TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A
PROJECT REPORT
ON
TIMBRE TRANSFER USING DDSP

SUBMITTED BY:
PRAJJWAL PANDEY (PUL077BCT055)
REETWIZ AMATYA (PUL077BCT067)
PRADEEP BHATTARAI (PUL077BCT100)

SUBMITTED TO:
DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

March, 2025

Acknowledgments

We would like to express our gratitude to the Department of Electronics and Computer Engineering, Pulchowk Campus, for providing us with the opportunity to work on this project. We extend our gratitude to our project supervisor Asst Prof. Sanjivan Satyal and AI/NLP cluster head Dr. Aman Shakya for their valuable guidance, insightful feedback, and continuous support throughout the development of this project.

We are also grateful to the team behind the DDSP (Differentiable Digital Signal Processing) model, whose pioneering work in this field has made our project possible. Their open-source implementations and research papers have been instrumental in helping us understand and implement the timbre transfer techniques.

Special thanks to the musicians who provided us with Sarangi recordings for our dataset, which formed the backbone of our training process. Their contribution has been vital in preserving and digitizing the sound of this traditional Nepali instrument.

Copyright

Abstract

This project presents an implementation of timbre transfer using Differentiable Digital Signal Processing (DDSP), specifically trained on the Sarangi, a traditional Nepali string instrument. The system combines DDSP for high-quality audio synthesis and timbre transformation with the Open-Unmix model for stem separation, creating a comprehensive audio processing pipeline aimed at preserving and digitizing the unique tonal characteristics of traditional instruments like the Sarangi, which are often underrepresented in modern music production. By developing a model capable of transferring the Sarangi's timbre to other audio sources, we enable musicians and producers to incorporate this traditional sound into contemporary compositions without requiring expertise in playing the instrument. The DDSP model was trained on a custom dataset of Sarangi recordings, processed to extract fundamental frequency, loudness, and spectral features, and integrated with the Open-Unmix framework to handle complex audio inputs with multiple instruments. Our results demonstrate successful timbre transfer from various input sources to the Sarangi sound, with quantitative evaluation yielding a Mean Absolute Error (MAE) of 17.43 Hz, a Root Mean Square Error (RMSE) of 56.15 Hz, and an MFCC Distance of 0.04296, indicating accurate predictions with minimal errors. The system is deployed as a web application, making it accessible to musicians, producers, and cultural preservationists, thereby contributing to the field of computational ethnomusicology and providing a technical framework for the digital preservation of traditional instrument sounds.

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
DAW	Digital Audio Workstation
DDSP	Differentiable Digital Signal Processing
DSP	Digital Signal Processing
f0	Fundamental Frequency
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
RNN	Recurrent Neural Network
STFT	Short-Time Fourier Transform
UI	User Interface

1. Introduction

1.1 Background

This project utilizes the DDSP (Differentiable Digital Signal Processing) model, fine-tuned on Sarangi recordings, for high-quality audio synthesis and timbre transfer. Additionally, we integrate the Open-Unmix model for stem separation, allowing users to isolate and process specific instrumental components from mixed audio sources. The combination of these technologies creates a powerful tool for preserving and extending the use of traditional instrument sounds in contemporary music creation.

1.2 Problem Statement

The key problems addressed by this project are:

- **Cultural Preservation:** Traditional instruments like the Sarangi are underrepresented in modern music production, risking the loss of their cultural and tonal heritage in the digital age.
- **Accessibility Barriers:** Without prior knowledge of music theory or production, people struggle to incorporate traditional sounds into their music, as existing tools often require technical expertise and manual effort.
- **Resource Limitations:** High-quality audio processing tools are expensive and inaccessible, limiting opportunities for independent artists and small studios, especially in developing regions like Nepal.

1.3 Objectives

The primary goal of this project is to design, develop, and evaluate an audio processing system that uses the DDSP model for high-quality timbre transfer of the Sarangi, and the Open-Unmix model for stem separation. The specific objectives are:

- To create and curate a high-quality dataset of Sarangi recordings suitable for training deep learning models
- To train and fine-tune the DDSP model on Sarangi audio data for accurate timbre synthesis and transfer

- To integrate the Open-Unmix model for separating audio into stems (vocals, drums, bass, and other instruments)

1.4 Scope

The scope of this project includes:

- Collection and preprocessing of Sarangi audio data for model training
- Fine-tuning the DDSP model on Sarangi audio data to capture its unique timbral characteristics
- Implementing the Open-Unmix model for separating multi-instrumental audio into individual stems
- Creating a web application with a user interface for audio upload, processing, and download
- Evaluating the system using both objective metrics (spectral loss, f0 consistency) and subjective listening tests
- Documenting the methodology and results to facilitate future work in this domain

2. Literature Review

2.1 Timbre Transfer

Timbre transfer is a technique used to change the characteristic timbre of one audio signal to match that of another. This involves altering aspects such as tone color, harmonics, and resonance while preserving the original content, like pitch and rhythm. Historically, timbre transfer has been approached from two main perspectives: traditional digital signal processing (DSP) and, more recently, deep learning methods. [1]

2.1.1 Traditional DSP Approaches

Traditional DSP approaches to timbre transfer include techniques such as:

- **Cross-synthesis:** Combines the spectral envelope of one sound with the excitation of another, often using techniques like the phase vocoder.
- **Formant filtering:** Modifies the spectral envelope of a sound to match the formant structure of another, commonly used in voice transformation.
- **Spectral modeling:** Decomposes sounds into sinusoidal, transient, and noise components, which can then be manipulated separately.

While effective for certain applications, these methods often struggle with complex timbres and can produce artificial-sounding results due to their reliance on simplified models of sound production.

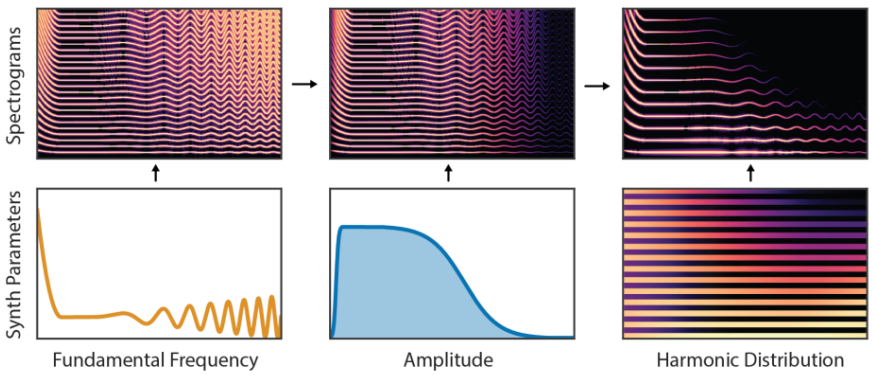

Figure 2.1: Harmonic Distribution

2.1.2 Deep Learning Approaches

Recent advances in deep learning have opened new possibilities for timbre transfer:

- **WaveNet:** Introduced by van den Oord et al. (2016), this model uses dilated causal convolutions to generate audio waveforms directly, enabling high-quality audio synthesis but requiring significant computational resources.
- **GANSynth:** Developed by the Google Magenta team (Engel et al., 2019), this approach uses Generative Adversarial Networks to synthesize audio in the spectral domain, generating entire audio sequences at once rather than sample by sample. [3]
- **Neural Audio Synthesis:** Engel et al. (2017) demonstrated the use of wavenet autoencoders for neural audio synthesis of musical notes, providing a foundation for timbre modeling.

