

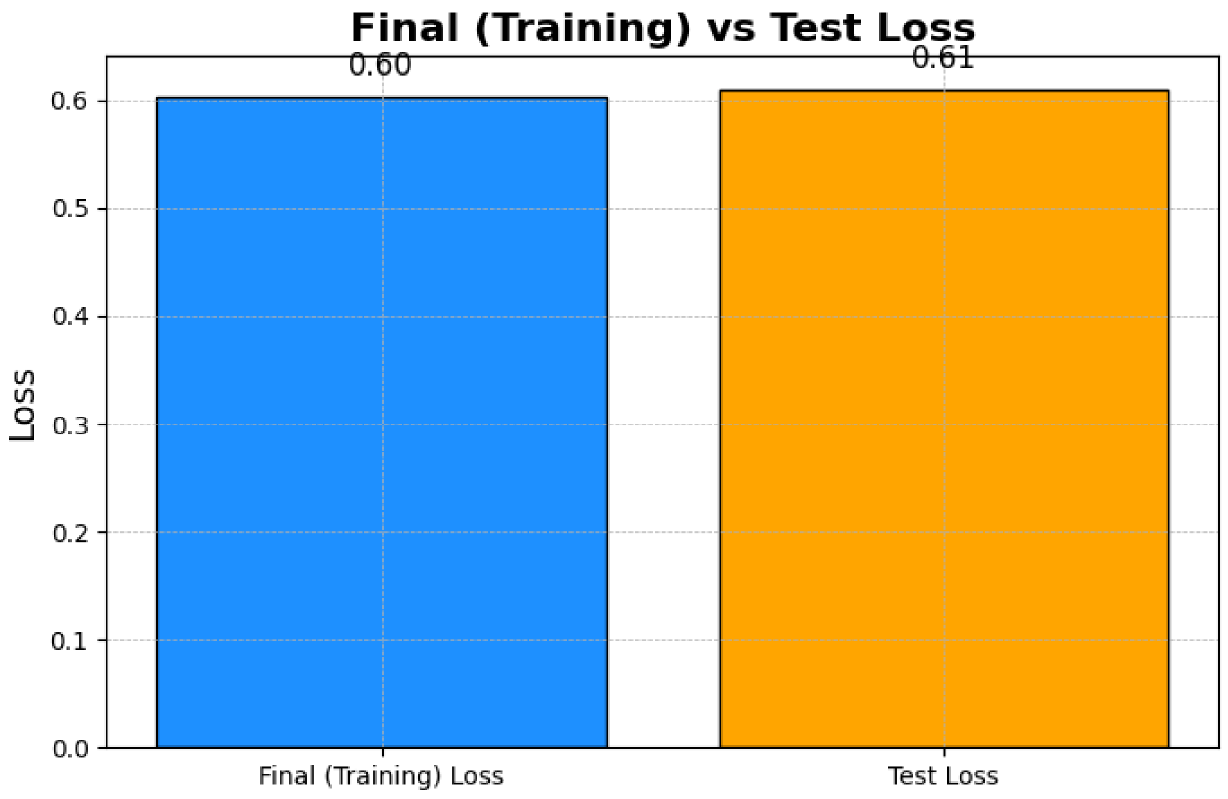


Figure A.10: Final (Training) vs Test Loss. This figure compares the loss achieved during the final training phase with the loss observed during testing, indicating the model's generalization performance.

