CHORDMOSAIC

First Author
Daming Wang
wdm1732418365@gmail.com

Second Author
Haolin Liu
haolinliu@uvic.ca

ABSTRACT

In this project, music understanding and visualization are combined to analyze songs and generate corresponding artwork. We present a system to automatically analyze song lyrics and audio chords to tag music and automatically visualize its mood and harmony. The system begins with some audio files, like WAV or MP3, and applies a music recognition API to recognize and process them. We use a lyrics-based auto-tagging using natural language processing (NLP) and a chord recognition method from audio using music information retrieval (MIR) techniques to preprocess the lyrics, tags, and chords. Then, multi-label classification will be used to generate labels into a text-to-image generation AI model. This will create an innovative picture of the song and describe the mood of the song. Moreover, we can analyze the chords and tones to output as Piet Mondrian-style color blocks based on the chords as an additional output result. In general, this project includes the processes of music analysis, deep learning, and generative art. The meaning of this project is to provide an efficient and accurate way to create a mood drawing or an album/song cover picture for songs.

1. INTRODUCTION

Music is an audio-visual experience in our daily lives that can enrich our lives and spirits. Music is also considered an auditory format of the artwork. However, another standard format of artwork that is often considered is painting and drawing. Therefore, our project wants to discuss the possibility of combining these two artworks together, making them feasible to convert from one way to another. Since we are focusing more on music and audio processing in this course, we will primarily work on converting the audio into drawings in some way. Some previous work on music information retrieval has tackled the classification of songs by genre, mood, and tags using audio signals or textual features [1][2]. However, combining lyric analysis and audio chord recognition to give generative visualizations is still a novel area. This project presents a unified system that can tag songs based on lyrics and chords and then use the tags to create visual art reflecting the song's mood and content.

This project aims to help listeners understand music more deeply by providing visual complements. For example, a song's lyrics usually suggest the main theme and emotions, while the chord progression represents the tonal mood. By combining these, we can generate an image, such as an album cover or dynamic visualization, that represents the song's mood and musical structure.

In general, this project aims to develop under three steps. The first step is to design a lyric analysis model for multi-label tag predictions on genre and themes and mood detection from raw lyrics. The second step is to develop an audio analysis model for automatic chord recognition from music audio and, finally, make a generative visualization module that creates artwork from the combined lyric and chord information. The purpose of this project is to create a visualization work from an auditory artwork.

2. RELATED WORK

2.1 Lyric-Based Auto Tagging and Mood Detection

Song lyrics have been used as textual data for music classification and tagging tasks. There are already some early studies that works on music auto-tagging as a multi-label text classification problem by applying techniques like bag-of-words, TF-IDF, and support vector machines to lyrics [1]. For example, Mayer et al. combined lyric features with audio features to classify genre [4], finding that a fusion of lyrics and audio improved accuracy [5]. Similarly, Mckay et al. evaluated lyrics versus audio and symbolic features for genre classification; nothing that lyrics alone can be indicative but often perform best when combined with audio [6]. Fell and Sporleder (2014) explored linguistic features for lyric-based music classification, highlighting the value of text analysis beyond simple word counts [2].

In addition to genre and tags, mood detection from lyrics has been another important topic for a long time. Hu and Downie (2010) showed that combining lyric test features with audio features can improve mood classification in music libraries [3]. Their paper shows that lyrics can provide many semantic contexts like happy, angry, romantic, etc. More recent deep learning approaches show NLP advances by using recurrent neural networks and attention mechanisms that can model lyric sequences to predict mood or theme. For example, Tsaptsinos (2017) introduced a hierarchical attention network for genre classification purely from lyrics [1], which achieves a good result by

considering the song's structure from words to lines to segments in the model. Additionally, Delbouys et al. (2018) extended lyrics-based mood detection with deep learning by using a bimodal approach to predict continuous mood dimensions [7].

Overall, lyric-based auto-tagging is feasible because of large lyric datasets with annotated lyrics. Deep learning models like CNNs and RNNs can effectively extract meaningful features from lyrics for the tag extraction task [8]. In our work, we will try to build a model using a multi-label text classifier to assign mood and theme tags to lyrics, which will then be processed to the next stage of image generation.

2.2 Lyric-Based Auto Tagging and Mood Detection

Another aspect we need to consider is the automatic chord recognition (ACR) from audio. It is a well-established MIR task with over two decades of research [15]. The goal of our work is to transcribe the chord sequences like C major, G7 and A minor from a music audio signal. Fujishima's work in 1999 introduced the use of chroma features like pitch class profiles for real-time chord detection [9]. That gives a basic typical chord recognition pipeline. Firstly, we need to compute a time-frequency representation such as the short-time Fourier transform or Constant-Q transform [18]. Then map the spectrum to 12-dimensional chroma vectors by summing energy into the 12 pitch classes of the octave and finally applying pattern matching or machine learning to identify the chord at each time frame [11]. Early models rely more on template matching and heuristics, sometimes with Hidden Markov Models (HMMs) to enforce the temporal continuity of chords [18]. An HMM can smooth frame-wise chord predictions by modelling transition probabilities between chords. This can be formulated as finding the chord sequence $c_{1:T}$ that maximizes a joint probability with transition penalties [19]:

$$\hat{c}_{1:T} = \arg\max_{c_{1:T}} \prod_{t=1}^{T} P(x_t \mid c_t)\, P(c_t \mid c_{t-1}) \qquad (1)$$

where x_t is the feature (chroma) at time t, $P(x_t|c_t)$ is the likelihood of chord c_t generating that feature, and $P(c_t|c_{t-1})$ is the transition probability from the previous chord. Solutions to (1) are typically found via the Viterbi algorithm in chord HMM frameworks [18].

