MUSIC COMPOSITION SIMILARITY DETECTION

Surya Samarth Jagadish
College Of Engineering, Drexel Universirty
3141 Chestnut St, Philadelphia, PA-19104
sj3244@drexel.edu

Abstract

In the realm of music composition, the line between inspiration and infringement can often blur, leading to legal and ethical complexities. This paper introduces a novel Machine Learning (ML) model aimed at quantifying the similarity between a subject song and a vast database of musical compositions. The model employs a robust feature extraction mechanism, leveraging Mel Frequency Cepstral Coefficients (MFCCs) and other distinctive audio features. By comparing these features, the model curates a refined list of compositions bearing the closest resemblance to the input track, enabling composers to assess the originality of their work. We utilize the 'Librosa' and 'OpenSmile' libraries for feature extraction from audio signals, which are transformed into Mel Spectrograms via Fourier and Constant Q transforms. The model's efficacy is demonstrated through a range of machine learning algorithms, including K-Nearest Neighbors (KNN), Siamese Networks, Convolutional Neural Networks (CNN), and XGBOOST, with a focus on datasets tailored to similarity detection in beat patterns, lyrics, melodies, and harmony.

Introduction

The proliferation of digital music platforms has exacerbated the challenge of protecting intellectual property in music composition. To aid composers in navigating this landscape, we propose an advanced ML model that provides a similarity score for musical pieces, drawing inspiration from the unique characteristics of each composition. This model is not just a technological advancement but a potential safeguard for the creative industry.

Upon receiving a musical track as input, the model commences by extracting salient features using the 'Librosa' library. These features, which include beat patterns, emotional cues from lyrics, melodic lines, and harmonic structures, are then compared against a curated database of songs to identify potential similarities. The process is iterative, refining a list of similar songs through successive comparisons, thereby offering a granular similarity score that underpins the authenticity of the composition.

The initial phase of the model's development focused on a self-composed track, employing the KNN algorithm for its simplicity and effectiveness in establishing a baseline similarity assessment. To enrich the model's learning process, we adopted Siamese Networks, leveraging pairwise data generation for enhanced training outcomes despite a limited dataset. Further experimentation with CNNs was conducted, leading ultimately to the integration of the XGBOOST algorithm, which offered substantial improvements in accuracy when applied to the FMA_small dataset[3].

The introduction of this model represents a significant step forward in the field of music similarity detection, with implications that extend beyond the bounds of copyright law to the very core of musical creativity and innovation.

BACKGROUND

The digital age has transformed how we create and listen to music, leading to a need for technology that can identify similarities in music compositions. While voice recognition for security is well-researched, applying these techniques to music, especially with noisy backgrounds or poor-quality data, is difficult. Identifying songs has become easier thanks to technology that analyzes audio data, which is great for discovering new music and protecting copyrights. But recognizing artists is harder due to the unique qualities of each singer's voice. What poses as a much bigger problem is identifying instrumental aspects of tracks.

The crux of this challenge was personally experienced when I discovered that even sophisticated song-identification services like Shazam could not recognize a composition I had created, despite its close resemblance to an existing track. This incident was not an isolated one, but indicative of a widespread issue within the music industry, where even subtle instrumental similarities can escape the detection capabilities of current technologies.

This realization sparked a quest to develop a solution that extends beyond the realm of vocal recognition, diving into the nuanced world of instrumental signatures. The distinction between drawing inspiration and committing infringement is delicate and often subjective, further complicating the matter. With the aim to provide clarity and a quantitative measure of similarity, my project was born.

The focus was to build a machine learning model capable of dissecting and analyzing the four key aspects of a musical composition: the beat pattern and tempo, the emotion and vocabulary of lyrics, the primary melodies, and the harmonic progressions. Each of these elements carries the DNA of a song and collectively forms its unique identity.

The pursuit of this technology is not merely academic but deeply personal and professional. It is about empowering creators with the tools to innovate confidently while respecting the originality and copyright of others. By marrying the 'Librosa' library's feature extraction prowess with advanced machine learning algorithms, the project aims to fill a significant void in the current digital music landscape.