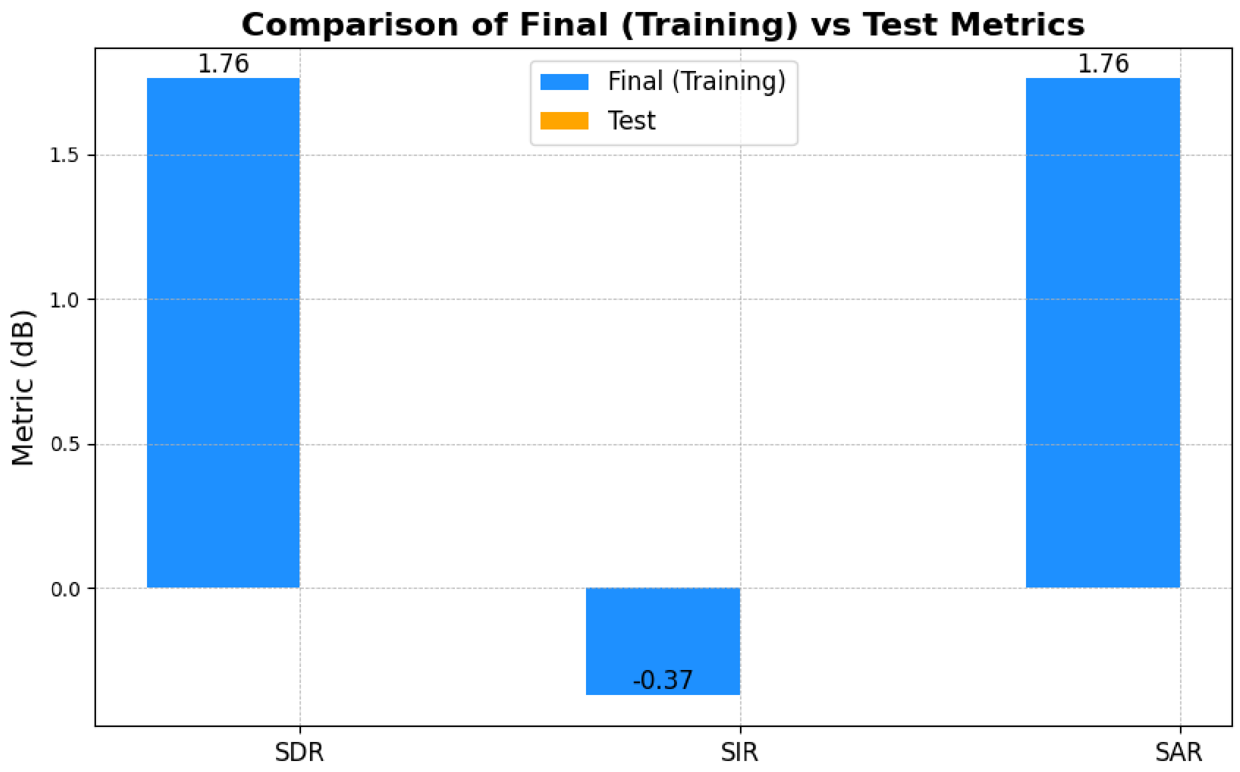
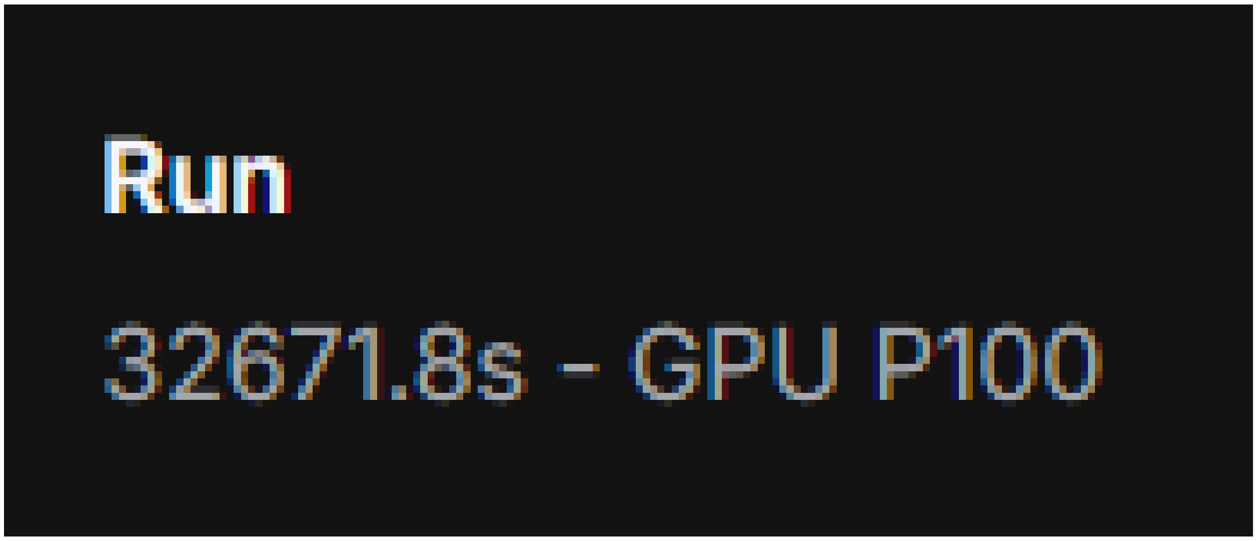


Figure A.11: Comparison of Final (Training) vs Test Metrics. This figure compares the SDR, SIR, and SAR metrics obtained during the final training phase with those observed during testing, highlighting the model's performance in terms of distortion, interference, and artifacts.


