MACHINE LEARNING EVALUATION OF SONG SIMILARITY USING SONG SEGMENTATION CLASSIFIERS

Ariana Pereira

ABSTRACT

This paper explores the feasibility of using machine learning techniques to recognize which segment of a song is most similar to another input audio sample. By leveraging Support Vector Machines (SVMs), and features such as tempo, Mel-frequency cepstral coefficients (MFCCs), chroma, and spectral contrast, the research investigates whether these machine learning models can accurately identify musicological concepts in audio clips by comparing them against labeled training dataset. The study presents an experimental framework for feature extraction, model training, and evaluation, using audio data annotated with segment labels as an example proof of concept. Results demonstrate the model's potential to capture structural similarities and the basis for evaluating song similarity to inform future applications in music information retrieval.

1. INTRODUCTION

Protection of musical intellectual property from copyright infringement is a crucial part of the music industry today, and is problem set ripe for machine learning applications. Especially as generative AI models can generate brand new songs based off of existing songs, understanding which part of an original song the newly generated song is replicating and to what extent the song has been replicated is crucial to build a case for copyright infringement if the AI model output is too similar to the training data (i.e. the original song).

While exploring the capabilities of generative AI music models to produce a song based on an audio input, I questioned to what extent the musical output was similar to the original input, and if that input was a copyright song, to what extent could the AI generated output be considered plagiarism?

To answer these questions, I looked to create a proof-of-concept machine-learning driven classifier to evaluate song similarity, using an original song (with the consent of the original composer) as the training dataset. This paper evaluates how manual analysis of song similarity would compare to a machine-learning model approach. To do this, portions of the original song were categorized into five distinct song segments (or labels): Verse A, Verse B, Chorus A, Chorus B, and Outro. This paper explores whether training a machine learning model on certain musical features can reliably classify song sections, with the goal of eventually using the model to analyze how similar AI-generated outputs adhere to the musicological principles and patterns of an original song used as training data.

2. RELATED WORK

Significant research has been dedicated to the segmentation and classification of musical structures using machine learning techniques. One notable study is "Unsupervised Learning of Deep Features for Music Segmentation," by McCallum which explores the use of unsupervised learning to identify boundaries between

distinct music segments, such as verses and choruses [2]. The researchers highlight the importance of feature representation in accurately segmenting music, emphasizing that the choice of audio features significantly impacts the performance of segmentation algorithms.

To this end, ensuring the right features are used to capture identifiable characteristics of the song section is crucial to ensure accurate segmentation. A similar study was performed by Lee et al. and demonstrated the effectiveness of feature extraction techniques utilizing long-term modulation spectral analysis of both spectral and cepstral features [1]. The spectral features include Octave-based Spectral Contrast (OSC), and the MPEG-7 Normalized Audio Spectrum Envelope (NASE), while the cepstral features involve Mel-Frequency Cepstral Coefficients (MFCCs). The authors introduce the concept of modulation spectrograms, which capture the time-varying or rhythmic information of music signals. By decomposing each modulation spectrum into logarithmically spaced subbands, they compute Modulation Spectral Contrast (MSC) and Modulation Spectral Valley (MSV) for each subband. Statistical aggregations of these MSCs and MSVs across all subbands yield effective and compact features for genre classification, which could also be applied to song segmentation when operating within the context of one song. This research highlights the significance of modulation spectral analysis in capturing the dynamic aspects of music signals, offering valuable insights for tasks such as music genre classification. The integration of spectral and cepstral features, along with the use of information fusion techniques, contributes to the development of more accurate and robust music classification systems.

Machine learning methods such as support vector machines (SVMs) have been known to work well for music-related classification tasks such as genre-classification, particularly when tempo is part of the features extracted from the song [3]. The application of SVM's to structural segmentation of songs remains underexplored. McCallum employs convolutional neural networks (CNN) to learn deep feature embeddings for music segmentation in an unsupervised manner [2] and demonstrates that employing these CNN-based embeddings in a classic music segmentation algorithm significantly enhances performance, achieving state-of-the-art results in unsupervised music segmentation. In the context of one song where training data is limited, an SVM approach is more suitable. This paper builds on these foundations by applying machine learning techniques, specifically SVM's to song segmentation within the context of one song with the goal of exploring further analysis of song similarity scoring.

---

3. METHODOLOGY

3.1 Dataset

The dataset consisted of nine audio samples from an original song, categorized into 5 song sections, "Verse A", "Verse B", "Chorus A", "Chorus B", and "Outro". Each sample was approximately 30 seconds long and defined by a distinct chord progression and melody.

3.2 Feature Extraction

Four key audio features were extracted for each training audio sample:

- **Tempo:** Beats per minute (BPM) to capture rhythmic characteristics.
- **MFCCs:** Representing the timbral aspects of the audio.
- **Chroma:** Capturing harmonic content.
- **Spectral Contrast:** Differentiating tonal and non-tonal content.

See Appendix 8.1 – Feature Extraction for further details

3.3 Machine Learning Models Selection

Several machine learning models were tested to classify the audio samples by splitting the training dataset into training and validation (80% training / 20% validation):

1. **K-Nearest Neighbors (K-NN):** K-NN yielded a 100% training error on the validation dataset.
2. **Linear Regression:** Encoding the five song section classifiers and running a linear regression yielded no improvement in accuracy vs. K-NN (100% training error)
3. **Support Vector Machine (SVM):** Achieved 0% training error on the validation dataset and was utilized for further analysis.
4. **Random Forest:** Given the output of Random Forest will change each time a new selection of features is suppressed during training, the model was assessed for accuracy 10 times and achieved an overall 30% training error on the validation dataset.

See Appendix 8.2 Machine Learning Model Selection for further details

3.4 Label Back testing

Each song section category was defined by a distinct chord progression and melody line. When analyzed manually, the chord and melody pairing would be key distinctive characteristics used to discern similarity from one song to another. To evaluate the influence of a chord progression and melody pairing on the model's prediction, I re-created each training audio sample of the original song by playing the chord progression with a MIDI grand piano and played the melody line with a Vocoder sound based on the original vocal audio. These samples were then run through the model to evaluate its accuracy and understand the extent to which harmonic content like chords and melody contributed to the predictions.

3.5 Feature Visualization

Each time the model is run on an inputted audio sample, a plotted visualization of the features extracted from that sample is produced to help contextualize the prediction. Plotted visualizations for the training dataset is available in the appendix of this paper as well for comparison purposes.

---

4. RESULTS

4.1 Initial Model Performance on the Validation Dataset

While the SVM model generated 100% accuracy on the validation dataset, the visualizations of the nine training data points helped to illuminate artifacts and outliers from the feature extraction process. These factors likely contributed to skewed results when testing the model on other audio samples outside of the training dataset.

Most notably, the average tempo extracted across the training dataset was measured at ~125 bpm which was not intended. The song was composed at ~80 bpm, but the original audio recording was recorded without a metronome and features a syncopated chord striking pattern, likely influencing the irregular tempo reading.

Training Audio Sample	Tempo in BPM
Verse 1 A	119.12
Verse 2 A	120.47
Verse 1 B	128.73
Verse 2 B	131.85
Chorus 1 A	131.40
Chorus 2 A	133.49
Chorus 1 B	122.19
Chorus 2 B	126.6
Outro	109.46
Average Tempo	**124.81**

Fig. 1. Extracted tempo in BPM per audio sample in the training dataset.

