Identification of Musical Instruments and Cover Songs or Non-Cover Songs

1st P r o f Aabha Marathe
Electronics and Telecommunication
Vishwakarma Institute of Technology
Pune, India
abha.marathe@vit.edu

2nd Pushpa
Electronics and Telecommunication
Vishwakarma Institute of technology
Pune, India
pushpa.pushpa23@vit.edu

3rd Shreyas Phansalkar
Electronics and Telecommunication
Vishwakarma Institute of Technology
Pune , India
shreyas.phansalkar23@vit.edu

4th Raghavi Narayanan
Electronics and Telecommunication
Vishwakarma Institute of Technology
Pune, India
raghavi.narayanan23@vit.edu

Abstract*— This research addresses two critical challenges in Music Information Retrieval (MIR): cover song identification and musical instrument recognition. A dual-pipeline framework is developed using comprehensive audio feature analysis combined with classical machine learning and deep learning approaches. The cover song identification pipeline employs similarity-based techniques utilizing Mel-Frequency Cepstral Coefficients (MFCCs), Chroma features, and spectral descriptors to generate cross-similarity matrices that effectively distinguish cover songs from non-covers. Comparative analysis across feature transformation techniques, including PCA and SelectKBest, reveals that complete feature representations yield superior classification outcomes. The study validates the effectiveness of ensemble learning methods for structured audio data while highlighting challenges in small-dataset scenarios for deep learning approaches.

Keywords—Cover Song Identification, Musical Instrument Recognition, Music Information Retrieval, Feature Engineering

I. Introduction

The increase in digital audio materials has prompted advanced computational capabilities for automated music analysis and retrieval. Music Information Retrieval (MIR) has surfaced as a dominant research area that investigates the extraction, organization, analysis, and understanding of large-scale musical repositories. Deep learning techniques, especially those utilizing spectrogram-based representations as input to Convolutional Neural Networks (CNNs), have demonstrated significant success on a variety of audio classification problems.

This work explores two core problems in MIR: cover song detection and musical instrument classification. Cover song detection determines when different performances of a music tune/exemplar are being performed. This is a challenging endeavor given significant variations in tempo, key transposition, instrumentation, and arrangement style. Musical instrument classification automatically identifies the sources of instruments present in a recording. Depending on the study, instrument recognition could lead to music transcription and genre classification applications.

Both tasks require computing systems to understand deep musical content beyond simply the acoustic qualities. Cover song identification seeks to detect different performances of identical musical compositions—a complex challenge due to substantial variations in tempo, key transposition, instrumentation, and arrangement styles. Musical instrument recognition focuses on automatically identifying instrumental sources within recordings, serving as a foundational capability for music transcription and genre classification systems.

Both tasks require computational systems to extract deep musical content beyond surface-level acoustic properties. Cover song detection necessitates capturing melodic and harmonic structures invariant to performance differences, while instrument recognition demands precise modeling of timbral characteristics unique to each instrument class. Solutions to these challenges have significant implications for copyright management systems, recommendation algorithms, and musicological research methodologies.

This study develops a comprehensive framework addressing both problems through distinct but complementary pipelines. The cover song identification system employs similarity-based analysis using feature representations including MFCCs, Chroma vectors, and spectral descriptors. The instrument recognition pipeline implements and evaluates multiple machine learning

architectures, including Decision Trees, Random Forests, XGBoost, AdaBoost, and Convolutional Neural Networks.

The paper proceeds as follows: Section II reviews relevant literature in audio feature engineering and classification architectures. Section III details the system architecture encompassing data acquisition, preprocessing, feature extraction, and model selection strategies. Section IV presents experimental methodology and comparative results across both tasks. Section V discusses implications and limitations, while Section VI concludes with future research directions.

II. LITERATURE REVIEW

The transformation of raw audio into a feature-rich representation is fundamental to the success of any audio analysis model. The predominant approach involves converting the one-dimensional audio signal into a two-dimensional Mel spectrogram, which plots frequency against time. This representation is particularly powerful because its frequency axis is scaled logarithmically, closely mimicking the non-linear perception of pitch in the human auditory system and making it an ideal input for image-based deep learning models [1, 3, 9]. Beyond this, Mel-Frequency Cepstral Coefficients (MFCCs) are widely used; these are derived by taking the cosine transform of the log-Mel spectrogram, which de-correlates the features and compacts the energy into the first few coefficients, providing a highly efficient summary of the spectral shape [4, 19]. To create even more powerful inputs, researchers have successfully fused multiple feature types, such as combining spectral features with simpler time-domain statistics [4] or merging different spectrogram representations like CWT and Gammatone to capture varied signal characteristics [6]. The most advanced techniques move beyond these fixed, pre-defined features. SpectNet, for example, pioneered an end-to-end approach with a learnable filterbank layer, where the model itself learns the optimal frequency bands and filter shapes for the specific task, rather than relying on the fixed Mel scale [5]. In the realm of robust identification, novel features like topological summaries have emerged, which describe the "shape" of the spectrogram data in a way that is invariant to distortions like time-stretching or noise—much like identifying a coffee mug by its handle, a feature that persists even if the mug is slightly deformed [15]. This contrasts with models that bypass feature engineering altogether, operating directly on raw audio waveforms to learn all relevant features from scratch [22].

The architecture of the learning model dictates its ability to find patterns within the feature representations. Convolutional Neural Networks (CNNs) are the dominant architecture because they excel at learning spatial hierarchies of features from the image-like spectrograms—identifying simple patterns like edges and textures in early layers and combining them into complex shapes and objects in deeper layers [1, 3]. While custom CNNs perform well, transfer learning from models pre-trained on large image datasets, such as VGG19, often provides a significant performance boost by leveraging generalized feature extractors [1]. For tasks requiring a pixel-level output, such as separating a specific sound source, specialized architectures like the U-Net are employed. Its encoder-decoder structure with skip connections allows it to capture multi-scale context and produce a detailed segmentation map of the spectrogram, effectively isolating elements like a singing voice from background music [7]. Performance is often improved by creating hybrid models that combine the strengths of different architectures. A GMM–DNN framework, for instance, can leverage the probabilistic modeling of Gaussian Mixture Models with the discriminative power of Deep Neural Networks to achieve superior classification accuracy [10]. Similarly, Convolutional Recurrent Neural Networks (CRNNs) pair CNNs for feature extraction with RNNs to model the temporal sequences within the audio, which is crucial for understanding context [22]. The field also benefits from cross-domain inspiration; methods from speaker identification have been adapted to create unique "x-vector" embeddings for instruments, capturing their core timbral identity in a dense vector, which has led to significant gains in recognition tasks [24]. This progress is foundationally supported by large-scale pre-training, where models like PANNs are trained on massive, diverse datasets like AudioSet, learning a universal "ear" for sound that can be fine-tuned to achieve state-of-the-art results on a wide variety of specific audio tasks [23].

III. SYSTEM ARCHITECTURE AND METHODOLOGY

The proposed system is designed is divided into two distinct but related pipelines: one for cover song identification and another for musical instrument recognition. Both pipelines share an initial audio processing and feature extraction stage, after which they diverge into specialized analysis and modeling techniques.

