Recognition and Classification of Musical Audio Signals

Andrea Ingoglia
University of Milan (Università Statale degli Studi di Milano)
Milan, Italy
andrea.ingoglia@studenti.unimi.it

***Abstract*—This project is an attempt to establish a basis for real-time musical chord recognition. The system initially classifies a signal as being harmonic or percussive based on DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering, and KNN (K-Nearest Neighbors) classification. Then, two models are developed to identify musical chords. The first one uses template matching with chroma vectors and Hamming distance. The second is a convolutional neural network (CNN) trained on spectrograms with interpretability enhanced by LIME (Local Interpretable Model-agnostic Explanations). The results demonstrate that the system can classify audio signals and recognize chords.**

I. Introduction

The identification of musical chords is a very challenging task in the area of audio signal processing. Chords are basic units of harmonic structures in music, so their accurate recognition is indispensable in many applications of music analysis, entertainment, and education. Real-time chord recognition can be difficult because of the variability in musical signals, background noise, and overlapping frequencies. These make it hard to create systems that are both efficient and accurate.

II. Dataset Description

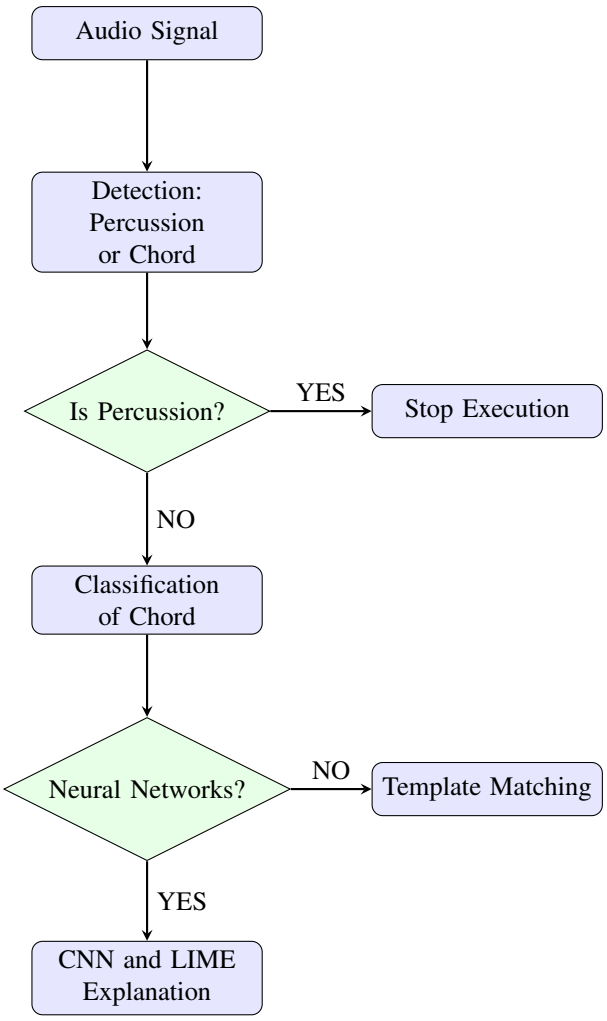
A. Audio Piano Triads Dataset

The dataset called *Audio Piano Triads Dataset*, available at **Zenodo** and created on 5 May 2021, by Agustn Macaya Valladares, contains 43,200 audio examples of piano triads in .wav format. The audio files are 4 seconds in length, with 3 seconds of the chord being played, followed by 1 second of release. The recordings were performed by a human player on a velocity-sensitive digital piano keyboard. The dataset contains chords played on three octaves (the 2nd, 3rd and 4th), using 12 base notes per octave (C, C$^\sharp$, D, D$^\sharp$, E, F, F$^\sharp$, G, G$^\sharp$, A, A$^\sharp$, B). For every base note, four types of triads are specified: major, minor, diminished and augmented. Each triad is recorded at three different dynamic levels: forte, mezzo-forte and piano. The filename contains embedded metadata about each chord, including octave, fundamental note, triad classification, and dynamic intensity.

B. 29k Samples Drums Dataset

The *29k Samples Drums Dataset*, available at **Zenodo** and created on 10 June 2021, by Macià Amorós i Cortiella, contains 29,000 single drum samples, classified into 22 different classes; each class represents a single or a combination of drums in a drum set: cymbals, toms, snare drums, kick drums, and hi-hats, thus providing a diverse range of percussive sound samples.

