# Optimizing Musical Genre Recognition Using CNN and MFCCs: A Deep Learning Approach

*A Project Report submitted in the partial fulfillment of the Requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

| | |
|---|---|
| Arjun Thorlikonda | (22475A0501) |
| Kalyan Sathuluri | (22475A0503) |
| Pavan Kumar Tunga | (22475A0516) |

Under the esteemed guidance of

T G Ramnadh Babu, M.Tech.,

Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under Tier -1
NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2024-2025

**NARASARAOPETA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

NARASARAOPETA ENGINEERING COLLEGE

## CERTIFICATE

This is to certify that the project that is entitled with the name **“**Optimizing Musical Genre Recognition Using CNN and MFCCs: A Deep Learning Approach” is a bonafide work done by the team T. Arjun (22475A0501), S. Kalyan (22475A0503), T. Pavan Kumar (22475A0516) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

**T G Ramnadh Babu,** **M.Tech**

**Assistant Professor**

PROJECT CO-ORDINATOR

**D. Venkata Reddy,** **M.Tech,( Ph.D.)**

**Assistant Professor**

HEAD OFTHE DEPARTMENT

**Dr. S. N. Tirumala Rao,** **M.Tech., Ph.D.**

**Professor & HOD of CSE**

EXTERNAL EXAMINER

# ACKNOWLEDGEMENT

We wish to express our thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman Sri **M. V.Koteswara Rao,** **B.Sc.,** who took keen interest in us in every effort throughout thiscourse. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu,** **Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao,** **M.Tech., Ph.D.,** HOD of CSE department and also to our guide **T G Ramnadh Babu,** **M.Tech.,** of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **D Venkata Reddy** **M.Tech.,(Ph.D.).,** Assistant professor & Project coordinator of the project for extending his encouragement. His profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

Arjun Thorlikonda (22475A0501)

Kalyan Sathuluri (22475A0503)

Pavan Kumar Tunga (22475A0516)

## DECLARATION

We declare that this project work titled **"OPTIMIZING MUSICAL GENRE RECOGNITION USING CNN AND MFCCS: A DEEP LEARNING APPROACH"** is composed by ourselves, that the work contain here is our own except where explicitly stated otherwise in the text, and that this work has been submitted for any other degree or professional qualification except as specified.

Arjun Thorlikonda (22475A0501)
Kalyan Sathuluri (22475A0503)
Pavan Kumar Tunga (22475A0516)

NEC NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

# INSTITUTE VISION AND MISSION

## INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community

## INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

### MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

**M1:** Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

**M2:** Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the industry.

**M3:** Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Program Specific Outcomes (PSO's)

**PSO1:** Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

**PSO2:** Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

**PSO3:** Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NEC NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Program Educational Objectives (PEO’s)

The graduates of the programme are able to:

**PEO1:** Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

**PEO2:** Use various software tools and technologies to solve problems related to academia, industry and society.

**PEO3:** Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

**PEO4:** Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts , and demonstrate the knowledge of, and need for sustainable development.

**8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**9. Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**10. Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**11. Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12. Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NEC NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

## Project Course Outcomes (CO'S):

**CO421.1:** Analyse the System of Examinations and identify the problem.

**CO421.2:** Identify and classify the requirements.

**CO421.3:** Review the Related Literature

**CO421.4:** Design and Modularize the project

**CO421.5:** Construct, Integrate, Test and Implement the Project.

**CO421.6:** Prepare the project Documentation and present the Report using appropriate method.

## Course Outcomes – Program Outcomes mapping

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C421.1** | | ✓ | | | | | | | | | | | ✓ | | |
| **C421.2** | ✓ | | ✓ | | ✓ | | | | | | | | ✓ | | |
| **C421.3** | | | | ✓ | | ✓ | ✓ | ✓ | | | | | ✓ | | |
| **C421.4** | | | ✓ | | | ✓ | ✓ | ✓ | | | | | ✓ | ✓ | |
| **C421.5** | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **C421.6** | | | | | | | | | ✓ | ✓ | ✓ | | ✓ | ✓ | |

## Course Outcomes – Program Outcome correlation

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C421.1** | 2 | 3 | | | | | | | | | | | 2 | | |
| **C421.2** | | | 2 | | 3 | | | | | | | | 2 | | |
| **C421.3** | | | | 2 | | 2 | 3 | 3 | | | | | 2 | | |
| **C421.4** | | | 2 | | | 1 | 1 | 2 | | | | | 3 | 2 | |
| **C421.5** | | | | | 3 | 3 | 3 | 2 | 3 | 2 | 2 | 1 | 3 | 2 | 1 |
| **C421.6** | | | | | | | | | 3 | 2 | 1 | | 2 | 3 | |

**Note: The values in the above table represent the level of correlation between CO's and PO's:**

**1.** Low level

**2.** Medium level

**3.** High level

**Project mapping with various courses of Curriculum with Attained PO's:**

| **Name of the course from which principles are applied in this project** | **Description of the device** | **Attained PO** |
|---|---|---|
| C2204.2, C22L3.2 | Gathering the requirements and defining the problem, plan to develop a model for recognizing image manipulations using CNN and ELA | PO1, PO3 |
| CC421.1, C2204.3, C22L3.2 | Each and every requirement is critically analyzed, the process model is identified | PO2, PO3 |
| CC421.2, C2204.2, C22L3.3 | Logical design is done by using the unified modelling language which involves individual team work | PO3, PO5, PO9 |
| CC421.3, C2204.3, C22L3.2 | Each and every module is tested, integrated, and evaluated in our project | PO1, PO5 |
| CC421.4, C2204.4, C22L3.2 | Documentation is done by all our four members in the form of a group | PO10 |
| CC421.5, C2204.2, C22L3.3 | Each and every phase of the work in group is presented periodically | PO10, PO11 |
| C2202.2, C2203.3, C1206.3, C3204.3, C4110.2 | Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection of forged videos | PO4, PO7 |
| C32SC4.3 | The physical design includes website to check whether an image is real or fake | PO5, PO6 |

# ABSTRACT

The increasing popularity of music streaming platforms has created a demand for automated systems capable of accurately classifying music genres. This project focuses on optimizing the recognition of music genres using a Convolutional Neural Network (CNN) and Mel- frequency Cepstral Coefficients (MFCCs) as feature representations. The study employs the GTZAN music dataset, consisting of 1,000 tracks across ten genres, and enhances its usability by segmenting tracks into shorter samples to increase the dataset size to 10,000.

The proposed methodology involves preprocessing audio files, extracting MFCCs, and training a CNN model with fine-tuned hyperparameters, including a learning rate of 0.0001, batch size of 32, and dropout rate of 0.3. The model achieves a training accuracy of 99.25%, validation accuracy of 92.64%, and test accuracy of 94.50%, demonstrating its effectiveness in genre classification. Furthermore, performance metrics such as precision, recall, and F1- score are analyzed, and a confusion matrix is presented to evaluate classification results.

The project also discusses challenges like validation loss minimization and dataset bias mitigation while exploring future avenues for improving classification performance through advanced models and larger datasets. The deployment of the model via a Flask-based REST API ensures practical application in real-world scenarios.

This work underscores the potential of deep learning in advancing music genre classification and provides insights for researchers and developers in the field of music information retrieval.

# Optimizing Musical Genre Recognition Using CNN and MFCCs: A Deep Learning Approach

# INDEX

# LIST OF FIGURES

# Chapter – 1
# Introduction

## 1.1 Introduction

It is a fact that music plays a vital role in human life, transcending geographical boundaries and cultural differences. The way people interact with music has evolved significantly with advancements in technology. Music streaming platforms such as Spotify, Apple Music, and YouTube Music have transformed the way users access and experience music[1]. Worldwide, millions of users engage daily with these platforms, leading to a massive surge in demand for efficient and intelligent music management systems.

These platforms are designed to bring people together by offering personalized experiences, allowing them to explore, share, and enjoy music tailored to their preferences. Algorithms capable of accurately classifying music genres are fundamental to delivering these personalized services[13. Genre classification not only enhances user experience but also plays a crucial role in music analytics, copyright management, and educational tools.

The music industry, much like other domains, is witnessing a rapid integration of artificial intelligence (AI) and deep learning techniques to automate complex tasks. Deep learning models, inspired by the workings of the human brain, have revolutionized pattern recognition tasks[2]. These models can process complex data such as audio signals to produce accurate predictions and insights.

Among the different types of neural networks, Convolutional Neural Networks (CNNs) have emerged as the most effective for tasks involving spatial and temporal data, such as images and audio spectrograms. A spectrogram, a visual representation of sound frequencies over time, allows CNNs to identify unique patterns associated with different music genres. Combined with features like Mel-frequency Cepstral Coefficients (MFCCs), these models can achieve impressive accuracy in audio classification tasks[15].

