DUAL MODE HAPTIC FEEDBACK GLOVE FOR ENHANCED GUITAR LEARNING

Surya Samarth Jagadish , Jayanth Suryaprakash
College Of Engineering, Drexel Universirty
3141 Chestnut St, Philadelphia, PA-19104
sj3244@drexel.edu , jas987@drexel.edu

Abstract

The "Dual-Mode Haptic Feedback Glove for Enhanced Guitar Learning" project combines haptic feedback with AI to transform guitar learning. The glove delivers real-time tactile feedback, offering positive reinforcement for correct notes and guidance for errors, which could reduce dropout rates among beginners. This paper outlines the development process, including the ROS/Gazebo setup and the creation of an ML model that assesses guitar note accuracy against a pentatonic note database. Utilizing 'Librosa' and 'OpenSmile' for feature extraction, the model analyzes audio signals transformed into Mel Spectrograms. The model's performance, verified using algorithms such as KNN, Siamese Networks, CNNs, and XGBOOST, focuses on similarity detection across various musical elements.

1. Introduction

Learning to play the guitar is a rewarding yet challenging endeavor that many find overwhelming. Traditional learning methods can often be monotonous and require significant investment of time, leading to high attrition rates among beginners. Inspired by advanced haptic and robotic systems portrayed in popular media and actualized in projects like MIT's "Move Me", our project presents a novel solution: a dual-mode haptic feedback glove that leverages the power of AI to streamline and enhance the guitar learning process. This glove is set to provide real-time guidance, correcting and confirming note execution through tactile sensation, thus embedding proper techniques into muscle memory more effectively.

Upon receiving a pentatonic audio 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 pentatonics to identify potential similarities. The process is iterative, refining a list of similar scales through successive comparisons, thereby offering a granular similarity score that underpins the authenticity of the users playing.

The initial phase of the model's development focused on a self-composed/played pentatonic scale audio 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 was in hopes to improve the accuracy.

The introduction of this model represents a significant step forward in the field of guitar learning.

2. Background

This project is underpinned by the challenges faced in conventional guitar training methods - their lack of interactivity and slow feedback mechanisms. Our background research delves into cognitive and motor skill learning theories, establishing the need for rapid and accurate feedback in skill acquisition. The project draws inspiration from existing technologies in haptic feedback and artificial intelligence, examining their application in educational contexts and identifying a gap in real-time, dual-mode feedback systems in music education. The following explains some key aspects which are crucial for this project.

2.1. MFCCs (Mel Frequency Cepstral Coefficients)

MFCCs, or Mel Frequency Cepstral Coefficients, represent a critical aspect of audio signal processing, especially in speech and music recognition tasks. They are grounded in the mel scale, a concept that mirrors how the human ear perceives different frequencies. By accurately reflecting the phonetic elements of sound, MFCCs are exceptionally useful for identifying diverse audio patterns.

2.2.Grasping the Concept of MFCCs

MFCCs emerge from transforming the power spectrum of an audio signal to fit the mel scale. This scale is uniquely designed based on human perception, where pitches are deemed equidistant by listeners. The method initiates by segmenting the audio signal into brief segments, premised on the idea that audio signals maintain stable spectral characteristics over short spans. For each segment, its power spectrum is calculated. Subsequently, this spectrum is filtered through the mel filter bank, isolating significant vocal tract resonances and energy concentrations within the signal. Following this, the logarithmic energy of each filter output is computed, and a discrete cosine transform (DCT) is applied. This sequence culminates in a set of coefficients that concisely encapsulate the distinct contour of the audio segment.

2.3.Relevance to our project

Within our project, MFCCs are pivotal in analyzing musical pieces. Their effectiveness lies in capturing the distinct timbre of instruments and the subtle qualities of different tones, crucial for differentiating instrumental compositions. Utilizing these coefficients allows our machine learning model to identify and compare patterns across various scales of the instrument guitar. This capability is instrumental in our objective to measure how closely instrumental pieces resemble each other, which helps us detect whether the user is actually playing the right set of scales/notes.

2.4.Application in the Dual Mode Haptic Glove Project

The integration of MFCCs into our Dual Mode Haptic Glove project encompasses a layered method to enhance tactile feedback. Initially, MFCCs are utilized to map the sonic profile of interactions, facilitating a broad spectrum analysis that allows the glove to recognize distinct textures and surfaces based on their acoustic signatures. This preliminary stage acts as a filter, segregating potential tactile patterns that the glove can simulate.