III. Method Workflow



The approach adopted in this work is twofold. First, the audio signals are classified as either harmonic or percussive using clustering and classification methods, namely, DBSCAN and KNN. Next, for harmonic signal classification, two types of models are adopted: a conventional template matching system based on the chroma vector and Hamming distance and a convolutional neural network (CNN) trained using

spectrograms. In order to increase the interpretability of the CNN, LIME is used.

IV. Feature Extraction and Clustering

The clustering process starts by applying DBSCAN, a method designed especially for identifying clusters based on the density of data points. DBSCAN separates the main signal types, chords and percussions, showing quite distinct regions in the 3D visualization. Chords form a cluster close to the origin—a dense one—because of their lower variability and more structured characteristics, while the percussive signals are more scattered, showing higher variability and irregular characteristics. The *epsilon* parameter ϵ defining the maximum allowed distance between points in the same cluster, was set to 0.2. The *min_samples* parameter, defining the minimum number of points to form a cluster, was set to 3. These parameter values ensure that a cluster will be created only if the features—Spectral Centroid, Zero-Crossing Rate, and Spectral Flux—are numerically close to each other, typically corresponding to harmonic signals (yellow cluster). Data points that do not meet these criteria are either assigned to a secondary cluster or labeled as noise, thus ensuring that irrelevant or incoherent signals are excluded from further analysis.

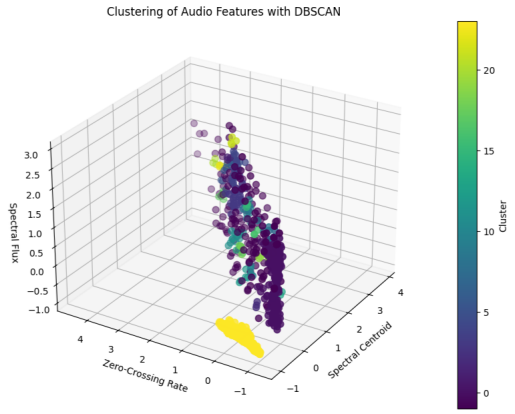


Fig. 1. DBSCAN Clustering

A. Overview on the Extracted Features

The features—Spectral Centroid, Zero-Crossing Rate, and Spectral Flux—proved to be effective in allowing DBSCAN to achieve a clear separation between harmonic and percussive classes. These features were then used as input to KNN classification:

- The **Spectral Centroid** describes the range of frequencies where most of the energy in a signal is found. For percussive signals, this measurement is typically high, as they often contain energy at higher frequencies. Harmonic signals, on the other hand, are usually dominated by lower frequencies—those of their fundamental tone and its harmonics;
- The **Zero-Crossing Rate** (ZCR) quantifies the frequency at which the waveform of a signal reverses its direction by intersecting the zero-amplitude axis. Percussive signals, noted for their abrupt changes in waveform, exhibit elevated ZCR values. In contrast, harmonic signals generate waveforms that are more fluid and continuous, leading to diminished ZCR values;
- **Spectral Flux** measures the rate of change in the frequency composition of a signal along a time axis, indicating how the stationarity is changing.Harmonic chords can be said to have lower spectral flux, as their frequency composition remains fairly invariable, with stable fundamental frequencies and harmonics.

V. Signal Detection and Classification

The classification of the harmonic and percussive signals is done by the K-Nearest Neighbors algorithm. The previously extracted features used in this classification are Spectral Centroid, Zero-Crossing Rate, and Spectral Flux. The algorithm classifies new data points based on their labels of its 4 nearest neighbors in the feature space, using weighted Euclidean distance to compute proximity. The KNN model was trained on a data split, setting aside 70% for training and keeping 30% for the test sets. 10-fold cross-validation was then performed on the training data. The trained KNN model was evaluated on the test set, and it gave almost perfect accuracy, proving that it can classify harmonic and percussive signals with very small errors.

This is likely due to the quiet and denoised nature of the dataset, which makes the signal practically perfect for classification without requiring complex preprocessing techniques before training. Its well-defined structure and lack of variability in the input data likely made the feature space easier to interpret, so that the KNN algorithm could perform its task perfectly. In addition, the features selected—strongly related to the physical properties of harmonic and percussive signals—may have further enhanced the discriminative power of the classes.

Although such elevated performance is commendable, it may suggest a restricted applicability to more varied or noisy datasets encountered in real-world scenarios. Subsequent researches could investigate the effects of incorporating noise to evaluate the model's resilience in less regulated environments.