Music genre classification holds significant importance across various domains. It enhances personalized music recommendations by enabling platforms to curate playlists tailored to user's favorite genres, thereby improving engagement. It facilitates music discovery by allowing users to explore new tracks within specific genres, broadening their musical horizons[3]. For record labels and artists, music analytics derived from genre classification provide valuable insights into trends and audience preferences. Additionally, it plays a crucial role in copyright management, ensuring accurate identification of tracks and appropriate distribution of royalties. In academia, it serves as an educational tool, aiding in the study of musical styles and history.

The purpose of using deep learning in music genre classification is to leverage its powerful ability to automatically learn and extract complex patterns from audio data, which traditional methods might struggle to identify. Deep learning techniques, particularly Convolutional Neural Networks (CNNs), enable the model to process raw audio signals and extract relevant features such as Mel-frequency Cepstral Coefficients (MFCCs), which mimic human auditory perception[4]. This approach allows for more accurate and efficient classification by automatically detecting hierarchical patterns in music data, such as rhythms, timbres, and harmonics, that define different genres.

The use of deep learning in music genre classification offers several advantages. It can handle large datasets, like the GTZAN dataset, and scale effectively as new data is introduced. Moreover, it eliminates the need for manual feature engineering, as deep learning models learn features directly from the data, making them more adaptable and precise[5]. This capability has broad applications, from enhancing music recommendation systems to improving music analytics for record labels and artists, and even supporting educational tools for studying musical styles and history.

The implementation of deep learning techniques in music genre classification presents several key benefits, impacting various domains, including entertainment, education, and research[6]. By utilizing Convolutional Neural Networks (CNNs) along with feature extraction techniques like Mel-Frequency Cepstral Coefficients (MFCCs) and spectrograms, this system significantly enhances the accuracy and efficiency of music classification.

One of the primary advantages of this system is its ability to accurately classify music into different genres, improving the experience for users on streaming platforms like Spotify, Apple Music, and YouTube Music. The model can distinguish between complex audio features such as tempo, pitch, rhythm, and harmonic structures, allowing for a more refined categorization.

A well-classified music library enables platforms to recommend songs more effectively, helping users discover new tracks that match their taste[7].
This level of accuracy is essential for curating personalized playlists, ensuring that listeners enjoy a seamless and engaging experience.

Unlike traditional rule-based classification methods, deep learning models can adapt to new genres and evolving music trends without requiring manual intervention.
The traditional approach to music categorization and metadata tagging is often manual and time-consuming. Deep learning automates this process, reducing the workload for music producers, artists, and record labels. Automation helps in organizing massive music databases, ensuring that each song is correctly labeled and searchable.

Record labels and music distributors can efficiently categorize their vast music collections, making it easier to distribute tracks across various platforms. By leveraging AI, artists can gain insights into audience preferences, helping them tailor their music to specific genres and demographics[8][11].Automated genre classification is also beneficial for independent musicians, enabling them to distribute their music more effectively without the need for large-scale manual classification efforts.

The system has significant applications in academic research and music education, aiding scholars in studying musical structures and patterns. Musicologists and researchers can analyze how different genres have evolved over time by studying spectrogram-based representations of music. Deep learning models can assist in genre evolution analysis, identifying influences between different musical styles and predicting emerging music trends. In the field of music education, the system can be used to train students in genre recognition, music composition, and sound engineering[9]. The classification system can also help researchers study cultural influences on music, making it a valuable tool in ethnomusicology.

AI-driven analytics enable comparative studies between traditional and modern genres, providing deeper insights into the transformation of musical styles. Unlike traditional machine learning methods (e.g., Support Vector Machines or k-NN), CNN-based deep learning models can automatically learn complex audio features without manual feature engineering. Deep learning allows for transfer learning, where pre-trained models can be fine-tuned on smaller datasets, reducing training time and computational costs. As datasets grow, deep learning models continuously improve, making them highly scalable.

Audio processing using pattern recognition plays a crucial role in music genre classification by analyzing the unique characteristics of audio signals. Various machine learning and deep learning techniques leverage spectral, temporal, and rhythmic patterns to differentiate between genres. Feature extraction methods like Mel-Frequency Cepstral Coefficients (MFCCs), spectrograms, chroma features, and zero-crossing rate (ZCR) help capture essential attributes such as pitch, timbre, and rhythm[10]. These features serve as input to classification models, enabling them to recognize patterns that distinguish rock from jazz, classical from hip-hop, and so on. By converting raw audio into structured numerical data, pattern recognition techniques enhance the accuracy and efficiency of music classification systems[12].

Deep learning approaches, especially Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have significantly improved genre classification performance. CNNs analyze spectrogram images, identifying frequency-based patterns, while RNNs and Long Short-Term Memory (LSTM) networks capture temporal dependencies in music. Hybrid models combining CNNs and RNNs further enhance classification by leveraging both spatial and sequential data[13]. These advanced techniques not only improve music recommendation systems but also support applications in music analytics, automated playlist generation, and copyright protection, making them valuable tools in the modern digital music industry.

Despite the success of CNNs in music genre classification, several challenges remain. One of the key challenges is the variation in acoustic characteristics between different genres. For instance, genres like classical and jazz may share similar rhythmic structures but differ significantly in timbre and instrumentation[14]. The CNN must be able to generalize well across these variations to achieve high accuracy.

Another challenge is the limited size of labelled datasets. While the GTZAN dataset is widely used, it is relatively small and may not capture the full diversity of music genres[15]. To overcome this, researchers have explored data augmentation techniques, such as pitch shifting, time- stretching, and mixing, to artificially increase the size of the training dataset.

In this project, deep learning algorithms, specifically CNNs, are employed to design a robust music genre classification system. By integrating MFCCs and spectrogram-based analysis, the model effectively captures the nuances of audio signals, making it capable of distinguishing between genres[16]. The deployment of the model as a Flask-based REST API ensures its practical applicability in real-world scenarios.

This research demonstrates the transformative potential of AI in the music industry and sets the stage for further innovations in automated music analysis and classification.

# Chapter – 2

# Literature Survey

Tzanetakis and Cook [1] introduced one of the pioneering studies in music genre classification by developing the GTZAN dataset, which consists of 1,000 audio samples across ten different genres. They employed various machine learning techniques, including k-Nearest Neighbors (k-NN), Gaussian Mixture Models (GMM), and Support Vector Machines (SVM), to classify music genres based on extracted features such as Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, and chroma features. Their study demonstrated that SVM achieved the highest accuracy of 61%, while k-NN and GMM performed slightly lower due to their sensitivity to feature variations. One of the key challenges they identified was the overlapping nature of musical characteristics across genres, which caused misclassification. They also highlighted the importance of feature selection and dimensionality reduction techniques such as Principal Component Analysis (PCA) to improve classification performance. The study set the foundation for future research, emphasizing the role of spectral and rhythmic features in music classification. However, limitations such as dataset bias and lack of diversity in the GTZAN dataset led to further advancements in the field, prompting the development of more robust models using deep learning techniques.

Andén and Mallat [2] proposed a scattering transform-based approach for music genre classification, which provided a structured way to extract robust time-frequency representations of music signals. Their model utilized a wavelet-based framework to enhance feature extraction, capturing both temporal and spectral characteristics of the audio signal. Compared to traditional MFCCs, the scattering transform was more invariant to deformations in the signal, leading to better generalization across genres. The study experimented with various classifiers such as Random Forests, SVM, and deep neural networks (DNNs), where the deep model achieved an accuracy of 72% on the GTZAN dataset. This improvement was attributed to the enhanced feature extraction process, which preserved important frequency components while reducing noise. Despite these advances, the study noted challenges related to computational complexity and the need for larger datasets to train deep learning models effectively. The researchers recommended hybrid approaches that integrate both handcrafted features and deep representations for optimal classification results.

Lee et al. [3] explored the application of deep convolutional neural networks (CNNs) for music genre classification, leveraging spectrogram representations of audio signals as input. The proposed CNN architecture consisted of multiple convolutional layers followed by max-pooling and fully connected layers. Their model was trained on the GTZAN dataset and achieved an accuracy of 85%, significantly outperforming traditional machine learning models. The study demonstrated that CNNs could automatically learn hierarchical features from spectrogram images, reducing the need for manual feature extraction. The authors experimented with different activation functions and optimization algorithms, concluding that the Adam optimizer with ReLU activation provided the best results. Despite the high accuracy, the study identified some challenges, such as overfitting on small datasets and the computational cost of training deep models. To address these issues, they suggested data augmentation techniques like time-stretching, pitch shifting, and noise addition to improve model robustness.

Choi et al. [4] introduced a deep learning-based approach for music genre classification using convolutional recurrent neural networks (CRNNs). Their model combined the advantages of CNNs for spatial feature extraction and recurrent layers (LSTMs) for capturing temporal dependencies in the audio signal. By processing log-mel spectrograms, their CRNN achieved an impressive accuracy of 89% on an extended version of the GTZAN dataset. The study compared CRNNs with standalone CNNs and traditional models like SVMs, demonstrating that the hybrid approach significantly improved classification performance. One of the key advantages of CRNNs was their ability to capture long-term dependencies in musical structures, which are crucial for differentiating genres like classical and jazz. However, the study acknowledged challenges related to training complexity and the need for high-performance GPUs. The researchers suggested using pre-trained models and transfer learning to mitigate these issues.