There are also some novel approaches that use machine learning and deep learning to improve chord recognition accuracy. Convolutional neural networks (CNNs) were applied to chromagrams to learn chord patterns robustly (Humphrey & Bello 2012) [12]. Recurrent neural networks (RNNs), including LSTMs, have been used to model temporal dependencies in chord sequences (Boulanger-Lewandowski et al. 2013) [10]. These network models see chord recognition as a sequence of labelling problems and output a probability distribution over chords at each time step. Sigtia et al. (2015) proposed a hybrid approach mixing an RNN and a temporal model for chord sequences to make it more accurate [13]. Additionally, in order to handle large chord vocabularies, Deng and Kwok (2017) presented an even-chance training scheme with deep models to balance chord class frequencies [20]. McVicar et al. (2014) mentioned that even advanced models benefit from good input representations [14], and Korzeniowski and Widmer (2016) developed a deep chroma extractor by learning an optimized chromagram from the raw audio using a neural network [15]. The chord recognition accuracy was improved by training the network to emphasize harmonic content and suppress noise. All these related works gave us the idea of using a pre-trained model or our own trained CNN/LSTM to output chord symbols from audio. We can then process the chord outputs to the next step.

2.3 Generative Music Visualization

Whenit comes to the phase of converting music into visuals, we can use generative AI, particularly text-to-image synthesis. Models like OpenAI's DALL-E can create images from textual descriptions by using large transformer-based generative models [16]. Recent diffusion models like Stable Diffusion can provide further improved quality and fidelity of generated images via latent diffusion processes [17]. We can use the tags and labels from previous steps and implement API connections to generate images that align with the song's emotional tone.

In addition to lyric-based imagery, we can also use chord progressions to map musical features into colors or shapes. Here, we propose a Mondrian-style visualization for chords since his work is characterized by rectangular blocks of primary colors with varying sizes. Our suggestion is to design a simple algorithm to convert the chord sequence into a neoplasticist art form, echoing Mondrian's style.

Overall, by combining all of the ideas on related works, we aim to produce a composite visualization that reflects both the lyrical and musical content of the song.

3. METHODOLOGY

3.1 Pipeline Overview

The system architecture (Figure 5) consists of three main stages arranged in a pipeline. First, the Lyric Analysis stage takes the raw lyrics of a song as input and produces both a set of tags and an inferred mood description. Next, the Chord Recognition stage processes the song's audio to output the chord progression. The lyrics are analyzed by an NLP classifier to predict tags like mood and genre. The audio is analyzed by a chord recognition model to yield the chord sequence. Finally, the Generative Visualization stage uses the information from the first two stages to create visual content: an AI-generated image guided by lyrical themes and a Mondrain-style artwork guided by chord progression. The two visual elements can be combined or presented side by side as the final visualization of the music. The result is a composite visual representation of the song.

3.2 Basic Setup

We used Shazam API to integrate the audio recognition process. Firstly, the user inputs an audio file, and Shazam recognizes it. Then, it can go to the next step and do further lyrics processing based on the metadata it gets. After that, the information(song_title and artist) is passed to the next step. We match the song with Music4All. If no songs are found in Music4All, we will use Musicmatch API to get the lyrics and last.fm API to get the tags of the song.

For the Chord Recognition part, we used McGill Billboard dataset that contains 890 songs with detailed chord annotations. We have downloaded the songs from YouTube and matched them with the CSV files that contain chord ID and a folder of each song's chord annotation TXT file. Now, we are prepared to train and evaluate the chord recognition model.

3.3 Lyrics Analyze Model

The lyrics analysis module is the key component responsible for converting textual lyrics into emotional expressions, like a mood detection module. To ensure that the AI model accurately understands the emotional and thematic nuances embedded in the lyrics, we have integrated advanced Natural Language Processing (NLP) techniques.

3.3.1 Lyrics data preprocessing

Before analyzing the lyrics, we perform data preprocessing to make sure that the textual content is clean and structured and remove any unnecessary information that may affect the model. Lyrics often contain punctuation marks, numbers, special characters and some meaningless repetitions. Thus we have to clean the text by removing all non-alphabetic characters and keeping the core information in the lyrics. Then we perform Tokenize the lyrics with NLTK, tokenizing the complete lyrics into single words and filtering out the Stopwords which are not semantically useful but occur very frequently, like "the, and, is," etc.

At this stage, we confirm that the model can emphasize the contents of the lyrics that really contains emotion and theme as opposed to being distracted by meaningless words. In addition, to ensure text consistency, we have text normalized all words by converting them to lowercase and removing spaces. This not only helps to reduce the noise of the data but also improves the model's ability to generalize across different lyrics data, ensuring that subsequent analyses can run more stably.

3.3.2 Keyword extraction

We clean the lyrics and homogeneously format it, and then we focus on extracting words using keyBERT, which shall help in automatically recognizing the topics of the lyrics. KeyBERT is a keyword extraction tool based on BERT (Bidirectional Encoder Representations from Transformers) that unites the statical method of TF-KeyBERT and the ability of BERT to understand context so that the significant words extracted not only have the importance of word frequency but also reflect the semantic relevance of the lyrics. This makes the extracted keywords more consistent with the real context of the lyrics.