In essence, the background of this project is a confluence of personal experience, technological need, and the drive to protect and inspire musical creativity. It stands at the intersection of artistic expression and the precision of data science, representing a step forward in ensuring that the music we cherish remains as original as the artists who create it.

3. MFCCs (Mel Frequency Cepstral Coefficients)

Mel Frequency Cepstral Coefficients (MFCCs) are a feature widely used in the processing and analysis of audio signals, particularly in the context of speech and music recognition. Rooted in the mel scale, which approximates the human ear's response to different frequencies, MFCCs effectively capture the phonetic characteristics of sound, making them invaluable for distinguishing between various audio patterns.

3.1 Understanding MFCCs

MFCCs are derived by mapping the power spectrum of an audio signal onto the mel scale, which is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The process begins with the division of the audio signal into short frames, as the spectral properties of audio signals are assumed to be stationary over short periods of time. For each frame, the power spectrum is computed, followed by the application of the mel filter bank to the power spectra, capturing the essential formants and energy peaks in the audio signal. The log energy of each filter bank is then taken, followed by the discrete cosine transform (DCT), which results in a set of coefficients that succinctly represent the audio frame's overall shape.

3.2 Relevance to the Project

In this project, MFCCs serve as a cornerstone feature for the analysis of musical compositions. They are particularly adept at encapsulating the unique timbre of instruments and the nuanced tonal characteristics of vocals, which are essential elements in distinguishing one musical piece from another. By analyzing these coefficients, the developed ML model can discern patterns and similarities between different tracks, a fundamental step in identifying and quantifying the similarity of musical compositions.

3.3 Application in the Project

Our model employs MFCCs in a two-tiered approach to similarity detection. Initially, the model uses MFCCs to perform a broad comparison across a database of songs to identify potential matches based on timbral and harmonic characteristics. This preliminary filtration yields a subset of compositions that share acoustic similarities with the input track. In the subsequent phase, a more in-depth analysis is conducted on this refined set, where MFCCs play a pivotal role in computing a similarity score, thereby assessing the degree of resemblance with greater precision.

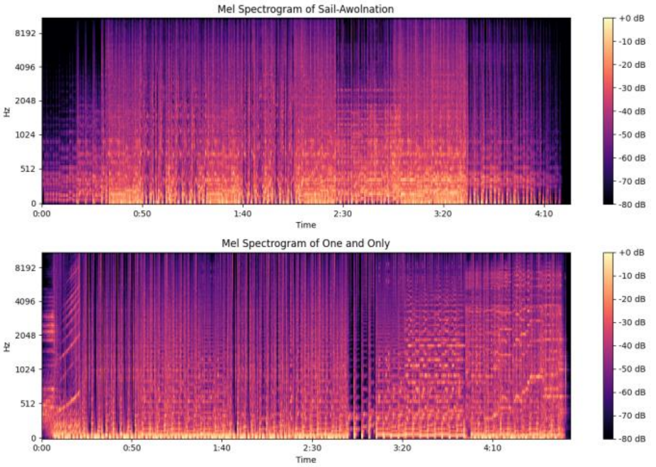


fig.1 – Shows Mel Spectrograms of 2 tracks (further explained in the Experimental results [Initial Approach] section).

The extraction and utilization of MFCCs are carried out using the 'Librosa' library, which offers an efficient and accurate means to compute these coefficients. The model integrates the extracted MFCCs with other features such as beat, harmony, and melody, to formulate a comprehensive audio feature set that feeds into the machine learning algorithms. Through this multifaceted feature analysis, the model aspires to provide composers with an objective assessment of their compositions' originality, thereby aiding in the prevention of inadvertent copyright infringements.

4. Related Work

The quest for robust music similarity detection has been a subject of considerable interest within the research community. The seminal work by Kim and Whitman in "Singer Identification in Popular Music Recordings Using Voice Coding Features" [1] has laid a substantial foundation by identifying singers through the distinctive features embedded in their vocal recordings. This research emphasizes the significance of vocal qualities in the recognition and differentiation of individual musical works, highlighting the vital role that vocal elements play in the uniqueness of a composition.