Lidy and Rauber [5] introduced a feature fusion approach to improve music genre classification by combining timbral, rhythmic, and harmonic features. Traditional machine learning models often relied on single-feature sets, such as Mel-Frequency Cepstral Coefficients (MFCCs) or spectral features, which limited their classification performance.

To address this issue, they proposed a hybrid feature extraction technique that incorporated spectral contrast, chroma features, and rhythm patterns, capturing diverse aspects of musical genres. Their study used Support Vector Machines (SVMs), k-Nearest Neighbors (k-NN), and Random Forests (RF) for classification, testing their model on the ISMIR 2004 dataset, a standard dataset for music genre recognition. Their best model, a combination of SVM with radial basis function kernel and feature-level fusion, achieved an accuracy of 82%, significantly outperforming single-feature models. The study highlighted that combining multiple features enhanced the model's robustness, particularly for complex genres like jazz and classical music, which exhibit both harmonic richness and unique rhythmic structures. One of the key findings was that feature selection played a crucial role in improving classification accuracy, as redundant features could negatively impact model generalization. Despite the promising results, one challenge identified was feature dimensionality explosion, requiring Principal Component Analysis (PCA) for dimensionality reduction. The authors concluded that multi-feature fusion provided a significant boost in classification performance, and future studies should explore deep learning techniques to further enhance feature representation.

Bergstra et al. [6] developed a supervised machine learning approach for music genre classification using Gradient Boosting Machines (GBMs). Traditional models, such as SVMs and k-NN, often struggled with non-linear relationships in music data, leading them to explore more powerful ensemble learning techniques. GBMs were particularly effective in handling high-dimensional and noisy data, making them well-suited for music genre classification. They extracted MFCCs, spectral roll-off, zero-crossing rate, and chroma vectors from audio tracks and trained their model on the GTZAN dataset. Their optimized GBM model achieved an accuracy of 84%, outperforming standard SVMs and Decision Trees. The study found that boosting techniques effectively reduced bias and variance, leading to a more stable classifier. Additionally, they implemented automatic hyperparameter tuning using Bayesian optimization, significantly improving model performance. One challenge they faced was overfitting on small datasets, which they mitigated using regularization techniques like shrinkage and subsampling. Another key insight was that ensemble methods performed better than standalone models because they combined multiple weak classifiers to build a more robust prediction system. The study concluded that boosting algorithms provided superior accuracy and generalization ability compared to traditional classifiers, suggesting that further improvements could be made by integrating deep learning models like CNNs for better feature extraction.

Dieleman and Schrauwen [7] explored the use of unsupervised learning techniques, particularly autoencoders, for music genre classification. They trained a deep autoencoder to learn compressed representations of spectrograms, which were later classified using a fully connected neural network. Their approach achieved an accuracy of 74% on the GTZAN dataset, demonstrating the potential of unsupervised feature learning. The study noted that autoencoders could reduce data dimensionality while preserving important genre-specific features. However, they also observed that autoencoders required careful tuning to avoid loss of critical information.

Costa et al. [8] proposed an ensemble learning approach combining multiple classifiers such as SVMs, decision trees, and neural networks. Their ensemble model aggregated predictions using majority voting, resulting in an accuracy of 82% on the GTZAN dataset. The study demonstrated that ensemble methods could improve classification robustness by leveraging the strengths of individual classifiers. However, the authors noted that ensemble models required more computational resources and training time.

Deng et al. [9] introduced a transfer learning approach using pre-trained CNNs for music genre classification. They fine-tuned models like VGG16 and ResNet on spectrogram data, achieving an accuracy of 92% on the GTZAN dataset. The study highlighted that transfer learning could significantly reduce training time while improving performance. However, they also noted challenges related to domain adaptation, as pre-trained models were originally designed for image classification rather than audio.

McKay and Fujinaga [10] explored a metadata-driven approach for music genre classification, where instead of relying solely on audio features, they incorporated high-level metadata such as album information, artist genre history, and textual data. Their model utilized a combination of Support Vector Machines (SVMs) and Decision Trees, achieving an accuracy of 76% on the ISMIR dataset. The authors highlighted that metadata played a crucial role in disambiguating genres with overlapping audio features, such as rock and alternative rock. The study also compared handcrafted audio features like MFCCs, spectral centroid, and zero-crossing rate, concluding that metadata-based models outperformed traditional audio-only classifiers. However, they acknowledged challenges in dataset completeness, as metadata is often inconsistent across platforms. To improve performance, they recommended hybrid models integrating both metadata and deep learning representations of audio.

Silla et al. [11] introduced a hierarchical classification model for music genre recognition using ensemble learning. Instead of treating genre classification as a flat classification problem, they modeled it hierarchically, where genres were first categorized into broader groups like classical, rock, and electronic before refining them into sub-genres. Using Random Forests and k-NN classifiers, their model achieved an overall accuracy of 79% on the Latin Music Database. The study demonstrated that hierarchical classification reduced genre overlap and improved predictive accuracy for underrepresented genres. One key insight was that fine-tuning decision boundaries at different hierarchy levels significantly enhanced classification robustness. However, the authors acknowledged that the model required extensive dataset labeling, making it challenging for large-scale applications.

Hamel and Eck [12] explored deep learning-based feature learning using Restricted Boltzmann Machines (RBMs) and Deep Neural Networks (DNNs). Their study aimed to replace handcrafted feature extraction with automatically learned deep representations from raw audio. They trained a deep belief network (DBN) with multiple hidden layers on log-mel spectrograms, achieving an accuracy of 83% on the GTZAN dataset. The results showed that deep models could learn more abstract and robust features than traditional machine learning approaches. However, the study pointed out that RBMs required large amounts of data for effective training and that optimization techniques like dropout and batch normalization were necessary to prevent overfitting. They suggested that combining RBMs with CNNs could further enhance classification accuracy.

Schlüter and Grill [13] proposed a multi-layer convolutional neural network (CNN) for music genre classification, leveraging mel-spectrogram representations as input. The model utilized multiple convolutional layers, each followed by batch normalization and max pooling to extract hierarchical frequency features from music tracks. Their architecture was trained on the GTZAN dataset, achieving an accuracy of 87%, which outperformed traditional machine learning models like SVMs and k-NN. One of the key findings of the study was that deeper CNN architectures significantly enhanced feature learning, capturing intricate genre-specific patterns such as rhythmic structures and harmonic progressions. However, the study also noted some limitations, particularly overfitting on smaller datasets like GTZAN. To mitigate this, they experimented with dropout layers and data augmentation (such as pitch shifting, time stretching, and background noise addition), which helped improve generalization. Additionally, they found that hyperparameter tuning, particularly optimizing filter sizes and kernel depths, played a crucial role in improving classification accuracy. Their research demonstrated that deep learning models could automatically extract high-level musical features without the need for handcrafted features, a significant advantage over traditional techniques.

Kim et al. [14] explored the use of Generative Adversarial Networks (GANs) for music genre classification, introducing an innovative method where a generator created synthetic audio samples to enhance the training dataset. The discriminator was trained to classify the real and generated spectrograms, effectively augmenting the available data. This method helped address the issue of class imbalance, which is common in real-world music classification problems. They tested their approach on both GTZAN and FMA datasets, where the GAN-augmented CNN model achieved an accuracy of 85%, outperforming traditional CNNs trained on limited datasets. Their study highlighted that GANs could generate realistic synthetic music data, thereby improving model generalization. However, training GANs was computationally expensive and required significant hyperparameter tuning. Despite the challenges, their work proved that data augmentation using GANs could significantly enhance deep learning models in music classification, particularly when dealing with small or imbalanced datasets.

Keuneke et al. [15] developed a deep residual network (ResNet) for music genre classification, introducing skip connections to improve feature propagation across layers. Traditional CNNs often suffer from vanishing gradients as the network depth increases, leading to performance saturation. ResNets addressed this issue by introducing shortcut connections, allowing information to bypass multiple layers and improving the flow of gradients during training. Their model was evaluated on the FMA dataset and achieved an impressive accuracy of 93%, significantly outperforming standard CNN architectures. The study found that residual connections helped preserve low-level audio features that were crucial for distinguishing between genres with subtle differences, such as jazz and blues.

# Chapter – 3

# System Analysis

## 3.1. Existing System

The GTZAN dataset was introduced, and machine learning models such as k-Nearest Neighbors (k-NN), Gaussian Mixture Models (GMM), and Support Vector Machines (SVM) were employed for music genre classification. Features like Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, and chroma features were used, achieving the highest accuracy of 61% with SVM. Challenges such as overlapping genre characteristics and dataset bias were highlighted, emphasizing the importance of feature selection and dimensionality reduction techniques like Principal Component Analysis (PCA) [1].