For the extracted keywords, we can improve their quality in practice by applying the Word Order Normalization and Set Duplication mechanism to guarantee that the final deduplicated keyword list does not contain a repeated element. By applying this optimized method integrating deep learning and traditional text processing techniques to obtain the representative theme of the lyrics text, it allows the automatic style classification and visualization generator of a song to use a more representative lyrics text feature set as the basis weight.

3.3.3 Multi-Label Classification

In order to analyze the lyrics on another level, we implement the Multi-Label Classification (MLC) system to provide automatic context-based labels such as style, mood and theme of a song. As a song may contains multiple features at the same time, the conventional single-label method of classification is no longer sufficient. Hence, we use TF-IDF for extracting features from text, and pair it with Random Forest for classification under multiple labels. TF-IDF is a method for effectively highlighting specific keywords of the content by calculating the frequency of each word, taking into account their relative uniqueness, often song lyrics due to its specificity, and thus allowing this model to effectively distinguish different music styles and themes. Random Forest is a powerful integrated learning algorithm that constructs multiple decision trees and combines them with a voting mechanism for classification. It is also able to reduce the overfitting danger, maintain good generalization in small/medium datasets, and perform with stability on multi-label problems. Furthermore, Random Forest is also highly interpretable, so we can see which words are most responsible for a given category prediction, making the classification results more trustworthy and interpretable.

Label co-occurrence and class imbalance are two major challenges in multi-label classification tasks. Some emotions, such as 'happy' and 'excited', tend to correlate closely, while others are somewhat independent, so treating each label independently may miss interdependencies. Moreover, the distribution of different styles of music and mood labels might be very unbalanced. The prediction results of the current classification system are mainly regarding mood and genre, which can be specifically manifested as mood classification and style prediction.

We evaluate the model using standard metrics such as Micro F1 Precision, Recall and Hamming Loss. Results and detailed analysis are presented in Section 3.3.5.

We first selected lyrics data from the dataset, implementing basic models such as support vector machines and multi-label logistic regression to classify the lyrics. These approaches are theoretically quite simple, but they turned out to be computationally inefficient in practice, with training times of 30 minutes to over 50 minutes, especially with higher dimensionality of features. To be able to scale and be responsive, we refactored our analysis into a TF-IDF +

Random Forest pipeline. This helped us keep the training time to less than 20 minutes while having a stable model. Faster iteration also allowed for easier tuning and integration within the full pipeline.

3.3.4 Lyric-to-Image Result

In order to show the visual effectiveness of the lyric model, we provide a complete example generated by the pipeline.The following image was generated based on a real lyric file using predicted tags, sentiment, and extracted keywords.

Every time we randomly choose 10 files in order to training. In order to demonstrate the full output from lyric inputs to generated annotations of our generation model, we list the outputs from passing the lyric file Fywe8EHE9KSD5Ulw.txt through our model as a representative example. This songs title is Only the End, and the artist is Djo. Once processed, the model had predicted the tags with “love,” “romantic,” “energetic,” and “major-key,” and classified the overall mood as "negative" with high confidence. KeyBERT then extracted semantically-relevant phrases from the cleaned lyrics, including, "need time," "talk time," "fault said," "stop time," and “ends taught.” All of these elements were compiled to create a final optimized prompt for image generation, “romantic, energetic, need time, negative, major-key, talk time, fault said, love, ends taught, stop time, highly detailed, cinematic lighting, ultra realistic, trending on artstation. The final image was generated with DALL-E 3, which utilizes its powerful natural language understanding to visually cohesively and cinematically explain lyrical themes, sentiments, and keywords.

Figure 1. Generated image based on predicted lyric model.

3.3.5 Evaluation of Lyric Model and Visual Consistency

To evaluate the effectiveness of our lyric classification model, we use standard metrics to assess its performance. After training on our dataset, we got the following scores. The Micro F1 score is 0.9373, Hamming Loss is 0.0023, Micro Precision is 1.0000, Micro Recall is 0.8821. The results reflect that the model has a very good prediction ability and very low error rates, such as shown by the Hamming Loss, perfect precision, and high recall. You can see that the microprecision is almost perfect, meaning that every time the model generates a tag, it is likely correct, and the recall is high, meaning that it finds most of the relevant tags even if it misses a few.

We also qualitatively inspect the system’s global consistency by checking whether the generated image fits with the predicted tags and mood of the model. In the case of the previous example given in Section 3.3.4, the system proposed the tags love, romantic, energetic, and major-key and classified the mood as negative. Indeed, the resulting image shows a warm and cinematic stage, filled with diffused light effects and soft landscape attributes that are reminiscent of a dreamy and emotive space. The “energetic” tag may not always be directly depicted in physical action or colors. However, we can interpret this through both dynamic lighting and upward motion in this scene. The lushness of purples and natural tones adds a sense of both romanticism and emotion, a wonderful counterpoint to the top-line lyric themes of “need time” and “ends taught.” In general, the image produced and the anticipated labels were semantically and aesthetically consistent, indicating that the system can translate textual and emotional information into textually coherent output.

For future work we intend to perform a small user study which would compare human-perceived emotion and genre labels produced versus the labels predicted by the system, further establishing how closely model output lines up with listener interpretation.

3.4 Chord Recognition Model

For the Chord Recognition Model, we choose to build our own model based on LSTM. Using an epoch of 100 and GPU to boost the accuracy and running speed. Ideally, the model will output a list of the main chords of the input songs.