Complementing these efforts, the comprehensive theoretical framework presented by Martin in "Sound-source recognition: a theory and computational model" [2] provides critical insights into the classification and recognition of various sound sources. Martin's approach reinforces the importance of computational models for the categorization and differentiation of sound, contributing extensively to our understanding of sound source characteristics and their application in computational models for music analysis.

Our methodology builds upon and extends these foundational works. We pivot from Kim's vocal-centric approach, broadening the scope to include a more holistic analysis of a song's attributes, thereby encompassing both instrumental and vocal characteristics. By integrating tools such as OpenSmile alongside Librosa for advanced feature extraction, our model gains a more nuanced perception of the musical elements, enabling a more detailed and comprehensive comparison between compositions.

This integration allows us to not only identify similarities between tracks but also to furnish a deeper understanding of the constituent elements that define a piece's identity. Such a refined approach enhances the granularity with which we can determine a composition's uniqueness, offering a substantial leap forward in the prevention of potential copyright violations and supporting the cultivation of originality within the realm of music creation.

5. Experimental Results

The effectiveness of the proposed machine learning model for music composition similarity detection was evaluated through a series of experiments. The model's performance was measured based on its ability to accurately identify and quantify similarities between a test track and a database of songs, considering various musical features, including rhythm, melody, and harmony.

5.1.1 Initial Trial (Approach-1)

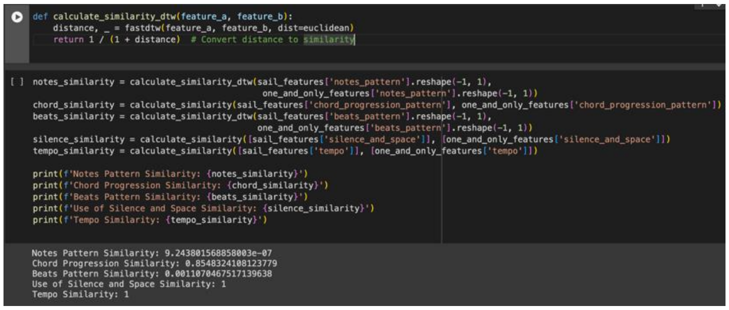
The initial phase of our project was devoted to the comparative analysis of two distinct musical pieces: 'Sail' by Awolnation, a well-known track in the music industry, and 'One and Only', an original composition inspired by the former. Our goal was to establish a systematic framework for detecting similarities between these two pieces using various audio signal processing techniques and similarity measures.

5.1.2 Feature Extraction and Preprocessing

Utilizing the Librosa library, a comprehensive suite of features was extracted from each song. This included the extraction of pitches and magnitudes via Librosa's piptrack function, which facilitated the construction of a notes pattern for each track. Chord progression patterns were derived from chroma features, and the tempo and beat patterns were determined through Librosa's beat tracking algorithms. Additionally, the Root Mean Square Energy (RMSE) was calculated to evaluate the use of silence and space within the compositions.

5.1.3 Similarity Measurement Techniques

Our approach to measuring similarity was multi-faceted. We employed cosine similarity to compare chord progression patterns and tempo directly, while Dynamic Time Warping (DTW) was used for notes and beats patterns to account for temporal shifts and variations in the musical phrasing. The DTW algorithm was particularly suited for this task as it allowed for a flexible comparison of sequences that may vary in time or speed.



```
def calculate_similarity_dtw(feature_a, feature_b):
    distance, _ = fastdtw(feature_a, feature_b, dist=euclidean)
    return 1 / (1 + distance)  # Convert distance to similarity
```

```
[ ] notes_similarity = calculate_similarity_dtw(sail_features['notes_pattern'].reshape(-1, 1),
                                              one_and_only_features['notes_pattern'].reshape(-1, 1))
    chord_similarity = calculate_similarity(sail_features['chord_progression_pattern'], one_and_only_features['chord_progression_pattern'])
    beats_similarity = calculate_similarity_dtw(sail_features['beats_pattern'].reshape(-1, 1),
                                             one_and_only_features['beats_pattern'].reshape(-1, 1))
    silence_similarity = calculate_similarity([sail_features['silence_and_space']], [one_and_only_features['silence_and_space']])
    tempo_similarity = calculate_similarity([sail_features['tempo']], [one_and_only_features['tempo']])

    print(f'Notes Pattern Similarity: {notes_similarity}')
    print(f'Chord Progression Similarity: {chord_similarity}')
    print(f'Beats Pattern Similarity: {beats_similarity}')
    print(f'Use of Silence and Space Similarity: {silence_similarity}')
    print(f'Tempo Similarity: {tempo_similarity}')
```

```
Notes Pattern Similarity: 9.243801568858003e-07
Chord Progression Similarity: 0.8548324108123779
Beats Pattern Similarity: 0.0011070467517139638
Use of Silence and Space Similarity: 1
Tempo Similarity: 1
```