Following the initial phase, a more detailed examination is carried out. Here, MFCCs are essential in refining the haptic glove's ability to reproduce textures with higher fidelity. By analyzing the similarity scores of the MFCCs, the glove's feedback mechanisms are fine-tuned to replicate the nuances of different materials and surfaces with greater accuracy.

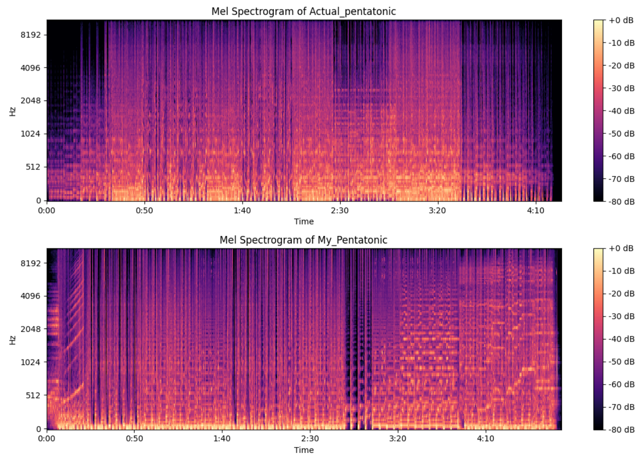


fig.1 – Shows Mel Spectrograms of 2 tracks (further explained in the Experimental results [Initial Approach] section).

The utilization of MFCCs is executed through advanced signal processing libraries, integrating them with other sensory input parameters such as pressure variance and surface temperature gradients. This combination creates a sophisticated feature set for the haptic glove, enabling it to deliver a comprehensive and realistic touch experience. With this detailed sensory input, the haptic glove is not only a tool for interaction but also becomes an advanced system for perceiving the subtleties of our physical world, providing users with a more immersive and tactilely rich experience.

Initially, the system leverages MFCCs for a general analysis across a collection of sound data, pinpointing potential matches by evaluating timbre and harmonic features. This initial screening produces a group of sounds that exhibit acoustic qualities similar to the sample input. In the following stage, a thorough investigation is performed on this curated selection. Here, MFCCs are crucial in determining a similarity index, thus providing a more exact measure of the acoustic likeness.

The process of extracting and applying MFCCs is executed through the 'Librosa' library, renowned for its precise and efficient computation of these coefficients. Our model integrates the extracted MFCCs with additional features such as harmony, melody and many more in order to formulate a comprehensive audio feature set that feeds into the machine learning algorithms. Through this multifaceted feature analysis, the model aspires to provide aspiring guitar learners with an objective assessment based on their playing style.

2.5.ROS/Gazebo Integration

- Initially, ROS provided the communication infrastructure for sensor data processing. Gazebo was used for its 3D simulation capabilities to prototype haptic interactions. However, due to frequent crashes and instability, we had to consider alternative solutions.

2.6.Shift to Construct . ai

- We adopted 'construct.ai' for its stability and comprehensive simulation services. This web-based platform allowed for continuous development cycles and provided a diverse range of testing environments without the instability issues of ROS/Gazebo.

2.7.Blender Implementation

- Utilizing Blender's sophisticated 3D modeling tools, we crafted a detailed model of the glove. This was crucial for accurate positioning of the embedded sensors and actuators, ensuring the fidelity of our haptic feedback mechanisms in the simulations.

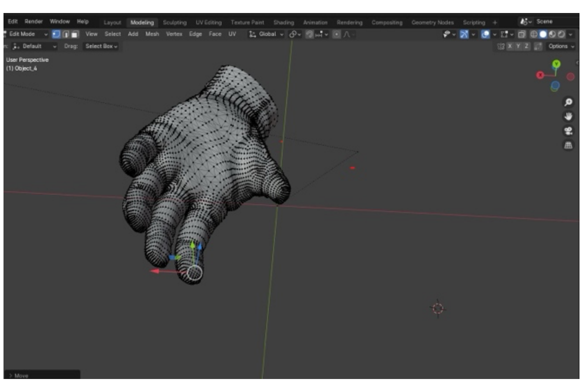


Fig 2. Illustrates 3D hand glove we created on Blender. (Angle-1)