3.4.1 Chord Preprocess

First, we delete all the rows with NULL value song_title and artist in the CSV file, and then we match the downloaded audio files with the dataset, if there are no audio files found, we also delete the rows. After that, update the chord annotation TXT files correspondingly, only keeping the rows in the updated dataset. Later, we simplify the content in the chord annotation files by only keeping a list of chords in the updated TXT files and saving them directly with their index number. Therefore, now we have all the audio files in one folder and all the simplified chord files in another folder and ready to start the training process. Both have around 880 songs.

3.4.2 Chord Training

We first used audio feature extraction with the Librosa audio analysis library. For each audio recording, we extracted

the first 30 seconds of each recording to maintain consistent input dimensionality and capture sufficient information for chord analysis. We extracted three chromagram representations: Chroma STFT, Chroma CQT and Chroma CENS to provide a different perspective on the harmonic content. In order to enhance the distinguishing of chord types with similar pitch content but with different timbral characteristics, we used Mel-frequency Cepstral Coefficients to capture spectral envelope information. Finally, zero norm is applied to normalize all features to improve training stability and convergence. In general, the final feature representation for each song is a sequence of 128 time frames, with each frame containing 12 dimensions times 3 chromagram types plus 7 MFCC coefficients.

Then we choose to use Bidirectional Long Short-Term Memory just as the project from krist311 uses [23]. The LSTM will make the model consider both past and future contexts when identifying chords. The input layer is, of course, 128 units with 43 features. For the first BiLSTM Layer, the process enables the model to analyze harmonic progressions in both directions. Each major layer is followed by batch normalization to stabilize training and dropout at a rate of 0.4 to prevent overfitting. Then the 64 units on the second layer will further tune the temporal representations extracted from the first layer. Then at dense layers, the first and second layer will translate the LSTM encodings into chord classification features with the help of ReLU activation functions. Finally, the output layer is a dense layer with softmax activation, providing probability distributions on all possible chord classes.

The data is splitting into 80% of training data and 20% of testing data. This process is done randomly since there are some likely occurrences of rare chord types. The batch processing is to help manage the memory efficiency with chunks of 32 songs at a time. When we do the training steps, we want to prevent overfitting, so we use early stopping with patience of 55 epochs on validation loss monitoring. Finally, we save the label-encoder and best-performing model based on the validation loss locally to easily do a chord recognition later.

3.4.3 Performance

The model stops at a total of 55 epochs. The initial training accuracy starts at 1.42%, and the final training accuracy reaches 65%. However, the results on evaluation sets are only 27.5% accurate. However, this is still a challenging issue since the model was classifying across 80 unique chord classes, making this a very complicated multi-class classification problem. Furthermore, many chord types share similar harmonic components, like G:maj and G:7 and G:maj7, this also creates ambiguity. However, the model can successfully identify some major chords, which suggests this is a good model for basic chord recognition. Most cases showing D:min7 predicted as D:maj shows the model confused on the chord quality. This shows that the model still has difficulty distinguishing harmonic extensions. Some other predictions, like A:maj predicted as Db:maj show significant errors. This may be due to shared harmonic overtones or timbral similarities in the audio. In general, chord recognition is a very complicated task in machine learning; we manually checked the accuracy of each evaluation set, and we found out that most recognitions are just small errors like D:min7 recognized as D:maj causing the low accuracy. However, this model is now still enough for us to do the further generation process since we don't need exactly 90 or 100 percent accuracy on the chord analysis.

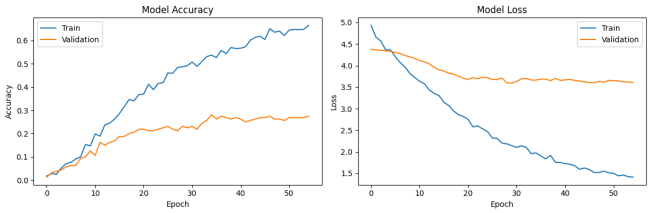


Figure 2. Chord recognization model result

In the future, we can update our training dataset size to include more rare chord types and consider a two-stage approach that identifies the root note and classifies the chord quality. Moreover, some additional features, like spectral and harmonic features, should be considered to capture chord quality distinctions. In addition, we can implement curriculum learning, starting with basic chord types and progressively introducing more complex chords.

3.4.4 Output

We generate the image using canvas to draw a 600x600 painting. The function calculates the total weight and then selects a random target fraction between 0.4 to 0.6 to decide where to split the array of chords. The reason behind choosing this range is to make a more square-like rectangle instead of a very long, stripy rectangle. Depending on the current rectangle's aspect ratio, it chooses either a vertical or horizontal split to favor shapes closer to a square. Once only a single chord remains, it just takes the last empty space. After partitioning, the colors are assigned randomly, with each primary color (red, blue, yellow) only appearing at least once and at most twice while leaving the reset blocks white.

Figures 3 and 4 show the percentage of chord results from a sample song: "Rise Up — Imagine Dragons." We can easily do this process again with different songs by just run the codes, and upload the song files with mp3 or wav file format, then it will generate the images for us with one showing with the chords name on the picture showing the ratio for rectangle size and the chord percentage, another one showing only the Mondrain-style painting.