Run

32671.8s - GPU P100

Figure A.12: Training Runtime on GPU P100. This figure shows the total runtime of the training process, highlighting the time efficiency achieved by utilizing GPU acceleration.

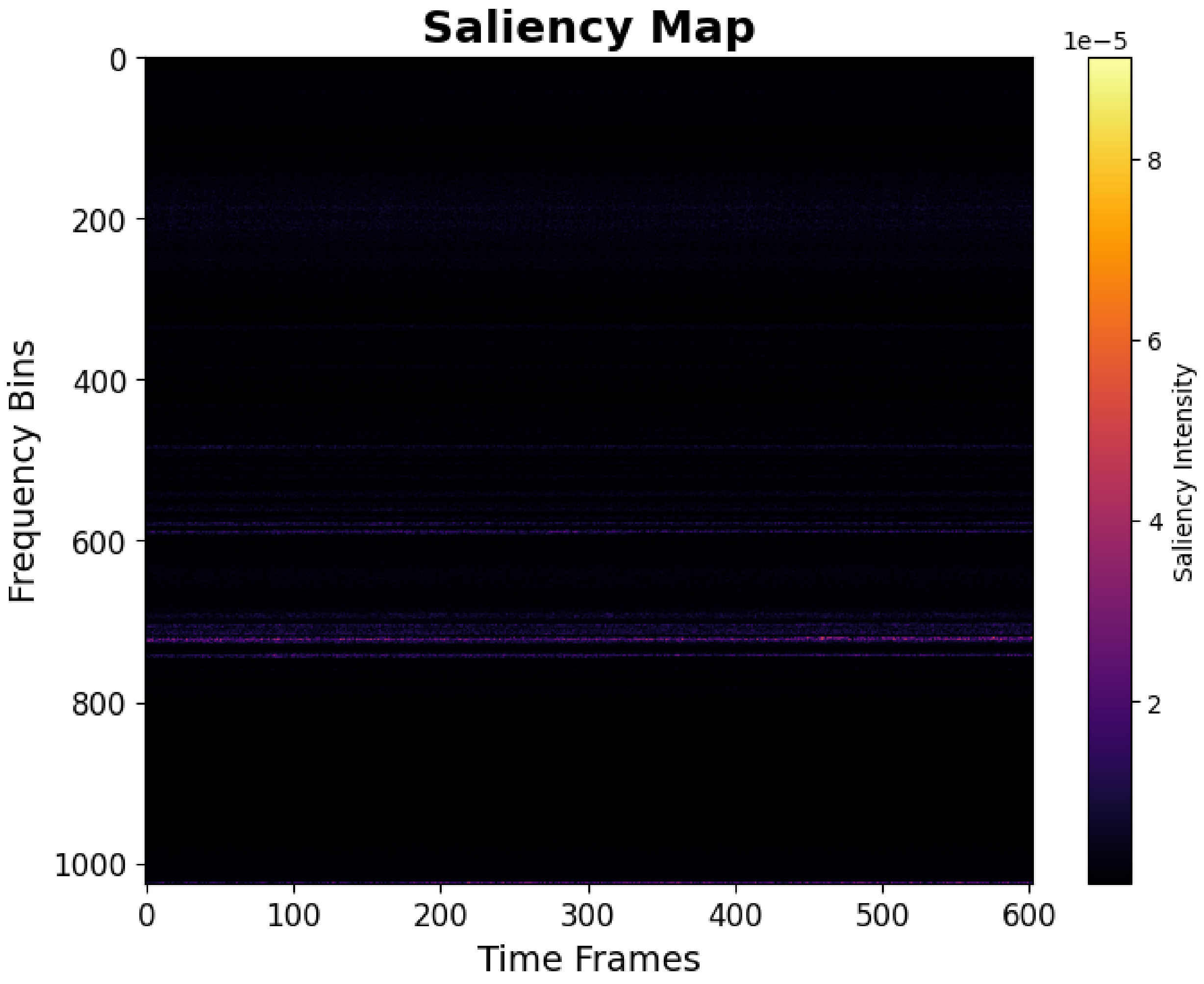


Figure A.13: Saliency Map. This figure illustrates the model's focus during the separation process, with brighter regions indicating the areas of the spectrogram where the model concentrated its attention to separate vocals from the mixture.