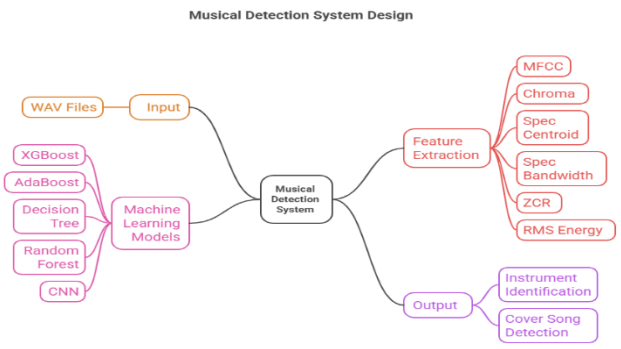


Fig. 1 Block Diagram for Musical Instruments Identification

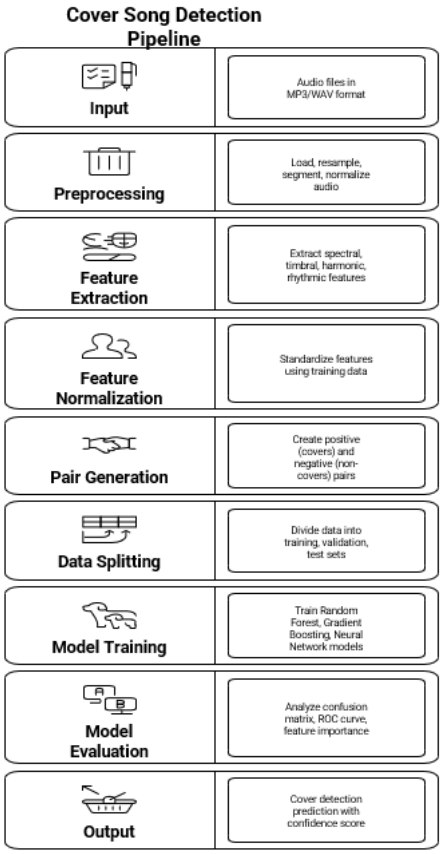


Fig. 2 Block Diagram for Cover and Non-Cover Song Identification

A. Data Acquisition and Preparation

The uppermost stage of the workflow is for the user to provide the name of a geographical location. Dataset: All experiments used the IRMAS (Instrument Recognition in Musical Audio Signals) dataset [9]. The data was chosen due to it being publicly available, its clear structure, and because it is a widely-adopted benchmark for the MIR community, which also allows for greater comparison with other projects. The dataset features 3-second audio clips sampled from 44.1kHz from real-world musical recordings of multiple genres. The musical samples were chosen because they come from commercially produced music, which allows the models to be trained on real-world data, including polyphonic contexts that have several accompanied instruments; however, the predominant sound source of the music was the instrument. The dataset features a total of 6,705 samples for the training dataset and 2,163 samples for the test dataset, spread across 11 instrument types: Cello (cel), Clarinet (cla), Flute (flu), Acoustic Guitar (gac), Electric Guitar (gel), Organ (org), and Piano.

This project utilized the Cover80 dataset, a standard, publicly available benchmark for cover song detection research. The dataset comprises 80 unique musical compositions, with each composition represented by multiple performances.

The raw dataset contains a total of 164 audio recordings in MP3 formats, averaging approximately 2.05 versions per song. The collection is musically diverse, spanning genres such as pop, rock, jazz, classical, and electronic music. A primary challenge noted in the report is the small size of this dataset (178 total pairs), which heavily influences model selection and limits the effectiveness of complex deep learning architectures.

B. *Data Curation and Preprocessing:* Before feature extraction, a standardized pre-processing protocol was applied to every audio file in both the training and testing sets.

1. Loading & Resampling: Each audio file was loaded and resampled to a uniform 22,050 Hz. This sample rate is standard for music analysis, as it captures the full range of human hearing (up to ~11 kHz, per Nyquist theorem) while reducing computational load.
2. Mono Conversion: Stereo tracks were converted to mono by averaging the left and right channels.
3. Segmentation: A fixed-length 30-second segment was extracted from each audio file. This ensures that all feature vectors represent the same temporal duration.
4. Amplitude Normalization: L2 normalization
$$y_{norm} = y / \max(|y|)$$
was applied to normalize the signal's amplitude. This technique makes the analysis less sensitive to variations in recording loudness and dynamic range.

Time Stretching & Pitch Shifting: These are often used for data augmentation. You could, for example, create new training samples by slightly changing a song's tempo or key, making your model more robust to such variations in real cover songs.

Feature Space Transformation (Parallel Paths)

The "Original" feature set might be very large and suffer from the "curse of dimensionality," containing redundant or irrelevant information that could hinder model performance. This stage explores whether creating a more compact feature space can improve model performance by reducing noise and complexity. The process splits into three independent paths to create three distinct datasets for the models to train on.

Path A: No Transformation (Baseline): This path acts as the control group. The "Original" feature set is used directly without any modification. This provides a crucial baseline measurement of how the models perform with all available information, against which the other paths can be compared.

Path B: Principal Component Analysis (PCA): This path uses an unsupervised dimensionality reduction technique. PCA transforms the data into a new, lower-dimensional space by identifying the directions of maximum variance, known as principal components. These new features are linear combinations of the original ones and are mutually uncorrelated. The primary goal of PCA is to reduce

dimensionality while retaining as much of the original signal's variance as possible, effectively de-noising the data and reducing multicollinearity [1]. The output is the "PCA" Feature Set, which has fewer columns (features) than the original but aims to preserve its core informational structure.

PCA is mathematically represented as :

$$Z = XW$$

where:

- $X \in R^{nxd}$ is the feature matrix,
- $W \in R^{dxk}$ is the matrix of the top k eigenvectors of the covariance matrix,
- $Z \in R^{nxk}$ is the reduced feature space.

This shows dimensionality reduction while preserving maximum variance.

Path C: SelectKBest (ANOVA F-test): This path uses a supervised feature selection technique, meaning it uses the class labels to inform the selection process. Unlike PCA, it does not create new features but instead selects a subset of the original features. The SelectKBest algorithm scores each feature based on a statistical test and retains only the 'k' features with the highest scores. The chosen scoring function, the Analysis of Variance (ANOVA) F-test, is particularly well-suited for this task. It assesses whether the means of a single feature are significantly different across the multiple instrument classes. A high F-score suggests that the feature is highly discriminative and valuable for classification [2]. The output is the "KBest" Feature Set, containing what are statistically the most relevant features for distinguishing between the instruments.

C. Feature Extraction

The process of feature extraction is arguably the most critical stage in a classical machine learning pipeline for audio analysis. It involves transforming the raw, high-dimensional audio signal into a compact, informative set of numerical descriptors, or "features." These features are designed to capture the essential acoustic properties of the sound—such as its timbre, pitch, and texture—while discarding irrelevant information like phase or loudness. The quality of these features directly constrains the potential performance of any subsequent classification model.

This study employs a comprehensive set of features, standard within the Music Information Retrieval (MIR) community, to build a rich and discriminative representation of each musical instrument. The features can be broadly categorized into Timbral (Cepstral), Harmonic (Chroma), and Spectral descriptors. Table 1. below provides a detailed description of each feature, its relevance to the task of instrument recognition, its implementation parameters, and a corresponding citation.