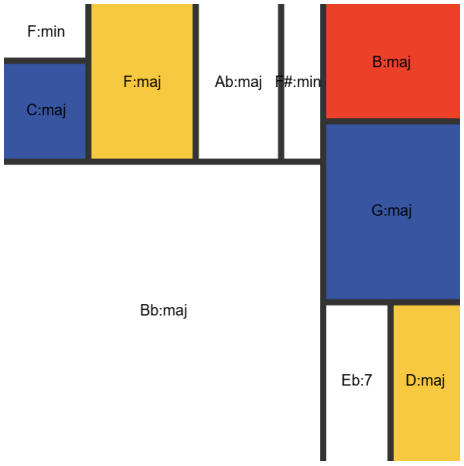


Figure 3. Generated image based on chord percentage

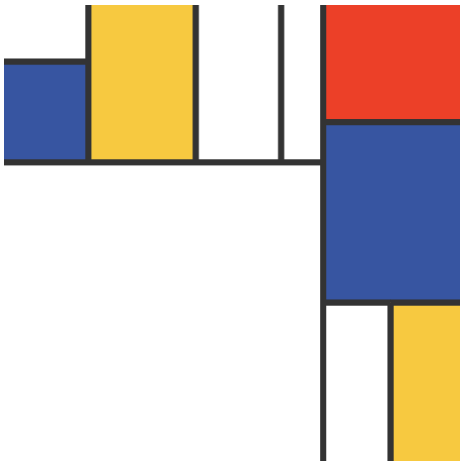


Figure 4. Generated image using Mondrian-style painting

3.5 Further Experiments

We did further experiments with peers on matching the songs with their generative images, like the figure below shows, with only a small sample of songs. At first, an image of a Mondrian-style painting and a text-to-image generated AI image are provided without the track name and artist name on the top but a list of all these song names. We need to find out which track stands for the corresponding images. Then, we checked the final accuracy once we had finished all the matching. However, the only mistakes made were on "Whatever It Takes" and "Believer" because they are all a list of related elements and relevant to the lyrics. I believe some of the songs can use the generated image as the album cover, especially for "I Drink Wine" from Adele. It turns out that some country songs and romantic songs are more suitable for this kind of generative images. Imagine Dragons' songs, on the other hand, are mostly all positive and full of positive moods, which makes the generated images similar to each other.

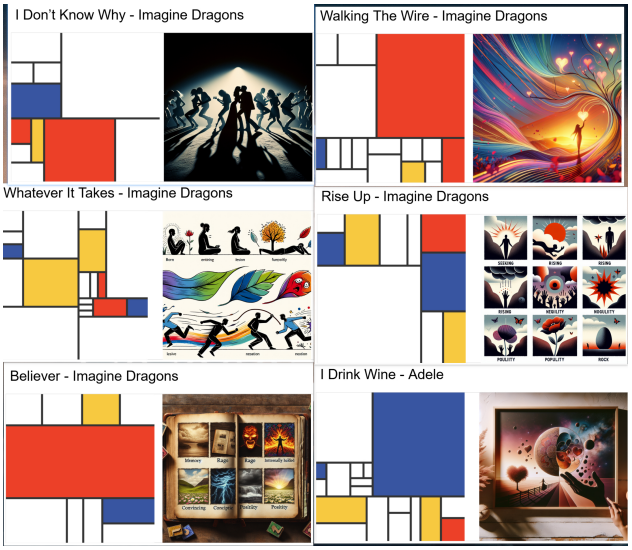


Figure 5. Generated image using Mondrian-style painting

3.6 Tools & Libraries

3.6.1 Program Language & Libraries

The project is implemented in the Colab notebook with Python. Python has a rich set of music information-related libraries, which makes it well-suited for the rapid development of machine learning and audio processing pipelines. We use TensorFlow to develop deep learning models. Specifically, we will use the LSTM model in TensorFlow to develop our chord recognition model since it's one of the best options for audio-related training models. Classic machine learning algorithms are implemented using Scikit-learn, a Python library with a wide range of ML algorithms. For analyzing musical audio, like extracting features and detecting chords and beats, we use Librosa. Librosa is a Python package for music and audio analysis that provides the building blocks for MIR systems. It offers functions for computing spectrograms, chroma features, onset strength, etc.

In order to create visualizations inspired by musical content, we can use text-to-image generation models. Specifically, we use OpenAI's DALL-E API and Stable Diffusion. By inputting descriptions of a song's mood, theme, or even transcriptions of its lyrics, the model can produce a corresponding image. These generative tools allow the project to visualize musical pieces in creative ways. For example, we can generate an album cover art from the song's emotional content.

For plotting and image handling, we use Matplotlib (if charts such as mood timelines or chord progression plots are involved) and Python Imaging Library for image processing tasks. These libraries help generate images with text or combine multiple visual elements to ensure the results are ready for analysis.

3.7 Datasets

Our methodology involves multiple sub-tasks, including lyrics mood classification, chord recognition, etc. Each

stage requires specialized datasets.

3.7.1 *Lyrics Classification & Mood Detection Datasets*

We use two primary datasets that provide lyrics, audio, and metadata for a large number of songs, enabling the training of lyric-based classifiers and mood detection models: Music4All [21] and Musicmatch datasets.

Music4All is a music database that contains a large variety of information per track: metadata (artist, album), user tags, genre labels, 30-second audio clips, and full lyrics for each song. Music4All is a large-scale dataset aimed at MIR tasks, and it satisfies key requirements like diversity of genres and inclusion of lyrics. We use Music4All for its lyrics and associated tags, which include genre and possibly mood indications, to train our lyric classification models.

Musicmatch can provide lyrics with free access using API. However, it can only provide the top 30 percent of the lyrics due to copyright issues. However, those are enough for a lyric analysis when the music can't be found in the Music4All database.

3.7.2 *Chord Recognition Datasets*

For training and evaluating the chord recognition component, we will mainly use McGill Billboard Dataset [22].

This is a dataset including 890 songs with valid detailed chord annotations which can help us train and evaluate the accuracy of our chord recognition model.

