DeepMelody: Personalized Music Recommendations Based on Instrumental Composition

Prasham Soni [1] **Shivam Shinde** [1] **Manav Mepani** [1]

Abstract

This project tackles the computational challenge of generating music recommendations rooted in instrumental composition. Using the Slakh2100 dataset, which includes multi-instrument stems for tracks featuring guitar, piano, bass, drums, and more, we focus on isolating and analyzing individual instruments. A U-Net-based model is employed for music source separation, isolating spectrograms for each instrument. These spectrograms are reconstructed into audio using the inverse short-time Fourier transform (ISTFT) and are used to compute "instrument embeddings" that represent the prominence of instruments in a audio track. This prominence is determined by the percentage of the total time the instrument plays.

The system generates recommendations by calculating the cosine similarity between these instrument embeddings and pre-calculated embeddings of audio tracks in the database. This allows the system to suggest songs with similar instrumental compositions. Our analyses demonstrate effective source separation and recommendation, showcasing the feasibility of using instrumental embeddings for personalized music discovery. This approach introduces a novel perspective by emphasizing instrumental attributes, offering a unique method of exploring music based on its compositional structure.

1. Introduction

The ability to recommend music based on user preferences has become a cornerstone of modern streaming platforms. However, most recommendation systems rely on genres, user listening history, or general acoustic characteristics to suggest new tracks. This approach, while effective in certain contexts, often fails to capture the nuanced role of instrumentation in defining the texture, mood, and appeal of a song. For example, a user learning to play the guitar might look for songs with a strong emphasis on the guitar, where the instrument's play percentage is particularly high. Traditional genre-based systems are not equipped to cater to such specific instrumental preferences.

Previous research has made strides in the separation of music sources and the recommendation systems as standalone tasks. However, these approaches have rarely been integrated into a cohesive framework for personalized music recommendation. Existing systems that focus on general audio embeddings or genre classification often overlook the details of instrumental composition.

To address this gap, we propose a novel approach that combines music source separation, and recommendation into a unified system. By isolating individual instruments from audio tracks and quantifying their prominence, our method enables recommendations based on the instrumental composition of a song. This provides a new dimension of personalization, allowing users to discover music aligned with their specific instrumental preferences. This work explores the development and evaluation of this approach, which aims to enhance music discovery by shifting the focus from genre to instrumental composition.

1.1. Research contributions

This work presents an approach to music recommendation by focusing on instrument-specific preferences, a dimension often overlooked in traditional genre- or user-history-based systems. The key contributions of this research to the academic and technological community are as follows:

[1]Worcester Polytechnic Institute. Correspondence to: Prasham Soni <psoni@wpi.edu>, Shivam shinde <sshinde@wpi.edu>, Manav Mepani <mmepani@wpi.edu>.

CS/DS 541 Deep Learning Final Project, Worcester Polytechnic Institute.

1. **Instrument-Centric Music Recommendation System**:

   This study uses a recommendation framework based on users' preferences for specific instruments. By quantifying instrumental prominence in songs, the system provides a unique way to personalize music discovery, diverging from conventional genre-based recommendations.

2. **Introduction of Instrument Embeddings**:

   A representation method, "instrument embeddings," is implemented to encode the presence and prominence of instruments in a track. These embeddings form the foundation for similarity calculations and recommendations, enabling a quantitative approach to music personalization.

3. **Novelty in User-Centric Interaction**:

   By focusing on specific instrument preferences rather than high-level music genres, this study proposes a fundamentally different way of interacting with recommendation systems

2. Related Work

In our literature review, we discovered several approaches to the music source separation task.

"Some of the latest top performing music source separation models are Open-Unmix [5], MMDenseLSTM [6], Demucs [7] and Meta TasNet [8]. While Open-Unmix and MMDenseLSTM models comprise LSTMs and operate on spectrogram input, the other two methods operate directly on the time-domain waveform. It is not possible to single out any one of these models as the best model because they differ in the number of trainable parameters, training time and performance metrics with respect to each of the sources in the mixture.[2]"

Most of these methods utilize either of the two types of inputs: time-domain audio representation (waveforms) or frequency-time domain audio representation (spectrograms).

"The spectrograms are compact representations of time-domain waveforms. The networks operating directly on the time-domain waveforms require larger convolution kernels than those operating on the spectrograms because of the higher time resolution in the time-domain waveforms. Hence, the number of trainable network parameters in the waveform-based models are generally higher than that of spectrogram-based models. In this way, the spectrogram-based models have less training cost than the waveform-based ones.[2]"

Given that spectrogram-based architectures require fewer training parameters, we chose to focus on music source separation using spectrogram-based neural network architectures.