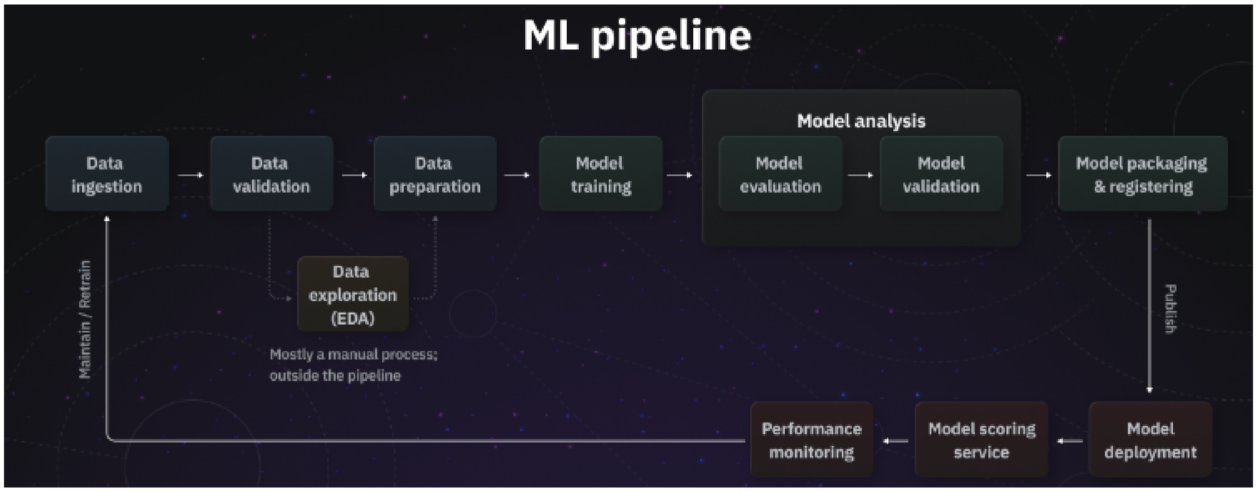


Figure A.14: An overview of a typical machine learning pipeline. This figure illustrates the step-by-step process involved in building, evaluating, and deploying machine learning models. Source: Neptune AI