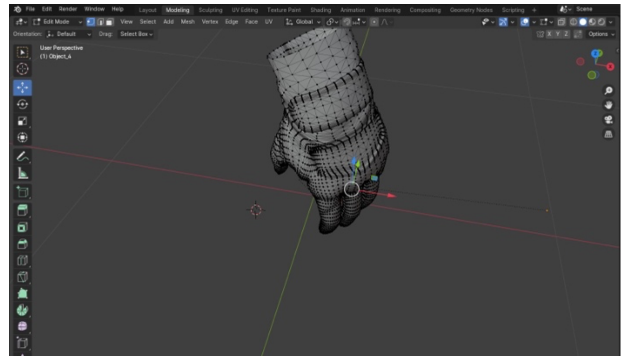


Fig 3. Illustrates 3D hand glove we created on Blender. (Angle-2)

2.8.Architecture and Technicalities

- Our system architecture incorporates a blend of physical sensors in the glove, data processing units, and feedback mechanisms. Sensor data is transmitted to a processing unit, which employs machine learning algorithms to interpret the gestures and command the haptic feedback actuators accordingly.

3. Related Work (add the links to previous work)

Our research draws from the evolving landscape of haptic technology and the rich tapestry of studies that contribute to this dynamic field, which intersects the spheres of human-computer interaction and tactile sensory augmentation. A particularly significant project that has impacted our development is MIT's "MoveMe," a venture that integrates 3D haptic support with expertly pre-recorded movements to instruct novices. This endeavor has been instrumental in advancing the synchronization of kinesthetic and tactile feedback, showcasing the capacity for intricate physical interactions to be digitally replicated and enhanced.

The "MoveMe" initiative is not just a reference point for our dual mode haptic glove project but also a foundation that, while comprehensive, has been identified to lack in areas our project targets. Notably, it does not incorporate AI-driven personalization or the nuanced dual feedback system that our proposed model integrates. Our glove design aims to blend passive and active haptic feedback to create a more immersive user experience in virtual environments. Inspired by the dual modes used in "MoveMe" to simulate various textures and resistances, we have incorporated these insights into the development of our glove's mechanisms.

In alignment with our literature review, which emphasizes the potential of machine learning to create adaptive and personalized learning environments, our project extends beyond "MoveMe" by leveraging AI to tailor the haptic learning experience to individual users. Through this approach, the glove not only simulates touch and provides force feedback but also adapts in real-time to the user's interactions.

Other works in the field, including those employing vibrotactile feedback and proprioceptive illusions, have also informed our approach, particularly in fine-tuning tactile sensations and understanding cognitive aspects of touch. This knowledge has been pivotal in designing our glove with advanced textile sensors and actuators.

By integrating machine learning algorithms, our dual mode haptic glove not only responds with enhanced sensitivity but also learns and adjusts to complex tactile stimuli, ensuring a high-fidelity haptic experience. This project not only compares favorably with systems like "MoveMe" but also innovates by adding a layer of machine learning-driven adaptability. As we forge ahead, we endeavor to enrich the corpus of knowledge in haptic technology, furthering the capabilities of touch-based interactions and underscoring the transformative influence of the "MoveMe" project on our work.

4. Operating System Selection

Ubuntu 20.04 LTS: A Foundation for Stability
Our project necessitates a reliable and secure operating system, which led us to select Ubuntu 20.04 LTS (Focal Fossa). Its robustness and extended support period make it an ideal choice for complex robotics projects requiring a long-term stable platform.

5. Robotics Software Framework

Integration of ROS Noetic Ninjemys - To facilitate our robotics development, we integrated the Robot Operating System (ROS) Noetic Ninjemys. As the last ROS 1 version compatible with Ubuntu 20.04, it brings a wealth of robotics software frameworks to our development toolkit.

6. Installation Process

Setting Up the ROS Repository - The project's initial setup involved configuring Ubuntu to recognize the ROS repository, ensuring access to a wide array of up-to-date ROS packages. This was meticulously achieved by adding the repository to Ubuntu's sources list and securely importing the ROS repository's authentication keys.