"Among the CNN based methods operating on spectrogram input, the U-Net [1] based methods [2][3][4] have been popular owing to their simplicity and ease of training.[2]"

For this reason, we decided to implement the U-Net (originally implemented by [2][3][4]) architecture for our music source separation task.

There are several spectrogram-based U-Net architectures currently used for music source separation tasks, including:

· A single neural network for predicting the source audio (Multi-Task Model) [2]

· Independent neural networks for predicting each source audio (Dedicated Models)

· A conditional neural network model for predicting the source audio (Conditional Model) [3]

Among the three approaches, we chose the method utilizing a single neural network for music source separation because it requires the least amount of training parameters while delivering comparable results [2].

This approach takes in the grayscale spectrograms as inputs and soft masks of the spectrograms of source audios as outputs.

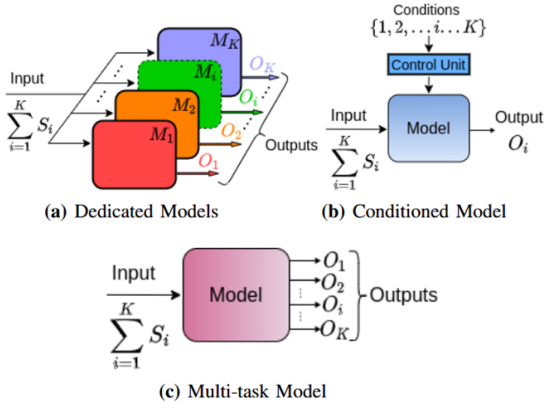

Figure 1. Different Approaches to Spectrogram and U-Net based Music Source Separation [2]

3. Proposed Method

Recommendation System Overview

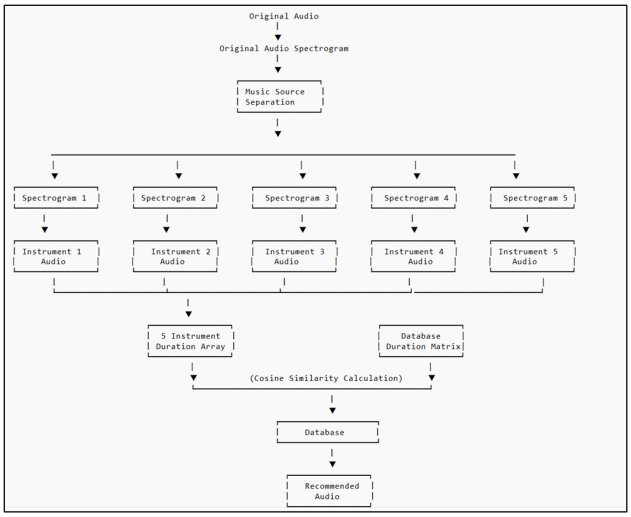

Figure 2. Project Workflow

Users will upload a song through our user interface, which will then be converted into a spectrogram and turned into grayscale. The spectrogram will pass through the trained model, which will predict five separate soft masks: one for each of the four instruments and one for all other instruments combined. The soft masks are transformed back into spectrograms by multiplying them with the input spectrogram magnitudes, followed by the addition of the phase information from the input spectrogram.

These spectrograms will then be converted back into audio files. Next, we will use the Librosa Python library to calculate the percentage of time each instrument plays in each audio file. These values will form an array representing the instrument's durations (or prominence). We will then compare this array with a pre-calculated duration (or prominence) matrix using cosine similarity. Finally, we will recommend the audio with the highest cosine similarity to the user. User interface was created using Streamlit python library.

Users can also select their preferred instrument. In this scenario, we will calculate the similarity score based on their chosen instrument. We will slice the database duration matrix and the instrument duration array according to the user's input, then compute the similarity between these two. The resulting similarity score will be used to generate recommendations.

User Interface (Figure 3 , Figure 4, Figure 5):