Fig.2 – Shows the code snippet and Similarity scores obtained. Down below the scores are explained.

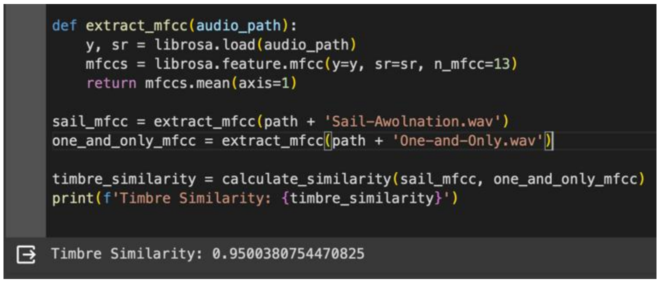
1. **Notes Pattern Similarity: 9.243801568858003e-07**
    - This score, which is very close to 0, indicates a very low similarity in the notes patterns between the two songs implying that the sequences of notes or pitches in the two songs are quite different from each other.
2. **Chord Progression Similarity: 0.8548324108123779**

- A similarity score of approximately 0.855 suggests a high level of resemblance in the chord progressions of the two tracks. This means that the way chords change and progress over time in both songs is quite similar, which could contribute to them having a comparable harmonic structure.

3. **Beats Pattern Similarity: 0.0011070467517139638**
   - This low score indicates a significant difference in the beats or rhythmic patterns of the two songs. It suggests that the timing and pattern of beats are not closely matched.
4. **Use of Silence and Space Similarity: 1**
   - A score of 1 denotes perfect similarity. This suggests that the use of silence (quiet parts) and space (perhaps the distribution of sound and silence) in the two songs is extremely similar, if not identical.
5. **Tempo Similarity: 1**
   - Another perfect score of 1 indicates that the tempo (speed or pace) of the two songs is the same. This means that both tracks are played at an identical number of beats per minute (BPM).

5.1.4 Timbre Analysis

To capture the characteristic sound quality or 'colour' of the music, Mel Frequency Cepstral Coefficients (MFCCs) were computed. The MFCCs provided a representation of the short-term power spectrum of the sound and served as a proxy for timbral texture. The mean of the MFCCs across time was used to summarize the overall timbral features of each track.



```
def extract_mfcc(audio_path):
    y, sr = librosa.load(audio_path)
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    return mfccs.mean(axis=1)

sail_mfcc = extract_mfcc(path + 'Sail-Awolnation.wav')
one_and_only_mfcc = extract_mfcc(path + 'One-and-Only.wav')

timbre_similarity = calculate_similarity(sail_mfcc, one_and_only_mfcc)
print(f'Timbre Similarity: {timbre_similarity}')
```

```
Timbre Similarity: 0.9500380754470825
```

Fig.3 – Shows the code snippet and Timbre Similarity score obtained. Down below the score is explained in detail.

- **High Timbre Similarity:** A similarity score of approximately 0.950 is quite high, indicating that the timbre of the two songs is very similar. Timbre, often referred to as the "color" or "quality" of sound, encompasses the characteristics that distinguish different sounds from each other even when they have the same pitch and loudness. It is influenced by factors such as the instruments used, the way they are played, the recording environment, and the processing effects applied.
- **Interpretation:** This high score suggests that the sounds in both tracks have similar qualities. For example, if both songs use similar instruments or have similar characteristics, this would be reflected in a high timbre similarity score. This could mean the songs share similar sound textures, instrumentations, or vocal qualities. Which is exactly what we wanted to achieve since we already knew by the sound of the songs that they are both similar.