See Appendix 8.3 Training Audio Sample Extracted Feature Visualization for further details

4.2 Chord and Melody-Only Analysis

Testing the SVM model on MIDI samples of isolated chord progression and melody line revealed high testing errors. Most samples were misclassified as "Verse B", likely because the chord striking pattern in Verse B is much less frequent and therefore more aligns with the way the MIDI sample was played, each chord struck only once.

The remaining error likely resulted from noise in the feature extraction data. Some notable issues which have arisen:

- **Tempo.** In the MIDI recordings, the tempo was consistently set to 80 bpm and the notes were quantized. However, the tempo extracted from the audio clips averaged closer to ~122 bpm.
- **Chroma**. The chroma features in the MIDI dataset also varied greatly compared to their respective counterparts in the training dataset, likely due to differences in the chord voicings and inversions used. For example, the Chroma visualization for Verse A in the MIDI sample indicated a strong presence of the C pitch, whereas the training dataset shows a heavier emphasis on the E pitch throughout Verse A. A similar phenomenon occurred when comparing Chorus A across the training and testing data sets.
- **MFCC.** While MFCC's have been known for their usefulness in speech recognition, the Vocoder melody lacks lyrical diction which is found in the original audio recording and training dataset and is likely contributing to the incorrect classifications.

This indicates that while chord progression and melody are important features for a human observer, it alone does not sufficiently distinguish between song sections using the selected features.

MIDI Sample Section	Extracted Tempo	Prediction
Verse A	98.80	Verse B
Verse B	130.55	Verse B
Chorus A	114.84	Verse B
Chorus B	127.89	Verse B
Outro	137.29	Verse B
Total Dataset	**Average Tempo: 121.87**	**Testing Error: 80%**

Fig. 2. Extracted tempo in BPM and classifier prediction from the SVM model

See to Appendix 8.4 MIDI Sample Extracted Feature Visualization for further details

5. Discussion

The results demonstrate that machine learning can identify recurring patterns in musical structures, though challenges remain in refining the features extracted to achieve more granular accuracy. The high error rate in chord and melody-only analysis suggests that chord voicing and rhythmic nuances play critical roles in distinguishing song sections when using the current set of features. Moreover, anomalies like the higher-than-expected BPM underscore the need for refining feature extraction techniques.

Future work could explore:

- Expanding the dataset to include more granular song subsections
- Recreating this analysis on another piece of music
- Incorporating deep learning models such as convolutional neural networks (CNNs) for more advanced feature learning
- Refining and expanding feature extraction to account for dynamics and phrasing in music or tailoring features to the style of music being tested
- Incorporating a linear regression model on top of the existing classifiers to expose the probability of each classifier and demonstrate how similar an inputted audio sample is to the overall original training dataset
- Incorporating semantic data from chord charts and lyrics as part of feature extraction *(See Appendix 8.5 – Original Composition Chord Chart and Lyrics for reference)*

6. Conclusion

This study highlights the potential of machine learning in music segmentation tasks, providing insights into the structural patterns of songs. While the SVM model showed promise, the findings underscore the importance of integrating multiple musical features for robust classification. These insights pave the way for developing tools that assist musicians and composers in analyzing and enhancing their work.

7. References

[1] Chang-Hsing Lee, Janez Kecic, and Jeng-Shyang Pan. “Automatic Music Genre Classification Based on Modulation Spectral Analysis of Spectral and Cepstral Features.” *IEEE Transactions on Multimedia*, vol. 11, no. 4, 2009, pp. 717-729. https://doi.org/10.1109/TMM.2009.2022917.

[2] Andrew McCallum. “Unsupervised Learning of Deep Features for Music Segmentation.” *International Society for Music Information Retrieval*, 2019, pp. 467-472.

[3] R. Thiruvengatanadhan. “Music Genre Classification Using SVM - IRJET.” *International Research*

Journal of Engineering and Technology (IRJET)., vol. 5 pp. 1058-1060, October 2018.

---

8. Appendix

8.1 Feature Extraction

```
import librosa
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score

# Load audio file and extract features
def extract_features(file_path):
    y, sr = librosa.load(file_path,duration=30)
    n_fft= min(2048,len(y))
    onset_env = librosa.onset.onset_strength(y=y, sr=sr)
    dtempo = librosa.feature.tempo(onset_envelope=onset_env, sr=sr, aggregate=None)
    avg_tempo = np.mean(dtempo)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13).T, axis=0)
    chroma = np.mean(librosa.feature.chroma_stft(y=y, sr=sr, n_fft=n_fft).T, axis=0)
    spectral_contrast = np.mean(librosa.feature.spectral_contrast(y=y, sr=sr, n_fft=n_fft).T, axis=0)
    return np.hstack([avg_tempo, mfcc, chroma, spectral_contrast])
```

8.2 Machine Learning Model Selection

K-NN Method

```
# Extract features
features = np.array([extract_features(file) for file in files])


# Train/test split
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

# Normalize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train k-NN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Predict and evaluate
y_pred = knn.predict(X_test)

print("Predictions:", y_pred[:])
print("True values:", y_test[:])

print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
```

K-NN Result

```
[In [1]: import KNN_Song_Component_Classifier
Predictions: ['Chorus A' 'Chorus A']
True values: ['Verse A', 'Chorus B']
Accuracy: 0.00
```

Linear Regression Approach

```
# Encode labels
label_encoder = LabelEncoder()
encoded_labels = label_encoder.fit_transform(labels)

# Extract features
features = np.array([extract_features(file) for file in files])

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(features, encoded_labels, test_size=0.2, random_state=42)

# Normalize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 7: Train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 8: Evaluate the model
y_pred = model.predict(X_test)

# Decode predictions back to labels
decoded_predictions = label_encoder.inverse_transform(y_pred.round().astype(int))
decoded_true = label_encoder.inverse_transform(y_test)


print("Predictions (decoded):", decoded_predictions[:])
print("True values (decoded):", decoded_true[:])


print(f"Accuracy: {accuracy_score(decoded_predictions, decoded_true):.2f}")
```

Linear Regression Result

```
[In [2]: import LinearRegression_Song_Component_Classifier
Predictions (decoded): ['Outro' 'Outro']
True values (decoded): ['Verse A' 'Chorus B']
Accuracy: 0.00
```

SVM Approach

```
# Extract features
features = np.array([extract_features(file) for file in files])

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

# Normalize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#Train SVM Model
model=SVC(kernel='linear')
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)

print("Predictions:", y_pred[:])
print("True values:", y_test[:])

print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
```

SVM Result

```
[In [4]: import SVM_Song_Component_Classifier
Predictions: ['Verse A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 1.00
```