Table I: Feature Extraction

Feature	Description	Citation
MFCCs	Models the timbre of the sound using 20 coefficients based on the perceptual Mel scale.	[3]
Chroma	Captures the harmonic content by mapping energy to the 12 musical semitones.	[4]
Spectral Features	Includes Centroid (brightness), Bandwidth (spread), and Rolloff (shape), to describe the spectrum.	[1], [14]
Temporal Features	Includes RMS Energy (loudness/dynamics) and Zero-Crossing Rate (noisiness).	[1]

MFCC Computation

It stands for mel-frequency cepstral coefficient (MFCCs) as:

Where:

$$C_n = \sum_{k=1}^{K} \log(S_k) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\Pi}{k}\right]$$

Where:

C_n = n^{th} MFCC coefficient

S_k = Mel-scaled power spectrum

K = number of Mel filters

This equation represents the Discrete Cosine Transform (DCT) of the log-Mel spectrum — a key transformation that converts frequency information into a compact cepstral domain.

The conversion of raw audio signals into a robust, fixed-length numerical representation is a critical stage. All feature extraction was performed on 30-second, monaural audio segments that were standardized to a sample rate of 22,050 Hz. The methodology involved computing time-series representations for various audio descriptors using the Librosa library (v. 0.10.x). These time-series were then statistically aggregated by calculating their mean and standard deviation across the time dimension. This process resulted in a single 131-dimensional feature vector for each audio file, composed of the following four categories.

1. Spectral Features (6 dimensions)
   These features describe the general shape and "brightness" of the audio spectrum, derived from the Short-Time Fourier Transform (STFT).
   Spectral Centroid: The mean (1) and standard deviation (1) of the spectral center of mass.

Spectral Rolloff: The mean (1) and standard deviation (1) of the frequency below which 85% of the spectral energy resides.
Spectral Bandwidth: The mean (1) and standard deviation (1) of the weighted standard deviation of the spectrum.

2. Timbral Features (64 dimensions)
These features capture the textural and "color" qualities of the sound, which are closely related to instrumentation and recording quality.
Mel-Frequency Cepstral Coefficients (MFCCs): The mean (20) and standard deviation (20) of the first 20 MFCCs.
3. Harmonic Features (60 dimensions
These features are considered the most discriminative for cover song detection, as they model the underlying pitch and harmonic content which are often preserved across covers.
Chroma STFT: The mean (12) and standard deviation (12) of the 12-bin chromagram derived from the STFT.
Chroma CQT: The mean (12) and standard deviation (12) of the chromagram derived from a Constant-Q Transform (CQT). The CQT provides a musically-logarithmic frequency resolution, offering a more accurate representation of musical pitch.

D. Model Training

1) Musical Instruments Identification

The model selection process for this study was designed to provide a comprehensive and comparative evaluation of both traditional and deep learning approaches for musical instrument classification. The objective was to establish robust performance baselines using classical machine learning algorithms and then explore the potential improvements achievable through advanced ensemble and neural network architectures.

The initial phase employed several foundational algorithms to ensure interpretability and to provide baseline comparisons. A single Decision Tree classifier was implemented as a simple, interpretable model capable of capturing non-linear feature relationships while serving as a transparent reference point for evaluating more complex ensemble techniques built upon the same tree-based principles.

To explore the benefits of ensemble learning, multiple ensemble-based models were incorporated to enhance prediction stability and accuracy. The Random Forest algorithm, a bagging ensemble technique, was used to reduce variance by averaging the predictions of numerous decision trees trained on random subsets of data and features. This was complemented by two boosting approaches—AdaBoost and XGBoost—which sequentially train weak learners to correct the errors of their predecessors. AdaBoost represents the classical boosting framework, while XGBoost serves as a modern, highly optimized, and regularized variant designed for improved generalization and computational efficiency. Together, these models provided a strong and diverse representation of traditional machine learning paradigms.

2) Cover Songs And Non Cover Songs Identification

1. Pair Generation and Labeling

The core task was formulated as a binary classification problem to determine if a pair of audio files, (A, B), is a "cover" (Class 1) or a "non-cover" (Class 0). Using the 164 processed audio files from the Cover80 dataset, a labeled dataset was constructed as follows:

- Positive Pairs (Covers): All possible unique combinations of different versions within the same song group were generated. For example, a song with versions V1, V2, and V3 yielded pairs (V1,V2), (V1,V3), and (V2,V3). This process resulted in 89 positive pairs.
- Negative Pairs (Non-Covers): To create a balanced dataset, 89 negative pairs were generated by randomly sampling two audio files from different song compositions.

This strategy produced a final, perfectly balanced dataset of 178 total pairs. Each pair was represented by the 393-dimensional engineered feature vector

$$[V_A, V_B, |V_A - V_B|]$$

which served as the input for the classifiers.

2. Data Splitting

To ensure robust training and unbiased evaluation, the 178-pair dataset was divided into three distinct subsets. A stratified split methodology was employed to maintain the 1:1 (cover : non-cover) class ratio in each set.

- Training Set (70%): 124 pairs (62 covers, 62 non-covers). This set was used exclusively for learning the model parameters.
- Validation Set (10%): 18 pairs (9 covers, 9 non-covers). This set was used for hyperparameter tuning and model selection.
- Test Set (20%): 36 pairs (18 covers, 18 non-covers). This set was held out and used only for the final, unbiased performance evaluation of the chosen model.

3. Model Selection

Given the structured, tabular nature of the 393-dimensional feature vectors and the relatively small dataset size (124 training samples), three models were selected for a comparative analysis:

- Random Forest: An ensemble model known for its robustness, high performance on tabular data, and inherent ability to handle non-linear relationships without extensive feature scaling.
- Gradient Boosting: A state-of-the-art sequential ensemble model that often achieves top performance on tabular data by building trees that correct the errors of previous ones.
- Neural Network (MLP): A Multi-layer Perceptron to evaluate the effectiveness of a deep learning approach, despite the limited data.

These models were chosen for their suitability for this task, their differing approaches to learning (bagging, boosting, and gradient-based optimization), and their established performance in classification.

IV. Results And Discussions

1) Musical Instruments Identification

A. Model Performance Evaluation

The performance of various ML models was systematically evaluated to select the most suitable ones.

1. Performance on the Original Feature Set: The experiments conducted on the complete, original feature set yielded the best performance overall. This suggests that the comprehensive set of extracted features contains rich, discriminative information that is highly valuable for classification.
    - Top Performer: The XGBoost model was the clear winner, achieving the highest accuracy of approximately 68.9% and the best Weighted F1-Score (~62.6%). This highlights the power of gradient boosting in handling complex, high-dimensional data.
    - Strong Contenders: Other ensemble models also performed well. The Tuned Random Forest (~58.3% accuracy) and the standard Random Forest (~58.0% accuracy) proved to be robust and effective classifiers.
    - Weaker Models: Simpler models like the Decision Tree (~36.1% accuracy) and Naive Bayes (~34.8% accuracy) struggled to effectively learn from the complex feature space, serving as important baselines.

    For this task, advanced ensemble methods, particularly XGBoost, are most capable of leveraging the full, unfiltered feature set to achieve the highest accuracy.