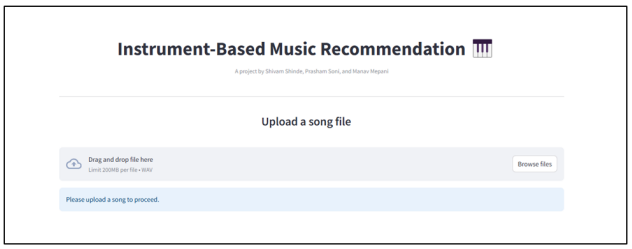

Figure 3. User Interface in Default State

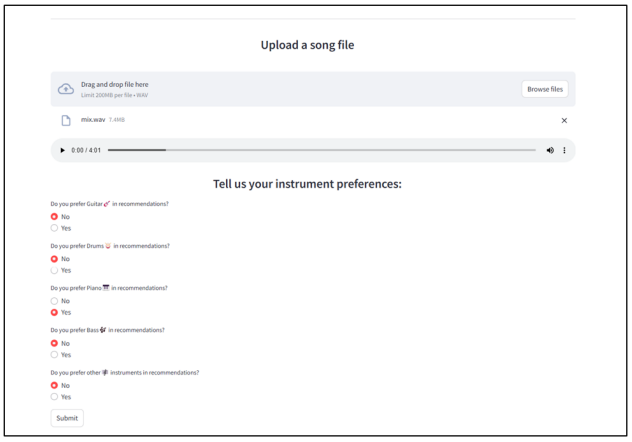

Figure 4. User Interface after Uploading an Audio Track

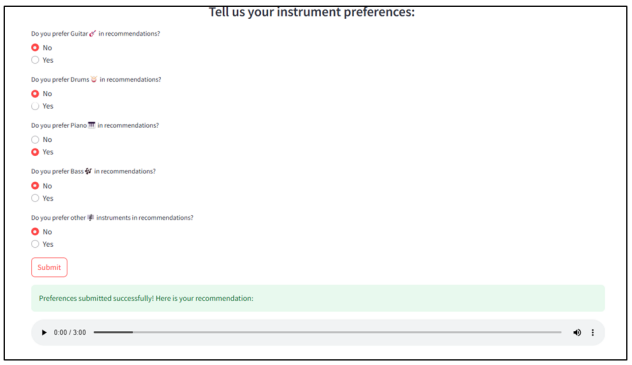

Figure 5. User Interface after recommending a song

Slakh2100 Dataset

We utilized the Slakh2100 dataset for our application. This dataset includes audio files featuring multiple instruments, along with separate audio files for each individual instrument. It contains 2100 audio tracks (as indicated by the name) in .FLAC format, totaling over 120 hours of audio data.

The dataset is pre-split into training, validation, and test subsets. We trained our model on the training data and validated it using the validation set.

Audio Pre-processing

We tailored our application to focus on four instruments: Guitar, Piano, Bass, and Drums. The audio files varied in duration, so we standardized their lengths to 180 seconds. If an audio file exceeded 180 seconds, it was trimmed to fit. Conversely, files shorter than 180 seconds were padded with zeros at the end to reach the desired length.

The dataset contained multiple audio files for each instrument, reflecting the variety within each category (e.g., different types of Guitars, Pianos, Basses, and Drums). To simplify, we consolidated all audio files of the same instrument into a single category. For instance, all Piano audio files were combined into one group. Similarly, the audio files of other instruments were merged into single files for each respective instrument. After merging, the dataset was reduced to five tracks: one each for piano, drums, bass, and guitar, and one additional track for all other instruments.

Additionally, our folder structure included multiple metadata files, each corresponding to a mixed audio file. We extracted the information from these metadata YAML files and consolidated it into a single CSV file for future use.

The original folder structure was as follows:

```
Slakh2100
├── train
|   ├── Track0001
|   |   ├── MIDI
|   |   |   ├── s00.mid
|   |   |   ├── s01.mid
|   |   |   ├── ...
|   |   ├── stems
|   |   |   ├── s00.flac
|   |   |   ├── s01.flac
|   |   |   ├── ...
|   |   ├── metadata.yaml
|   |   ├── mix.flac
|   ├── Track0002
|   |   ├── MIDI
|   |   |   ├── s00.mid
|   |   |   ├── s01.mid
|   |   |   ├── ...
|   |   ├── stems
|   |   |   ├── s00.flac
|   |   |   ├── s01.flac
|   |   |   ├── ...
|   |   ├── metadata.yaml
|   |   ├── mix.flac
|   ├── ...
├── test (same structure as train)
├── validation (same structure as train)
```

Figure 6. Original Data Folder Structure

The original folder structure was not compatible with PyTorch data loading framework. Therefore, we modified it to the structure shown in figure 7. Also, FLAC files were converted to wav files.

```
Audio_Dataset
├── train
|   ├── Input
|   |   ├── Track0001_mix.wav
|   |   ├── Track0002_mix.wav
|   |   └── ...
|   ├── Output
|   |   ├── Track0001
|   |   |   ├── Guitar.wav
|   |   |   ├── Bass.wav
|   |   |   ├── Drums.wav
|   |   |   ├── Piano.wav
|   |   |   └── Others.wav
|   |   ├── Track0002
|   |   |   ├── Guitar.wav
|   |   |   ├── Bass.wav
|   |   |   ├── Drums.wav
|   |   |   ├── Piano.wav
|   |   |   └── Others.wav
|   |   └── ...
├── test
|   ├── Input (same structure as train)
|   └── Output (same structure as train)
├── validation
|   ├── Input (same structure as train)
|   └── Output (same structure as train)
```

Figure 7. Audio Dataset New Folder Structure

Next, we focused on converting the audio files into spectrograms. A window length of 1022, a hop length of 512, and a sample rate of 10880 were used to generate the spectrograms for the audio tracks.These parameters were adapted from [2].