Random Forest Approach

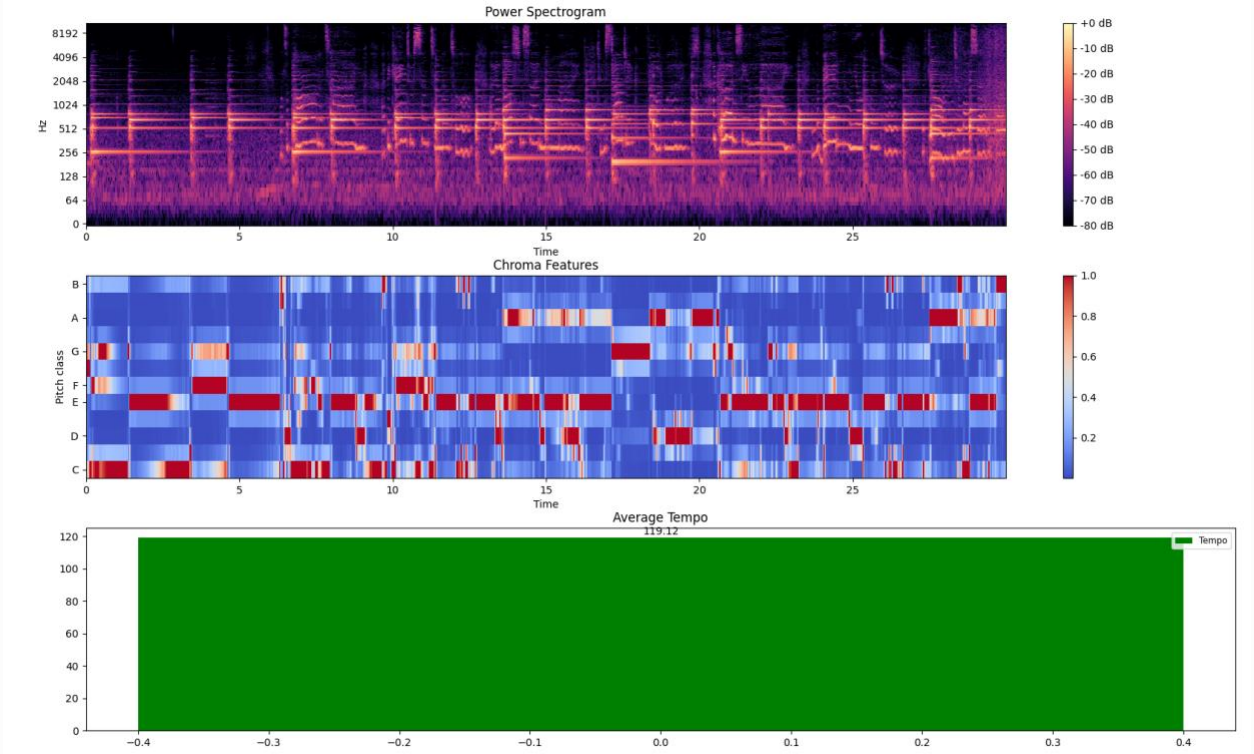
```
# Extract features
features = np.array([extract_features(file) for file in files])

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

# Normalize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#Train Random Forest
model=RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)

print("Predictions:", y_pred[:])
print("True values:", y_test[:])

print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
```

Random Forest Result – over the course of 10 code executions

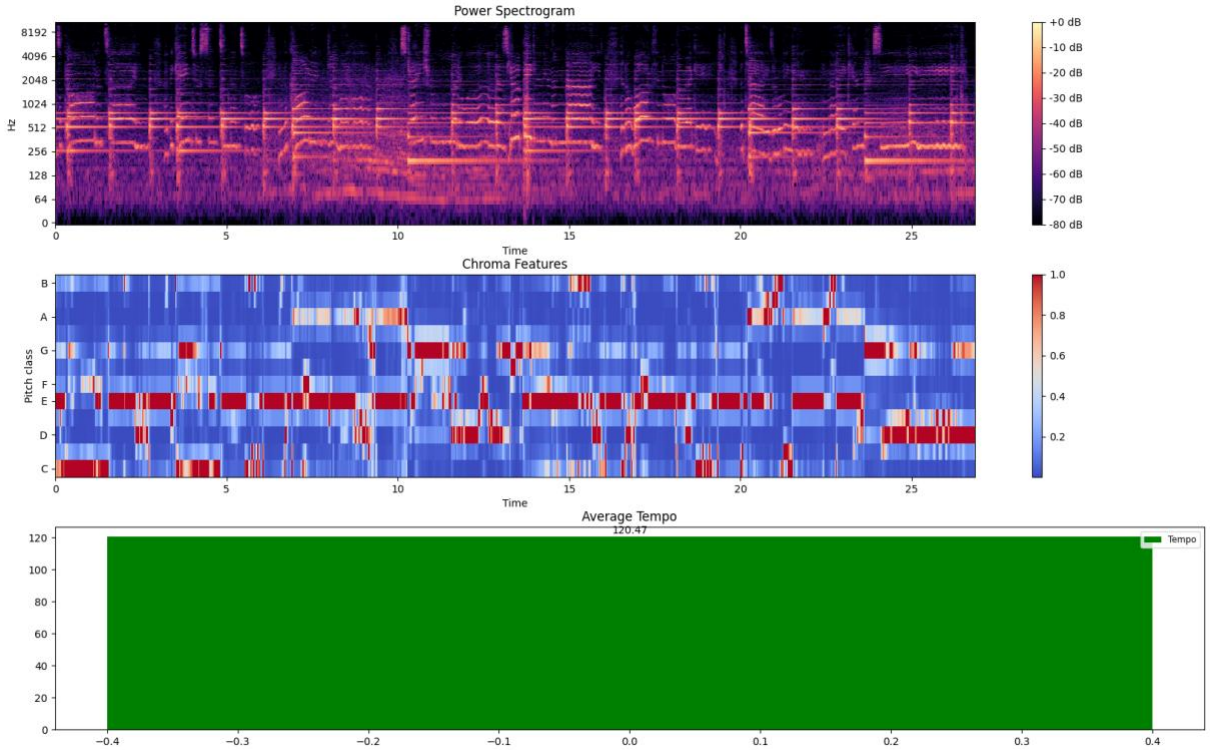
```
In [17]: import Random_Forest_Song_Component_Classifier
Predictions: ['Chorus A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 0.50

In [18]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Verse A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 1.00
Out[18]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [19]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Chorus A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 0.50
Out[19]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [20]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Verse A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 1.00
Out[20]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [21]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Chorus A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 0.50
Out[21]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [22]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Chorus A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 0.50
Out[22]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [23]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Chorus A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 0.50
Out[23]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [24]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Verse A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 1.00
Out[24]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [25]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Verse A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 1.00
Out[25]: <module 'Random_Forest_Song_Component_Classifier' from '/U

In [26]: importlib.reload(Random_Forest_Song_Component_Classifier)
Predictions: ['Chorus A' 'Chorus B']
True values: ['Verse A', 'Chorus B']
Accuracy: 0.50
```