Refreshing the System - A system-wide update followed, refreshing the package index. This step was crucial for maintaining access to the latest software releases and ensuring compatibility with our development requirements.

Choosing the Right ROS Package - For comprehensive support in our robotics venture, we selected the full desktop version of ROS Noetic. This package not only includes core ROS functionalities but also additional tools such as Gazebo for simulation, Rviz for visualization, and various libraries, expanding our project's capabilities from simulation to real-world deployment.

7. Development Environment Optimization

Automating ROS Variables - To streamline our development operations, we automated the inclusion of ROS environment variables into every new shell session, enhancing accessibility to ROS's vast toolset and packages.

8. Managing Package Dependencies

Ensuring Smooth Package Builds- The final phase of our setup focused on dependency management. We installed Python-based tools for efficient package handling and initiated rosdep, a key tool for installing system dependencies required by ROS packages. By updating rosdep's package database, we secured a seamless and streamlined development process for our robotics project.
?9

9. Experimental Setup & Evaluation

9.1. Virtual Glove and Guitar String Simulation (ROS/ GAZEBO Experiments)

9.1.1. Model Creation and Simulation in Gazebo

To initiate the simulation process, we tried our hand at designing simplistic yet functional 3D models representing the hand's motion and a guitar string. Using Blender and sourcing from the 3D Warehouse, we tried to developed models that could be integrated into Gazebo for advanced interaction simulations. Hypothetically our aim was to program the 3D hand model to simulate basic strumming and picking actions associated with guitar play, while the string model vibrated or moved in response, emulating the reaction of a real guitar string being strummed.

9.1.2. Exporting and Testing the Models

Further our procedure included the aspect of exporting the created models as .dae or .sdf files, suitable for Gazebo, ensuring that both the physical and visual properties were accurately reflected in the simulation environment. Next step was to ensure our models responded correctly to simulated physical interactions, effectively representing the tactile feedback that would be provided by the dual mode haptic glove.

9.1.3. System Integration & Architecture

Glove Interface Development and Sensor Integration

Our development process involved embedding the following sensors :-

- flex sensors,
- accelerometers
- gyroscopes.

These sensors were tasked with detecting finger movements, hand speed, and orientation, transforming raw data into actionable inputs for our decision-making unit.

9.1.4. Haptic Feedback Control System Evaluation

Our glove incorporated vibration motors that provided real-time feedback based on the processing unit's analysis. Users could switch between passive and active feedback modes, enabling a comprehensive evaluation of the glove's effectiveness in various simulated environments. The control logic embedded within the microcontroller's firmware interpreted commands from the decision-making unit and managed the haptic feedback accordingly.

9.1.5. Decision-Making Unit and Note Detection

(Machine learning Model Experiments)

The microcontroller, acting as the intermediary, gathered the sensor data and interfaced with a ROS2 system for further processing.

Stage-1 :- Before we jump into using a reinforcement learning model, we need to first use basic supervised learning models in order to setup a foundation for the purpose of clear understanding of the scales and sounds of the guitar pentatonics.

Stage-2 :- A dedicated processing unit ideally should be running a reinforcement learning model which theoretically evaluates the data against pre-recorded strumming patterns and chords to ascertain the accuracy of the user’s guitar playing in real-time.

9.2. Supervised Learning model exploration

For the sake of simplicity, we first decided to go with basic audio recordings of the “A” scale pentatonics which were self-played and recorded in order to perform analysis.

The effectiveness of the machine learning model for pentatonic notes 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 pentatonic scale recordings, considering various audio features, including rhythm, melody, and harmony.

9.2.1. Initial Trial (Approach-1)

The initial phase of our project was devoted to the comparative analysis of two distinct musical pieces: 'Actual_pentatonic' which was found on souncloud scale dataset [], and

'My_Pentatonic', which was an original recording personally played by us inspired by the former. It is important to mention that we deliberately played one note wrong in the recording to check for difference detection. Our goal was to establish a systematic framework for detecting similarities between these two pieces using various audio signal processing techniques and similarity measures.

9.2.2. 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 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.

9.2.3.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.

```
[9] !pip install fastdtw
    from fastdtw import fastdtw
    from scipy.spatial.distance import euclidean

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

    Requirement already satisfied: fastdtw in /usr/local/lib/python3.10/dist-packages (0.3.4)
    Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from fastdtw) (1.25.2)

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'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
Use of Silence and Space Similarity: 1
Tempo Similarity: 1
```

Fig.4 – Shows the code snippet and Similarity scores obtained. Down below the scores are explained.