In addition, a Hann window was applied during the processing. The resulting spectrograms were stored in a folder structure similar to that shown in Figure 1.

Next, all input mixed audio spectrograms were resampled to a shape of (512, 512) and converted to grayscale.

Subsequently, the output spectrograms were transformed into soft masks. After completing all these preprocessing steps, the inputs consisted of grayscale spectrograms, with corresponding soft masks as outputs for each instrument.

Model Architecture:

For the music source separation component of our application, we employed a neural network architecture called U-Net [1]. The encoder consisted of five contracting blocks, each containing two convolutional layers with a kernel size of 3, padding of 1, and ReLU activation, followed by a max-pooling layer. The decoder had five expanding blocks, with each block consisting of a transpose convolution layer, followed by two sets of convolutional layers with a kernel size of 3, padding of 1, and ReLU activation. Additionally, skip connections were used between the encoder and decoder parts of the network. Finally, after the entire network, we applied a convolutional layer to adjust the output channels to 5 (equal to number of sources). This architecture was inspired by the paper [1][2][4].

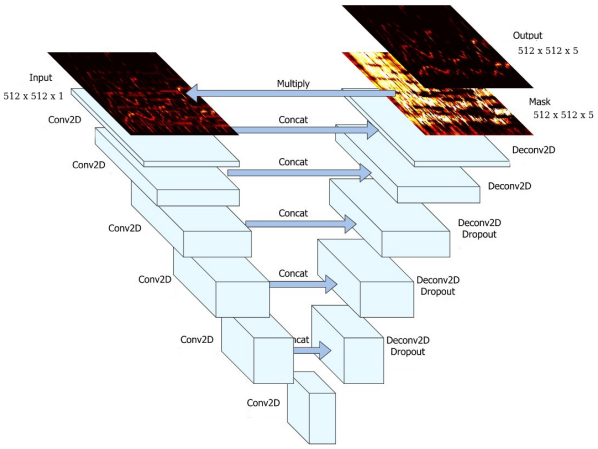

Figure 8. UNet [4]

Our model architecture (Figure 8) closely resembles the one depicted in the figure.

Model Training

For model training, we utilized an energy-based loss function, adapted from the approach presented in [2].

In audio files, certain instruments may be louder than others. If we were to use a non-weighted loss function, the neural network might overlook the quieter instruments. To address this, we assign weights that are inversely proportional to each instrument's energy in the audio file. These weights are then applied to the output loss for each instrument. We tried multiple hyperparameter combination for model training. You can find those in experiment section.

4. Experiment

To validate our approach, we conducted a series of experiments designed to identify optimal training settings and assess the model's performance. The experiments were performed on 1,500 audio tracks, each preprocessed into their corresponding spectrogram representations. Our primary goal was to determine how different hyperparameter choices influenced the training stability, convergence rate, and eventual generalization quality of the model.

We systematically evaluated three key hyperparameters:

1. **Learning Rate:** We tested learning rates of 0.1, 0.01, and 0.001 to observe their effects on convergence speed and training stability.
2. **Optimizer:** We compared Adam and Stochastic Gradient Descent (SGD) to determine which optimizer helps the model to converge faster.
3. **Weight Decay:** We experimented with weight decay values of 0.1, 0.01, and 0.001.

Each experimental configuration involved training the model for a fixed number of epochs while monitoring the weighted mean square loss on a validation subset. By altering one hyperparameter at a time and keeping all other factors constant, we were able to isolate its impact on performance.The model achieved the lowest weighted mean square loss when trained for 15 epochs using a learning rate of 0.001 , a weight decay of 0.01, and the Adam optimizer.

5. Results

With the chosen hyper-parameters (as mentioned in experiments section), we trained the model on 1,500 audio tracks. On the validation dataset, the model achieved a weighted mean square loss of 0.002210, indicating effective generalization and stable convergence.