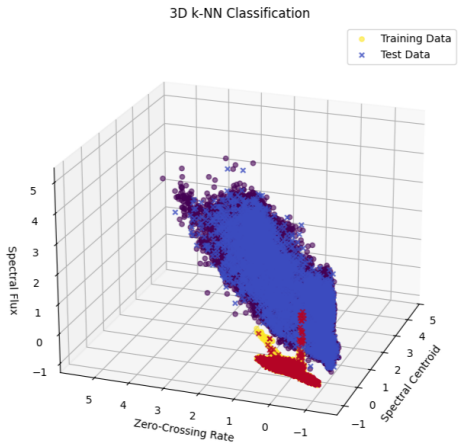


Fig. 2. k-Nearest Neighbors Classification

Presented below is the confusion matrix, detailing the classification and misclassification outcomes:

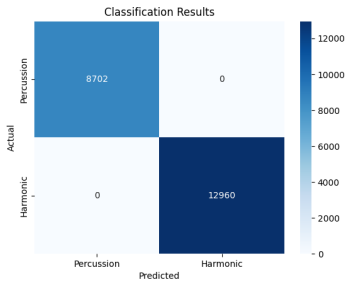


Fig. 3. Classification Results

VI. Template Matching for Chord Classification

As a first approach, the template matching method is used for harmonic signals. This method is inspired by the paper "Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music" [1] by Professor Takuya Fujishima at CCRMA, Stanford University.

The model utilizes chroma vectors, an efficient way to represent the pitches of a chord, aligning its fundamental and harmonics within a single 12-pitch class cycle, allowing for comparisons independent of frequency. To identify chords, the system compares an input binarized chroma vector—representing triads by three active pitches set to 1—to a set of predefined chord templates using the Hamming distance, selecting the template with the shortest distance to the input vector as the recognized chord.

The reliability of this methodology depends entirely on the quality of the input signal and the level of noise it contains. In fact, unless the preprocessing used is strong enough to isolate pertinent information and weaken the influence of noise, then the accuracy and consistency of the system may both be significantly degraded.

The table shows the accuracy, precision, recall, and F1-score values achieved by the template matching algorithm on the specific audio tracks used to validate the approach:

TABLE I
Template Matching Performance

Metric	Value
Accuracy	0.8147
Precision	0.9275
Recall	0.8147
F1 Score	0.7729

A. Design Decisions

1) The technique to estimate chromagrams is based on the application of constant Q-transform which yields a spectrum with logarithmically spaced frequencies. It was proposed by J. P. Bello and J. Pickens in [2], in their article, "A Robust mid-level representation for harmonic content in music signals". This process generates a log-spectrum where the frequencies are not linearly placed like in the STFT and, for that matter, are brought closer to the frequency resolution of the human ear. The log spectrum is then folded to attain a representation showing the 12 pitch classes;
2) The CQT is calculated at a high resolution to facilitate detailed frequency analysis. This becomes important since the chord samples are 4 seconds long. The shorter the audio samples, the less temporal information they contain, and thus, the more critical it becomes to grab as much of the frequency detail as possible in that limited time;
3) To reduce noise, a dynamic threshold attenuates low-energy components of the spectrum, thus retaining only the critical frequencies related to musical tones. The filtered CQT is then used to extract the chroma vector, which is applied as a median filter, that smooths the chroma representation by reducing rapid variations caused transient frequencies, improving the consistency of the detected pitch;
4) After smoothing, the aggregated chroma values for each pitch class are used to identify the three most important pitches. A binary vector is created by setting the indices of these dominant pitches to a value of 1 and all other indices to a value of 0;
5) The binary vectors are compared against pre-computed chord templates extracted from a ground truth data set through the Hamming distance, that measures the number of diverging elements between two binary vectors; accordingly, the chord that has the minimum Hamming distance compared to the input vector is the predicted chord.

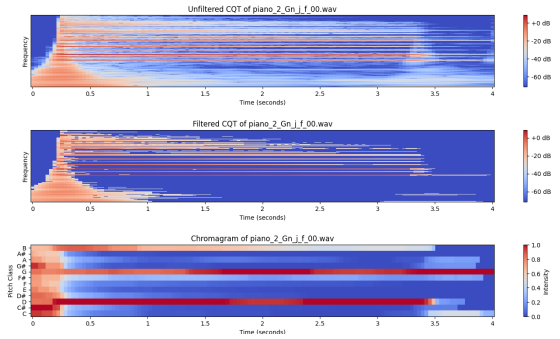


Fig. 4. Resulting Chroma Vector of G Major Chord (G, B and D Notes)

VII. Convolutional Neural Network for Chord Classification

A. Design of CNN

The aim of this section was to develop a Convolutional Neural Network (CNN) capable of distinguishing between various chords by using their visual representations. This approach is inspired from some findings demonstrated in the work "Transfer Learning for Audio Waveform to Guitar Chord Spectrograms" [3].