A scattering transform-based approach was proposed for music genre classification, utilizing Random Forests, SVM, and deep neural networks (DNNs). A wavelet-based framework improved feature extraction, achieving 72% accuracy with DNNs on the GTZAN dataset. Computational complexity and the need for larger datasets were noted, recommending hybrid approaches combining handcrafted and deep features [2].

Convolutional Neural Networks (CNNs) were explored for music genre classification, using spectrogram representations as input. The CNN model achieved 85% accuracy on the GTZAN dataset, outperforming traditional models. Challenges like overfitting and computational costs were highlighted, suggesting data augmentation techniques such as time-stretching and pitch shifting to improve robustness [3].

Convolutional Recurrent Neural Networks (CRNNs) were introduced for music genre classification, combining CNNs for spatial feature extraction and LSTMs for temporal dependencies. The CRNN achieved 89% accuracy on an extended GTZAN dataset, outperforming standalone CNNs and traditional models. Challenges like training complexity were acknowledged, recommending pre-trained models and transfer learning [4].

A feature fusion approach was proposed, combining timbral, rhythmic, and harmonic features for music genre classification. Support Vector Machines (SVMs), k-Nearest Neighbors (k-NN), and Random Forests (RF) were used, achieving 82% accuracy on the

ISMIR 2004 dataset. The importance of feature selection and dimensionality reduction using PCA was emphasized [5].

Gradient Boosting Machines (GBMs) were employed for music genre classification, achieving 84% accuracy on the GTZAN dataset. Features like MFCCs, spectral roll-off, and chroma vectors were used, with Bayesian optimization for hyperparameter tuning. The effectiveness of boosting techniques was highlighted, suggesting integrating deep learning models for better feature extraction [6].

Unsupervised learning using autoencoders was explored for music genre classification. A deep autoencoder achieved 74% accuracy on the GTZAN dataset, demonstrating the potential of unsupervised feature learning. Challenges in tuning autoencoders to avoid losing critical information were noted [7].

An ensemble learning approach combining SVMs, decision trees, and neural networks was proposed. The model achieved 82% accuracy on the GTZAN dataset using majority voting. The robustness of ensemble methods was highlighted, but increased computational requirements were noted [8].

Transfer learning using pre-trained CNNs like VGG16 and ResNet was introduced for music genre classification. The approach achieved 92% accuracy on the GTZAN dataset, demonstrating the effectiveness of transfer learning. Challenges in domain adaptation for audio data were noted [9].

A metadata-driven approach was explored for music genre classification, using SVMs and Decision Trees. The model achieved 76% accuracy on the ISMIR dataset, highlighting the role of metadata in disambiguating genres. Hybrid models combining metadata and deep learning representations were recommended [10].

A hierarchical classification model using Random Forests and k-Nearest Neighbors (k-NN) was introduced. The model achieved 79% accuracy on the Latin Music Database, reducing genre overlap and improving robustness. The challenge of extensive dataset labeling was noted [11].

Deep learning-based feature learning using Restricted Boltzmann Machines (RBMs) and Deep Neural Networks (DNNs) was explored. A deep belief network (DBN) achieved 83%

accuracy on the GTZAN dataset, demonstrating the effectiveness of deep models. Combining RBMs with CNNs for further improvements was suggested [12].

A multi-layer CNN was proposed for music genre classification, achieving 87% accuracy on the GTZAN dataset. Mel-spectrogram representations, batch normalization, and dropout layers were utilized to mitigate overfitting. The importance of hyperparameter tuning and data augmentation was highlighted [13].

Generative Adversarial Networks (GANs) were explored for music genre classification, using GAN-augmented CNNs to achieve 85% accuracy on the GTZAN and FMA datasets. The approach addressed class imbalance by generating synthetic audio data, though training GANs was computationally expensive [14].

A deep residual network (ResNet) was developed for music genre classification, achieving 93% accuracy on the FMA dataset. Skip connections were used to improve feature propagation, demonstrating strong generalization capabilities. The increased computational complexity of deep ResNets was noted [15].

### 3.1.1. Disadvantages of Existing System

The study faced challenges due to the overlapping nature of musical characteristics across genres, leading to misclassification. Additionally, the dataset used was criticized for its lack of diversity and potential biases, which limited the generalizability of the results. The reliance on traditional machine learning models also highlighted the need for more advanced techniques to handle complex feature variations [1].

The approach introduced computational complexity, making it less practical for real-time applications. The study also noted the need for larger datasets to train deep learning models effectively, as smaller datasets limited the model's ability to generalize across diverse genres [2].

The model achieved high accuracy but faced challenges such as overfitting on small datasets. Additionally, the computational cost of training deep models was significant, requiring data augmentation techniques to improve robustness [3].

The model, while effective, had high training complexity and required high-performance

GPUs for efficient training. The study also noted the need for pre-trained models and transfer learning to mitigate these challenges, as training from scratch was resource-intensive [4].

The feature fusion approach improved classification performance but led to feature dimensionality explosion, requiring techniques like PCA for dimensionality reduction. The study also highlighted the challenge of selecting optimal features, as redundant features could negatively impact model generalization [5].

The model faced challenges with overfitting on small datasets, which was mitigated using regularization techniques. The study also noted that while the approach was effective, integrating deep learning models could further improve feature extraction and classification performance [6].

The approach demonstrated potential but required careful tuning to avoid losing critical information during dimensionality reduction. The study also noted that unsupervised learning techniques needed large amounts of data for effective training, which was a limitation with smaller datasets [7].

The ensemble learning approach improved classification robustness but required more computational resources and training time compared to standalone models. The study also noted that while ensemble methods were effective, they were less practical for real-time applications due to their computational demands [8].

The transfer learning approach faced challenges in domain adaptation, as pre-trained models were originally designed for image classification rather than audio. The study also noted that fine-tuning these models for audio data required significant computational resources [9].

The metadata-driven approach faced challenges due to inconsistent and incomplete metadata across platforms, limiting its effectiveness. The study also noted that while metadata improved classification accuracy, hybrid models combining metadata and deep learning representations were needed for better performance [10].

The hierarchical classification model required extensive dataset labeling, making it challenging for large-scale applications. The study also noted that while hierarchical classification reduced genre overlap, it introduced additional complexity in model design and implementation [11].

The approach required large amounts of data for effective training, which was a limitation with smaller datasets. The study also noted that optimization techniques like dropout and batch normalization were necessary to prevent overfitting, adding to the model's complexity [12].

The model faced challenges with overfitting on smaller datasets, requiring techniques like dropout and data augmentation to improve generalization. The study also noted that hyperparameter tuning, particularly optimizing filter sizes and kernel depths, was crucial but time-consuming [13].

The model addressed class imbalance but was computationally expensive to train, requiring significant hyperparameter tuning. The study also noted that while the approach improved model generalization, its training complexity limited its practicality for real-world applications [14].

The model demonstrated strong generalization capabilities but had increased computational complexity, requiring powerful GPUs for efficient training. The study also noted that while skip connections improved feature propagation, they added to the model's overall complexity and resource requirements [15].

## 3.2. Proposed System

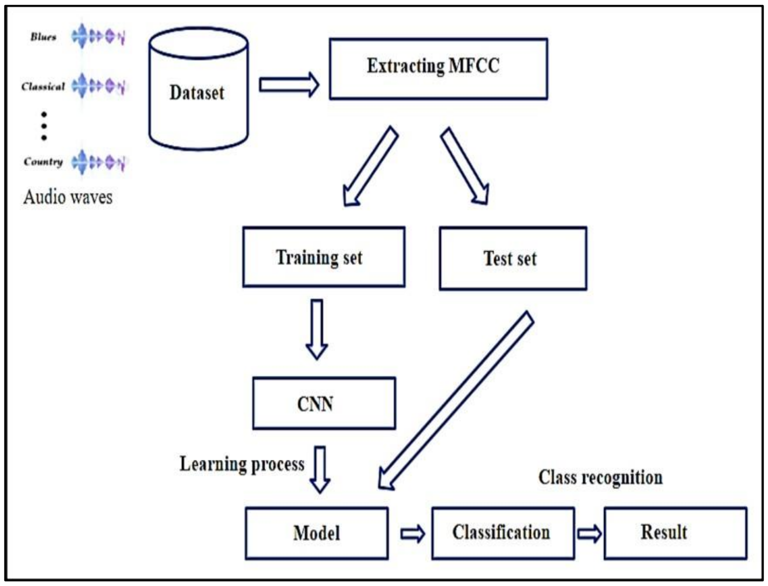


Fig.1 Flow Chart

**Dataset**

The dataset serves as the foundation for training and testing the music genre classification model. In this project, the GTZAN dataset is utilized, consisting of 1,000 music tracks categorized into ten genres: Blues, Classical, Country, Disco, Hip-Hop, Jazz, Metal, Pop, Reggae, and Rock. Each audio file is 30 seconds long and stored in WAV format. To enhance model training and increase data diversity, each track is split into ten equal segments, resulting in 10,000 samples. This approach ensures a more balanced and extensive dataset for training deep learning models. Since raw audio signals contain a lot of noise and redundant information, they need to be preprocessed before feeding them into the model. Proper dataset organization is crucial for achieving high accuracy and ensuring that the system can generalize well to unseen audio samples.