6. Discussion

While the results obtained demonstrate the potential of our approach, there are several limitations and considerations that warrant discussion. First, the model's performance was assessed on a controlled and relatively limited dataset. Although this allowed us to establish proof-of-concept outcomes, applying the approach to more diverse and larger-scale audio collections may pose additional challenges. For instance, the computational cost could increase significantly, and the chosen hyperparameters or model configurations may not directly translate to new data domains. Techniques such as domain adaptation, transfer learning, or curriculum learning could be investigated to help the model generalize more effectively to novel musical genres, cultural contexts, or recording conditions.

Another limitation lies in the reliance on a single type of input representation. While spectrogram-based U-Net architectures have proven effective, certain instruments or complex audio textures may benefit from alternative or more sophisticated representations, such as mel-spectrograms, constant-Q transforms, or learned filter banks. Exploring these representations could enhance the model's ability to capture subtle timbral nuances and dynamic variations.

From a methodological standpoint, our use of energy-weighted loss functions[2] successfully addressed quieter instruments, yet it remains possible that other factors—such as temporal structure, harmonic complexity, or spatial cues—could further improve instrument separation. Investigating advanced architectures (e.g., transformers or diffusion-based models) or incorporating multi-channel input data may offer new insights.

Beyond the technical scope, the implications of this work

for music recommendation systems are significant. By focusing on instrumental composition rather than broad meta-tags like genre or artist identity, we use a axis of personalization. This suggests that emphasizing inherent musical attributes—such as instrument duration—could be beneficial in other content-based recommendation tasks

In essence, our work adds to the understanding that modeling intrinsic properties of a signal (in this case, instrumental composition) can yield valuable representations for downstream tasks. Going forward, incorporating more diverse instruments, experimenting with alternative data representations, and evaluating performance on larger and more varied datasets may broaden the applicability and impact of our approach.

7. Acknowledgment

This work was conducted using computational resources provided by the Academic & Research Computing group at Worcester Polytechnic Institute. We gratefully acknowledge their support and assistance.

8. Conclusions and Future Work

In this work, we used a spectrogram-based U-Net[1] approach to music source separation and personalized music recommendation, emphasizing the underlying instrumental composition of tracks. By employing energy-weighted loss functions[2] and carefully chosen hyper-parameters (moderate learning rate, Adam optimizer, and moderate weight decay), we achieved stable convergence and robust performance, as demonstrated by low weighted mean square loss values on the validation set. The resulting instrument embeddings effectively captured each track's instrumental signature, enabling content-based recommendations that move beyond conventional genre- or history-based methods.

Looking ahead, we intend to expand both the size and diversity of our dataset, thereby improving the model's ability to handle a broader range of musical styles and contexts. Moreover, we aim to increase the number of instruments represented within the recommendation pipeline, further enhancing the system's capacity to address varied user preferences. In addition to exploring more advanced spectral representations (e.g., mel-spectrograms or adaptive transforms) for improved separation quality, future research could integrate these embeddings with user profiles, social graphs, or other metadata, ultimately offering more nuanced and user-centric music discovery experiences.

9. References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, vol. 9351, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., in Lecture Notes in Computer Science, vol. 9351. , Cham: Springer International Publishing, 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28.

[2] V. S. Kadandale, J. F. Montesinos, G. Haro, and E. Gómez, "Multi-channel U-Net for Music Source Separation," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2020, pp. 1–6. doi: 10.1109/MMSP48831.2020.9287108.

[3] G. Meseguer-Brocal and G. Peeters, "Conditioned-U-Net: Introducing a Control Mechanism in the U-Net for Multiple Source Separations," Nov. 21, 2019, *arXiv*: arXiv:1907.01277. Accessed: Nov. 10, 2024. [Online]. Available: http://arxiv.org/abs/1907.01277

[4] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "SINGING VOICE SEPARATION WITH DEEP U-NET CONVOLUTIONAL NETWORKS".

[5] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix: A reference implementation for music source separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, 2019. Available at: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C22&q=F.-R.+St+%CC%88oter%2C+S.+Uhlich%2C+A.+Liutkus%2C+and+Y.+Mitsufuji%2C+%E2%80%9COpen-unmix-a+reference+implementation+for+music+source+separation%2C%E2%80%9D+Journal+of+Open+Source+Software%2C+vol.+4%2C+no.+41%2C+p.+1667%2C+2019.&btnG=

[6] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An Efficient Combination of Convolutional and Recurrent Neural Networks for Audio Source Separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, Sep. 2018, pp. 106–110. doi: 10.1109/IWAENC.2018.8521383.

[7] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed," Sep. 03, 2019, *arXiv*: arXiv:1909.01174. doi: 10.48550/arXiv.1909.01174.

[8] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-Learning Extractors for Music Source Separation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 816–820. doi:

10.1109/ICASSP40776.2020.9053513.