The first step included creating spectrograms from audio files containing different chords, saved as grayscale images with a resolution of 128x128 pixels. Since the audio files were recorded with very few distortions and resulted in similar spectrograms, data augmentation techniques were used on the train dataset to avoid data leakage and enhance the model's generalization ability. These transformations were designed to simulate real-world variations, such as timing delays, pitch changes, and recording differences, while preserving the essential patterns in the spectrograms:

- horizontal shifts up to 20% of the image width;
- vertical shifts up to 10% of the height;
- zooming in or out by up to 20%;
- additionally, pixel values were normalized to the range [0, 1].

The CNN, designed for grayscale spectrogram images of size 128x128x1, has three convolutional layers with an increasing number of filters: 32, 64, and 128 with size 3x3 and the ReLU activation function, ensuring the extraction of spatial features from the spectrograms. Immediately after each convolutional layer, a max-pooling layer was used with a 2x2 window to downsample the spatial dimensions in order to retain important features. Then, a dense layer of 128 neurons was followed by a dropout layer with a 50% rate, with the purpose of reducing overfitting and build knowledge. And finally, the output layer used a softmax activation function to map those features to classes of chords.

B. Training phase

The model was trained using the categorical cross-entropy loss function, which is suitable for tasks where multi-class classification is performed. Additionally, early stopping was applied in order to prevent overfitting by monitoring validation accuracy with a patience parameter of 5 epochs. The training regime was carried out for up to 20 epochs using the augmented dataset.

In order to evaluate the performance of the model, various metrics—accuracy, loss, and error—were observed throughout the training process, providing insights into how well the model generalizes in terms of preventing overfitting.

The model's training and validation performance demonstrates a high level of generalization: the validation accuracy is always higher than the training accuracy, while the validation loss is always lower than the training loss. Such trends can be traced back to the use of data augmentation and dropout regularization, which inject added complexity and noise into the training process. The overall reduction in both training and validation loss, coupled with the convergence of the training and validation errors, indicates that the model does not overfit and learns generalization, being effective on previously unseen data.