8.3 Training Audio Samples Extracted Feature Visualization

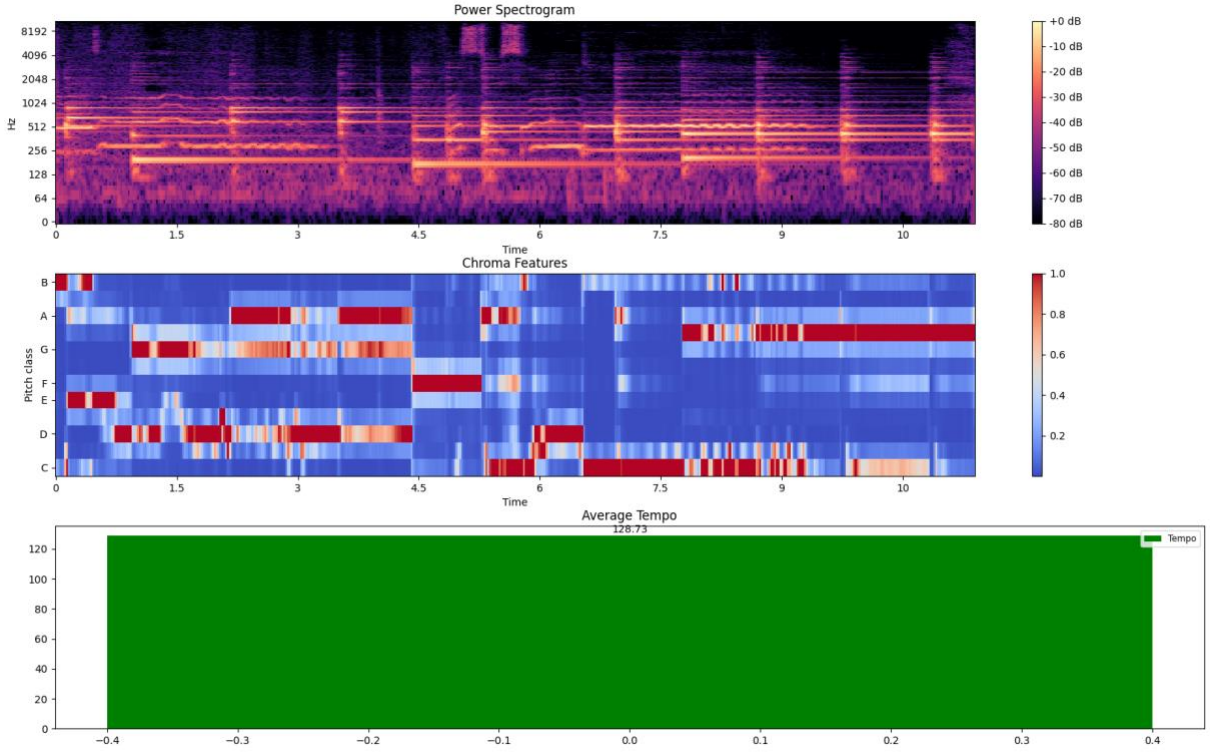
Verse 1 A



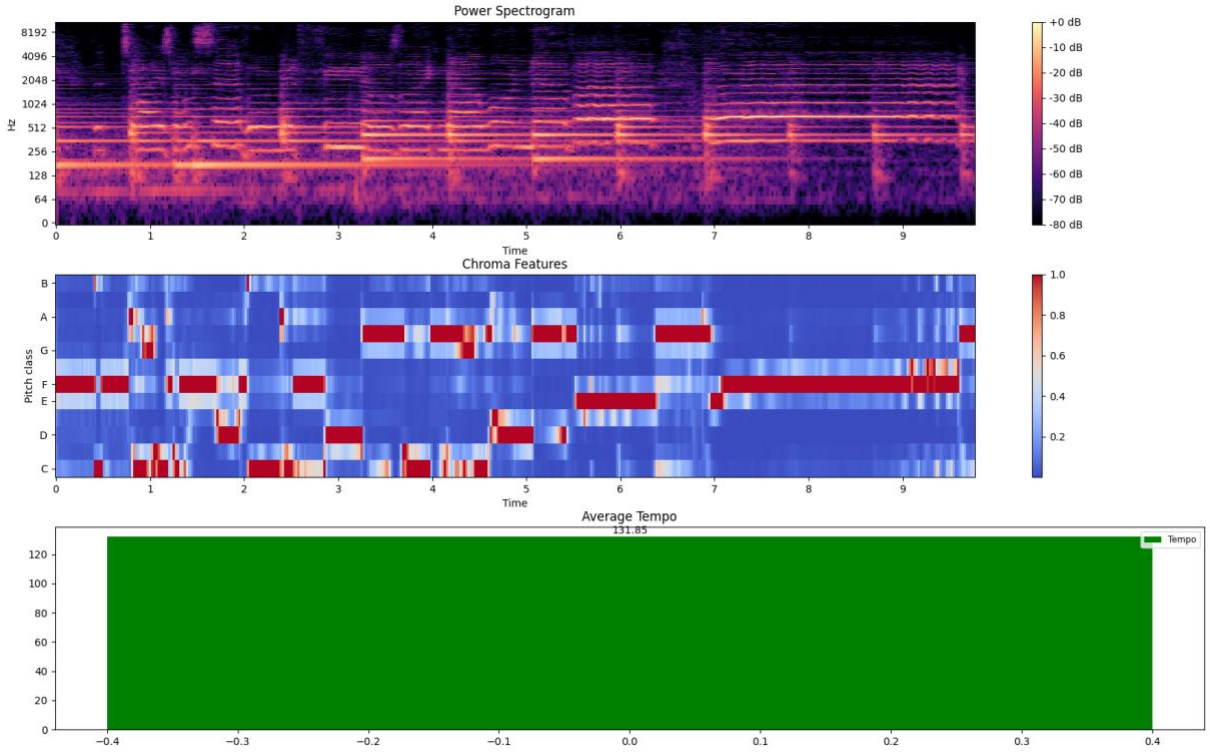
Verse 2 A



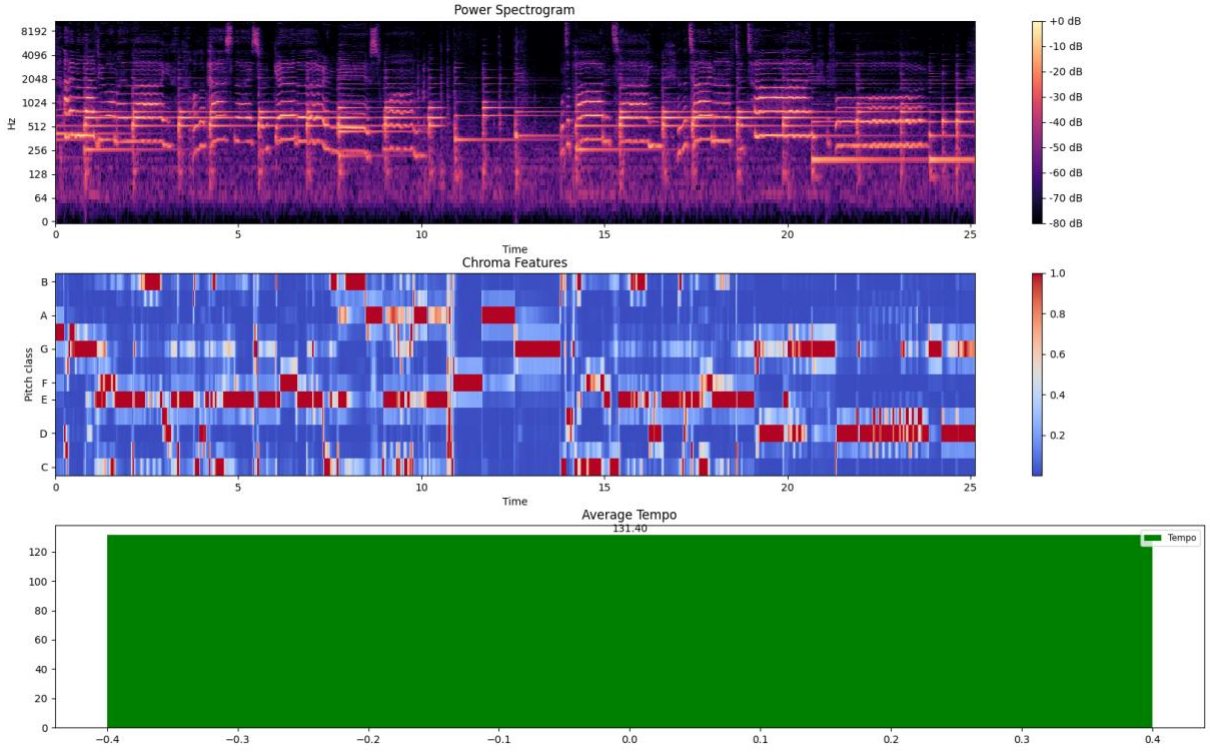
Verse 1 B