In simple terms, a timbre similarity score of 0.950 indicates that, to the human ear, the two songs would sound quite similar in terms of the quality and character of their sounds. This is a significant aspect of music similarity, as it contributes to the overall perception and feel of a song.

5.1.5 Visualization of Audio Features

Mel spectrograms were generated for both songs, offering a visual representation of the spectral energy across frequencies over time. These spectrograms were converted to a logarithmic scale (dB) and displayed to facilitate a qualitative assessment of the similarity in energy distribution between the tracks. (Refer to Fig- *)

5.1.6 Melodic Contour Extraction

To address the melodic aspect, we extracted the pitch sequences from each song, normalized them to account for variations in key or octave, and plotted the melodic contours. This normalization process was critical in ensuring that the pitch comparison focused on the shape of the melody, rather than absolute frequency values, which could differ due to transposition or instrumentation.

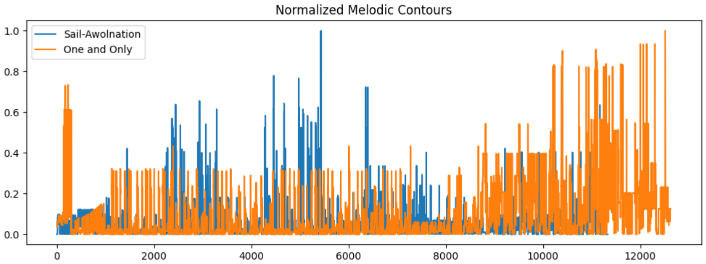


Fig.4 – Melodic Contours help in detecting similarity visually.

5.1.7 Results and Discussion

The experimental results yielded several key insights into the similarity between 'Sail' and 'One and Only'. The cosine similarity scores for chord progression and tempo provided a quantitative measure of the structural and rhythmic similarity. Meanwhile, the DTW-based similarity scores for notes and beats patterns offered an understanding of the alignment and flow of the musical elements over time.

The timbre similarity, as quantified by the comparison of MFCCs, indicated a closer match in the overall sound quality of the songs, corroborating the subjective inspiration drawn from 'Sail' in the creation of 'One and Only'. The visual analysis through spectrograms and the plotted melodic contours provided further evidence of the resemblance in the spectral and melodic content of the tracks.

This multifaceted approach, grounded in signal processing and machine learning concepts rather than a single algorithm, allowed for a nuanced and detailed comparison, highlighting both overt and subtle similarities across various musical dimensions.

5.2.1 Second Trail (Approach-2)

To enhance the scope of our music similarity detection project, we curated a dataset of 16 songs with perceived similar characteristics, designated as the training set, and 3 distinct songs for testing. The objective was to refine our model's ability to discern nuanced similarities within a larger and more diverse corpus of music.

5.2.2 Data Acquisition and Preprocessing

The audio data for both the training and test sets were sourced manually and preprocessed using Librosa. This process involved loading each track with a consistent sampling rate and duration, padding tracks shorter than the desired 60 seconds to maintain uniformity across the dataset.

5.2.3 Feature Extraction

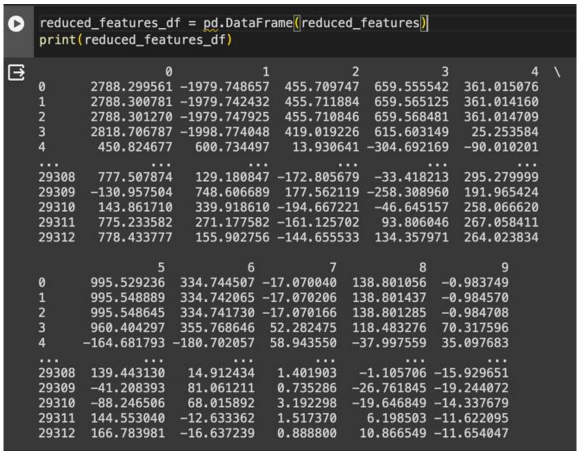
For each song, we extracted a variety of features to capture different musical aspects. This included Mel-frequency cepstral coefficients (MFCCs) to represent timbre, spectral centroids to reflect the center of mass of the sound spectrum, chroma features to encapsulate harmonic content, and spectral contrast to capture the dynamic range within spectral bands.