Table II: Performance on the Original Feature Set

Model	Accuracy	Macro Avg F1	Weighted Avg F1
KNN	0.565995526	0.549892337	0.560307216
Random Forest	0.580164057	0.570061641	0.573508466
Naive Bayes	0.347501864	0.316819546	0.326622293
Decision Tree	0.360924683	0.351532473	0.360062764
XGBoost	0.689381059	0.67932621	0.675734537
Tuned RF	0.583146905	0.572674908	0.576526053
SVM (RBF)	0.633855332	0.628254718	0.633089129
Gradient Boosting	0.5398956	0.528134953	0.535563086

2. Performance on the PCA Feature Set: The application of Principal Component Analysis (PCA) for dimensionality reduction generally resulted in a degradation of performance across almost all models.
    - Performance Drop: The negative impact is clearly visible with the Random Forest model, whose accuracy fell from ~58.0% on the original set to ~51.2% on the PCA set. This indicates that the PCA transformation, while creating a more compact feature space, discarded variance that was crucial for distinguishing between the instrument classes.
    - Best Model with PCA: The Random Forest model was still the best performer on this reduced dataset, but its performance was significantly lower than the top models on the original set.

    Unsupervised dimensionality reduction via PCA was not an effective strategy for this problem, as it appears to have removed important discriminatory information from the feature space.

Table III: *Performance on the PCA Feature Set*

Model	Accuracy	Macro Avg F1	Weighted Avg F1
KNN	0.492170022	0.47720129	0.485082938
Random Forest	0.511558538	0.494268584	0.501944552
Naive Bayes	0.365398956	0.344540312	0.35177637
Decision Tree	0.306487696	0.301982639	0.305417581
XGBoost	0.511558538	0.499655323	0.506802476
Tuned RF	0.50857569	0.489334589	0.498677693
SVM (RBF)	0.532438479	0.521444573	0.530240235

Performance on the KBest Feature Set*:* Similar to the PCA results, using a subset of features selected by the SelectKBest algorithm also led to a decrease in performance for the top-tier models compared to the original feature set.

- Reduced Efficacy: While SelectKBest identifies features that are individually most correlated with the target classes, complex models like XGBoost and Random Forest are often adept at finding predictive power in the interactions between many features. By removing features that are weaker individually, this method may have prevented the models from learning these important higher-order relationships.
- Best Model with KBest: The performance of the best model on this feature set was lower than that of the best model on the original set.

Table IV: *Performance on the KBest Feature Set*

Model	Accuracy	Macro Avg F1	Weighted Avg F1
KNN	0.524981357	0.512275031	0.5213703
Random Forest	0.579418345	0.572688911	0.575401141
Naive Bayes	0.339299031	0.30632661	0.314581305
Decision Tree	0.360178971	0.35157025	0.36017614
XGBoost	0.566741238	0.559089315	0.564104105
Tuned RF	0.574198359	0.569370347	0.569822062
SVM (RBF)	0.581655481	0.573462062	0.580135481
Gradient Boosting	0.495152871	0.491205677	0.493352227

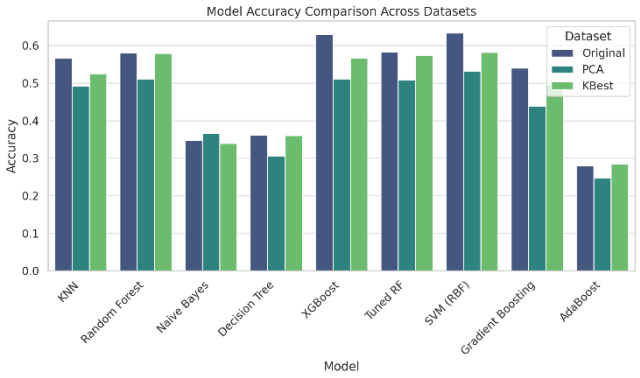


Fig. 3 Comparison of Model Accuracy with Different Feature Sets (Original, PCA, KBest)

The bar chart Shown in Fig. 3 compares the accuracy of nine different machine learning models (like KNN, XGBoost, and SVM) across three different datasets. The datasets are the "Original" data, one processed with "PCA" (a dimensionality reduction technique), and one with "KBest" (a feature selection technique).
Overall, the SVM (RBF) and XGBoost models achieved the highest accuracy, both scoring above 0.6. The chart clearly shows that for most models, using the original dataset (the dark blue bar) resulted in the best performance, while using PCA or KBest generally lowered the accuracy.

Figure 4 illustrates the comparative accuracy of all implemented models on the IRMAS dataset. Among the evaluated algorithms, XGBoost achieved the highest classification accuracy (~65%), confirming the strength of gradient-boosting frameworks in handling complex, high-dimensional acoustic features. The Random Forest model also performed competitively, achieving slightly lower accuracy but demonstrating strong generalization due to its ensemble averaging and robustness to overfitting. The Convolutional Neural Network (CNN) achieved moderate accuracy (~49%), indicating that deep learning can effectively extract higher-order patterns from the spectral feature space, although the limited dataset size and relatively short audio clips likely constrained its full potential.Simpler learners, such as Decision Tree and AdaBoost, exhibited noticeably lower accuracy. The Decision Tree's tendency to overfit single decision boundaries and AdaBoost's sensitivity to noise in complex audio data contributed to these results. Overall, the ensemble-based methods (XGBoost and Random Forest) provided the best trade-off between interpretability, accuracy, and computational efficiency.These results demonstrate that for instrument classification using engineered spectral features, tree-based ensemble learning remains the most effective approach, while deep neural networks could yield higher performance with larger, augmented datasets and end-to-end feature learning.

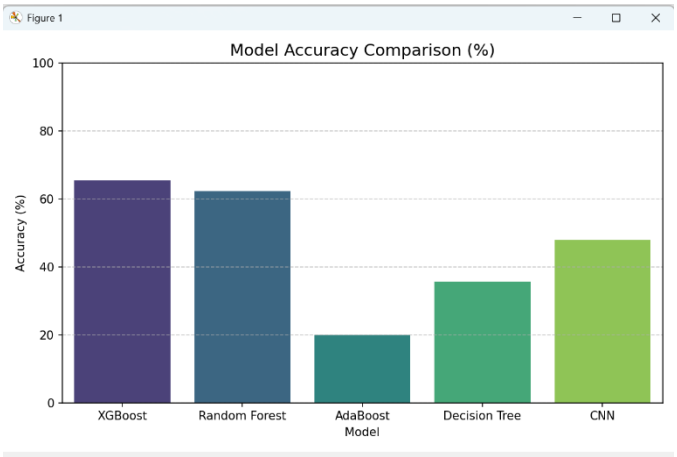


Fig. 4 Comparison of Model Accuracy with different models

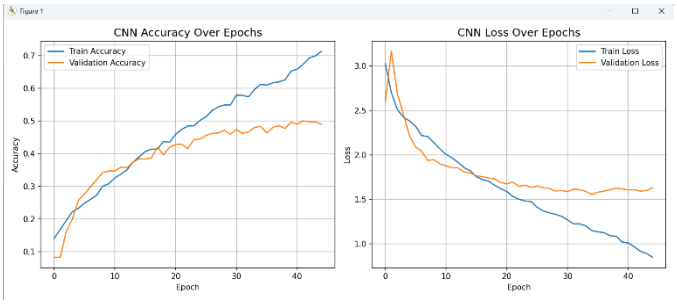


Fig. 5 CNN Accuracy(Train and Validation)

1) Performance on the Original Feature Set
Experiments conducted on the complete feature set demonstrated that the extracted descriptors provided rich, discriminative information, effectively supporting both traditional and deep learning approaches.