3.7.3 *Integration and New Dataset Construction*

We have merged all the separate tables in Music4All to one data frame based on the unique ID for easier search and information extraction purpose.

4. PROBLEM AND DIFFICULTY

First, we tried the Musixmatch API, and it turns out it only supports 30% of the lyrics output. Therefore, we turned to Music4All, and it proves that Music4All has much more information including lyrics. However, the database is too large, so we have to unzip it without the audio folder to Google Drive and integrate it with Colab afterward. We tried some other APIs like Lyricsgenius, but it has a limitation of free account access, and it keeps giving a 403 error. As for the Spotify API, we wanted to acquire the audio features from Spotify. However, they stopped updating their API, and these audio feature parts are no longer supported due to decryption issues, as their official forum explained. Therefore, Spotify has basically abandoned the Spotify API because it now can only get the most basic information like album, cover, etc.

When run the Shazam API and then ran the Pandas data frame process, we met the "TypeError: Cannot convert numpy.ndarray to numpy.ndarray" error. After research, we found out that the shazamio library uses a different version of the numpy library, which causes this issue in Colab. It would be easier to fix locally, but it is a difficult issue in Colab. Therefore, we solve it by running the Audio Recognition in one Colab notebook and starting the process in another notebook. It not only ensures the individuability of the running environment but also clears the steps we need to do in this whole big project. We split the notebooks into three to match each step.

When we used the generative AI model, at first, we used the Lykon/dreamshaper-7 Stable Diffusion model, but we found that the image quality was quite low with this small and free model. For example, the person generated had a weird body shape and uncountable fingers. Therefore, we turned to OpenAI's image AI API, which is Dall-E 3. The result turns out that the image quality is excellent, and the AI generator can better understand what we want. Although the API costs some money, the quality is worth the cost.

5. TIMELINE

- The literature review and database set up was done in the first week. We gather the required datasets for lyrics with tags or annotations. The basic dataset setup was done with the correct implementation of using and requiring data from datasets. We extracted the lyrics and tag information from Music4all and chord annotations from McGill Billboard datasets.

- The lyric preprocessing took 2 weeks from March 9th to March 22nd. We developed a multi-label lyric classifier, which includes preprocessing lyrics and training the model on a labeled dataset. Also, we implemented a mood detection model. We have a working script that takes songs as input and outputs predicted tags and moods from these songs.

- We did the training and evaluation of the chord annotation info in one week from March 23rd to March 29th. The output will be a list of the main chords in the song sorted by frequency/importance. We did the lyric analysis finalization in the same week. The output is a list of prompt words that describe the song and can be used to generate AI images.

- The evaluation phase took a week from March 31st to April 5th. Finished the testing phase and evaluated the results qualitatively and quantitatively. We asked peers to guess the mood of the song from the image to see if it correlates.

- Finished the final report and presentation before April 10th.

6. MILESTONES

6.1 Basic Milestones

- **Literature Review & Data Setup:**
  - Gathered and preprocessed the required datasets (Music4All, McGill Billboard).

  - Completed an initial literature review to establish baseline methods for lyric tagging and chord recognition and also some previous related work.

- **Initial Lyric Analysis Model:**
  - Implemented a basic multi-label lyric classifier, including text cleaning, tokenization, and keyword extraction, and the result with mood and genre predictions.
- **Basic Chord Recognition Model:**
  - Developed a preliminary chord recognition pipeline using pre-trained models from GitHub to output basic chord sequences.

6.2 Expected Milestones (Core Functionality)

- **Enhanced Lyric Analysis:**
  - Optimized text normalization, keyword extraction (using KeyBERT), and multi-label classification (TF-IDF and Random Forest).
  - Validated the system with standard evaluation metrics, including Micro F1, Precision, Recall, and Hamming Loss.
  - Conducted user studies to validate the alignment between the generated images and the song.
  - Produced artwork that reflects the predicted tags, mood and lyrics.
- **Enhanced Chord Recognition:**
  - Trained and evaluated a deep learning model (LSTM network) to improve chord identification accuracy.

6.3 Advanced Milestones (Enhanced Features in the future)

- Adapt the system for low-latency, real-time audio analysis.
- Scale the pipeline to process many more music genres and larger datasets, including instrumental music.
- Refine both lyric and chord recognition models using extended datasets and advanced training strategies, especially to improve the accuracy for chord recognition.
- Combine lyric and chord analysis outputs with a text-to-image generation module.
- Explore diverse painting styles beyond Mondrian-style paintings.
- Integrate comprehensive error handling and performance monitoring for a robust end-to-end system.

7. TEAM CONTRIBUTIONS

7.1 Daming Wang

Daming Wang mainly charges for the audio chord recognition model implementation and the Mondrian-style art generation and is responsible for overall system integration (APIs). In addition, he will also contribute to the final report and check grammar.

7.2 Haolin Liu

Haolin Liu will lead the lyric-based model training, testing, and evaluation phases. She also needs to finalize the AI image generation step and write reports.

8. REFERENCES

[1] A. Tsaptsinos, "Lyrics-Based Music Genre Classification Using a Hierarchical Attention Network," in *Proc. Intl. Society for Music Information Retrieval (ISMIR)*, pp. 553–559, 2017.

[2] M. Fell and C. Sporleder, "Lyrics-based Analysis and Classification of Music," in *Proc. COLING*, pp. 620–631, 2014.

[3] X. Hu and J. S. Downie, "Improving Mood Classification in Music Digital Libraries by Combining Lyrics and Audio," in *Proc. ACM/IEEE Joint Conf. Digital Libraries*, pp. 159–168, 2010.