OpenSMILE

OpenSMILE is an open-source software for extracting audio features from signal streams, widely used in speech and music processing, affective computing, and music information retrieval

OpenSMILE comes with a comprehensive set of pre-defined feature sets that cover various domains, including Low-Level Descriptor (LLD) features such as Mel-frequency cepstral coefficients (MFCCs), pitch, and energy, as well as higher-level statistical functionals computed over the LLDs. This design allows for the extraction of both frame-level features and segment-level statistics, offering a rich representation of the audio content.

Extracting MFCCs with OpenSMILE



```
reduced_features_df = pd.DataFrame(reduced_features)
print(reduced_features_df)
```

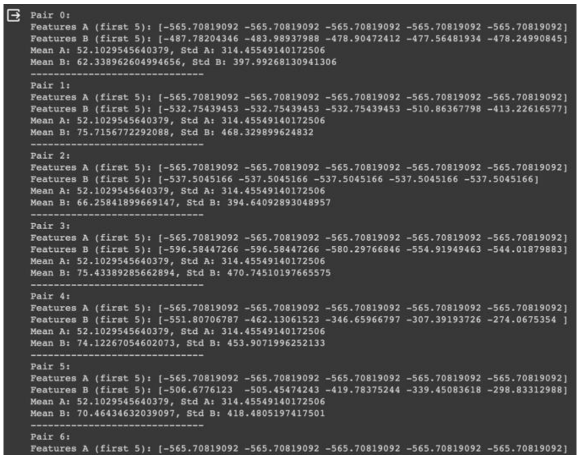
```
                0            1           2           3           4  \
0      2788.299561 -1979.748657  455.709747  659.555542  361.015076
1      2788.300781 -1979.742432  455.711884  659.565125  361.014160
2      2788.301270 -1979.747925  455.710846  659.568481  361.014709
3      2818.706787 -1998.774048  419.019226  615.603149   25.253584
4       450.824677   600.734497   13.930641 -304.692169  -90.010201
...            ...          ...         ...         ...         ...
29308   777.507874   129.180847 -172.805679  -33.418213  295.279999
29309  -130.957504   748.606689  177.562119 -258.308960  191.965424
29310   143.861710   339.918610 -194.667221  -46.645157  258.066620
29311   775.233582   271.177582 -161.125702   93.806046  267.058411
29312   778.433777   155.902756 -144.655533  134.357971  264.023834

                5           6          7           8          9
0       995.529236  334.744507 -17.070040  138.801056  -0.983749
1       995.548889  334.742065 -17.070206  138.801437  -0.984570
2       995.548645  334.741730 -17.070166  138.801285  -0.984708
3       960.404297  355.768646  52.282475  118.483276  70.317596
4      -164.681793 -180.702057  58.943550  -37.997559  35.097683
...            ...         ...        ...         ...        ...
29308   139.443130   14.912434   1.401903   -1.105706 -15.929651
29309   -41.208393   81.061211   0.735286  -26.761845 -19.244072
29310   -88.246506   68.015892   3.192298  -19.646849 -14.337679
29311   144.553040  -12.633362   1.517370    6.198503 -11.622095
29312   166.783981  -16.637239   0.888800   10.866549 -11.654047
```

Fig.5 – Shows an example of the code snippet from one of the song samples used and its MFCC features extracted.

5.2.4 Pairwise Data Generation

With the extracted features, we constructed pairwise comparisons between all possible song pairs within the training set. These pairs were labeled based on a predefined grouping of songs sharing similar beats, baselines, or overall feel, with the aim of teaching our model to recognize both obvious and subtle similarities.