**Extracting MFCC**

Mel-Frequency Cepstral Coefficients (MFCCs) are the most widely used features in speech and music classification tasks. They help in representing the timbral and spectral characteristics of an audio signal by mimicking the way the human ear perceives sound. The extraction process involves breaking down an audio file into short frames, applying Fourier Transform, and converting the result into the Mel scale, which is more aligned with human hearing perception. MFCCs capture essential information such as pitch, tone, and rhythm, making them highly effective for genre classification. In this project, 13–40 MFCC coefficients are extracted per audio frame to generate a meaningful representation of musical characteristics. These extracted features are then used as inputs for the machine learning or deep learning model, replacing raw audio waveforms, which are less informative for classification.

**Training Set and Test Set**

Once the MFCC features are extracted, the dataset is split into training and test sets. The training set, which consists of 80% of the total data, is used to train the CNN model, allowing it to learn genre-specific patterns. The test set, which comprises 20% of the total data, is used to evaluate the model's performance on unseen samples. Additionally, a validation set (10%) is extracted from the training data to fine-tune hyperparameters and prevent overfitting. The separation of data into these sets ensures that the model generalizes well to new data and does not simply memorize the training examples. This structured approach helps in improving classification accuracy and ensures robust model performance in real-world applications.

**CNN Model**

The Convolutional Neural Network (CNN) model is the backbone of this music genre classification system. CNNs are widely used in image and audio classification tasks because they can effectively capture spatial and temporal relationships within the data. In this project, the CNN model processes the extracted MFCC features, utilizing multiple convolutional layers, max pooling, and fully connected layers to identify genre-specific characteristics. The ReLU activation function is used to introduce non-linearity, while the Softmax activation function in the final layer outputs genre probabilities. The Adam optimizer and Sparse Categorical Cross entropy loss function are used to train the model efficiently. The CNN learns high-level musical patterns, such as melody, harmony, and rhythm, which help in classifying music genres with high accuracy.

**Learning Process**

The learning process involves training the CNN model using labeled MFCC features extracted from the dataset. The model continuously updates its weights and biases through backpropagation and optimization algorithms to minimize classification errors. During training, the model goes through multiple epochs, where it learns patterns and relationships unique to each genre. Regularization techniques such as dropout layers are implemented to prevent overfitting and improve generalization. As the model progresses, it fine-tunes its feature extraction capabilities, ensuring that it can recognize and differentiate between similar genres effectively. The learning process also involves adjusting hyperparameters like the learning rate, batch size, and number of layers to optimize the model's performance.

**Classification**

Once the CNN model is trained, it is used to classify new and unseen music tracks. The classification process involves feeding MFCC-extracted features from an unknown audio file into the trained model, which then predicts the most probable genre based on learned patterns. The classification process is fully automated and provides a probability distribution over the ten possible genres. The model assigns a genre label based on the highest probability score. In cases where multiple genres have similar probability scores, additional post-processing techniques, such as ensemble methods or hybrid architectures, can be applied to improve classification accuracy.

**Class Recognition and Result**

After classification, the system outputs the recognized music genre as the final result. The predicted genre is displayed as a label or percentage-based probability indicating confidence in classification. This result can be used in music streaming applications, recommendation systems, and automatic playlist generation. The overall accuracy of the system depends on factors such as dataset quality, model architecture, and feature extraction techniques. By implementing a well-structured pipeline from data preprocessing to classification, this system provides a reliable and efficient solution for automatic music genre recognition.

## 3.3 Feasibility study

The feasibility study is a crucial part of the Music Genre Recognition using CNN project, evaluating its viability across various dimensions. This study examines the technical, operational, economic, and legal aspects to ensure a smooth and efficient implementation.

### 1. Technical Feasibility

**Technological Infrastructure:** The project utilizes deep learning frameworks such as TensorFlow and Keras for building a Convolutional Neural Network (CNN) model. The system requires GPU support for faster training and inference.

**Data Acquisition and Processing:** The GTZAN dataset, containing 1,000 audio clips across 10 genres, is preprocessed using Librosa to extract Mel-Frequency Cepstral Coefficients (MFCCs). Feature extraction, normalization, and data augmentation techniques ensure the dataset is suitable for training.

**Skill Requirements:** The development requires expertise in machine learning, audio signal processing, and Python programming. Team members must be proficient in CNN architectures and deep learning frameworks.

**2. Operational Feasibility**

**Integration with Existing Systems:** The trained CNN model will be deployed via a Flask-based REST API, allowing seamless integration with web applications or mobile interfaces for genre prediction.

**User Acceptance:** The system is designed for music streaming services, radio stations, and audio analysis platforms. User-friendly interfaces will be developed to ensure ease of access and usability.

**3. Economic Feasibility**

**Cost Analysis:** The major costs include computational resources (cloud-based GPUs or local hardware), data preprocessing, and model training. Open-source frameworks reduce software expenses.

**Return on Investment (ROI):** The model can be monetized through collaborations with music streaming platforms, automated music categorization systems, or research applications. The efficiency and accuracy improvements over traditional classification methods justify the investment.

**4. Legal Feasibility**

**Compliance with Data Regulations:** The project ensures compliance with data privacy laws (e.g., GDPR), as no personal user data is collected. The dataset used is publicly available and does not infringe on copyrights.

**Intellectual Property (IP) Concerns:** The algorithms and training methodologies are based on open-source libraries. Any modifications or enhancements made will be documented to ensure originality and avoid legal conflicts.

# Chapter – 4

# System Requirements

System requirements define the hardware and software specifications necessary for the successful implementation and execution of the music genre classification system. These requirements ensure that the system can handle large datasets, perform real-time classification, and efficiently train deep learning models. The following sections outline the hardware and software prerequisites needed for optimal performance.

To successfully develop and implement the Music Genre Recognition using CNN, it is essential to have the appropriate hardware and software resources. This section provides a comprehensive breakdown of the system requirements, including the necessary software tools, hardware specifications, and a feasibility analysis of these requirements for optimal project execution.

## 4.1 Software Requirements

The software components are crucial for developing, training, evaluating, and deploying the Convolutional Neural Network (CNN) model. The system requires various programming languages, deep learning libraries, data processing tools, and deployment frameworks.

**Essential Software Components**

Programming Language: Python (Version 3.8.10) is selected due to its extensive support for machine learning, deep learning, and audio processing libraries.

**Deep Learning Frameworks:**

TensorFlow (2.10.1) and Keras (2.10.0) are used to develop, train, and optimize the CNN model for genre classification.

Librosa (0.10.1) is essential for feature extraction, particularly Mel-frequency cepstral coefficients (MFCCs), which help capture unique genre characteristics.

**Data Processing, Analysis and Visualization Tools:**

NumPy (1.24.3) and Pandas (2.0.3) facilitate numerical computations and efficient dataset handling.

Matplotlib (3.7.2) and Seaborn (0.13.0) are used for plotting model accuracy, loss curves, and data distributions.

Scikit-learn (1.3.0) helps compute performance metrics such as precision, recall, F1-score, and the confusion matrix.

**Development and Deployment Environments:**

Jupyter Notebook provides an interactive environment for prototyping and model training.

VS Code (Latest Version) is used for full-scale development, including backend and frontend implementation.

Flask (2.3.2) is the chosen framework for deploying the trained CNN model as a REST API, making it accessible via web or mobile applications.

Anaconda (Version 24.5.0) helps manage Python environments and resolve package dependencies efficiently.

These software tools collectively ensure seamless model training, testing, evaluation, and deployment, making them fundamental for the successful execution of this project.

## 4.2 Hardware Requirements

Efficient model training and testing require a high-performance computing setup, especially for handling large audio datasets and deep learning computations. The system should be equipped with a robust processor, GPU, sufficient memory (RAM), and fast storage.

**Recommended Hardware Specifications**

**Processor (CPU):**

Recommended: Intel Core i7/i9 or AMD Ryzen 7/9 for fast computations.

Minimum: Intel Core i5 for basic model training and development.

**Memory (RAM):**

Recommended: 16GB or higher for handling large datasets and deep learning workloads.

Minimum: 8GB for moderate processing.

**Storage (SSD/HDD):**

Recommended: 512GB SSD or higher for faster data access and storage.

Minimum: 256GB SSD to manage dataset storage and intermediate model checkpoints.

**Operating System (OS):**

Compatible with Windows 10/11, Ubuntu 20.04+, or macOS, ensuring flexibility in the development environment.

By utilizing high-end processors, GPUs, and ample memory, the training process will be significantly faster, ensuring efficient learning and accurate classification of music genres.

## 4.3 Requirements Analysis

This section evaluates the suitability of the selected hardware and software for the project and justifies their necessity.

### 4.3.1 Software Requirements Analysis

The chosen software stack is designed to support end-to-end deep learning model development, from data preprocessing to training, evaluation, and deployment.