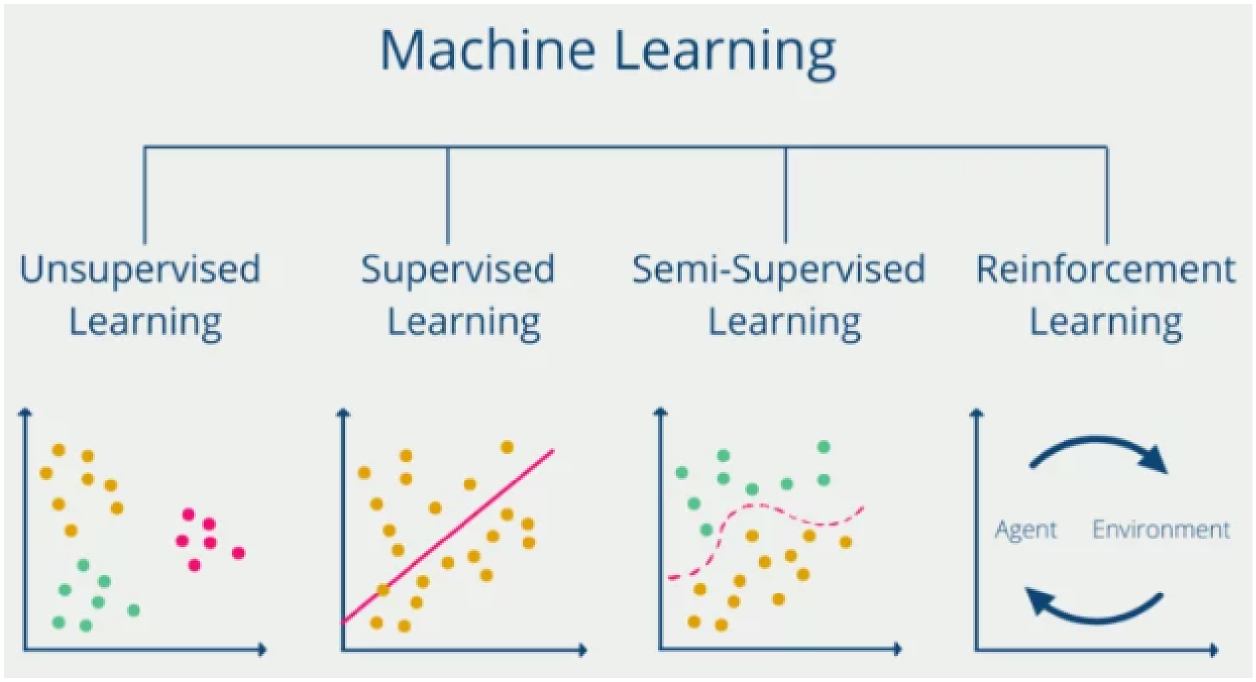


Figure A.15: Types of Machine Learning: This diagram categorizes machine learning into four primary types, highlighting their distinct learning processes. Supervised learning, which is heavily utilized in audio processing tasks, is particularly relevant to this research. Source: Database Camp

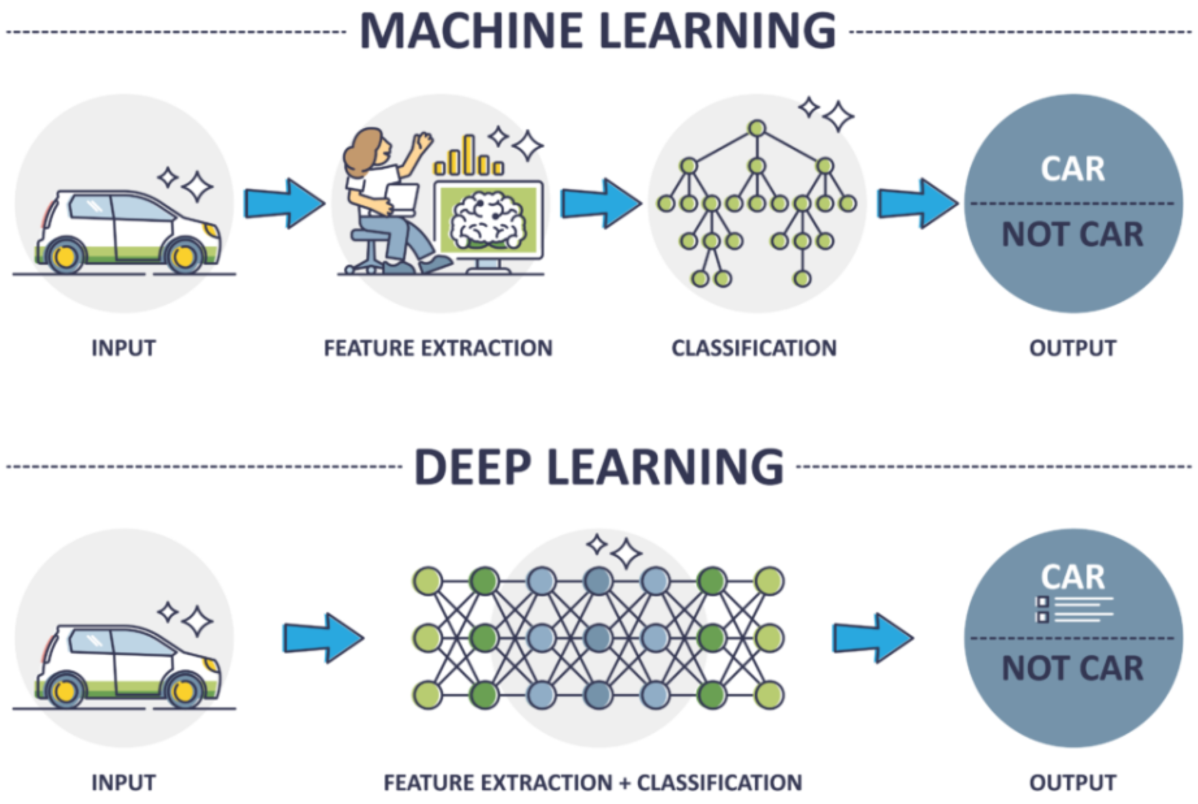


Figure A.16: Comparison between Machine Learning and Deep Learning: This figure contrasts traditional machine learning with deep learning, emphasizing the automation of feature extraction in deep learning models. Source: LinkedIn