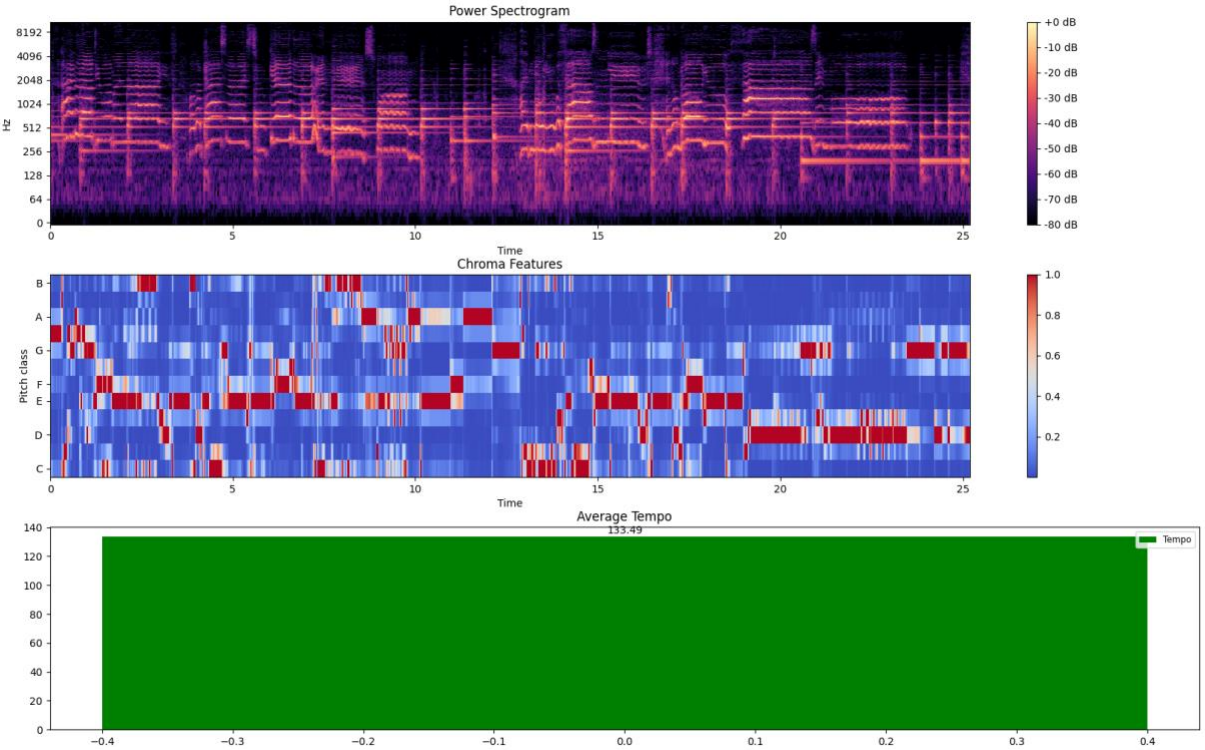
Verse 2 B



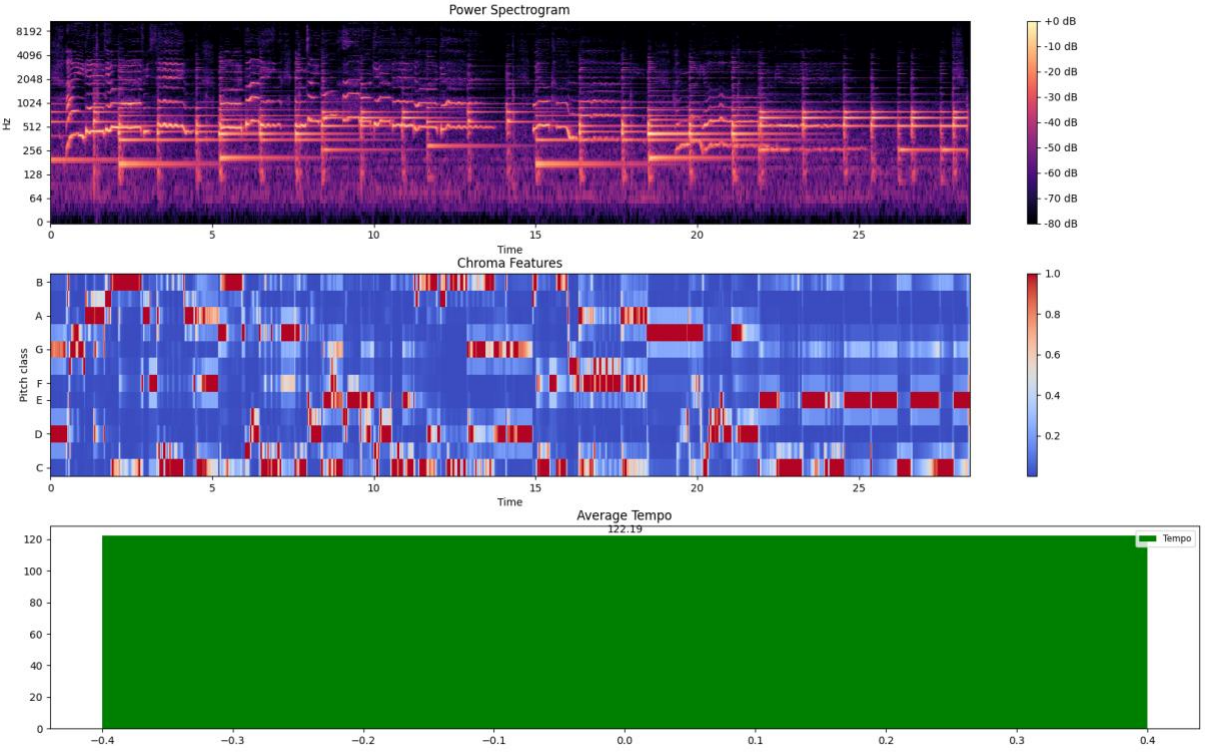
Chorus 1 A



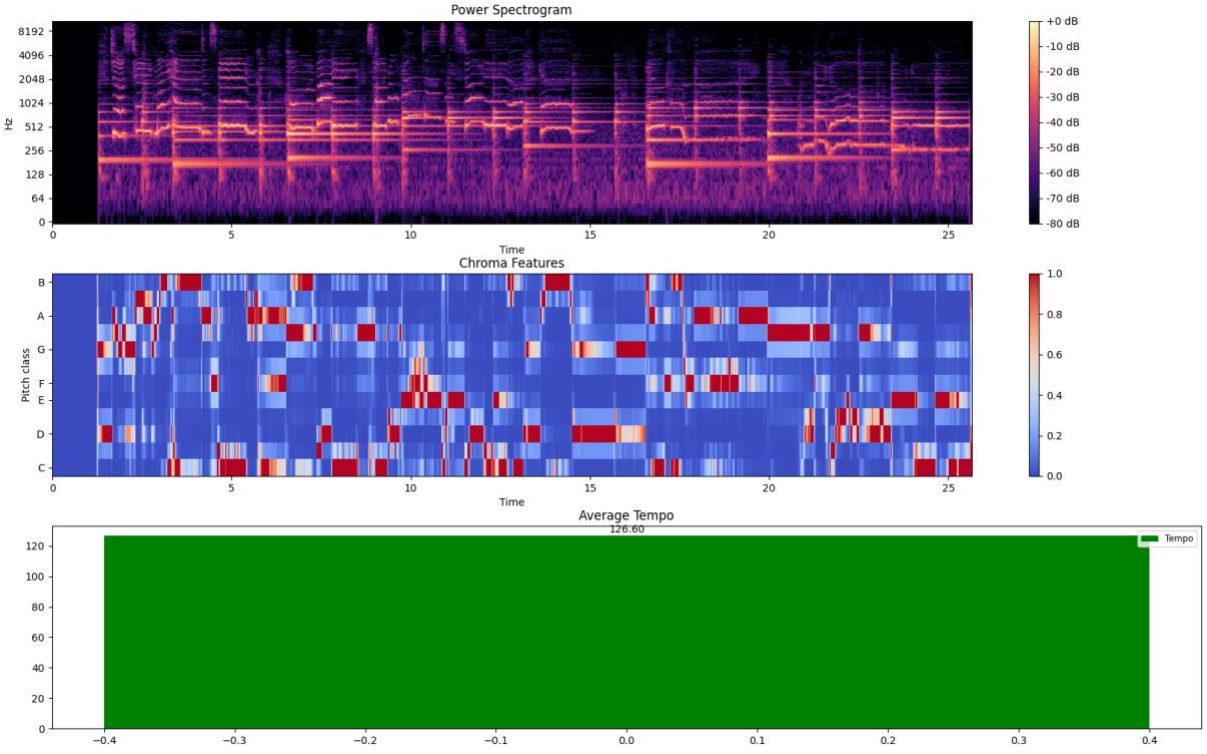
Chorus 2 A


Power Spectrogram
Hz
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB
Chroma Features