- **Notes Pattern Similarity: 243801568858003e-07**
  - This score, which is very close to 0, indicates a very low similarity in the notes patterns between the two audio tracks implying that the sequences of notes or pitches in comparison are quite different from each other.
- **Chord Progression Similarity: 854832410812377**
  - A similarity score of approximately 0.855 suggests a high level of resemblance in the chord progressions of the audio tracks. This means that the way chords change and progress over time in both tracks is quite similar, which could contribute to them having a comparable harmonic structure.
- **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 tracks are extremely similar, if not identical.
- **Tempo Similarity: 1**
  - Another perfect score of 1 indicates that the tempo (speed or pace) of the two tracks is the same. This means that both tracks are played at an identical number of beats per minute (BPM).

- **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.

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

Timbre Similarity: 0.9500380754470825
```

Fig.5 – 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 tracks 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 uniqueness of the guitarist, 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 the pentatonics have similar characteristics, this would be reflected in a high timbre similarity score. This could mean the tracks share similar sound textures, instrumentation qualities, which is exactly what we wanted to achieve since we already knew by the sound of the tracks that they are both almost similar.

In simple terms, a timbre similarity score of 0.950 indicates that, to the human ear, the two tracks 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.

9.2.4. Visualization of Audio Features

Mel spectrograms were generated for both tracks, 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- *)

9.2.5. Melodic Contour Extraction

To address the melodic aspect, we extracted the pitch sequences from each track, 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.

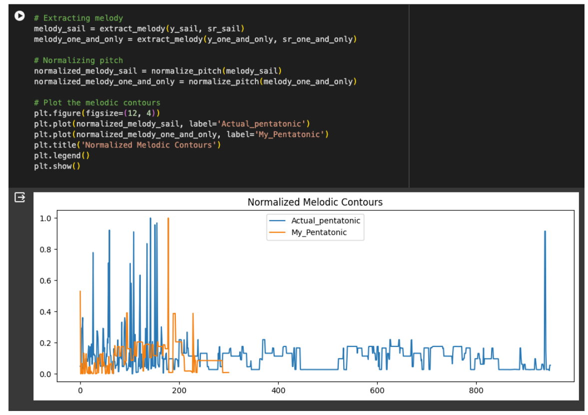


Fig.6 – Melodic Contours help in detecting similarity visually.

9.2.6. Results and Discussion

The experimental results yielded several key insights into the similarity between 'Actual_pentatonic' and 'My_Pentatonic'. 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 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, corroborating the subjective inspiration drawn from 'Actual_pentatonic' in the creation of 'My_pentatonic'. 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.

9.3. Second Trail (Approach-2)

To enhance the scope of our pentatonic similarity detection, we acquired a guitar lessosn dataset found on soundcloud with perceived similar characteristics, designated as the training set, and 3 distinct custom recorded guitar pentatonics for testing. The objective was to refine our model's ability to discern nuanced similarities within a larger and more diverse corpus of music.

9.3.1. 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.

9.3.2. Feature Extraction

For each track, 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.7 – Shows an example of the code snippet from one of the song samples used and its MFCC features extracted.

9.3.3. Pairwise Data Generation

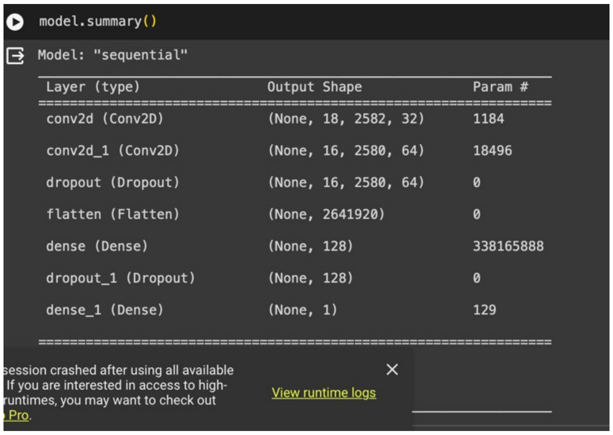
With the extracted features, we constructed pairwise comparisons between all possible audio pairs within the training set. These pairs were labeled based on a predefined grouping of tracks sharing similar tempo, baselines, or overall feel, with the aim of teaching our model to recognize both obvious 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.8 – Shows the pairwise data generated

9.3.4. 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

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


Fig.9 – Model Architecture (Before corrections were made eventually)

9.3.5. 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.

9.3.6. Testing and Evaluation

Post-training, we conducted evaluations using the test set. Each test track 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 pentatonics as similar or not based on the model's output.

9.3.7. 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.

10. DISCOVERY OF LIMITATIONS

In the course of developing our dual mode haptic glove, we have encountered various challenges that have served to both temper our expectations and refine our approach. The following points outline the limitations encountered:

10.1. Lack of a Perfect Dataset