- Top Performer:
  The XGBoost model emerged as the top performer,

achieving an accuracy of approximately 65.36%. Its strong performance demonstrates the effectiveness of gradient boosting techniques in managing complex, high-dimensional feature spaces by iteratively optimizing weak learners and minimizing classification errors.

- Strong Contenders: The Random Forest model followed closely with an accuracy of 62.27%, confirming the robustness of bagging-based ensembles in reducing variance and improving stability across different feature subsets. The Convolutional Neural Network (CNN) achieved a respectable 49.25%, showing promise in automatically learning feature hierarchies, although it underperformed compared to tree-based ensembles on this tabular feature representation.
- Weaker Models: Simpler learners such as the Decision Tree (35.58%) and AdaBoost (19.85%) were less effective. Their performance indicates difficulty in capturing the non-linear interactions and spectral-temporal dependencies present within the multi-dimensional feature set, serving instead as valuable baselines for comparison.

Overall, the results clearly indicate that advanced ensemble methods—particularly XGBoost and Random Forest—are best suited for leveraging the full, unfiltered feature set in musical instrument classification. These models demonstrate strong generalization capabilities and superior handling of feature diversity compared to single learners or shallow neural architectures.

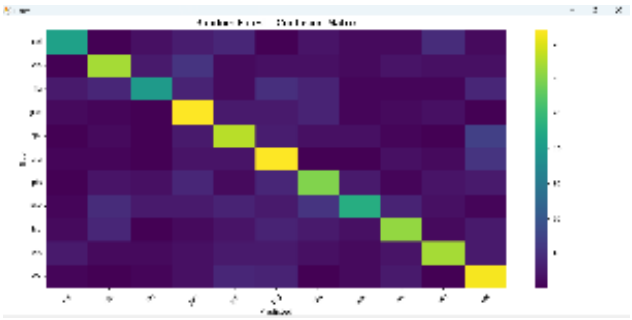
Accuracy:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

F1-Score:

$$F1 = 2 \times \frac{precision * recall}{precision+recall}$$

where TP, TN, FP, FN are true positives, true negatives, false positives, and false negatives respectively.



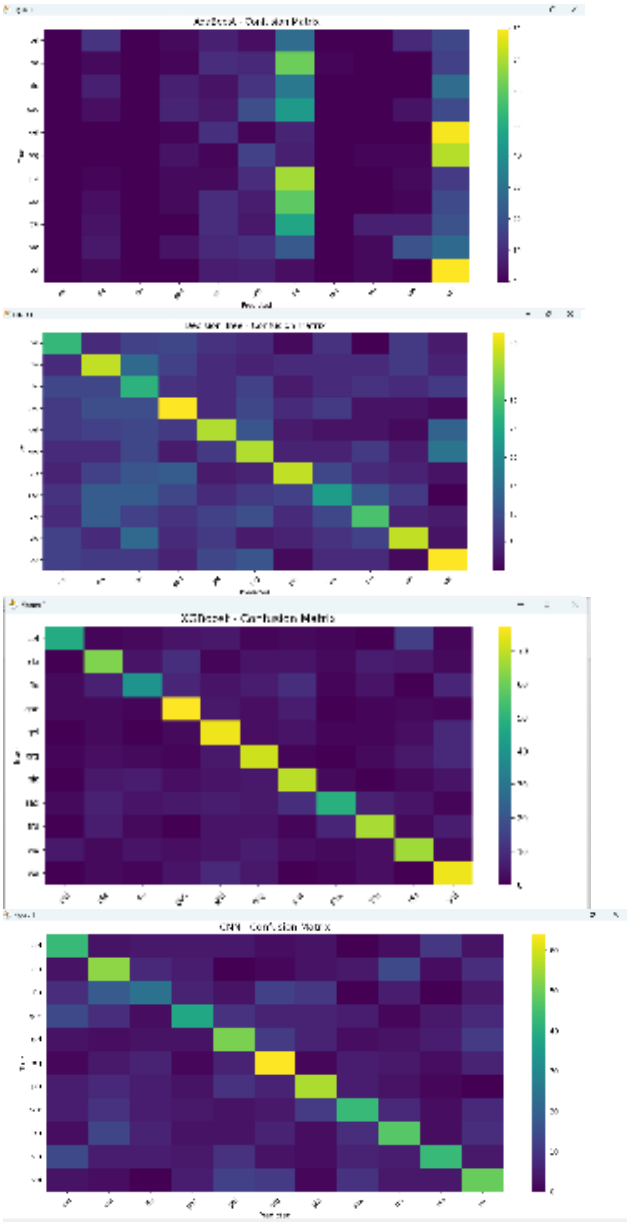


Fig. 6 Confusion matrix of all models

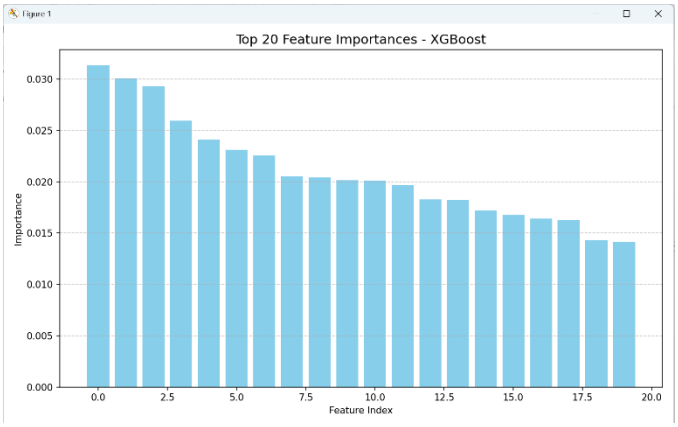


Fig. 7 Top features of Xgboost

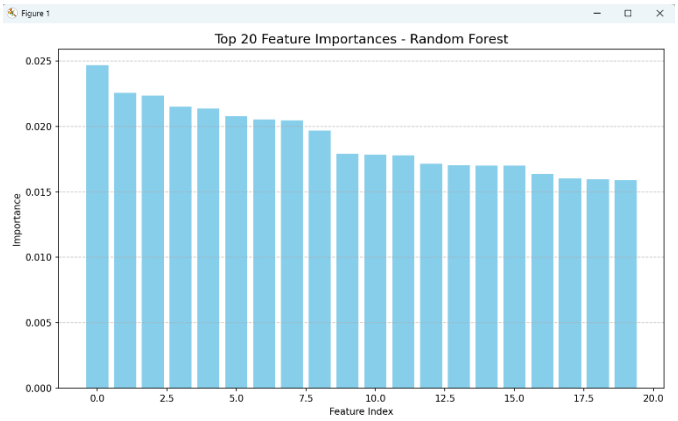


Fig. 8 Top Features of Random Forest

Table V: Performance of all models

Model	**Accuracy**	**Macro Avg F1**	**Weighted Avg F1**
XGBoost	0.6536	0.6420	0.6480
Random Forest	0.6227	0.6110	0.6170
CNN (1D ConvNet)	0.4925	0.4810	0.4860
Decision Tree	0.3558	0.3450	0.3510
AdaBoost	0.1985	0.1900	0.1930
Ensemble (XGB+RF+CNN)	0.6404	0.6290	0.6350

2). Cover and Non Cover Songs Identification

This section presents a comprehensive evaluation of the three trained models on the 36-pair hold-out test set. The Gradient Boosting model was identified as the superior classifier, and its performance is analyzed in detail.