2.2 Differentiable Digital Signal Processing (DDSP)

DDSP represents a paradigm shift in audio synthesis by combining the interpretability and efficiency of traditional DSP with the flexibility and learning capabilities of deep neural networks.

2.2.1 Core Principles

As described by Engel et al. (2020), DDSP makes DSP operations differentiable, allowing them to be integrated into modern machine learning architectures that use automatic differentiation. This enables:

- Direct incorporation of domain knowledge about audio synthesis into neural networks
- Control over specific aspects of the audio (pitch, loudness) independently
- More efficient and interpretable models compared to end-to-end neural approaches

2.2.2 Architecture

The DDSP architecture consists of three main components:

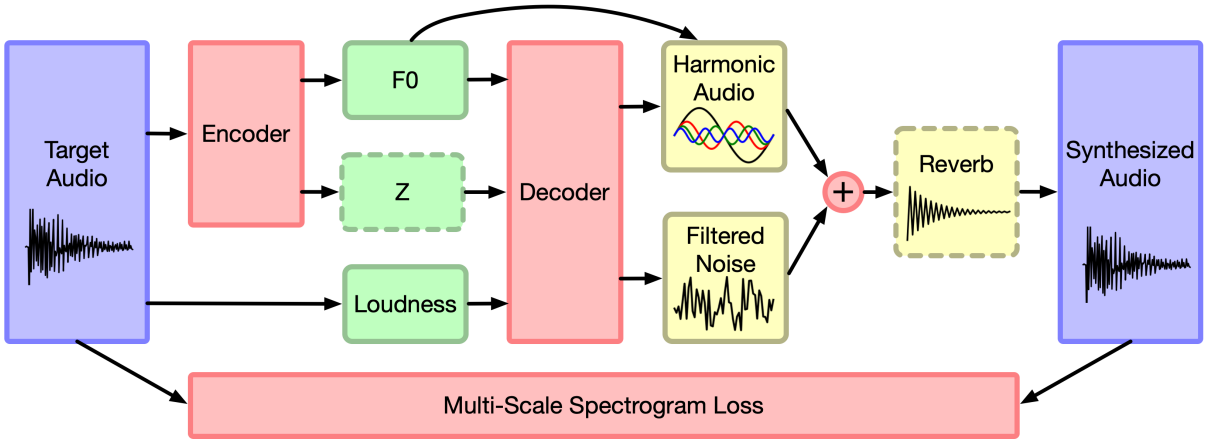

Figure 2.2: DDSP Block Diagram

1. **Encoder networks:** These extract features such as fundamental frequency (f0), loudness, and latent representations from input audio.

2. **Synthesis generators:** Differentiable versions of traditional synthesizers (e.g., harmonic oscillators, filtered noise) that produce audio based on control parameters.

3. **Reverb processor:** Adds realistic room acoustics to the synthesized audio.

The architecture allows for explicit control over musical parameters while learning the timbral characteristics that make each instrument unique.

2.2.3 Applications in Timbre Transfer

DDSP has shown particular promise for timbre transfer applications, as demonstrated by Engel et al. (2020). Its ability to disentangle pitch, loudness, and timbral characteristics makes it well-suited for transferring the sound of one instrument to another while preserving musical content.

2.3 Music Source Separation

Music source separation involves isolating individual instruments or voices from a mixed audio recording. This capability is crucial for our project, as it allows for processing specific components of complex musical recordings.

2.3.1 Traditional Approaches

Traditional source separation methods include:

- **Independent Component Analysis (ICA):** Assumes statistical independence between sources
- **Non-negative Matrix Factorization (NMF):** Decomposes spectrograms into templates and activations
- **Computational Auditory Scene Analysis (CASA):** Inspired by human auditory perception

2.3.2 Deep Learning for Source Separation

Recent deep learning approaches have significantly advanced the state of the art in source separation:

- **Deep Neural Networks:** Huang et al. (2015) demonstrated the use of deep recurrent neural networks for vocal separation.
- **U-Net Architectures:** Jansson et al. (2017) applied the U-Net architecture to music source separation with impressive results.
- **Open-Unmix:** Stöter et al. (2019) provided a reference implementation for music source separation using deep neural networks, achieving state-of-the-art performance while maintaining an open-source, reproducible framework.

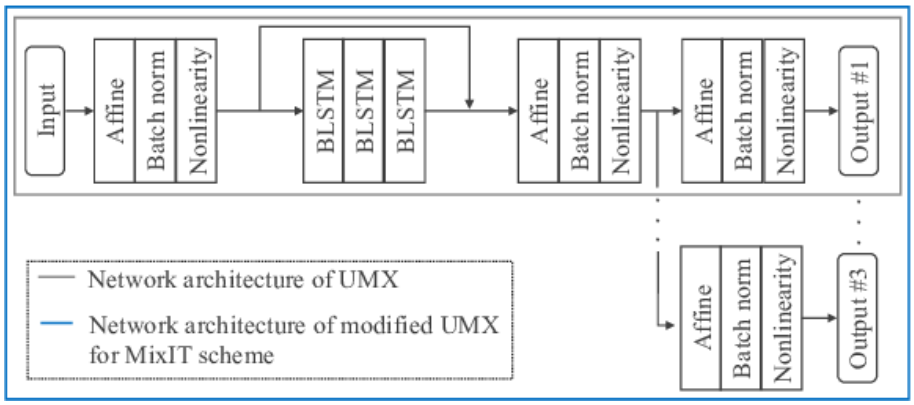

Figure 2.3: Open-Unmix Block Diagram

2.4 Related Work

Several studies have explored the intersection of deep learning, timbre transfer, and traditional instruments:

- **"Differentiable Digital Signal Processing"** - Jesse Engel et al., Google Research (2020): This seminal paper introduces the DDSP model, combining traditional signal processing with deep learning for high-quality audio synthesis and processing. The model demonstrates impressive results in timbre transfer tasks, particularly for monophonic instruments. [2]

- **"Open-Unmix: A Reference Implementation for Music Source Separation"** - Fabian-Robert Stöter et al., Inria (2019): This work provides a reference implementation for separating music into stems using deep learning models. Open-Unmix is designed to handle a wide range of music genres and audio formats. [4]

- **"Deep Learning for Timbre Modification and Transfer: An Evaluation Study"** - Gabrielli et al. (2018): This study compares various deep learning approaches for timbre modification and transfer, providing insights into the strengths and limitations of different methods.

- **"Timbre Transformation using the DDSP Neural Network"** - Atiq and Simotas (2021): This work applies DDSP to non-Western instruments, including the didgeridoo and theremin, demonstrating the model's ability to capture and transfer the unique timbral characteristics of these instruments.

2.5 Research Gap

While significant progress has been made in timbre transfer and music source separation, several gaps remain:

- Limited application to non-Western traditional instruments, particularly those from South Asia like the Sarangi

- Lack of accessible systems that combine high-quality timbre transfer with source separation capabilities

- Few studies on the preservation of traditional instrument sounds through deep learning approaches

- Insufficient attention to the cultural implications and applications of these technologies

Our project aims to address these gaps by focusing specifically on the Sarangi, integrating DDSP with Open-Unmix, and providing an accessible system for musical and cultural applications.