The specificity of guitar pentatonic scales requires a dataset that is both comprehensive and nuanced. The absence of such a dataset poses a significant hurdle, as it impacts the machine learning model's ability to accurately learn and predict the wide range of possible note combinations and variations that a guitarist can produce.

10.2. ROS2-Gazebo Integration Issues

Interfacing ROS2 with Gazebo for simulation purposes has not been without issues. Random crashes during simulations have disrupted the workflow, presenting an obstacle to continuous development and testing.

10.3. Complexity of Sound Analysis

The task of accurately recognizing and classifying complex guitar notes is intricate, involving detailed audio processing that becomes increasingly challenging with the subtleties of pentatonic scales played with different styles and techniques. The complexity lies in distinguishing the correct from incorrect notes, which is further complicated by individualistic playing styles, where variations in plucking, strumming, and fingerstyle techniques are prevalent.

10.4. Variability in Playing Style

Guitar playing is a highly individualistic endeavor. Each musician brings a unique touch to their instrument, influencing the way notes and chords are played. This variability makes it challenging to create a one-size-fits-all model that accurately captures and responds to the nuances of individual playing styles.

10.5. Sensor Sensitivity and Precision

Our project's success hinges on the precision of sensors to detect fine movements and pressure changes. Current off-the-shelf sensor technology may not always meet the high level of sensitivity required for nuanced musical tasks, leading to potential inaccuracies in feedback.

10.6. Latency Issues

The interactive nature of our glove demands near-instantaneous feedback to be effective. Any noticeable delay between playing a note and receiving feedback can disrupt the learning and playing experience. Achieving low latency is crucial and is affected by both hardware limitations and software efficiency.

10.7. Algorithmic Bias and Overfitting

Machine learning models, particularly those involving reinforcement learning, can develop biases based on the training data. This predisposition can lead to overfitting, where the model performs well on training data but fails to generalize to new, unseen data.

10.8. User Adaptability

The diversity in users' responses to haptic feedback poses another challenge. Personalizing the system to optimize learning for each individual is complex and requires not only a dataset that encapsulates the full range of possible interactions but also sophisticated algorithms capable of adapting to user feedback.

10.9. Data Annotation Challenges

Creating a perfect dataset necessitates accurate labeling. The manual process of annotating audio data with correct and incorrect plays is labor-intensive and prone to human error. This can introduce inaccuracies in the data, which in turn affect the training and performance of the machine learning model.

These limitations, while presenting significant challenges, also offer avenues for future research and development. They underscore the importance of continued innovation in haptic technology, machine learning, and sensor development to overcome these hurdles and improve the fidelity of our dual mode haptic glove system.

11. Results and Discussion

While the project is theoretical, we predict the outcomes through simulated environments and hypothesized testing. We envision that the glove will yield faster learning times, increased retention rates, and greater student engagement compared to traditional learning methods. Discussions will include potential challenges, such as ensuring the system's adaptability to various learning styles and technical specifications required to process the musical data accurately.

12. Conclusion

In conclusion, the "Dual-Mode Haptic Feedback Glove for Enhanced Guitar Learning" represents a cutting-edge intersection of haptic feedback technology and AI-driven educational tools. It stands to not only expedite and enrich the guitar learning process but also pave the way for future innovations in skill-based learning across various disciplines. The project holds promise to significantly diminish the time it takes to become proficient in guitar, potentially disrupting traditional methods and setting a new standard in musical pedagogy.

This theoretical exploration hopes to inspire further research and development, leading to tangible advancements in the way we approach learning complex skills like musical instrument proficient.

13. Discussions