[4] R. Mayer, R. Neumayer, and A. Rauber, "Combination of Audio and Lyrics Features for Genre Classification in Digital Audio Collections," in *Proc. ACM Int. Conf. on Multimedia*, pp. 159–168, 2008.

[5] R. Mayer and A. Rauber, "Musical Genre Classification by Ensembles of Audio and Lyrics Features," in *Proc. ISMIR*, pp. 675–680, 2011.

[6] C. McKay, J. A. Burgoyne, J. Hockman, J. B. L. Smith, G. Vigliensoni, and I. Fujinaga, "Evaluating the Genre Classification Performance of Lyrical Features Relative to Audio, Symbolic and Cultural Features," in *Proc. ISMIR*, pp. 213–218, 2010.

[7] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, "Music Mood Detection Based on Audio and Lyrics with Deep Neural Nets," in *Proc. ISMIR*, pp. 475–481, 2018.

[8] K. Choi, G. Fazekas, and M. Sandler, "Automatic Tagging Using Deep Convolutional Neural Networks," arXiv:1606.00298, 2016.

[9] T. Fujishima, "Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music," in *Proc. Int. Computer Music Conf. (ICMC)*, pp. 464–467, 1999.

[10] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio Chord Recognition with Recurrent Neural Networks," in *Proc. ISMIR*, pp. 335–340, 2013.

[11] T. Cho and J. P. Bello, "On the Relative Importance of Individual Components of Chord Recognition Systems," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 22, no. 2, pp. 477–492, 2014.

[12] E. J. Humphrey and J. P. Bello, "Rethinking Automatic Chord Recognition with Convolutional Neural Networks," in *Proc. Intl. Conf. on Machine Learning and Applications (ICMLA)*, pp. 357–362, 2012.

[13] S. Sigtia, N. Boulanger-Lewandowski, and S. Dixon, "Audio Chord Recognition with a Hybrid Recurrent Neural Network," in *Proc. ISMIR*, pp. 127–133, 2015.

[14] M. McVicar, R. Santos-Rodríguez, Y. Ni, and T. De Bie, "Automatic Chord Estimation from Audio: A Review of the State of the Art," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 22, no. 2, pp. 556–575, 2014.

[15] F. Korzeniowski and G. Widmer, "Feature Learning for Chord Recognition: The Deep Chroma Extractor," in *Proc. ISMIR*, pp. 37–43, 2016.

[16] A. Ramesh *et al.*, "Zero-Shot Text-to-Image Generation," arXiv:2102.12092, 2021.

[17] R. Rombach *et al.*, "High-Resolution Image Synthesis with Latent Diffusion Models," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

[18] J. Pauwels, K. O'Hanlon, E. Gómez, and M. B. Sandler, "20 Years of Automatic Chord Recognition from Audio," in *Proc. Intl. Conf. on Music Information Retrieval (ISMIR)*, [Location], pp. [pages], [Year].

[19] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[20] J. Deng and Y.-K. Kwok, "Large vocabulary automatic chord estimation using bidirectional

[21] J. A. Burgoyne, J. Wild, and I. Fujinaga, "An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, A. Klapuri and C. Leider, Eds., Miami, FL, 2011, pp. 633–638.

[22] I. A. P. Santana, F. Pinhelli, J. Donini, L. Catharin, R. B. Mangolin, Y. M. e G. da Costa, V. D. Feltrim, and M. A. Domingues, "Music4All: A New Music Database and Its Applications," in *Proceedings of the 27th International Conference on Systems, Signals and Image Processing (IWSSIP 2020)*, Niterói, Brazil, 2020, pp. 1–6.

[23] krist311, "chords-recognition," GitHub repository. [Online]. Available: `https://github.com/krist311/chords-recognition`. [Accessed: Mar. 18, 2025].

9. FIGURES AND EQUATIONS

9.1 Equations

$$\hat{c}_{1:T} = \arg\max_{c_{1:T}} \prod_{t=1}^{T} P(x_t \mid c_t)\, P(c_t \mid c_{t-1}) \qquad (1)$$

9.2 Figures

Figure 1. Generated image based on predicted lyric model.