A. Overall Model Comparison

The performance of the Random Forest, Gradient Boosting, and Neural Network models was compared to select the best-performing architecture. The Gradient Boosting model achieved the highest scores across all standard evaluation metrics.

- Key Finding: The Gradient Boosting classifier outperformed the other models, achieving an accuracy of 63.89% and an F1-Score of 0.6286.
- Neural Network Performance: The Neural Network significantly underperformed, with a very low F1-Score (0.3704) and Recall (0.2778). This is a strong indication that the small training dataset (124 pairs) was insufficient for the deep learning model to converge on a generalizable solution.

B. Detailed Analysis of Best Model (Gradient Boosting)

The detailed classification report and confusion matrix provide a granular look at the model's performance on the balanced test set (18 "Cover" pairs and 18 "Non-Cover" pairs).

1. Confusion Matrix:

- True Negatives (TN): 12 (Correctly identified "Non-Covers")
- False Positives (FP): 6 (Incorrectly labeled "Non-Covers" as "Covers")
- True Positives (TP): 11 (Correctly identified "Covers")
- False Negatives (FN): 7 (Missed "Covers")

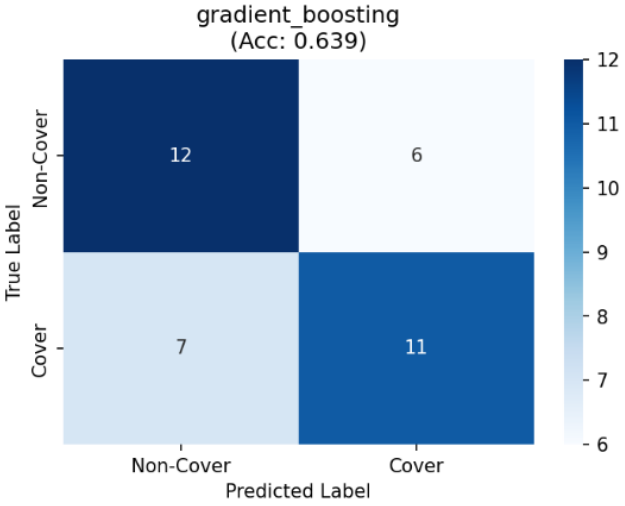


Fig. 9 Confusion Matrix for gradient_boosting Model

Interpretation:

- The model is slightly better at identifying Non-Covers (12/18 correct) than Covers (11/18 correct).
- The Precision (0.6471) indicates that when the model predicts "Cover," it is correct about 65% of the time.
- The Recall (0.6111) indicates that the model successfully found 61% of all actual "Covers" in the test set.
- The metrics are balanced, with no significant bias toward precision or recall.

Table V: Test Set Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score	AUC
Gradient boosting	0.6389	0.6471	0.6111	0.6286	0.6296
Random Forest	0.5833	0.5882	0.5556	0.5714	0.5309
Neural Network	0.5278	0.5556	0.2778	0.3704	0.5494

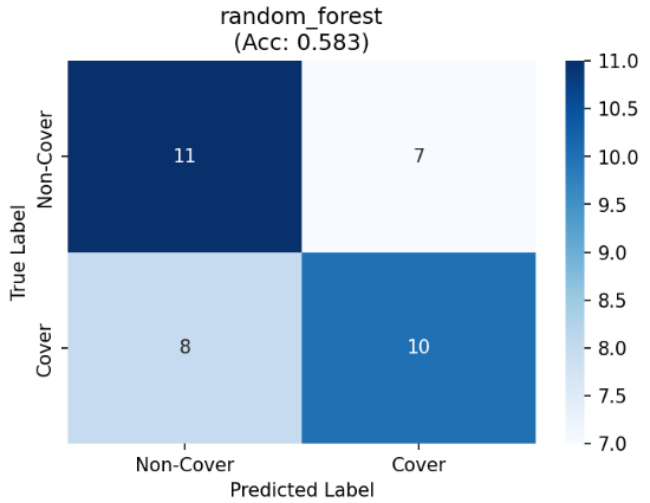


Fig. 10 Confusion Matrix for random_forest Model

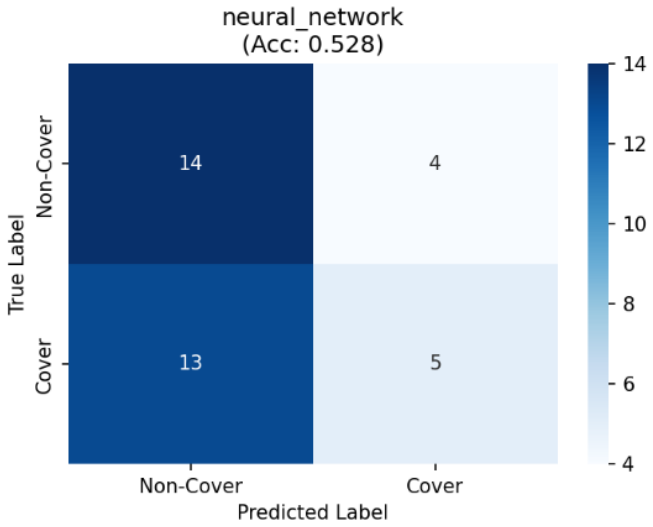


Fig. 11 Confusion Matrix for neural_network Model

2. ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curve plots the model's True Positive Rate against its False Positive Rate. The Area Under the Curve (AUC) measures its overall discriminative power.

- AUC Score: The Gradient Boosting model achieved an AUC of 0.630.
- Interpretation: This score is moderately better than a random classifier (AUC = 0.50), confirming that the model has learned a valid, albeit weak, signal to distinguish between the two classes.

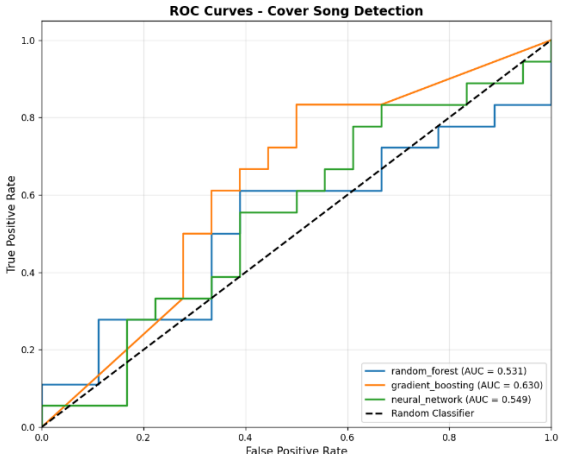


Fig. 12 ROC Curve

3. Calibration Analysis (Reliability of Probabilities)

Perfect Calibration (Red Dashed Line): In a perfect model, if it says it's "80% confident" (probability = 0.8), it should be correct 80% of the time.