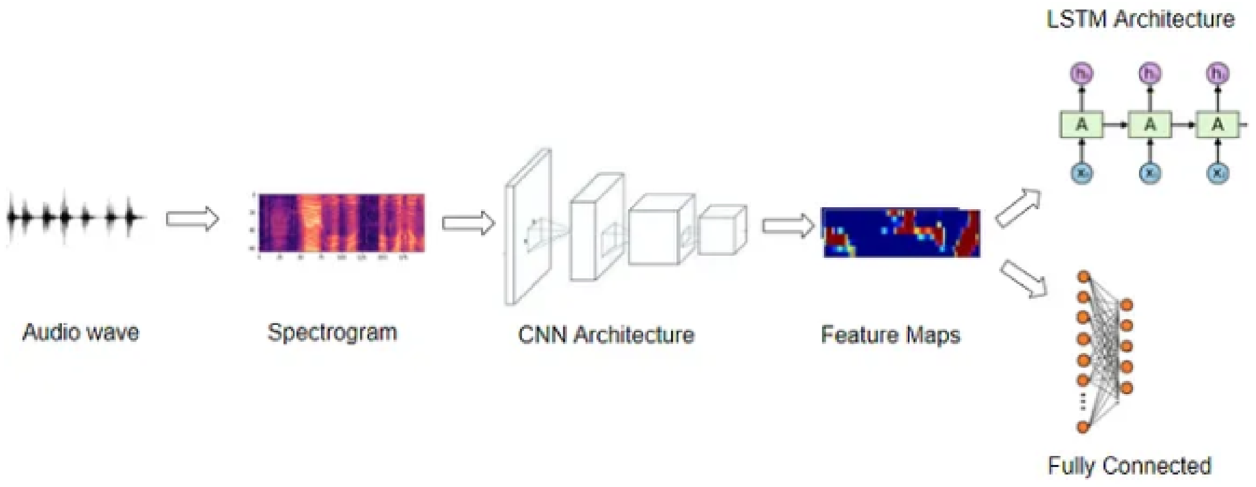


Figure A.17: Hybrid CNN-LSTM Architecture for Audio Processing: This figure illustrates the hybrid approach combining CNNs and LSTMs, where CNNs are used for feature extraction from spectrograms and LSTMs for capturing temporal dependencies. Source: Towards Data Science

A.2 Tables

Table A.1: Best Hyperparameters Found by Optuna

Hyperparameter	Value
Learning Rate	0.001
Batch Size	32
Hidden Size	128
Epochs	100
Dropout Rate	0.3
Optimizer	Adam

Table A.2: Final Model Performance Metrics (Test)

Metric	Value
SDR (Signal-to-Distortion Ratio)	-inf dB
SIR (Signal-to-Interference Ratio)	-inf dB
SAR (Signal-to-Artifact Ratio)	-inf dB
Test Loss	0.61

Table A.3: Final Model Training Metrics

Metric	Value
Final Training Loss	0.60
Validation Loss	0.61
SDR (Training)	1.76 dB
SIR (Training)	-0.37 dB
SAR (Training)	1.76 dB

Table A.4: Computational Resource Constraints

Resource	Value
GPU Runtime	9 hours
Time per Epoch	10-12 minutes
Total Training Time	12 hours
GPU Memory Usage	13.6 GB
CPU Memory Usage	24-29 GB

A.3 Code and Reproducibility

For full reproducibility of the experiments and to allow other researchers to build upon this work, the complete code used in this thesis is available on Kaggle. The notebook contains all the steps from data preprocessing to model training, evaluation, and post-processing.

- **Kaggle Notebook**: My Thesis on Kaggle

The code is organized in a clear and modular manner, with each section corresponding to different parts of the thesis. This allows for easy navigation and understanding of the implementation details. Users can replicate the results presented in this thesis or modify the code to experiment with different models, hyperparameters, or datasets.

Researchers and practitioners are encouraged to use this resource to further explore the field of music source separation, and to adapt the provided code for their own research or applications.