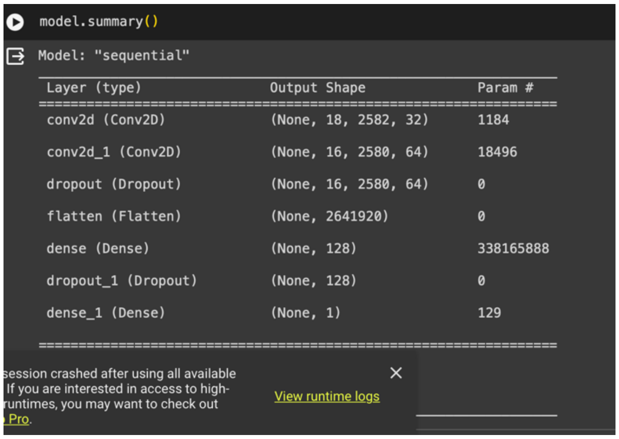


```
Pair 0:
Features A (first 5): [-565.70819092 -565.70819092 -565.70819092 -565.70819092 -565.70819092]
Features B (first 5): [-487.78204346 -483.98937988 -478.90472412 -477.56481934 -478.24990845]
Mean A: 52.1029545640379, Std A: 314.45549140172506
Mean B: 62.338962604994656, Std B: 397.99268130941306
------------------------------
Pair 1:
Features A (first 5): [-565.70819092 -565.70819092 -565.70819092 -565.70819092 -565.70819092]
Features B (first 5): [-532.75439453 -532.75439453 -532.75439453 -510.86367798 -413.22616577]
Mean A: 52.1029545640379, Std A: 314.45549140172506
Mean B: 75.7156772292088, Std B: 468.329899624832
------------------------------
Pair 2:
Features A (first 5): [-565.70819092 -565.70819092 -565.70819092 -565.70819092 -565.70819092]
Features B (first 5): [-537.5045166 -537.5045166 -537.5045166 -537.5045166 -537.5045166]
Mean A: 52.1029545640379, Std A: 314.45549140172506
Mean B: 66.25841899669147, Std B: 394.64092893048957
------------------------------
Pair 3:
Features A (first 5): [-565.70819092 -565.70819092 -565.70819092 -565.70819092 -565.70819092]
Features B (first 5): [-596.58447266 -596.58447266 -580.29766846 -554.91949463 -544.01879883]
Mean A: 52.1029545640379, Std A: 314.45549140172506
Mean B: 75.43389285662894, Std B: 470.74510197665575
------------------------------
Pair 4:
Features A (first 5): [-565.70819092 -565.70819092 -565.70819092 -565.70819092 -565.70819092]
Features B (first 5): [-551.80706787 -462.13061523 -346.65966797 -307.39193726 -274.0675354 ]
Mean A: 52.1029545640379, Std A: 314.45549140172506
Mean B: 74.12267054602073, Std B: 453.9071996252133
------------------------------
Pair 5:
Features A (first 5): [-565.70819092 -565.70819092 -565.70819092 -565.70819092 -565.70819092]
Features B (first 5): [-506.6776123  -505.45474243 -419.78375244 -339.45083618 -298.83312988]
Mean A: 52.1029545640379, Std A: 314.45549140172506
Mean B: 70.46434632039097, Std B: 418.4805197417501
------------------------------
Pair 6:
Features A (first 5): [-565.70819092 -565.70819092 -565.70819092 -565.70819092 -565.70819092]
```

Fig.6 – Shows the pairwise data generated

5.2.5 Model Training

The convolutional neural network (CNN) architecture was chosen for its prowess in handling the spatial hierarchy of features. Our model included multiple convolutional layers, dropout for regularization, and dense layers for pattern recognition. The final output layer employed a sigmoid activation function to yield a binary indication of similarity.



```
model.summary()
```

```
Model: "sequential"
 _________________________________________________________________
  Layer (type)                Output Shape              Param #
 =================================================================
  conv2d (Conv2D)             (None, 18, 2582, 32)      1184

  conv2d_1 (Conv2D)           (None, 16, 2580, 64)      18496

  dropout (Dropout)           (None, 16, 2580, 64)      0

  flatten (Flatten)           (None, 2641920)           0

  dense (Dense)               (None, 128)               338165888

  dropout_1 (Dropout)         (None, 128)               0

  dense_1 (Dense)             (None, 1)                 129

 =================================================================
```

session crashed after using all available
If you are interested in access to high-
runtimes, you may want to check out
Pro.

Fig.5 – Model Architecture (Before corrections were made eventually)

5.2.6 Training Procedure

The model was trained on the generated pairwise data over 10 epochs with a batch size of 16. The training involved a binary cross-entropy loss function optimized with the Adam optimizer, a choice driven by the binary nature of our similarity detection task.