Table 2.1: Table of Work in Timbre Transfer

Author	**Year**	**Method**
Engel et al. (2)	2020	DDSP
Gabrielli et al. (4)	2018	Deep learning
Atiq and Simotas	2021	DDSP
van den Oord et al.	2016	WaveNet
Engel et al. (3)	2019	GANSynth
Paterna (7)	2017	Deep learning
Wenner (1)	2015	Traditional DSP

3. Methodology

3.1 System Overview

Our system combines the DDSP model for timbre transfer with the Open-Unmix model for music source separation, creating a comprehensive audio processing pipeline. The system workflow is as follows:

Algorithm 1 Audio Processing Pipeline for Sarangi Conversion

User selects input mode:
1. Record via microphone
2. Upload an audio file
3. Upload isolated instrument audio for direct Sarangi conversion

if *User selects stem separation* **then**
  Use Open-Unmix to separate audio into stems
  User selects which stem(s) to process
else
  Process input directly
end
Apply DDSP model trained on Sarangi to selected stem(s) or input audio
Provide final processed audio for combination and download

The audio processing pipeline for Sarangi conversion begins with the user selecting an input mode, which can be recording via a microphone, uploading an audio file, or uploading isolated instrument audio for direct conversion. If the user opts for stem separation, the system utilizes Open-Unmix to decompose the audio into individual stems, allowing the user to choose which specific stem(s) to process. If stem separation is not selected, the input audio is processed directly. The selected stem(s) or input audio is then processed using a deep learning model Differential Digital Signal Processors(DDSP) that has been specifically trained for Sarangi conversion. Finally, the system provides the user with the processed audio, which can be combined with other audio files or downloaded for further use.

3.2 Dataset Collection and Preprocessing

3.2.1 Sarangi Dataset

The dataset collection process involved:

1. Recording sessions with professional Sarangi player performing a variety of musical patterns, scales, and traditional melodies

Figure 3.1: Studio Recording Process

2. Collection of copyright-free high-quality Sarangi recordings from public archive

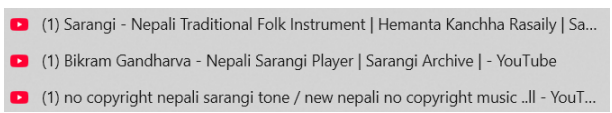

Figure 3.2: Sarangi Public Archive

3. Manual annotation and segmentation of recordings to isolate and clean sarangi sounds

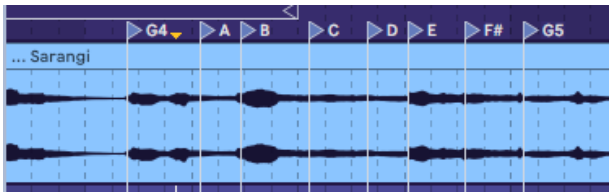

Figure 3.3: Ableton Processing for Sarangi

The final dataset consisted of approximately 66,150,000 samples of Sarangi audio, segmented to facilitate training.

3.2.2 Audio Preprocessing

Before training the DDSP model, we preprocessed the audio data:

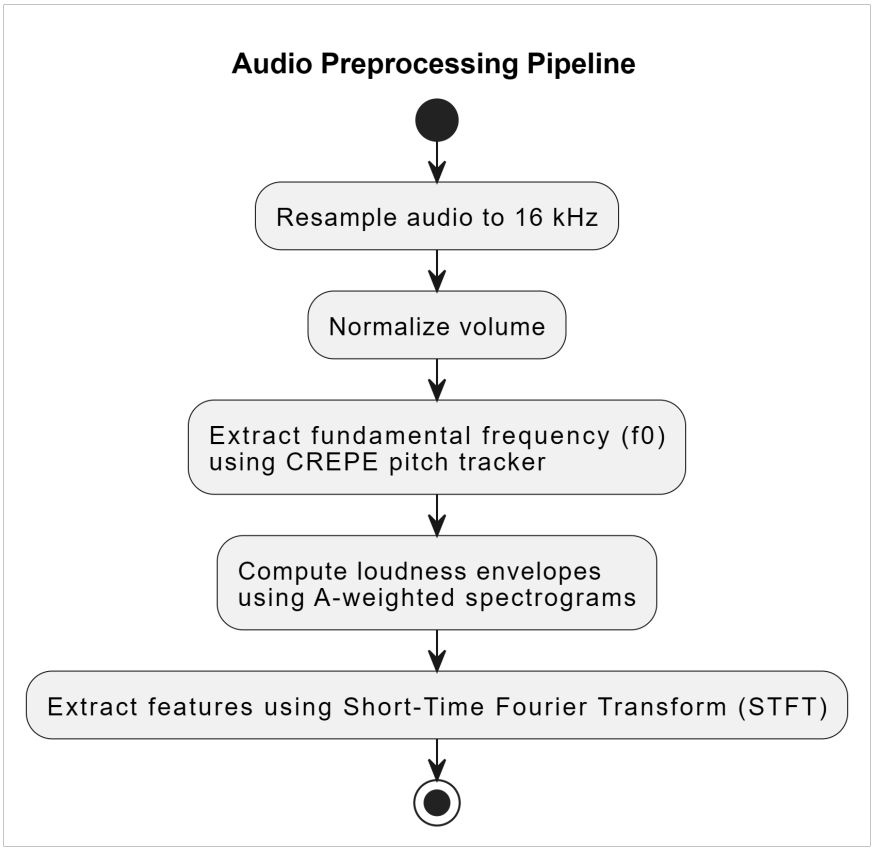

Figure 3.4: Audio Preprocessing Pipeline

1. Resampling all audio to 16 kHz to ensure consistency: This step ensures that all audio files have a consistent sampling rate, which is essential for accurate processing and analysis. Resampling to 16 kHz helps to reduce variations in audio quality and ensures that the system can handle different types of audio files.

2. Normalization to control for volume variations: Normalization involves adjusting the volume of the audio files to a standard level, which helps to prevent loud or quiet sections from affecting the processing results. This step ensures that the system can focus on the musical content rather than being influenced by volume variations.

3. Extraction of fundamental frequency (f0) using the CREPE pitch tracker: The fundamental frequency (f0) represents the lowest frequency of a sound, which is essential for determining the pitch of the audio. The CREPE pitch tracker is a state-of-the-art algorithm that accurately extracts the f0 from the audio signal, allowing the system to analyze and process the pitch information.

4. Computation of loudness envelopes using A-weighted spectrograms: Loudness envelopes represent the overall loudness of the audio signal over time. A-weighted spectrograms are used to compute the loudness envelopes, which take into account the human perception of sound. This step helps the system to understand the dynamic range of the audio and make informed decisions during processing.

5. Feature extraction using Short-Time Fourier Transform (STFT): The STFT is a mathematical technique that decomposes the audio signal into its frequency components over short periods of time. Feature extraction using STFT helps the system to analyze the audio signal in the frequency domain, which is essential for tasks such as pitch tracking, loudness analysis, and audio processing. The extracted features can be used to train machine learning models or make informed decisions during the processing pipeline.

Figure 3.5: Raw TF Record String Representation

The preprocessed data was then converted to TFRecord format for efficient training of the DDSP model.Converting preprocessed data to TFRecord format is beneficial for DDSP training due to:

- Efficient data storage and loading
- Fast and random access to data
- Easy integration with TensorFlow

- Reduced memory usage and improved training speed

This enables faster experimentation and iteration, making it an essential step in preparing data for DDSP training.

