Chromesthesia Chord Engine

Course: Data Science Final Project

Date: June 11, 2025

Team Members:

1. 나우팔 카마루딘 **(2024320084)**
2. 무함마드 이즈미르 빈 수하이미 **(2022320113)**
3. 아흐마드 나우팔 **(2023320085)**

Introduction / Problem Statement

Chromesthesia in a nutshell is a condition where some people could perceive colors when they hear music or sound. These sound-color associations are unique to each individual, for instance, one person might “see” the notes of C major as blue and A minor as red. Some of the famous musicians like Franz Liszt, Duke Ellington, and Billie Eilish have this condition, with Billie even mentioned that she could literally see colors when listening to music.

The Chromesthesia Chord Engine we have created aims to simulate the senses that these people experience. Using machine learning techniques, we could analyze audio input, identify the chords that are being played, and generate corresponding visual outputs, such as colors and shapes which translate to different chords. Our goal is to let users “see” the music in real time, even if they don’t naturally have synesthesia.

This tool could help both creatively and educationally as it can help novice musicians grasp chord progressions and offer a new way to experience music visually. Major chords are associated with brightness and happiness, and they are mapped to warm colors and soft shapes, while minor chords which are perceived as sad are translated to cool colors and angular shapes. This project blends music, perception, and machine learning into a unique audiovisual experience for all of us to experience.

Dataset

We could not find any public “chords to colors” dataset, therefore we created our own by generating synthetic audio samples of musical chords. Each sample is a short clip of a few seconds containing a single chord, labeled manually such as C Major or A Minor. We focused on the common major and minor triads across the 12-note chromatic scale, resulting in dozens of chord classes.

Key Dataset Details:

- **Chords Covered:** Major and minor chords for all root notes (C to B), providing a wide range of targets for classification tasks.
- **Sample Generation:** We used pure sine waves to synthesize each chord's notes (e.g., C, E, G for C Major) and combined them into audio clips using the librosa library. Differences in octave and timbre also added more diversity to the dataset.
- **Feature Extraction:** Rather than using raw audio, we have extracted chroma features which are 12-dimensional vectors representing energy in each pitch class. For example, an F Major chord would show high values at F, A, and C. These chroma vectors became the input X, while the corresponding chord names were used as the labels y.
- **Train/Test Split:** The dataset was split (e.g., 80/20) into training and testing to evaluate the model's generalization capabilities. Because we've used multiple samples per chord, this ensures that the model learns patterns instead of memorizing individual recordings.

The color and shape mappings were not part of the dataset or training labels. They were applied later using a manual lookup table based on the predicted chord, as described later in the Methodology part of our report.

Methodology

Using this implementation we are able to identify chords from audio using machine learning, as well as mapping them to specific colors and shapes for visualization. The workflow comprises of five main steps:

1. Audio Synthesis & Data Collection

We generated the dataset by synthesizing chord audio samples using sine wave combinations for each note in a chord. This was done with librosa and NumPy, giving us clean and noise-controlled audio. We created both major and minor triads (24 total chord classes), and generated 100 unique samples per chord (2400 in total), and introducing variation in:

- Duration (0.5s to 1.5s)
- Volume levels
- Background noise

This controlled variable improved generalization and made the model more accurate in detecting chord..

2. Feature Extraction (Chroma Representation)

Each audio sample was converted into a Mel spectrogram, a 2D time–frequency representation capturing pitch and timbral structure. Mel spectrograms are widely used in music and speech processing because they emphasize perceptual pitch spacing.

Processing steps are include:

- Using librosa.feature.melspectrogram() followed by power_to_db() to convert to log scale
- Normalizing the spectrograms
- Padding them to a consistent width to ensure uniform input shape

These spectrograms served as image-like inputs to the neural network, preserving the temporal and harmonic structure of the audio.