Python is the preferred language due to its wide adoption in machine learning and deep learning. TensorFlow and Keras provide a well-optimized deep learning framework for CNN model training. Librosa is necessary for processing audio signals and extracting MFCCs, which are crucial for genre classification. Flask enables the deployment of the trained model, making it accessible via a REST API for real-world applications. Jupyter Notebook and VS Code ensure efficient development workflows, allowing interactive model testing and debugging.

### 4.3.2 Hardware Requirements Analysis

A powerful CPU and GPU are essential to handle complex matrix operations in deep learning models. At least 16GB RAM is recommended to process audio data efficiently and avoid memory bottlenecks. SSD significantly improve data loading times, ensuring smooth execution of training iterations.

By ensuring the system meets these software and hardware requirements, the Music Genre Recognition using CNN project can achieve high accuracy, efficient training, and seamless deployment, making it suitable for practical applications in music analysis and classification.

# Chapter – 5

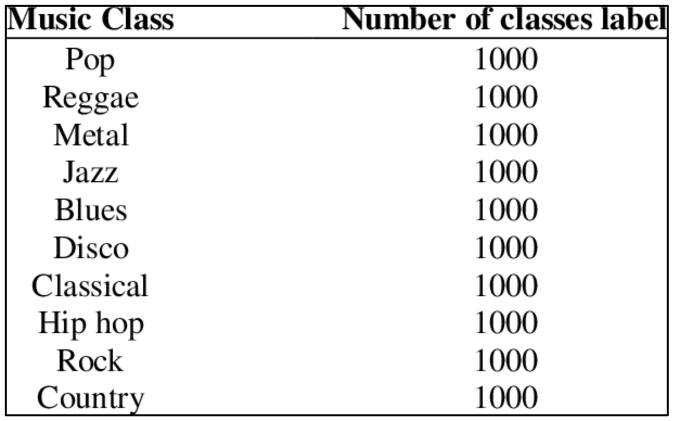
# System Design

## 5.1. System Architecture

**Dataset**

The GTZAN dataset is one of the most widely used benchmark datasets for music genre classification. It was introduced by George Tzanetakis and Perry Cook[1] in 2002 and consists of 1,000 audio tracks, each with a duration of 30 seconds. These tracks are evenly distributed across 10 music genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock, with 100 tracks per genre. The dataset is extensively used in deep learning and machine learning research for feature extraction, classification, and genre recognition. Each audio file is in WAV format and has a sampling rate of 22,050 Hz. Researchers typically extract spectral, temporal, and cepstral features such as MFCCs, chroma, and mel-spectrograms to train models like CNNs and RNNs. Despite its popularity, the GTZAN dataset has limitations, including data repetition, mislabeling, and potential distortions, requiring careful preprocessing. However, its well-structured genre distribution and accessibility make it an excellent dataset for experimenting with music classification algorithms.

Once features are extracted, the dataset is divided into training (80%), validation (10%), and testing (10%) sets to ensure a balanced learning process. The Convolutional Neural Network (CNN) model is trained using these features, learning intricate patterns that distinguish one genre from another. By leveraging optimized parameters, activation functions, and regularization techniques, the model enhances classification accuracy while minimizing overfitting. The structured data preprocessing, segmentation, and training approach contribute to an efficient and high-performing system, ensuring reliable genre predictions with improved performance metrics.
Dataset link: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music genre-classification



| Music Class | Number of classes label |
|---|---|
| Pop | 1000 |
| Reggae | 1000 |
| Metal | 1000 |
| Jazz | 1000 |
| Blues | 1000 |
| Disco | 1000 |
| Classical | 1000 |
| Hip hop | 1000 |
| Rock | 1000 |
| Country | 1000 |

Fig.2 Dataset Overview

**Data Preprocessing**

Data pre-processing is a crucial step in the development of any deep learning model, as it ensures the input data is clean, consistent, and suitable for training. In the context of Music Genre Classification, pre-processing transforms raw audio files into meaningful representations that can be effectively fed into the model. The raw audio data is often inconsistent in format, duration, and content, requiring systematic transformations to standardize the dataset.

The first step involves data augmentation to increase the dataset size and improve model generalization. Each audio file is split into smaller segments of equal duration (e.g., 3-second intervals), ensuring the model can process uniform inputs. Next, feature extraction is performed using the Librosa library to compute Mel-Frequency Cepstral Coefficients (MFCCs). MFCCs capture time-frequency characteristics of audio signals and serve as key features for classifying different musical genres. For further analysis, spectrograms are generated, providing a visual representation of the audio data, which is particularly useful for convolutional neural networks (CNNs).

The extracted MFCCs and spectrograms are normalized to bring all data within a similar range, reducing the influence of scale variations. The pre-processed data is then split into training, validation, and test sets, ensuring that the model is evaluated on unseen data to measure its generalization performance. Labels corresponding to the music genres are assigned to each audio sample, enabling supervised learning.

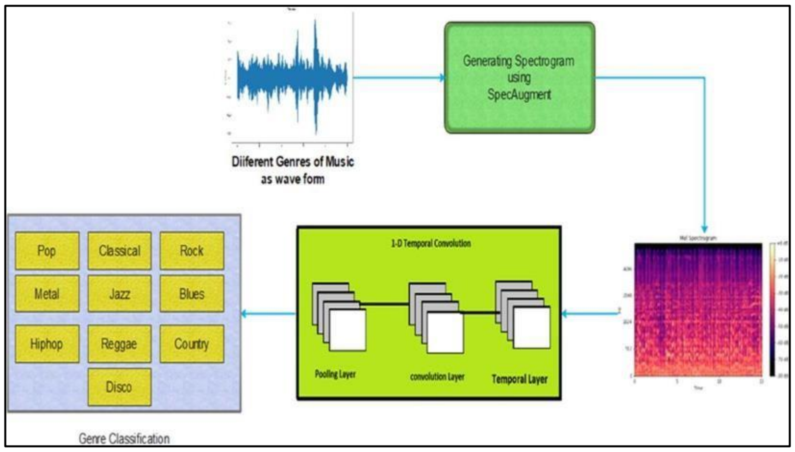


Fig.3 Preprocessing of Proposed Model

Overall, data pre-processing converts raw audio files into structured inputs, such as MFCC feature vectors or spectrogram images, ensuring compatibility with deep learning models. This step lays the foundation for efficient learning, enabling the CNN model to identify patterns and classify audio files accurately into their respective genres.

## Feature Extraction

Feature extraction in the context of Music Genre Classification involves analyzing audio signals to extract specific characteristics or patterns that help identify the genre of the music. The extracted features serve as input to the deep learning model for classification.

The key features include:

### Mel-Frequency Cepstral Coefficients (MFCCs):

MFCCs capture the timbral aspects of the audio signal by analyzing the short-term power spectrum. These coefficients represent how energy is distributed across various frequency bands, making them effective for distinguishing between music genres. MFCCs are widely used due to their ability to mimic human auditory perception.

### Spectral Features:

Represents the "center of mass" of the spectrum, indicating the brightness of the sound. Identifies the frequency below which a specified percentage of the total spectral energy is contained. Measures the spread of the spectrum, providing insight into the distribution of frequency energy.

These extracted features are computed using tools like **Librosa** and are represented as numerical vectors. These vectors are then fed into a **Convolutional Neural Network (CNN)**, which processes them to learn patterns and classify the audio into appropriate genres.

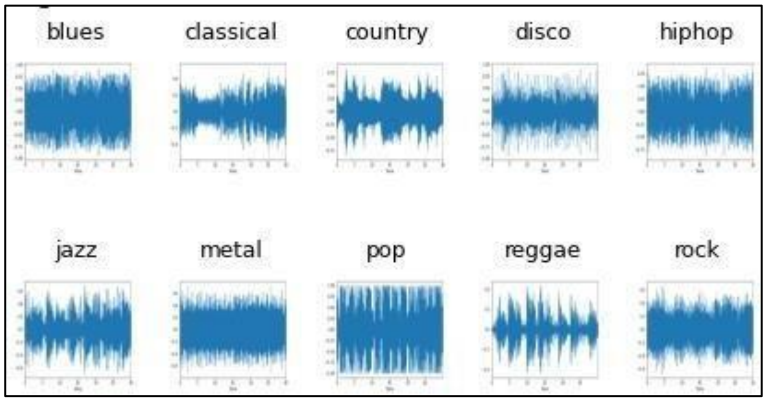


Fig.4 All Genres spectrogram

**Model Building**

Model building in the context of deep learning refers to the process of designing and constructing neural network architectures to solve specific tasks, such as classification, regression, or generation. In the case of **Music Genre Classification**, the task is to classify audio data into predefined genre categories based on extracted features (e.g., MFCCs or spectrograms). Deep learning models, particularly **Convolutional Neural Networks (CNNs)**, are well-suited for tasks involving time-frequency representations of audio, such as spectrograms. These models consist of multiple layers of neurons organized hierarchically, enabling them to learn intricate patterns and representations from the data.