Pitch class
Time
Average Tempo
133.49
Tempo

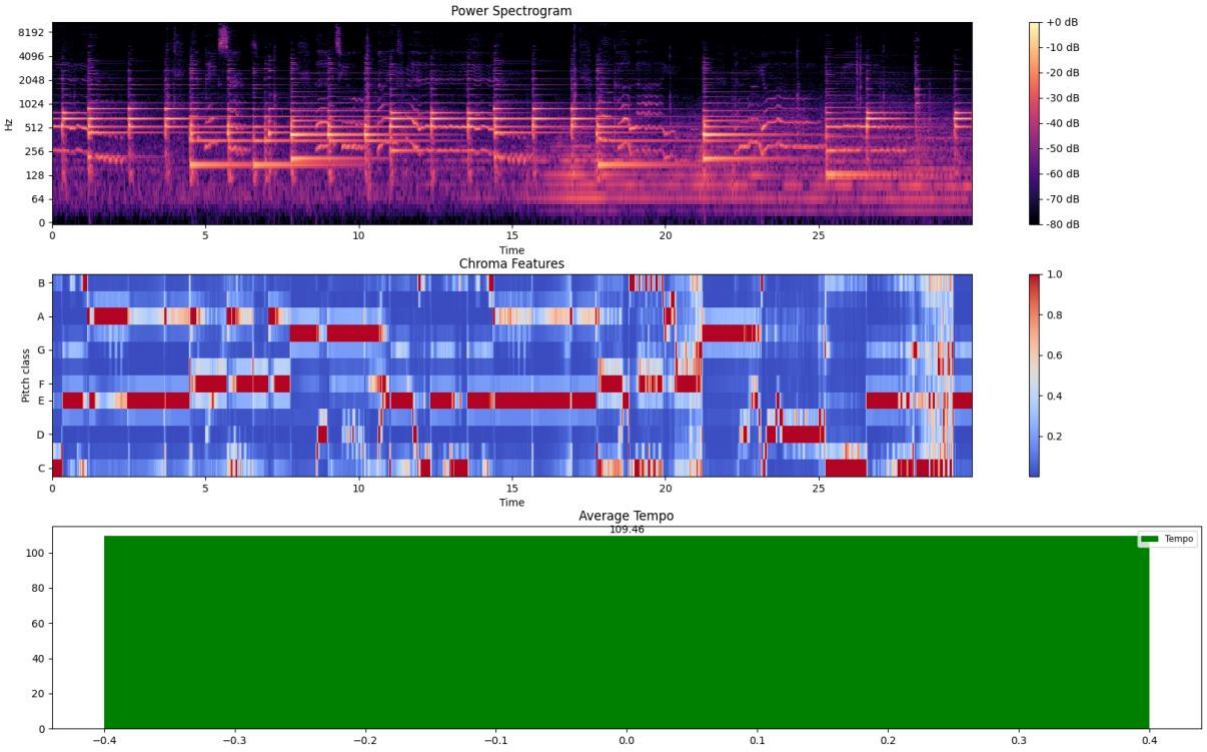
Chorus 1 B


Power Spectrogram
Hz
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB
Chroma Features
Pitch class
Time
Average Tempo
122.19
Tempo