3. Model Selection and Training

We selected a Convolutional Neural Network (CNN) architecture for its ability to learn spatial hierarchies in spectrogram data.

The model consisted of:

- 3 convolutional layers with increasing filters (32 → 64 → 128)
- Max-pooling layers for downsampling
- A dense layer with dropout for regularization
- A softmax output layer for multi-class classification

Training used the adam optimizer and sparse_categorical_crossentropy loss. We use validation taken from the 80% training data during training to monitor generalization. The model was trained for 50 epochs with a batch size of 32.

4. Model Evaluation

We evaluated the CNN on a test set (20% of the total data) that was held out during training. The model achieved 100% accuracy on this test set, reflecting perfect classification and this high performance is attributed to:

- The clean and balanced dataset
- The CNN's strength in extracting local and global patterns from spectrograms
- Effective use of augmentation (duration, volume, and noise)

Minimal confusion occurred between harmonically similar chords due to the model's strong ability to distinguish subtle patterns in time and frequency.

5. Chord-to-Color Mapping (Visualization)

After predicting the chord for each beat, the system assigns a visual identity to the chord using a static mapping table. This visual identity consists of:

- Color: Represents emotional tone
    - Major chords → warm colors (e.g., yellow, orange, red)
    - Minor chords → cool colors (e.g., blue, purple, teal)

This mapping is not learned by the model but instead, it's a manual design decision based on common synesthetic associations, where certain sounds evoke visual sensations like color or shape.

The visual mapping ensures:

- The same chord always appears with the same color and shape.
- Viewers can intuitively distinguish chord types without reading the label.
- Adds emotional and aesthetic value to the visualization.

These visual attributes are then used in the next step to draw animation frames in sync with the music's chord progression.

Results & Evaluation

Model	Accuracy	Evaluation
MLP (Multi-Layer Perceptron)	71%	This struggled due to treating spectrograms as flat vectors. It lacks the ability to capture time-frequency patterns, which is essential in audio data.
RFC (Random Forest)	85%	This performed better by handling noise and feature variation, but still limited by its inability to process spatial structure in spectrograms.
CNN (Convolutional Neural Network)	100%	This performed best by leveraging the 2D structure of spectrograms and aptured both local and global features, leads to accurate classification.

While MLP and RFC struggled, the CNN's achieve the best performance due to its ability to process 2D time-frequency patterns makes it the most effective model for chord detection in this project.

Discussion

Our results confirm that the model could effectively learn recognizing chords from chroma features. It identifies key note patterns, such as detecting F, A, and C as "F Major" similar to how a musician would. Feature importance analysis showed that certain notes like B were helpful in distinguishing between similar chords, aligning with music theory.

The visual mapping created an intuitive representation of emotions in the music. So, even without knowing the song, viewers could grasp emotional shifts just by observing the visuals.

Limitations

- **Real-world generalization**: The model was trained on clean, isolated chords. In real music, overlapping notes, inversions, or instrument noise can confuse the classifier.

However, additional training on diverse and noisy data would help a long way in improving the model's generalization performance.

- **Subjective mapping**: Color-shape assignments are not universal. Synesthetic experiences vary between the individuals with synesthesia, and the choices that we made for the visual representations are based on general emotion-tone logic. In the future, allowing user customization could definitely improve personalization.
- **Chord variety**: We have only modeled major and minor triads. Real music includes many other types (7ths, diminished, etc) that weren't accounted for. A system that could classify more complex chords will be an interesting approach in the future.
- **Simple visuals**: The current visual output is static. While clear, it's limited. Adding animations like pulsating effects, spinning triangles, or smoother transitions could better reflect musical movement and enhance the chromesthetic experience.

Despite these limitations, this project showcases how a simple ML pipeline with musically informed features can produce meaningful, engaging results.