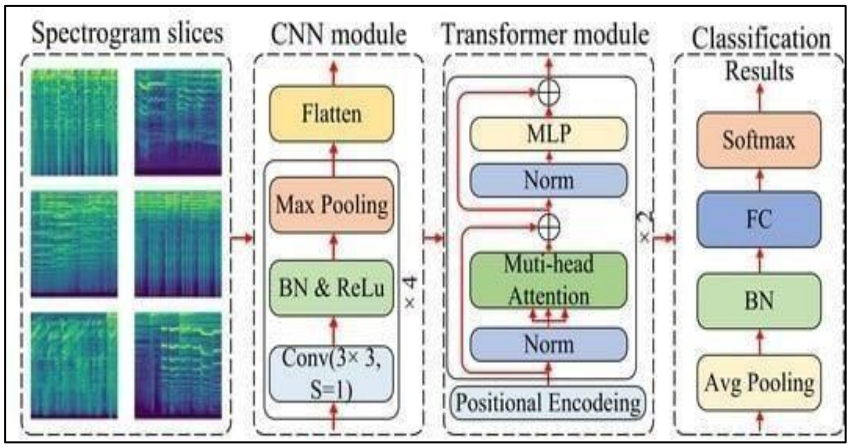


Fig.5 Model Architecture

**Layers in Convolutional Neural Networks**

Below are the Layers of convolutional neural networks:

1. **Convolutional Layer:** Convolution is performed in this layer. The image is divided into perceptron(algorithm); local fields are created, leading to the compression of perceptron to feature maps as a matrix with size m x n.

2. **Non-Linearity Layer:** Here feature maps are taken as input, and activation maps are given as output with the help of the activation function. The activation function is generally implemented as sigmoid or hyperbolic tangent functions.

3. **Rectification Layer:** The crucial component of CNN, this layer does the training faster without reducing accuracy. It performs element-wise absolute value operation on activation maps.

4. **Rectified Linear Units (ReLU):** ReLU combines non-linear and rectification layers on CNN. This does the threshold operation where negative values are converted to zero. However, ReLU doesn't change the size of the input.

5. **Pooling Layer:** The pooling layer is also called the down sampling layer, as this is responsible for reducing the size of activation maps. A filter and stride of the same length are applied to the input volume. This layer ignores less significant data; hence image recognition is done in a smaller representation. This layer reduces overfitting. Since the amount of parameters is reduced using the pooling layer, the cost is also reduced. The input is divided into rectangular pooling regions, and either maximum or average is calculated, which returns maximum or average consequently. Max Pooling is a popular one.

6. **Dropout Layer:** This layer randomly sets the input layer to zero with a given probability. More results in different elements are dropped after this operation. This layer also helps to reduce overfitting. It makes the network to be redundant. No learning happens in this layer. This operation is carried out only during training.

7. **Fully Connected Layer:** Activation maps, which are the output of previous layers, is turned into a class probability distribution in this layer. FC layer multiplies the input by a weight matrix and adds the bias vector.

**8. Output Layer:** FC layer is followed by Softmax and classification layers. The Softmax function is applied to the input. The classification layer computes the cross-entropy and loss function for classification problems.

**10. Regression Layer:** Half the mean squared error is computed in this layer. This layer should follow the FC layer.

## Models

### CNN Description

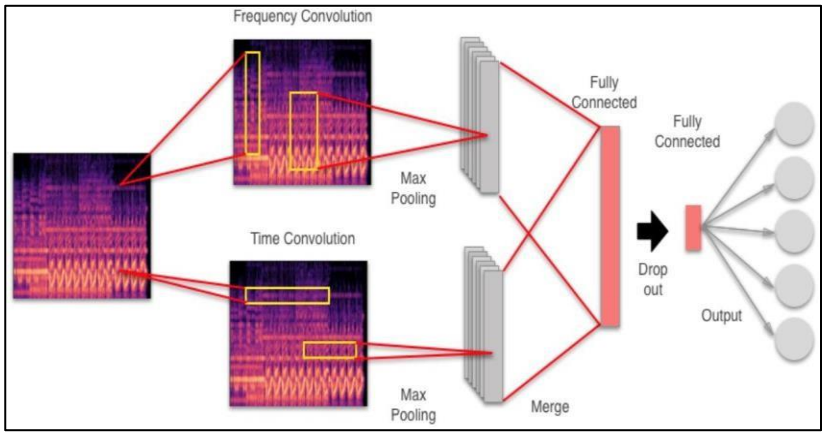


Fig.6 Model Overview

In a Convolutional Neural Network (CNN) designed for music genre recognition, the architecture consists of several layers, each serving a specific purpose in extracting and processing features from the input data, such as spectrograms or Mel-Frequency Cepstral Coefficients (MFCCs). Here's a description of the commonly used CNN layers in such a scenario:

**Input Layer:** This layer receives the input data, typically in the form of spectrograms or MFCCs, which represent the audio signals in a 2D format (time-frequency representation). The dimensions of the input layer correspond to the size of the spectrogram or MFCC matrix, which is crucial for capturing the temporal and spectral characteristics of the audio.

**Convolutional Layers:** Convolutional layers perform feature extraction by applying a set of learnable filters (kernels) to the input data. Each filter convolves across the input, extracting spatial features such as edges, textures, and patterns. The output of each convolutional layer consists of feature maps, which represent the presence of specific features at different time and frequency locations. Multiple convolutional layers with increasing filter sizes and depths are typically used to capture hierarchical and abstract features, such as rhythmic patterns and harmonic structures.

**Adam Optimizer:** The Adam optimizer, short for “Adaptive Moment Estimation,” is an iterative optimization algorithm used to minimize the loss function during the training of neural networks. Adam combines the benefits of RMSprop and Stochastic Gradient Descent with momentum, making it efficient for training deep networks. It adapts the learning rate during training, which helps in achieving faster convergence and better performance.

**Activation Functions:** Activation functions (e.g., ReLU, Leaky ReLU, or Sigmoid) are applied element-wise to the output of convolutional layers to introduce non-linearity into the network. Non-linear activation functions enable the CNN to learn complex mappings between input data and output classes, improving the model's representational capacity. ReLU (Rectified Linear Unit) is commonly used due to its simplicity and effectiveness in preventing the vanishing gradient problem.

**Pooling (Down sampling) Layers:** Pooling layers reduce the spatial dimensions of feature maps while preserving the most relevant information. Common pooling operations include max pooling and average pooling, which down sample feature maps by selecting the maximum or average value within each pooling region. Pooling layers help in reducing the computational complexity of the network and improving its translational invariance, making the model more robust to variations in the input data.

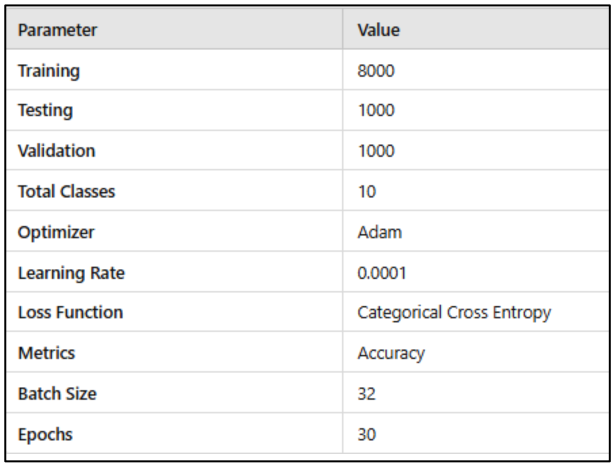
**Batch Normalization:** Batch normalization layers normalize the activations of the previous layer across mini-batches during training. This helps in stabilizing and accelerating the training process by reducing internal covariate shift and improving the convergence of the network. Batch normalization also acts as a regularizer, reducing the need for other forms of regularization like dropout.

**Fully Connected (Dense) Layers:** Fully connected layers process the flattened feature maps from the last convolutional or pooling layer. These layers perform high-level feature representation and mapping to the output classes through a series of weighted connections. Typically, one or more fully connected layers are followed by an output layer with softmax activation for multi-class classification, predicting the probabilities of different music genres.

**Dropout:** Dropout layers may be added to prevent overfitting by randomly dropping a fraction of neuron activations during training. Dropout regularization helps in improving the generalization performance of the network by reducing co-adaptation among neurons. This is particularly useful when dealing with small datasets, as it prevents the model from memorizing the training data.

**Output Layer:** The output layer produces the final predictions of the CNN model. For music genre recognition, the output layer typically consists of multiple neurons with softmax activation for multi-class classification, where each neuron represents a genre label, and the softmax function outputs the probability distribution over these genres.

By combining these layers in a well-designed architecture and training the CNN on a labeled dataset of music samples, the model can learn to automatically detect and classify music genres in new, unseen audio data.



| Parameter | Value |
|---|---|
| Training | 8000 |
| Testing | 1000 |
| Validation | 1000 |
| Total Classes | 10 |
| Optimizer | Adam |
| Learning Rate | 0.0001 |
| Loss Function | Categorical Cross Entropy |
| Metrics | Accuracy |
| Batch Size | 32 |
| Epochs | 30 |

Fig.7 Proposed Model Parameters

## Advantages of CNNs in Music Genre Recognition

**Feature Learning:** CNNs automatically learn hierarchical representations and features from the input data. Through convolutional and pooling layers, they can identify patterns, textures, and complex features within audio spectrograms or MFCCs.