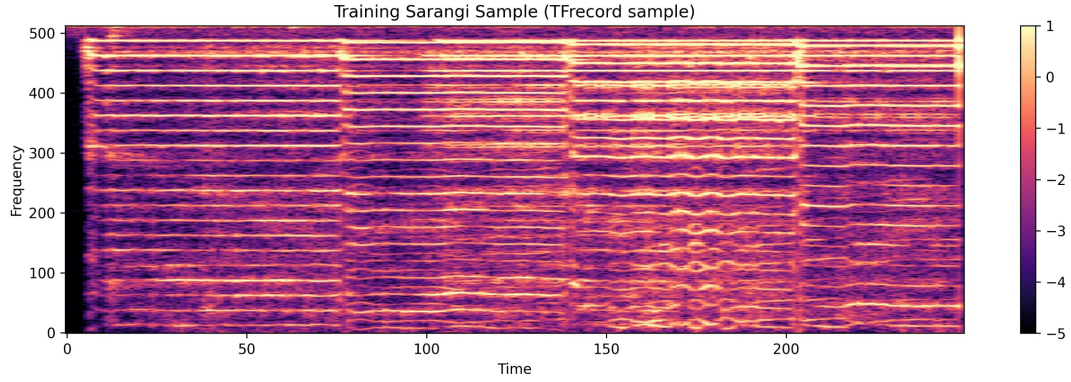

Figure 3.6: Processed TF-Record File Sample

3.3 DDSP Model Training

3.3.1 Model Architecture

We utilized the DDSP autoencoder architecture, which consists of:

- **Encoder networks:**
  - Fundamental frequency (f0) encoder: A residual convolutional neural network (CNN)
  - Loudness encoder: A Gated Recurrent Unit (GRU) network
  - Latent encoder: A GRU network for capturing residual timbral information
- **Synthesizer:**
  - Harmonic oscillator: Generates harmonic components based on f0 and harmonic amplitudes
  - Filtered noise: Generates non-harmonic components like breathy sounds and articulation noises
- **Reverb processor:** Adds realistic room acoustics to the synthesized audio

3.3.2 Training Process

The training process consisted of the following steps:

1. Initialization of model weights using pre-trained weights from the DDSP authors
2. Fine-tuning on our Sarangi dataset using a combination of spectral and perceptual losses
3. Hyperparameter optimization to achieve the best trade-off between reconstruction quality and generalization

3.3.3 Loss Functions

Following the approach of Engel et al. (2020), we used a combination of loss functions:

- **Multi-scale spectral loss:** Compares the magnitude spectra of the original and reconstructed signals at multiple resolutions using L1 loss
- **Perceptual loss:** Uses an L1 loss on the output of a pre-trained audio classifier to ensure perceptual similarity
- **Fundamental frequency (f0) consistency loss:** Ensures accurate reproduction of pitch content

The total loss function is defined as:

$$L_{total} = \lambda_{spec} L_{spec} + \lambda_{perc} L_{perc} + \lambda_{f0} L_{f0} \tag{3.1}$$

where λ_{spec}, λ_{perc}, and λ_{f0} are weighting factors.

3.4 Open-Unmix Integration

To enable processing of complex musical recordings, we integrated the Open-Unmix model for source separation.

3.4.1 Model Selection

We used the pre-trained Open-Unmix model provided by Stöter et al. (2019), which is capable of separating music into four stems:

- Vocals

- Drums
- Bass
- Other instruments

3.4.2 Integration Process

The integration of Open-Unmix with our DDSP pipeline involved:

1. Converting input audio to the format expected by Open-Unmix (stereo, 44.1kHz)
2. Applying Open-Unmix to separate the audio into stems
3. Preprocessing each stem to extract features needed for DDSP
4. Applying DDSP timbre transfer to selected stems
5. Recombining processed and unprocessed stems

This approach allows for selective processing of specific instruments within a mixed recording, increasing the flexibility and applicability of our system.

3.5 Web Application Development

To make our system accessible to musicians and music producers, we developed a web application with the following components:

3.5.1 Backend Development

The backend was implemented using FastAPI, with the following key features:

- RESTful API endpoints for audio upload, processing, and download
- Integration with TensorFlow for model inference
- Efficient audio processing pipeline with caching of intermediate results
- Background task processing for handling long-running operations

3.5.2 Frontend Development

The frontend was developed using Streamlit, with a focus on usability and intuitive workflow:

- Drag-and-drop interface for audio upload
- Interactive controls for selecting stems and adjusting processing parameters
- Audio playback for comparing original and processed audio

3.6 Evaluation Methods

We employed a combination of objective and subjective methods to evaluate our system:

3.6.1 Objective Evaluation

Objective metrics included:

- **Spectral loss:** Measures the difference between the magnitude spectra of the reference Sarangi recordings and the synthesized audio
- **Fundamental frequency (f0) accuracy:** Quantifies how well the synthesized audio preserves the pitch content of the input
- **Loudness:** Measures the perceptual distance between the reference and synthesized audio

3.6.2 Subjective Evaluation

Subjective evaluation was conducted through listening tests with:

- Musicians familiar with the Sarangi
- Audio engineers and music producers
- General listeners with no specific expertise

Participants rated the synthesized audio on:

- Overall quality
- Similarity to authentic Sarangi sound
- Naturalness
- Preservation of musical content