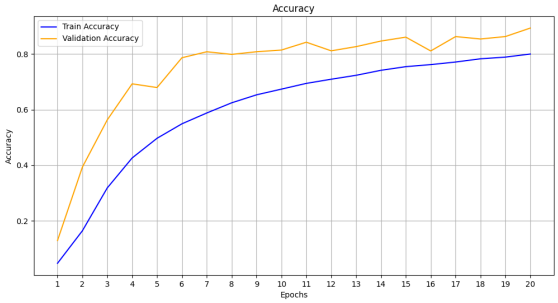


Fig. 5. CNN Accuracy

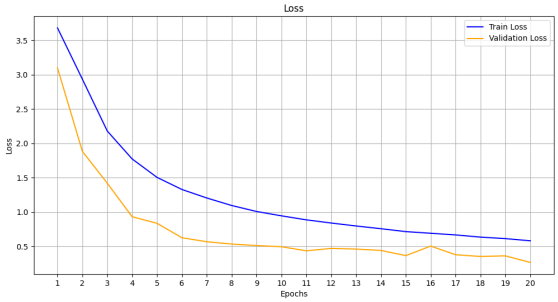


Fig. 6. CNN Loss

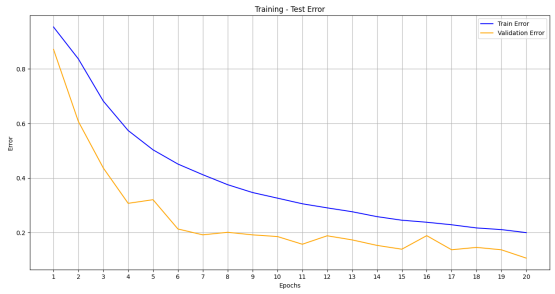


Fig. 7. CNN Train and Test Errors

TABLE II
TRAINING AND VALIDATION METRICS PER EPOCH

Epoch	Accuracy	Loss	Val. Accuracy	Val. Loss	Time (s)
1	0.0314	3.8060	0.1292	3.0966	167
2	0.1245	3.1473	0.3933	1.8827	194
3	0.2811	2.3318	0.5631	1.4191	192
4	0.4052	1.8578	0.6927	0.9308	172
5	0.4794	1.5693	0.6794	0.8353	170
6	0.5376	1.3695	0.7865	0.6249	171
7	0.5803	1.2299	0.8080	0.5669	171
8	0.6154	1.1130	0.7986	0.5329	174
9	0.6423	1.0375	0.8081	0.5115	172
10	0.6736	0.9422	0.8142	0.4945	172
11	0.6869	0.9069	0.8424	0.4344	172
12	0.7064	0.8463	0.8114	0.4698	179
13	0.7198	0.8089	0.8265	0.4602	201
14	0.7400	0.7587	0.8466	0.4400	213
15	0.7503	0.7289	0.8606	0.3648	200
16	0.7587	0.7007	0.8110	0.5036	182
17	0.7673	0.6779	0.8627	0.3774	186
18	0.7828	0.6385	0.8539	0.3527	204
19	0.7847	0.6309	0.8627	0.3613	179
20	0.8012	0.5770	0.8932	0.2657	180

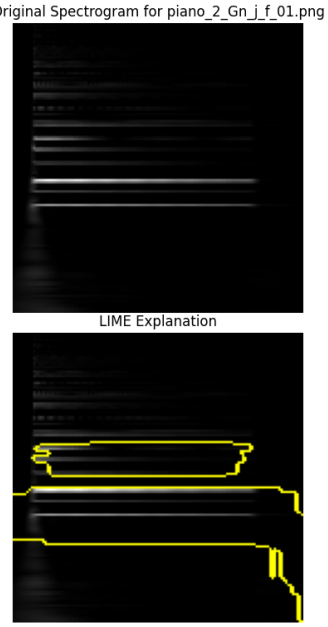
Metric	Test Loss	Test Accuracy
Final Test Results	0.2593	0.8969

VIII. CNN EXPLAINABILITY

LIME (Local Interpretable Model-agnostic Explanations) is a methodology that provides localized explanations for predictions made by machine learning models, by highlighting the most important areas within the input image. It works by slightly modifying the input image, partitioning it into ”superpixels” (homogeneous areas) and examining the impact of local perturbations on the model predictions. This approach reveals which parts of the spectrogram most affect chord recognition. The highlighted areas correspond roughly to spatial regions that contain the most dominant spectral components to which the model’s predictions are sensitive.

IX. CONCLUSION

The present project investigated various ways of dealing with the task of chord classification from audio signals, merging traditional signal processing with data mining methodologies. A crucial step in the workflow was separating percussive elements from chords by using DBSCAN (Density-Based



[!h]

Fig. 8. LIME Explaination for G Major Chord

Spatial Clustering of Applications with Noise) and K-Nearest Neighbors. This preprocessing step guaranteed that the input data for the chord classification module would contain only harmonic content, hence filtering out the percussive tracks. These two classification methods were then implemented and compared after this preprocessing stage: Template Matching and Convolutional Neural Networks (CNNs). Template Matching offered a rule-based paradigm including the comparison of input audio features to predefined templates. This method, although robust in controlled environments, proved vulnerable to variations in the data, like distortions or noise, due to which it lacked adaptability. The CNN took spectrograms as input and automatically learned meaningful features for the classification of chords. Using data augmentation techniques, the CNN showed a good generalization ability in dealing with variations of timing, pitch, and recording conditions. Prospective research efforts can focus on integrating these methods in a real-time system for chord recognition, which allows for dynamic analysis of the audio signal as soon as it is received. Further improvement in performance can also be expected through advanced preprocessing techniques and by improving the quality of input data. Also, expanding the dataset to include more variety in the set of chords, along with the addition of noise to mimic real-world conditions, would further strengthen the system’s robustness and generalization.

REFERENCES

[1] T. Fujishima, ”Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music”, 1999.
[2] J. P. Bello and J. Pickens, ”A Robust mid-level representation for harmonic content in music signals” in ISMIR, 2005, vol. 5, pp. 304-311
[3] Y. Jadhav, A. Patel, R. H. Jhaveri, R. Raut, S. Hakak ”Transfer Learning for Audio Waveform to Guitar Chord Spectrograms Using the Convolution Neural Network”, 2022