Chorus 2 B

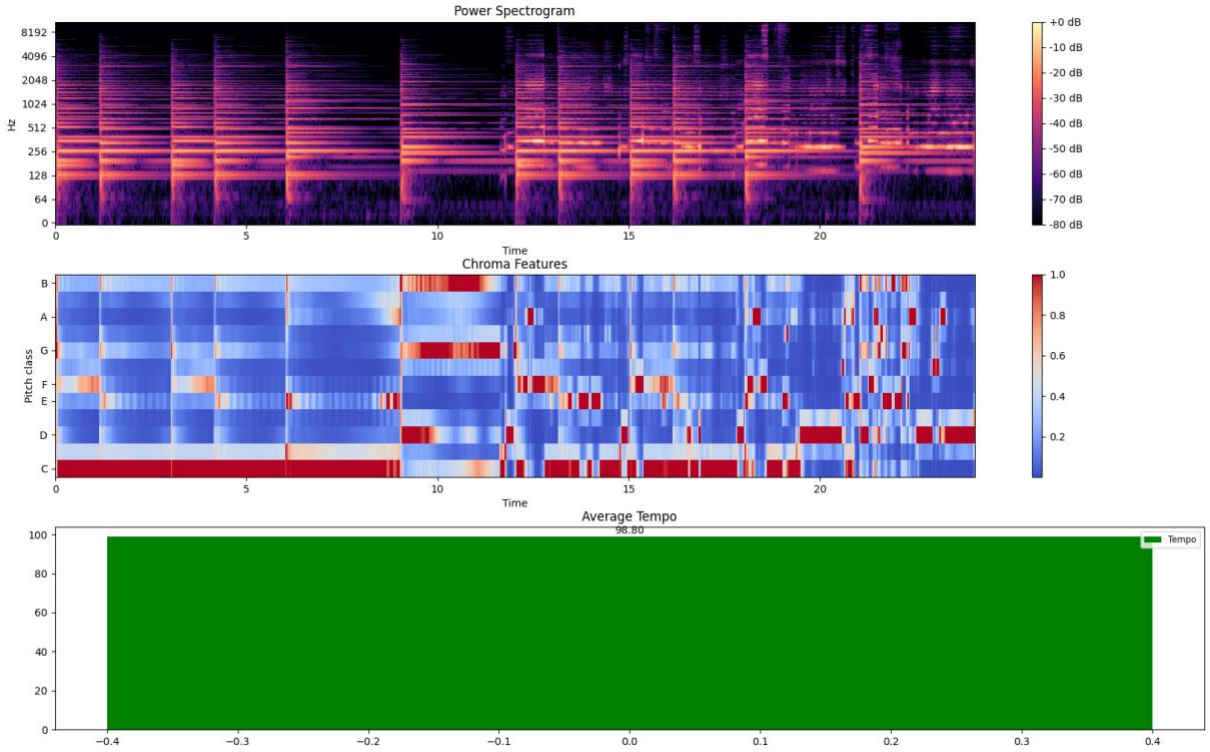


Outro

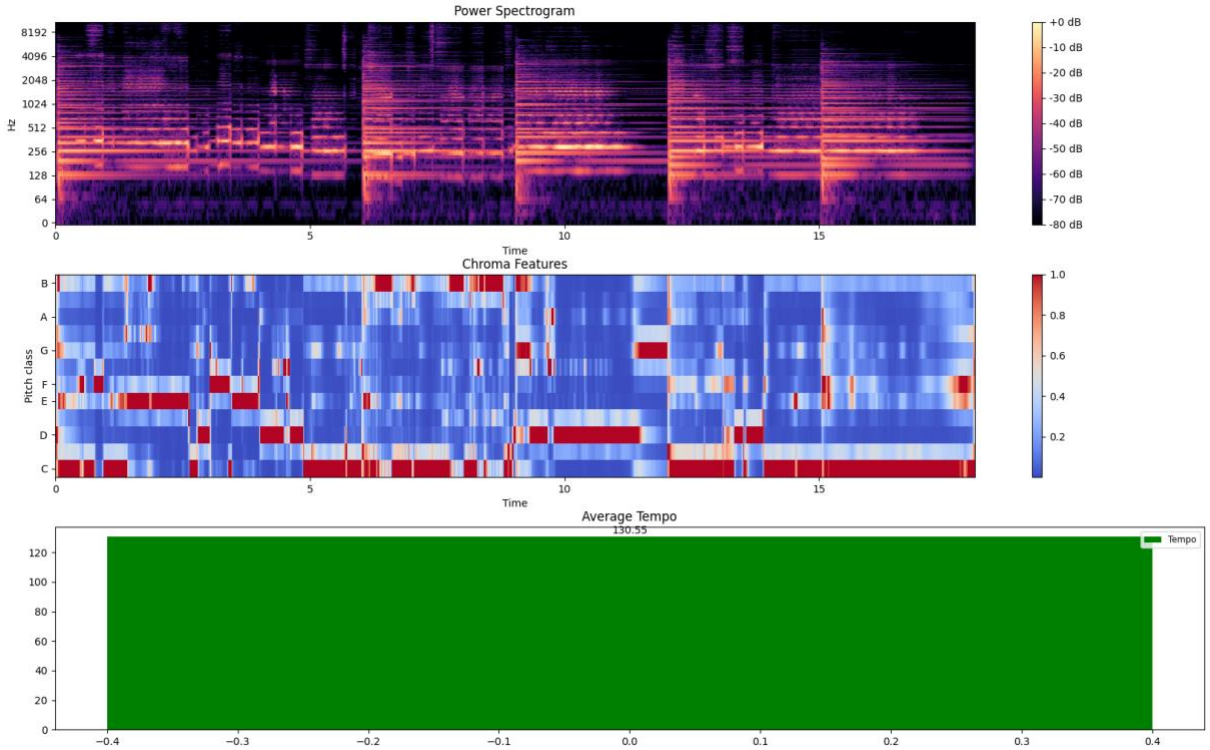


8.4 MIDI Sample Extracted Feature Visualization

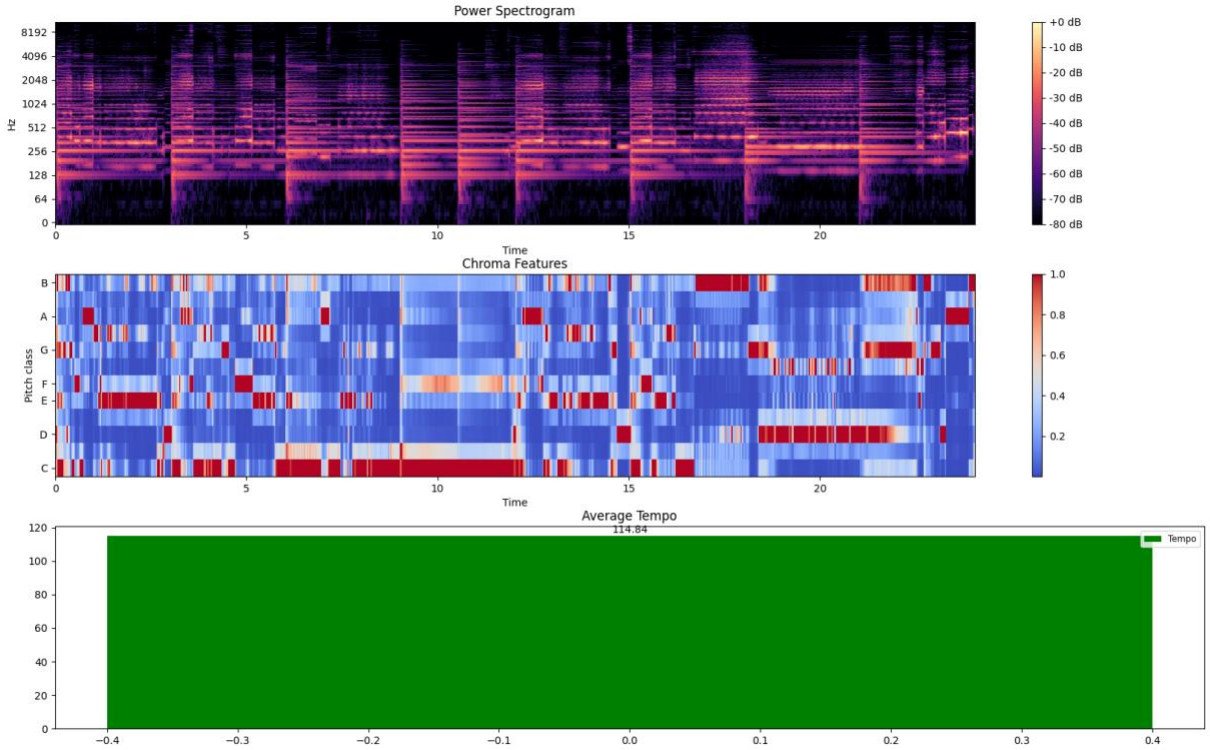
Verse A



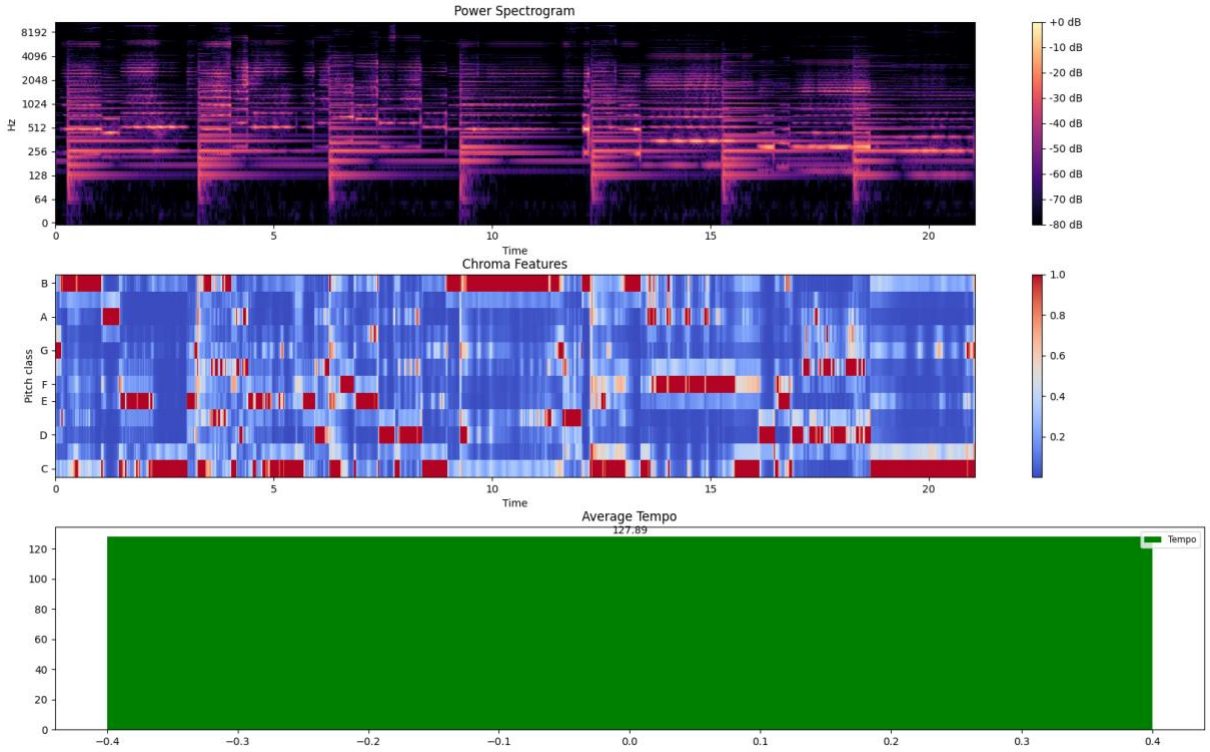
Verse B


Power Spectrogram
Hz
Time
Chroma Features
Pitch class
Time
Average Tempo
130.55
Tempo

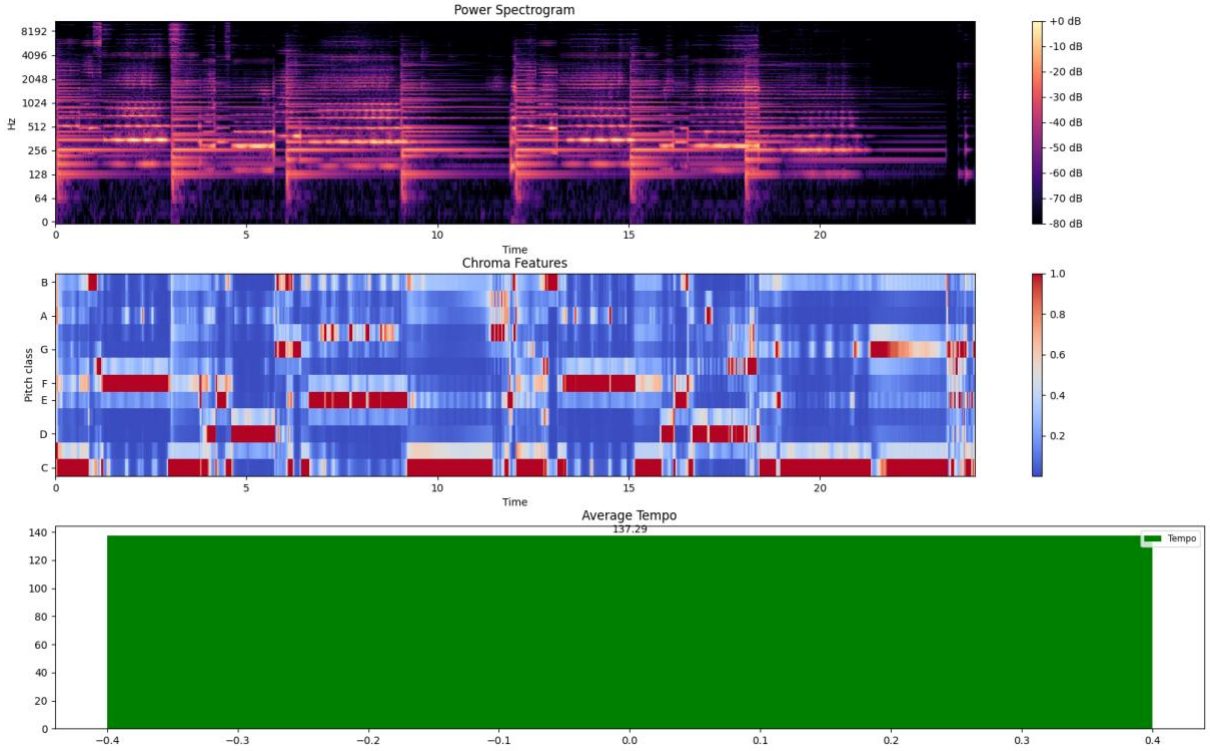
Chorus A


Power Spectrogram
Hz
Time
Chroma Features
Pitch class
Time
Average Tempo
114.84
Tempo

Chorus B



Outro



8.5 Original Composition Chord Chart and Lyrics

Lyrics by Kylie Lefkowitz
Musical Composition by Avina Pereira

Verse 1 A

Csus C Csus C Am G
[Instrumental]

Csus C
Once upon a time
Csus C
I became too sentimental
Am
I got lost inside my mind
G
Too nostalgic for what was

Verse 1 B

Csus C
I couldn't see the light
Csus C
In the bitter New York winters
Am
Withdrawn into myself
G
I was alone
F Fm
Forced to surrender

Chorus 1 A

C/E
I've loved you all my life
Am F Fm
All our past lives together and this one
C G
Once upon a time and the time after time before

Chorus 1 B

F Fm C G
I gave you everything, orbiting, and I know that I'll give it again
F Fm
we'll collide
C
When we're young again

Verse 2 A

Csus C

Once upon a time
Csus C
I became too sentimental
Am
In the carnage of my heart
G
Estranged from what it was

Verse 2 B

Csus C
You're a beacon in the night
Csus C
And a warmth in all the wreckage
Am G
The tides are rolling in and I can see

F Fm
The ties won't sever it feels

Like now or never

Chorus 2 A

C/E
I've loved you all my life
Am F Fm
All our past lives together and this one
C G
I've known you a thousands years And I'll know you 1000 more

Chorus 2 B

F Fm C G
Just take my hand and spin, orbiting, Spiral round as we start it again
F Fm
we'll collide
C Am
When we're young again

Outro

F Fm
It makes sense
C
That it's you my friend
F Fm C
We'll collide when we're young again