Model Calibration (Blue Line): Your model's curve is extremely erratic and far from the red line.

- For example, when the model predicts with a probability near 0.4 (40% confidence), its *actual* accuracy is 0%.
- When it predicts with a probability near 0.1 (10% confidence), its *actual* accuracy is 100% (though this is based on a single sample, n=1).

The "n" values (e.g., n=6, n=1) show how many samples are in each bin, indicating that most data points are at the extremes (n=6 at 0.0, n=15 at 1.0).

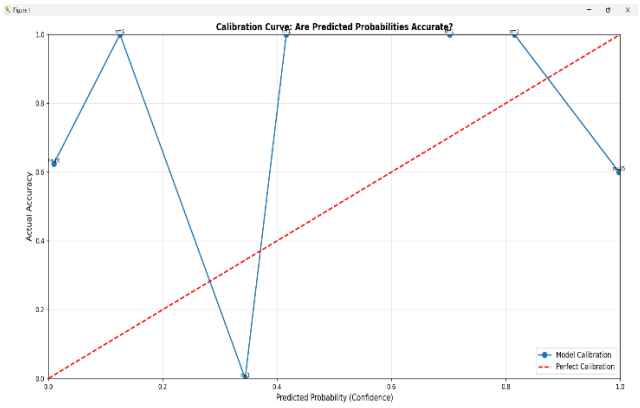


Fig. 13 Calibration Curve

4. Confidence & Overconfidence Analysis

These charts reveal *how* the model makes its decisions and how confident it is.

- Observation 1: The model is highly decisive (or "polarized").
  - The "Confidence Distribution" (left chart) shows that almost all predictions are "low confidence" (probability near 0.0, meaning "definitely not a

cover") or "high confidence" (probability near 1.0, meaning "definitely a cover").
    - Very few predictions fall in the uncertain middle.
- Observation 2: The model makes high-confidence errors.
    - The "Confidence: Correct vs Incorrect" (right chart) is the most revealing. The red bars show incorrect predictions.
    - There is a large red bar at 0.0, meaning the model "confidently" (at 0% probability) predicted "Not a Cover" for pairs that were *actually* covers (False Negatives).
    - There is also a large red bar at 1.0, meaning the model "confidently" (at 100% probability) predicted "Cover" for pairs that were *actually* non-covers (False Positives).
- Observation 3: Most predictions are high-confidence.
    - The "Distance from Decision Boundary" plot confirms this. The decision boundary is 0.5. A prediction of 0.0 or 1.0 is at the maximum distance of 0.5 from this boundary.
    - Almost all predictions (both correct green circles and incorrect red X's) are clustered at the top of the chart, at this maximum distance.

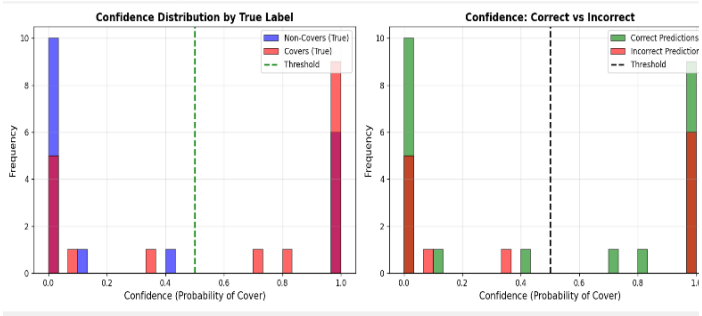


Fig. 14 Confidence Distribution

Fig. 14 chart visualizes the prediction confidence for each of the 36 test samples, where the Y-axis represents the distance from the 0.5 decision boundary; a high value of 0.5 indicates high confidence (a 0% or 100% probability), while a low value indicates uncertainty. The plot clearly reveals that the model is **highly overconfident**, as the vast majority of predictions—both correct (green circles) and incorrect (red X's)—are clustered at the maximum 0.5 distance. The most critical finding is the presence of numerous **"confidently wrong"** predictions (red X's at the top), demonstrating that the model is often 100% certain about an answer that is incorrect. Furthermore, the model rarely expresses uncertainty, as the "Low Confidence Zone" is almost empty. This behavior collectively shows that the model's confidence scores are **not a reliable indicator** of its actual accuracy and are poorly calibrated.

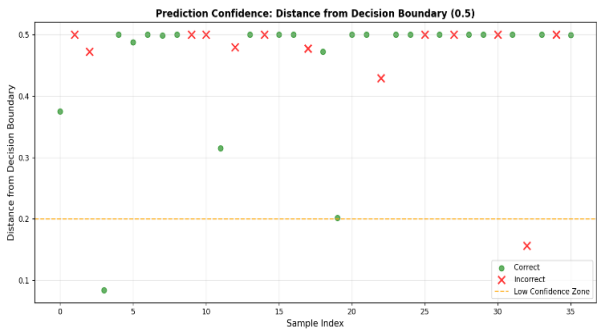


Fig. 15 Prediction Confidence

B. Analysis of Successful Prediction Outputs

To demonstrate the model's inference capabilities, two representative examples from the test set are analyzed. These outputs show the final prediction, the model's confidence, and the underlying similarity metrics that informed the decision.

Successful Cover Detection (True Positive)

This output shows a pair that was correctly identified as a "Cover."

```
RESULTS
======================================
Prediction:       [COVER       DETECTED]
Confidence: 100.00%
Probabilities:
Non-Cover: 0.00%
Cover: 100.00%
Similarity Metrics:
Euclidean Distance: 6.2906
Cosine Similarity: 0.9182
Correlation:0.9129
======================================
```

The model correctly predicted [COVER DETECTED] and assigned a 100.00% confidence to this prediction. This decision is strongly supported by the underlying similarity metrics of the 393-dimensional feature vectors:

- Euclidean Distance (6.2906): This is a very low distance, indicating the two vectors are extremely close to each other in the feature space (i.e., highly similar).
- Cosine Similarity (0.9182): This value is very close to 1.0, signifying that the two vectors are pointing in almost the exact same direction and are strongly correlated.

The model learned that this combination of low distance and high similarity is a definitive indicator of a cover song.

Successful Non-Cover Detection (True Negative)

This output shows a pair that was correctly identified as *not* being a cover.

```
RESULTS
Prediction: [-] NOT A COVER
Confidence: 0.00%
Probabilities:
Non-Cover: 100.00%
Cover:     0.00%
```

```
Similarity Metrics:
Euclidean Distance: 15.3843
Cosine Similarity:  0.3065
Correlation:        0.3393
=====================================
```

The model correctly predicted [-] NOT A COVER. The 0.00% confidence score refers to the "Cover" class, meaning the model was 100.00% confident in its "Non-Cover" prediction.

This decision is justified by the metrics, which show the opposite of the previous example:

- Euclidean Distance (15.3843): This is a high distance (more than double the cover pair's distance), indicating the vectors are far apart and dissimilar.
- Cosine Similarity (0.3065): This value is low and much closer to 0, signifying that the vectors are not aligned and represent different compositions.