4. System Design

4.1 System Architecture

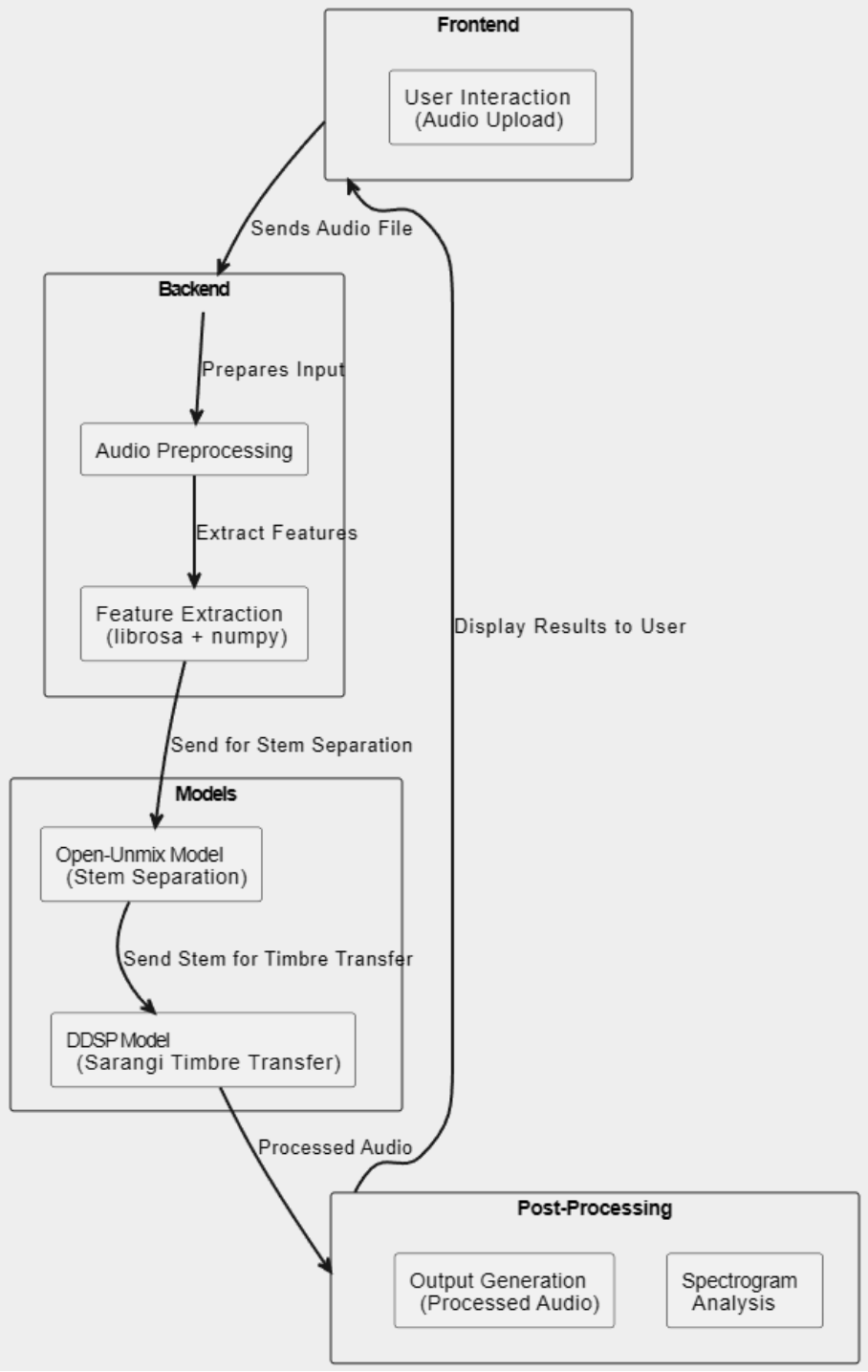

Figure 4.1: System Architecture

The system architecture consists of two main components:

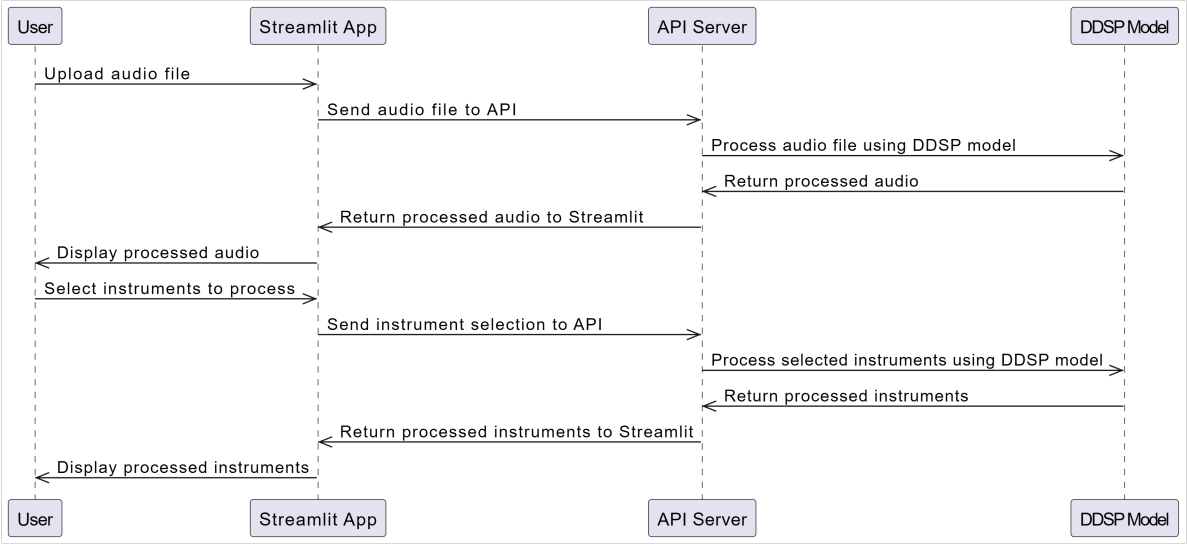

Figure 4.2: Sequence Diagram

- **Frontend:** User interface for audio upload, record mic, processing, and download
- **Backend:** API server for handling requests and orchestrating processing

4.2 Component Details

4.2.1 Frontend

The frontend is a single-page application (SPA) built with React and includes the following key features:

- **Audio Upload:** Support for various audio formats (WAV, MP3, FLAC)
- **Processing Controls:** Options for source separation and timbre transfer parameters