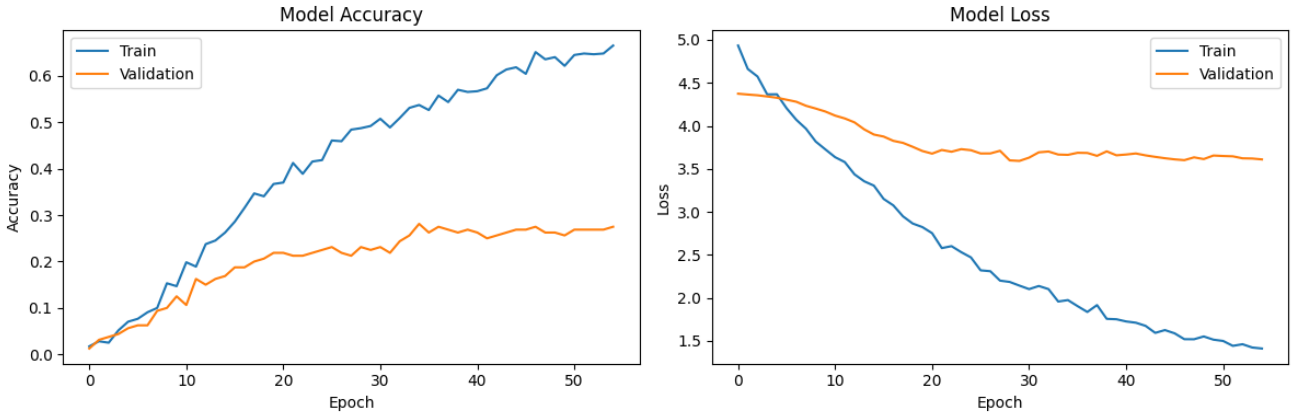


Figure 2. Chord recognization model result

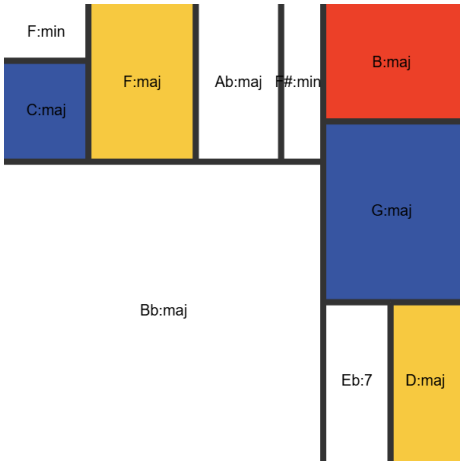


Figure 3. Generated image based on chord percentage

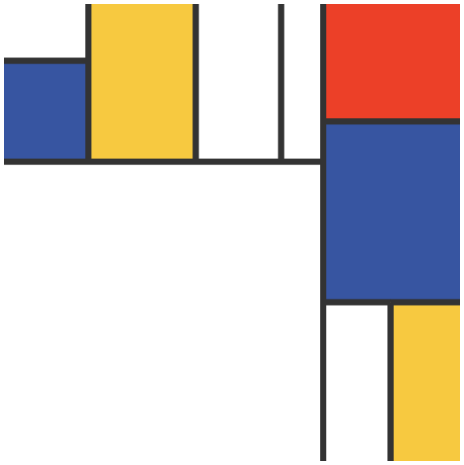


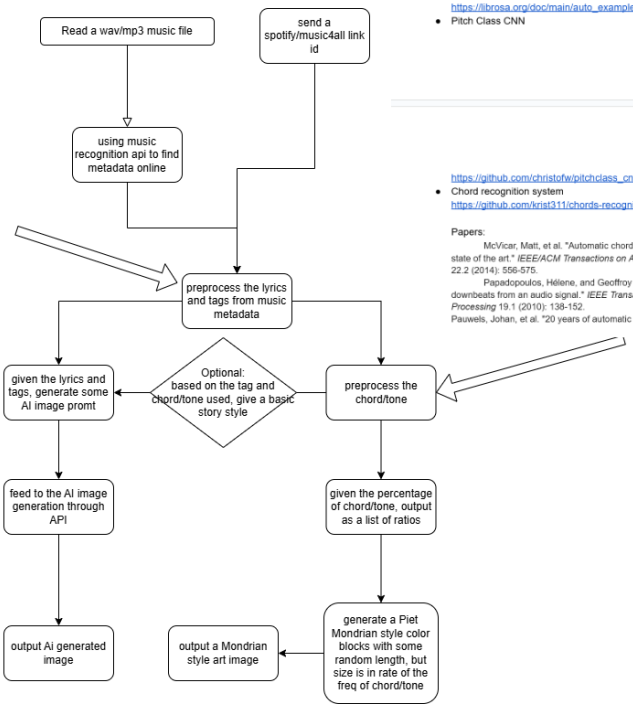
Figure 4. Generated image using Mondrian-style painting

- **DP1 - Lyrics-based auto-tagging**
  Datasets:
  Music4All: I have a local copy
  https://sites.google.com/view/contact4music4all

  Multi-label classification:
  https://scikit-learn.org/stable/modules/multiclass.html

  Text processing:
  https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

  Papers:
  - Turnbull, Douglas, et al. "Semantic annotation and retrieval of music and sound effects." *IEEE Transactions on Audio, Speech, and Language Processing* 16.2 (2008): 467-476.
  - Mayer, Rudolf, and Andreas Rauber. "Musical genre classification by ensembles of audio and lyrics features." *Proceedings of internationa Jim I conference on music information retrieval*. 2011.
  - Santana, Igor André Pegoraro, et al. "Music4all: A new music database and its applications." *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020.
  - Choi, Keunwoo, et al. "Convolutional recurrent neural networks for music classification." *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017.



- **DP4 - Chord recognition from Audio:**
  Datasets:
  - Beatles (I have the audio):
    http://isophonics.net/content/reference-annotations-beatles
  - Chordify
    https://github.com/chordify/CASD
  - Choco
    https://github.com/smashub/choco
  - Jazz Audio Aligned Harmony Dataset
    https://github.com/MTG/JAAH

  Code:
  - Chroma librosa:
    https://librosa.org/doc/main/auto_examples/plot_chroma.html
  - Pitch Class CNN
    https://github.com/christofw/pitchclass_cnn
  - Chord recognition system
    https://github.com/krist311/chords-recognition

  Papers:
  McVicar, Matt, et al. "Automatic chord estimation from audio: A review of the state of the art." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.2 (2014): 556-575.
  Papadopoulos, Hélene, and Geoffroy Peeters. "Joint estimation of chords and downbeats from an audio signal." *IEEE Transactions on Audio, Speech, and Language Processing* 19.1 (2010): 138-152.
  Pauwels, Johan, et al. "20 years of automatic chord recognition from audio." (2019).

Figure 5. Project Pipeline