Conclusion

We have successfully developed the Chromesthesia Chord Engine, a system that analyzes musical chords and visualizes them as color-shape combinations, simulating a chromesthetic experience through machine learning. By combining chroma-based feature extraction, Random Forest classifier, and a visualizer, our model accurately identifies chords from synthetic audio and provides an interesting visual experience that shows how people with synesthesia 'sees' music.

The project successfully integrated several core data science concepts such as feature engineering, classification, evaluation, and creative output. It demonstrated how a subjective idea like "hearing colors" can be captured with structured design and ML tools.

Future Improvements

- **Expanded Chord Coverage**: Include 7ths, diminished, and more complex chords to improve recognition accuracy in music.
- **Enhanced Visuals**: Add animations or effects and allow user-customized mappings or feedback-based learning.
- **Live Instrument Integration**: Build an interactive version for real-time use with MIDI instruments or microphones.
- **Multisensory Extension**: Explore other types of chromesthesia conditions beyond visuals, such as vibration or scent, to further emulate synesthetic perception.

Ultimately, this project captures the essence of creative machine learning which uses simple tools to produce expressive, intuitive results. It's a foundation for future exploration into how data science can bridge sensory boundaries and how it could be somewhat useful for future learning purposes.

Appendix

Project Files

All code, model files, and datasets are included in the submitted GitHub/Google Drive folder, along with setup instructions and dependencies (librosa, scikit-learn, etc.).

Chord-to-Color Mapping

Chord	Assigned Color	Assigned Shape	Rationale (Emotion)
C Major	Bright Yellow	Circle	Happy, “sunny” resolution
A Minor	Deep Blue	Triangle	Sad or mellow
G Major	Orange-Red	Circle	Lively, warm
E Minor	Purple (violet hue)	Triangle	Somber but rich
D Major	Vibrant Green	Circle	Triumphant, fresh

Table 1: *Sample of the chord-to-visual mapping used in the Chromesthesia Chord Engine. Major chords were mapped to brighter, warmer colors and smooth circular shapes, while minor chords were mapped to cooler colors and angular shapes (triangles). This design was chosen to reflect the general emotional quality of the chords.*

Evaluation Report

```
Epoch 1/50

48/48 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26s 486ms/step - accuracy: 0.0390
- loss: 6.2902 - val_accuracy: 0.0417 - val_loss: 3.1727

Epoch 2/50

48/48 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23s 489ms/step - accuracy: 0.0340
- loss: 3.1821 - val_accuracy: 0.0391 - val_loss: 3.1772

Epoch 49/50

48/48 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23s 474ms/step - accuracy: 0.9941
- loss: 0.0251 - val_accuracy: 1.0000 - val_loss: 1.0865e-08

Epoch 50/50
```

```
48/48 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41s 474ms/step - accuracy: 0.9892
- loss: 0.0298 - val_accuracy: 1.0000 - val_loss: 6.5192e-08

Test Loss: 0.0000

Test Accuracy: 1.0000
```

Code Snippet – Feature Extraction + CNN Prediction per Beat

```
predicted_chords = []

for beat_time in beat_times:

    start = int(max(0, sr * (beat_time - 0.25)))

    end = int(min(len(y), sr * (beat_time + 0.25)))

    segment = y[start:end]

    if len(segment) == 0:

        continue

    mel = extract_mel_spectrogram(segment, sr=sr, max_len=128)

    mel_input = np.expand_dims(mel, axis=(0, -1))  # Add batch and channel dims

    prediction = model.predict(mel_input)

    label = label_encoder.inverse_transform([np.argmax(prediction)])[0]

    predicted_chords.append((beat_time, label))
```

Reference

The Synesthesia Tree. (2021, February). *Chord-colour*. Retrieved June 9, 2025, from https://www.thesynesthesiatree.com/2021/02/chord-colour.html

The London Piano Institute. (n.d.). *Synesthesia and piano: How some musicians see colours when they play*. Retrieved June 9, 2025, from https://www.londonpianoinstitute.co.uk/synesthesia-and-piano/