Throughout our journey, we've gleaned the critical influence of an extensive, well-curated dataset in the intricate realm of music similarity detection. The path was challenging, highlighting the need for a generous timeline to thoroughly refine our model and move closer to a fully-realized prototype. We've learned that the richness of the data we feed into our system is paramount—it is the difference between surface-level similarity and deep, nuanced musical analysis.

Our project's stride could be amplified with the expertise of a more diverse team. We aim to capitalize on a multiplicity of skills and insights from various fields to quicken our progress. By embracing an open-source approach and transitioning our project to a GitHub repository, we invite the global community to contribute, bringing us closer to our goal through collective ingenuity.

Moreover, the cross-disciplinary partnerships would be a beacon, shedding light on the immense potential we could have by means of collaboration with educational, technological and music industry domains. These alliances are more than just resource pools; they are incubators for groundbreaking ideas that could propel us to the forefront of music technology.

This reflective process has been invaluable, teaching us that beyond the algorithms and data, the collaborative spirit and a multi-faceted approach are what will drive innovation in our project and beyond.

14. Individual Contributions

Jayanth Suryaprakash -

- **Blender Modeling:** Spearheaded the creation of the 3D model of the haptic glove using Blender, setting the foundation for our virtual simulations.
- **Supervised Model (1st approach):** Developed the initial supervised learning model architecture, critical for the project's ML component.
- **ROS Noetic Integration:** Played a key role in the integration of ROS Noetic, in order to enhance the robotic simulation capabilities.
- **Construct AI Research:** Led the research and found out about 'construct.ai', which facilitated our transition, ensuring a more stable simulation environment.
- **Deciding System Integration and Architecture:** Took charge in the decisive planning of system integration, laying out the architecture for our glove which includes haptic feedback research.
- **Haptic feedback research –** Went through multiple papers in order to understand the haptic motors better.

Surya Samarth's Role -

- **MFCC Research:** Conducted in-depth research into Mel Frequency Cepstral Coefficients (MFCCs), which are vital for the audio analysis in our ML models.
- **Supervised Learning (2ns approach):** Developed the second approach which involves making use of OpenSMILE and Librosa libraries focusing on pattern recognition aspect.
- **Data Creation and Processing:** Personally recorded guitar pentatonics and created a set of data to pursue this project. Further, managed the processing of the custom made datasets, ensuring quality and relevance for training the ML models.
- **CNN Architecture:** Constructed the Convolutional Neural Network (CNN) architecture, optimizing it for efficient audio signal processing.
- **Audio Libraries Exploration:** Found out about 'OpenSmile' tool and 'Librosa' library which played a vital role for feature extraction, significantly contributing to the project's audio processing capabilities.
- **Melodic Contour Extraction:** Focused on extracting melodic contours, a crucial step towards achieving accurate similarity detection in musical patterns.

Group Efforts

- **Limitations and Discussions:** As a collective, we identified and analyzed the project's limitations as we steered deep into this concept, engaging in comprehensive discussions to strategize our future works.
- **ROS/Gazebo Implementation:** As a team we were instrumental in the ROS/Gazebo implementation, contributing to the installation and configuration processes.
- **Future Works:** Together, we envisioned the trajectory of the project, proposing

extensions in the timeline and collaborations that could enrich our project.

15. References

- **Move Me (1st link) -** https://www.media.mit.edu/publications/moveme-3d-haptic-support-for-a-musical-instrument/
- **Move Me (2nd link) -** https://www.media.mit.edu/publications/moveme-3d-haptic-support-for-a-musical-instrument/
- **Opensmile -** https://www.audeering.com/research/opensmile/
- **Blender –** https://www.blender.org/download/
- **XGBOOST -** https://www.kaggle.com/code/batuhansenerr/music -recommendation-xgboost-over-90-accuracy
- **MFCCs -** https://librosa.org/doc/main/generated/librosa.featur e.mfcc.html
- **Haptic Feedback for virtual reality -** https://www.researchgate.net/publication/2356993_Haptic_Feedback_for_Virtual_Reality
- **Design and Evaluation of a wearable haptic device paper -** **https://ieeexplore.ieee.org/abstract/document/8304760**