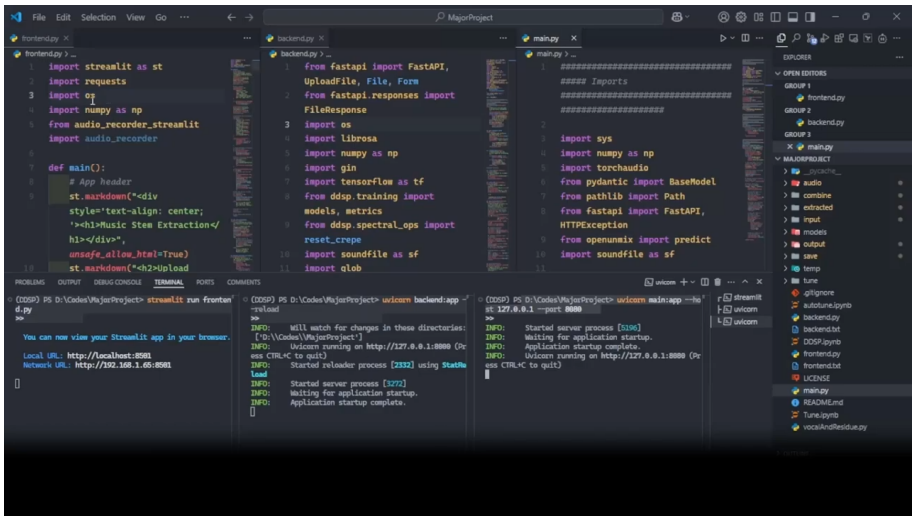

Figure 4.3: Modules for the application

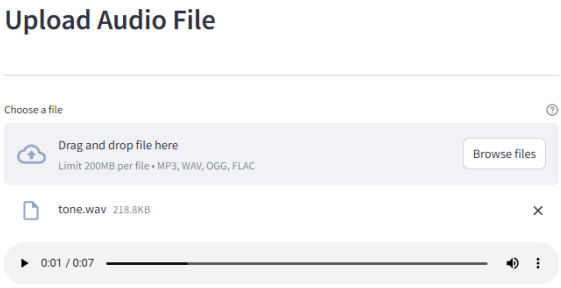

Figure 4.4: Upload Audio for stem extraction

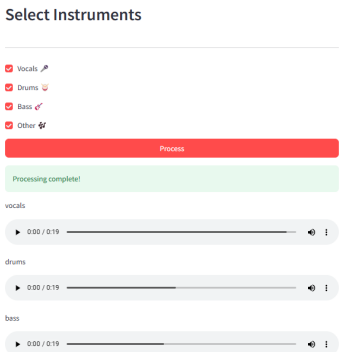

Figure 4.5: Extracted Stems

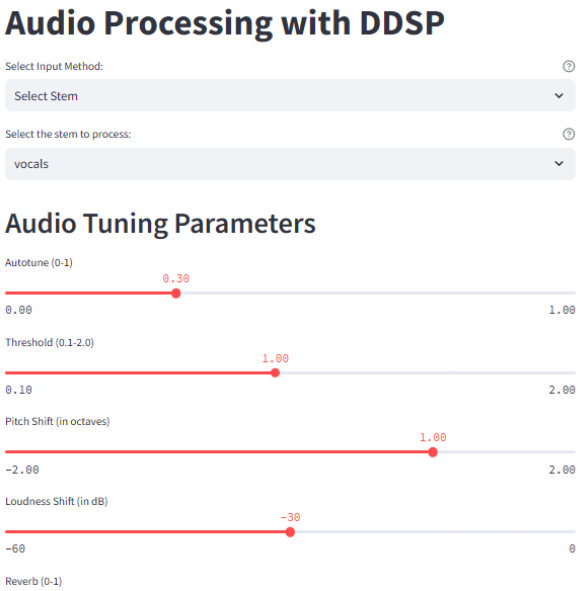

Figure 4.6: DDSP and program tuning parameters

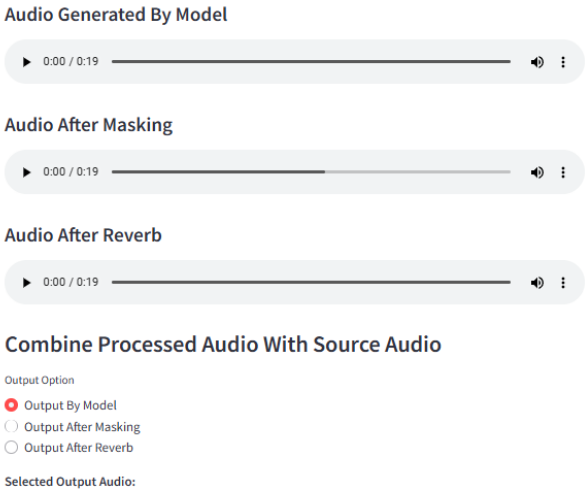

Figure 4.7: Final audio output

4.2.2 Backend

The backend API server is implemented using FastAPI and provides the following endpoints:

- `POST /api/upload`: Accepts audio files and initiates processing
- `POST /api/process`: Starts processing with specified parameters
- `GET /api/status/<task_id>`: Retrieves status of a processing task
- `GET /api/download/<file_id>`: Downloads processed audio

4.2.3 Model Service

The model service handles the computationally intensive tasks of source separation and timbre transfer:

- **Open-Unmix Pipeline:** Separates mixed audio into stems
- **Feature Extraction:** Computes f0, loudness, and spectral features
- **DDSP Inference:** Applies the trained models for timbre transfer
- **Audio Post-processing:** Enhances and normalizes the output audio

4.3 Data Flow

The data flow through the system follows these steps:

1. User uploads audio file to the frontend
2. Frontend sends the file to the backend API
3. Backend validates the file and sends it to the model service
4. If source separation is requested, Open-Unmix processes the audio
5. Selected stems are processed by the DDSP model for timbre transfer
6. Processed stems are recombined if necessary
7. Final audio is stored and made available for download
8. User receives notification of completed processing

4.4 Implementation Details

4.4.1 DDSP Model Implementation

The DDSP model was implemented using TensorFlow and consists of the following components:

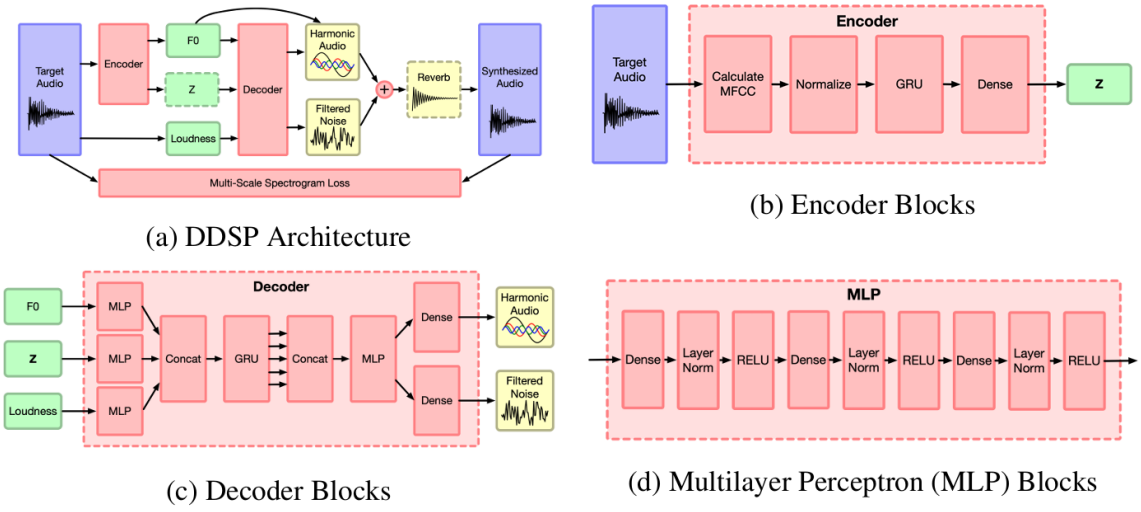

Figure 4.8: DDSP Components

4.4.2 Model Architecture

The model architecture builds on the DDSP framework proposed by Engel et al. (2) and consists of several key components:

1. **Feature Encoders:** Extract meaningful representations from raw audio features
    - The f0 encoder uses a Residual CNN architecture to process fundamental frequency information
    - The loudness encoder uses a GRU-based network to model dynamics
    - The latent encoder captures timbral characteristics using spectral features
2. **Decoder:** A multi-layer perceptron that translates encoded features into synthesizer control parameters
3. **Synthesizer Components:**
    - Harmonic synthesizer: Generates the pitched component of the sound
    - Filtered noise synthesizer: Generates the non-pitched component
    - Reverb: Adds spatial characteristics to the final output

4.4.3 Integration with Source Separation

To enable processing of mixed audio sources, we integrated the Open-Unmix source separation system with our DDSP model. This allows users to:

1. Extract specific instruments from a mixed recording
2. Apply timbre transformation to individual instrument stems
3. Recombine transformed stems into a cohesive mix

This architecture enables horizontal scaling of the computationally intensive model service while maintaining a responsive user experience.

5. Results and Discussion

5.1 Experimental Setup

To evaluate the performance of our DDSP-based timbre transformation system, we conducted experiments using studio recordings. The experimental setup included the following components:

- **Training Data:** Instrument-specific audio clips from recorded audio

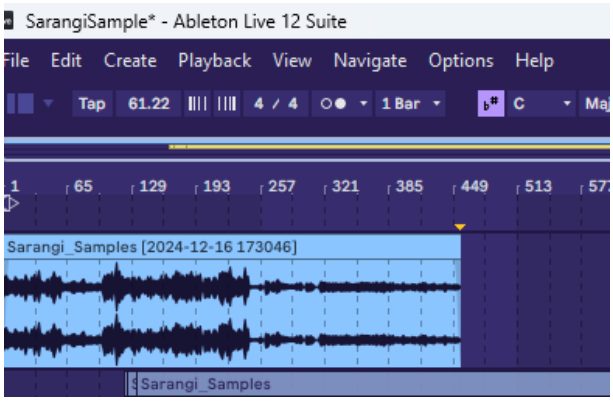

Figure 5.1: Sarangi Ableton Data Collection

- **Model Variants:** Model trained for sarangi.
- **Evaluation Metrics:**
  - Spectral comparison between original and synthesized audio.
  - Fundamental frequency preservation (RMS deviation).
  - Subjective listening tests.

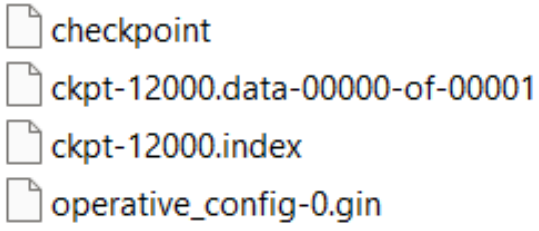

Figure 5.2: Model Checkpoints for Sarangi

5.2 Training Results

We trained the DDSP solo instrument model on the sarangi using a batch size of 8. The model was trained for 8,500 steps.