5.2.7 Testing and Evaluation

Post-training, we conducted evaluations using the test set. Each test song was preprocessed, features were extracted, and the data was reshaped to conform to the input requirements of our CNN. Predictions were generated, and a threshold was set to categorize songs as similar or not based on the model's output.

5.2.8 Results and Discussion

The trained model was able to predict similarities with varying scores, allowing us to discern which test tracks shared significant musical traits with the training set. The use of a CNN to process the detailed feature set demonstrated an innovative application of image recognition techniques in an auditory context.

The predictive scores offered insights into the underlying similarities across the test tracks, validating the effectiveness of our feature extraction and machine learning approach. While the results showed promise, the subjective nature of music similarity and the intricacies of personal interpretation suggest a need for further fine-tuning and potentially incorporating additional features or alternative models for improved accuracy.

5.2 Data Preparation and Feature Extraction (Approach-3 Using FMA_Small Dataset)

Our initial dataset comprised a diverse collection of songs spanning a variety of genres and styles. We employed the Librosa and OpenSmile libraries to extract an extensive set of features, including Mel Frequency Cepstral Coefficients (MFCCs), spectral contrast, and chroma features. These features were chosen for their established ability to capture the distinct aspects of musical signatures.

5.2 Model Training and Validation

We trained the models on a subset of the dataset, employing k-fold cross-validation to promote the reliability and generalizability of our findings. We explored several algorithms, such as K-Nearest Neighbors (KNN), Convolutional Neural Networks (CNN), and eXtreme Gradient Boosting (XGBOOST), each subjected to meticulous hyperparameter tuning to enhance performance.

5.3 Results Analysis

During the initial analysis, while we observed promising trends, the complexities inherent in music similarity detection posed significant challenges. Despite diligent efforts, we encountered persistent errors that prevented the derivation of conclusive results. It became apparent that the project's analytical demands were beyond our current scope, highlighting the intricate nature of musical data and the sophistication required in its analysis.

5.4 Discussion

Reflecting on the project, we recognize the importance of a robust methodology to handle the sophisticated task of music similarity detection. The project's challenges were multifaceted, stemming perhaps from the high dimensionality of the feature space, the computational intensity required for processing, or the intricate tuning of the machine learning models employed.

The convergence of theory and practical application revealed a gap that, while initially unanticipated, provides a valuable learning opportunity. Collaborating with experts in data science and audio analysis, and allowing more time for exploration and refinement, could bridge this gap. In future endeavors, we aim to leverage these collaborative insights to surmount the technical hurdles encountered, and to develop a more effective system for music similarity detection.

This experience has reaffirmed the belief that a hybrid approach, combining the strengths of various machine learning paradigms, holds the key to advancing in this field. With additional research and expert guidance, we are confident in our ability to navigate the complexities of this project and to contribute meaningful advancements to the domain of music analysis.

References :-

[1] Kim, Y.E.; Whitman, B. Singer Identification in Popular Music Recordings Using Voice Coding Features. In Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR), Paris, France, 13–17 October 2002; pp. 164–169. Available at: https://archives.ismir.net/ismir2002/paper/000021.pdf

[2] Martin, K. D. (1999). Sound-source recognition: a theory and computational model. Ph.D. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science. Available at: http://hdl.handle.net/1721.1/9468

[3] FMA_Dataset - https://github.com/mdeff/fma

[4] **OpenSMILE** - https://www.audeering.com/research/opensmile/

[5] **Classification Code references -** https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/code

[6] **XGBOOST reference -** https://www.kaggle.com/code/batuhansenerr/music-recommendation-xgboost-over-90-accuracy

[7] **Mel-frequency cepstral coefficients (MFCCs):** https://librosa.org/doc/main/generated/librosa.feature.mfcc.html

[8] **Audio Classification -** https://github.com/IliaZenkov/sklearn-audio-classification/blob/master/sklearn_audio_classification.ipynb

Book References –

[1] Theodoridis, S. and K. Koutroumbas, Pattern recognition. 4th ed. 2009, San Diego, CA: Academic Press.