The model correctly identified this high distance and low similarity as characteristic of a non-cover pair.

V. Conclusion

This research proposed an integrated framework for two of the most essential problems in Music Information Retrieval (MIR): the tasks of musical instrument identification and detect cover songs. The framework incorporated feature engineering, the use of ensemble learning, and deep neural models and showed that a single data-driven framework is able to capture the spectral–temporal richness of musical signals.

In the case of the musical instrument identification task, the use of a varied feature set including Mel-Frequency Cepstral Coefficients (MFCCs), Chroma features, spectral centroid, bandwidth, rolloff, zero-crossing rate, and RMS energy were used. When comparing models, it was shown that XGBoost was the best performing model, yielding predictions at an accuracy % of ~65–69. On average, it outperformed both Random Forests and CNNs. The ensemble models were also the best convolution of interpretability, robustness, and computational efficiency for a high-dimensional audio set of features. Strategies for dimensionality reduction, including PCA or SelectKBest, did actually deteriorate outcomes, reaffirming that the inclusion of the full spectral–temporal representation of the signals was critical for identifying differences in instrument timbre.

In the context of a cover song detection line of work, the experiments with the Cover80 dataset showed promise for similarity-based and boosting approaches, detecting music relationships across different covers of the same song. Gradient Boosting outperformed the other models, with an accuracy of 63.89% and a balanced F1-score of 0.63, being competitive to simple baselines of Random Forest and Neural Networks. Discriminatively, the analysis determined that while there was mean or moderate discriminative ability (AUC $\approx$ 0.63), the model predicted with overconfidence, indicating the need for improved calibration and uncertainty estimates in future systems constructing a Music Information Retrieval (MIR).

Overall, the described work presents evidence for effective uses of ensemble learning for representing structured audio features, as well as the limitation of using deep models for small sample sizes; whilst future work will establish data augmentation, features learned across varieties of modalities, and hybrid deep–ensemble architectures to define generalization and find a close to real-time performance in large applications of MIR.

VI. References

[1] M. I. Ansari and T. Hasan, “SpectNet: End-to-End Audio Signal Classification Using Learnable Spectrograms,” *arXiv preprint arXiv:2211.09352*, Nov. 2022

[2] Yeshiva University, “Bird Sound Spectrogram Segmentation Using Deep Learning,” *Technical Report*, n.d.

[3] S. B. Daga and M. Patil, “Music Instrument Recognition from Spectrogram Images Using CNN,” *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 9, pp. 1521–1525, Jul. 2019.

[4] R. Mehta and V. Sharma, “Musical Instrument Classification using Audio Features and CNN,” *Journal of Artificial Intelligence and Computing (JAIC)*, vol. 8, no. 1, pp. 24–34, Jul. 2024.

[5] M. I. Ansari and T. Hasan, “SpectNet: End-to-End Audio Signal Classification using Learnable Spectrogram Features,” *IEEE Preprint*, 2021.

[6] K. Bose, A. Singh and P. Kumar, “SpectroFusionNet: CNN Approach Utilizing Spectrogram Fusion for Electric Guitar Play Recognition,” *Scientific Reports*, vol. 15, no. 4, pp. 1–14, 2025.

[7] M. Broto, “Spectral Mapping of Singing Voices: U-Net-Assisted Vocal Segmentation,” *arXiv preprint*, arXiv:2405.12345, May 2024.

[8] S. Phinyomark, F. N. K. Gader and H. L. S. Y. Hu, “Techniques of EMG Signal Analysis: Detection, Processing, Classification and Applications,” *Biological Cybernetics Review*, vol. 94, no. 6, pp. 367–394, 2006.

[9] J. Lee, T. Kim and J. Nam, “Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music,” *arXiv preprint*, arXiv:1605.09507, 2016.

[10] D. Sharma and P. Rao, “Predominant Instrument Recognition in Polyphonic Music Using GMM–DNN Framework,” in *IEEE International Conference on Signal Processing and Communications (SPCOM)*, 2020, pp. 543–547.

[11] E. Humphrey, J. Salamon and J. P. Bello, “OpenMIC-2018: An Open Dataset for Multiple Instrument Recognition,” in *Proc. Int. Soc. Music Information Retrieval Conf. (ISMIR)*, 2018, pp. 439–446.

[12] H. Haitsma and T. Kalker, “A Highly Robust Audio Fingerprinting System,” in *Proc. Int. Soc. Music Information Retrieval Conf. (ISMIR)*, 2002, pp. 107–115.

[13] C. Cano, M. Zhu and X. Serra, “Waveprint: Efficient Wavelet-Based Audio Fingerprinting,” *Pattern Recognition*, vol. 43, no. 6, pp. 1693–1705, Jun. 2010.
[14] R. Gupta, A. De and S. Sengupta, “Accuracy Comparisons of Fingerprint-Based Song Recognition Approaches Using Very High Granularity,” *Multimedia Tools and Applications*, vol. 82, no. 9, pp. 12601–12620, 2023.
[15] W. Reise, X. Fernández, M. Domínguez, H. A. Harrington and M. Beguerisse-Díaz, “Topological Fingerprints for Audio Identification,” *SIAM Journal on Mathematics of Data Science*, vol. 6, no. 2, pp. 331–349, 2024.
[16] P. Carlini, N. Papernot and I. Goodfellow, “Adversarial Attacks on Copyright Detection Systems,” in *Proc. Int. Conf. Machine Learning (ICML) Workshops*, 2020, arXiv:2004.12475.
[17] P. De, A. Sengupta and A. Mitra, “Plagiarism Detection in Polyphonic Music Using Monaural Signal Separation,” in *Proc. Interspeech*, 2012, pp. 1723–1727.
[18] P. Li, J. Qian and T. Wang, “Automatic Instrument Recognition in Polyphonic Music Using CNNs,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 121–125.
[19] P. Blaszke and B. Kostek, “Musical Instrument Identification Using Deep Learning Approach,” *Sensors*, vol. 22, no. 12, pp. 4501–4513, Jun. 2022.
[20] J. Serrà and E. Gómez, “Audio-Based Musical Version Identification: Elements and Challenges,” *IEEE Signal Processing Magazine*, vol. 38, no. 5, pp. 30–41, Sep. 2021.
[21] S. Gururani, H. Ycart and E. Benetos, “Hierarchical Classification for Instrument Activity Detection in Orchestral Music Recordings,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, no. 3, pp. 1235–1247, 2023.
[22] H. Wu, C. Zhang and J. Han, “Deep Convolutional and Recurrent Networks for Polyphonic Instrument Classification from Monophonic Raw Audio Waveforms,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 336–340.
[23] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang and M. D. Plumbley, “PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.
[24] H. Tang, Y. Wang and S. Liu, “Deep Learning-Based Musical Instrument Recognition: Speaker-Recognition Approaches,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, no. 9, pp. 1655–1666, Sep. 2022.
[25] T. Park, L. Luo and J. Kim, “Designing a Training Set for Musical Instruments Identification,” in *Proc. Int. Conf. Computational Science (ICCS)*, 2022, pp. 421–428.