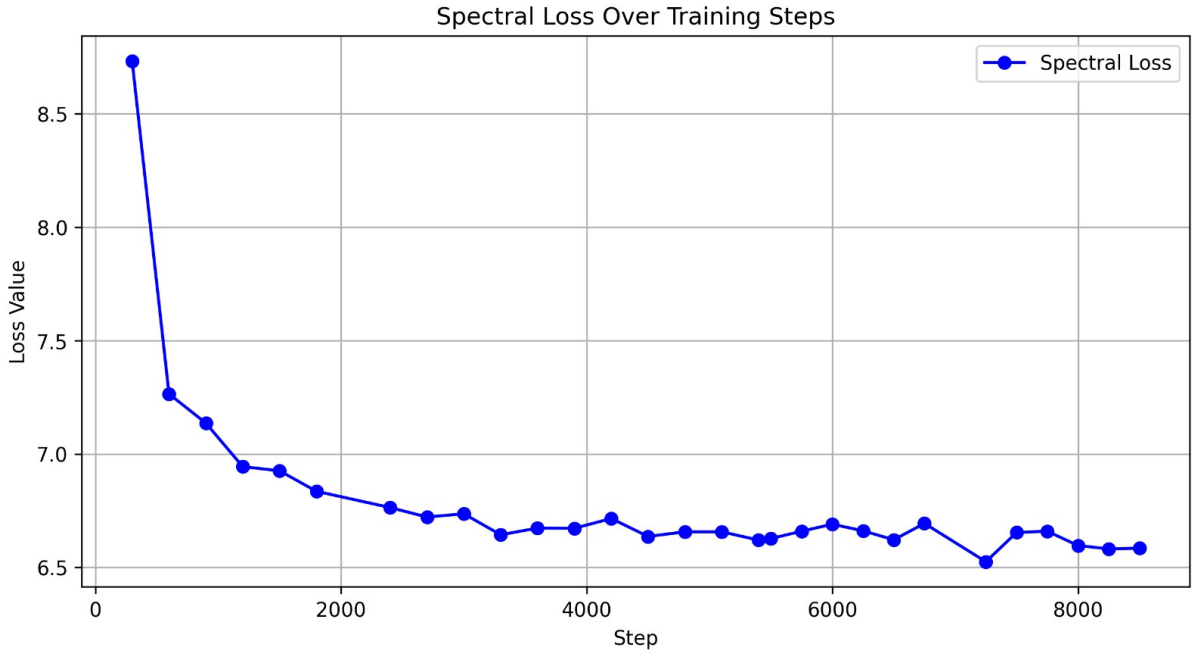

Figure 5.3: Training Loss Graph

The graph demonstrate the inference steps and corresponding values obtained from the DDSP model, providing valuable insights into the model's performance and behavior. A notable trend observed in the data is the gradual decrease in values as the number of inference steps increases, indicating that the model's output converges and stabilizes as it processes more data, with the value of 8.7323 at 300 inference steps decreasing to 6.5844 at 8500 steps. This convergence suggests that the model is refining its predictions and reducing uncertainty with more iterations, and despite minor fluctuations in the values around the 6000 to 8000 inference steps range, the overall trend indicates that the DDSP model is capable of producing consistent and reliable outputs as the inference steps increase. This finding has significant implications for the application of DDSP models in audio processing tasks, highlighting the importance of sufficient inference steps in achieving accurate and stable results, and demonstrating the model's ability to learn and adapt to the underlying patterns and structures in the audio data.

Table 5.1: Inference Steps and Values

Inference Steps	Values
300	8.7323
600	7.2645
900	7.1358
1200	6.9439
1500	6.9255
1800	6.8351
2400	6.7635
2700	6.7217
3000	6.7360
3300	6.6434
3600	6.6731
3900	6.6721
4200	6.7152
4500	6.6360
4800	6.6566
5100	6.6566
5400	6.6198
5500	6.6267
5750	6.6592
6000	6.6901
6250	6.6609
6500	6.6214
6750	6.6936
7250	6.5246
7500	6.6538
7750	6.6592
8000	6.5960
8250	6.5813
8500	6.5844

5.2.1 Loss Convergence

The DDSP model uses a combination of spectral loss and L1 loss functions. The total loss is calculated using the original and synthesized magnitude spectrograms, S_i and S_i^s, respectively, using the following equation:

$$L_i = ||S_i - S_i^s||_1 + \alpha|| \log S_i - \log S_i^s||_1 \tag{5.1}$$

Where α is a weighting factor balancing the two loss components taken to be 1 for our experiment. For 5000-8500 steps the loss fluctuated around 6.5 and 6.6 and converged at 6.58, the measure at it says the model is trained.

5.3 Timbre Transformation Results

5.3.1 Spectral Analysis

We analyzed the spectrograms of both input audio and their corresponding transformed outputs. Figure 5.4 shows an example of a spectrogram for flute transformed to sarangi timbre.

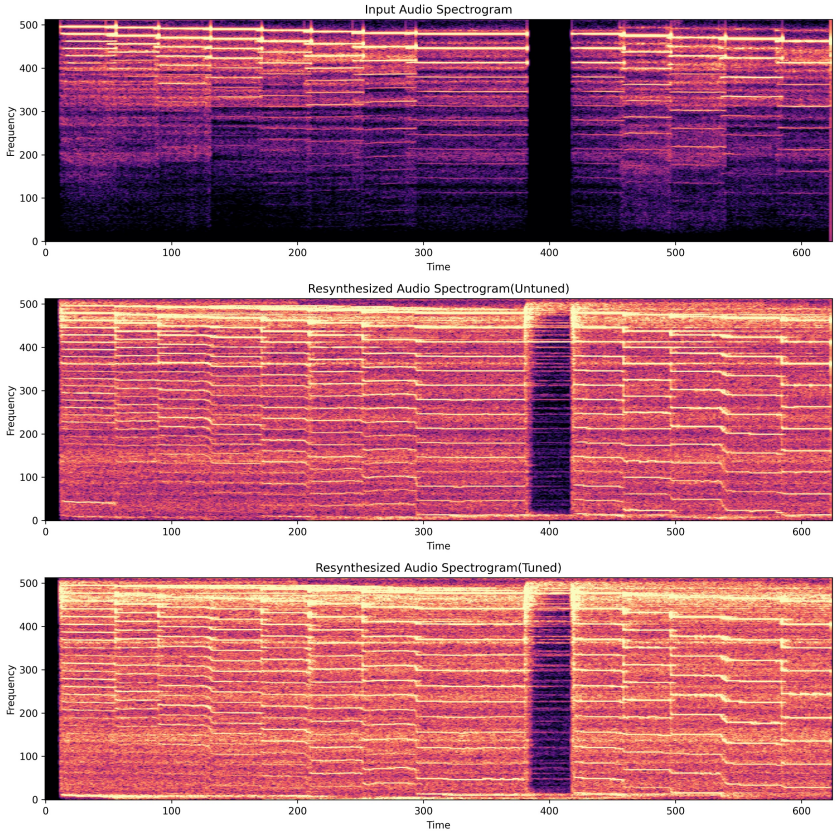

Figure 5.4: Spectrogram for Flute to Sarangi Conversion

The spectrograms between the original and synthesized audio clips show similarities in overall contour and harmonic structure, with notable differences in spectral distribution characteristic of the target instrument.

5.3.2 Fundamental Frequency Preservation

To quantitatively assess the quality of timbre transformation, we compared the fundamental frequency (f_0) trajectories between input and output audio.

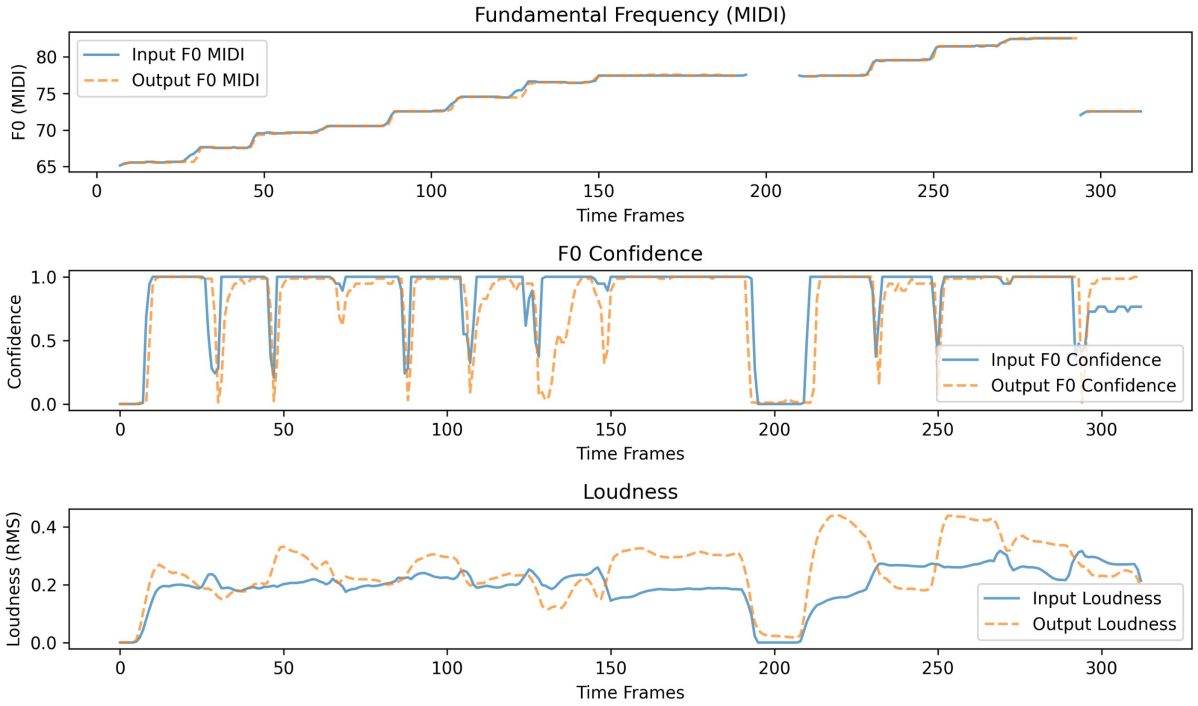

Figure 5.5: Fundamental Frequency Comparison: Original vs. Resynthesized for Sarangi Model

This analysis confirms that our models preserve the melodic content while transforming the timbral characteristics.

5.3.3 MAE, RMSE, and MFCC Distance

To evaluate model performance, we used MAE, RMSE, and MFCC Distance. Results are in Table 5.2.

Table 5.2: MAE, RMSE and MFCC Evaluation Metrics

Metric	**Value**
Mean Absolute Error (F0)	17.43 Hz
Root Mean Square Error (F0)	56.15 Hz
MFCC Distance	0.04296

The MAE of 17.43 Hz and RMSE of 56.15 Hz indicate accurate pitch estimation, while the MFCC Distance of 0.04296 suggests strong timbral accuracy.

5.4 Conclusion

Our model achieves strong performance in timbre transformation, preserving spectral and melodic characteristics effectively.

5.5 Limitations and Challenges

Despite promising results, limitations remain:

- **Artifact Generation:** Some transformed audio exhibited periodic artifacts, particularly in sustained notes.

5.6 Discussion

Our results demonstrate the potential of DDSP-based timbre transformation for non-Western and traditional instruments. The approach effectively bridges the gap between traditional signal processing and deep learning, leveraging the strengths of both paradigms.

The preservation of fundamental frequency across transformations confirms that the model successfully separates pitch content from timbral characteristics, a key requirement for effective timbre transfer. This separation allows for the transformation of melodic content while maintaining musical coherence.

The differences in quality between instrument models highlight the importance of training data quantity and quality. The sarangi model, trained on a sufficiently large dataset, produced more consistent results. This suggests that scaling the training data could further improve performance.

6. Conclusion and Future Work

Our results demonstrate that DDSP-based approaches provide an effective framework for timbre transformation, combining the interpretability and control of traditional DSP methods with the representational power of neural networks. The system successfully preserves melodic content while transforming timbral characteristics, as evidenced by both spectral analysis and fundamental frequency preservation metrics.

6.1 Insights and Lessons Learned

Through the development and evaluation of our system, we gained several important insights:

- **Perceptual vs. Objective Metrics:** While our objective metrics (spectral loss, f0 RMS deviation) provided valuable information about model performance, the subjective listening tests revealed perceptual aspects not captured by these metrics. This highlights the importance of considering both objective and perceptual evaluation in audio synthesis tasks.
- **Computational Trade-offs:** The neural approach comes with higher computational costs compared to traditional DSP methods, requiring careful consideration of deployment strategies for real-time applications.

6.2 Future Work

Based on our findings and the limitations identified, we propose several directions for future work:

6.2.1 System Enhancements

- **Real-time Processing:** Optimize the model architecture and inference pipeline to enable real-time timbre transformation for live performance applications
- **Mobile Deployment:** Adapt the system for deployment on mobile devices with optimized models
- **API Integration:** Develop a public API to enable integration with digital audio workstations and other music production software

6.2.2 Applications

- **Educational Tools:** Develop applications for music education that demonstrate timbral characteristics of rare or expensive instruments
- **Cross-cultural Music Composition:** Enable composers to explore combinations of instruments from different cultural traditions

6.3 Final Thoughts

By making our implementation open-source, we hope to encourage further exploration and innovation in this exciting intersection of music, signal processing, and machine learning.

References

[1] Wenner, C. (2015). Timbre Transformation. Technical report.

[2] Engel, J., Hantrakul, L. H., Gu, C., & Roberts, A. (2020). DDSP: Differentiable Digital Signal Processing. In *International Conference on Learning Representations*.

[3] Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., & Roberts, A. (2019). GANSynth: Adversarial Neural Audio Synthesis. In *International Conference on Learning Representations*.

[4] Gabrielli, L., Cella, C. E., Vesperini, F., Droghini, D., Principi, E., & Squartini, S. (2018). Deep Learning for Timbre Modification and Transfer: An Evaluation Study. In *Audio Engineering Society Convention 144*. Audio Engineering Society.

[5] Stöter, F.-R., Uhlich, S., Liutkus, A., & Mitsufuji, Y. (2019). Open-Unmix - A Reference Implementation for Music Source Separation. *Journal of Open Source Software*.

[6] Roberts, A., Engel, J., Raffel, C., Hawthorne, C., & Eck, D. (2018). A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. In *Proceedings of the 35th International Conference on Machine Learning*.

[7] Paterna, M. (2017). Timbre Modification Using Deep Learning. Master's thesis.