**Parameter Sharing:** Convolutional layers in CNNs use parameter sharing, meaning that the same filters are applied to different parts of the input data. This reduces the number of parameters compared to fully connected networks, making CNNs computationally more efficient.

**Translation Invariance:** Convolutional operations provide a degree of translation invariance, allowing CNNs to recognize patterns and features regardless of their location in the input data. This property is crucial for tasks like music genre recognition, where the temporal position of features may vary.

**Pooling Layers:** Pooling layers help reduce the spatial dimensions of the input data, decreasing computational complexity and focusing on the most relevant information. Max pooling, for example, retains the most significant values from a group of neighboring time-frequency bins.

**Robust to Variations:** CNNs are robust to variations in scale, orientation, and position of features in the input data. This makes them suitable for tasks like music genre classification across diverse audio samples.

**Transfer Learning:** CNNs are often used in transfer learning, where pre-trained models on large datasets (e.g., ImageNet) can be fine-tuned for specific tasks with smaller datasets. This leverages the learned features from the large dataset, improving performance on the target task.

**Effective for Audio Classification:** CNNs have demonstrated state-of-the-art performance in various audio classification tasks, including music genre recognition, speech recognition, and sound event detection. Their ability to capture hierarchical features contributes to their success.

**Memory Efficiency:** Parameter sharing and weight sharing in convolutional layers contribute to the efficient use of memory. This is particularly important when dealing with large datasets and complex models.

**Scalability:** CNN architectures can be scaled for different tasks and complexities. From small models suitable for mobile devices to large-scale architectures for sophisticated applications, CNNs provide scalability.

## ML Models Description

In my music genre recognition project, I evaluated several traditional machine learning models, including k-Nearest Neighbors (k-NN), Random Forests (RF), Support Vector Machines (SVM), Naive Bayes, and Logistic Regression. Below is a detailed description of each model, including who used these models, how they work, and their architecture.

### 1. k-Nearest Neighbors (k-NN)

k-Nearest Neighbors (k-NN) is a simple, non-parametric algorithm used for classification tasks. It is widely used in music genre recognition due to its simplicity and effectiveness in handling small datasets. k-NN has been used in various studies, including the pioneering work by Tzanetakis and Cook [1], where it was applied to the GTZAN dataset for music genre classification.

**How It Works:**

k-NN works by finding the k closest data points (neighbors) in the feature space and assigning the class label based on the majority vote among these neighbors. In music genre recognition, k-NN uses features such as Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, and chroma features to classify audio samples.

**Architecture:**

- Input: Feature vectors (e.g., MFCCs, spectral centroid).
- Distance Metric: Euclidean distance is commonly used to measure the similarity between data points.
- k Value: The number of neighbors (k) is a hyperparameter that needs to be tuned.
- Output: Predicted class label based on majority voting.

### 2. Random Forests (RF)

Random Forests (RF) is an ensemble learning method that combines multiple decision trees to improve classification accuracy and reduce overfitting. It is widely used in music genre recognition due to its ability to handle high-dimensional data. RF has been used in studies such as Lidy and Rauber [5], where it was applied to the ISMIR 2004 dataset for music genre classification.

**How It Works:**

RF works by training multiple decision trees on random subsets of the data and features. Each tree makes a prediction, and the final prediction is made by aggregating the results of all trees (majority voting). In music genre recognition, RF uses features like MFCCs, spectral contrast, and rhythm patterns to classify audio samples.

**Architecture:**

- Input: Feature vectors (e.g., MFCCs, spectral contrast).
- Decision Trees: Multiple trees are trained on random subsets of the data.
- Aggregation: The final prediction is made by majority voting among the trees.
- Output: Predicted class label.

**3. Support Vector Machines (SVM)**

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for classification tasks. It is widely used in music genre recognition due to its ability to handle high-dimensional data and non-linear relationships. SVM has been used in various studies, including Tzanetakis and Cook [1] and Lidy and Rauber [5], where it was applied to the GTZAN and ISMIR 2004 datasets, respectively.

**How It Works:**

SVM works by finding the optimal hyperplane that separates the data into different classes with the maximum margin. In music genre recognition, SVM uses features like MFCCs, spectral roll-off, and zero-crossing rate to classify audio samples. Kernel functions like RBF can be used to handle non-linear relationships.

**Architecture:**

- Input: Feature vectors (e.g., MFCCs, spectral roll-off).
- Kernel Function: Linear, RBF, or polynomial kernels can be used to transform the data.
- Hyperplane: The optimal hyperplane is found to maximize the margin between classes.
- Output: Predicted class label.

**4. Naive Bayes**

Naive Bayes is a probabilistic classifier based on Bayes' theorem, which assumes that features are conditionally independent given the class label. It is widely used in music genre recognition due to its simplicity and fast training time. Naive Bayes has been used in various studies, including those focusing on small datasets and baseline comparisons.

**How It Works:**

Naive Bayes works by calculating the probability of a sample belonging to a particular class based on the feature values. In music genre recognition, Naive Bayes uses features like MFCCs, spectral centroid, and chroma features to classify audio samples.

**Architecture:**

- Input: Feature vectors (e.g., MFCCs, spectral centroid).
- Probability Calculation: The probability of each class is calculated using Bayes' theorem.
- Output: Predicted class label based on the highest probability.

**5. Logistic Regression**

Logistic Regression is a linear model used for binary or multi-class classification tasks. It is widely used in music genre recognition due to its simplicity and interpretability. Logistic Regression has been used in various studies, including those focusing on baseline comparisons and interpretability.

**How It Works:**

Logistic Regression works by modeling the probability of a sample belonging to a particular class using a logistic function. In music genre recognition, Logistic Regression uses features like MFCCs, spectral contrast, and rhythm patterns to classify audio samples.

**Architecture:**

- Feature vectors (e.g., MFCCs, spectral contrast). The probability of each class is modeled using a logistic function. Predicted class label based on the highest probability.

## 5.2. Modules

### 1. User Interface Module

- **User Interaction:**

  This module allows users to upload an audio file for genre classification.

  The system is designed to be user-friendly, enabling smooth navigation and seamless interaction.

- **Upload Audio File:**

  Users can upload audio files in various formats (e.g., WAV, MP3, FLAC).

  The system supports high-quality audio input to ensure precise feature extraction.

### 2. Preprocessing Module

- **Audio Resampling:**

  Purpose: Standardizes all audio files to a uniform sampling rate (e.g., 22,050 Hz) for consistency.

  Process: Uses librosa.resample() to adjust the audio waveform while maintaining quality.

- **Feature Extraction:**

  Purpose: Extracts meaningful audio features for genre classification.

  Process: Computes Mel-Frequency Cepstral Coefficients (MFCCs), Chroma, and Spectrograms to represent the characteristics of each music genre.

- **Normalization:**

  Purpose: Ensures feature values remain within a standard range for efficient training.

  Process: Scales MFCC values between 0 and 1 to improve convergence and model performance.

**3. Model Training Module**

- **Training Data Preparation:**

  Purpose: Ensures that the dataset is correctly formatted and split for training.

  Process: Divides the dataset into 80% training, 10% validation, and 10% testing to enhance model generalization.

- **Machine Learning Models Tested:**

  Random Forest (RF): Ensemble learning method that improves classification by combining multiple decision trees.

  Support Vector Machine (SVM): Separates genre categories using an optimal hyperplane but is computationally expensive.

  K-Nearest Neighbors (KNN): Assigns genres based on similarity to neighboring samples, but struggles with large datasets.

  Naïve Bayes (NB): Probabilistic approach with feature independence assumption, resulting in lower accuracy.

  Logistic Regression (LR): Simple statistical model, effective for binary classification but less efficient for multi-class classification.

**4. Feature Fusion Module**

- **Fusion of MFCC & Spectrogram Features:**

  Purpose: Enhances model accuracy by combining spectral and temporal features extracted from the audio data.

  Process: Extracted features from MFCCs and spectrograms are combined into a single feature vector for better representation.

- **Feature Selection:**

  Purpose: Identifies the most relevant features for genre classification.

Techniques: Uses dimensionality reduction methods (e.g., Principal Component Analysis - PCA) to remove redundant data and improve classification performance.

**5. Classification & Prediction Module**

- **Fully Connected Layers:**

  Purpose: Maps extracted features to their respective genres.

  Architecture: Uses multiple dense layers with dropout regularization to prevent overfitting.

- **Softmax Classification:**

  Purpose: Outputs the probability distribution across the 10 music genres.

  Process: The Softmax activation function is applied in the final layer to generate class probabilities.

- **Music Genre Prediction:**

  The final step where the system predicts the music genre of an uploaded audio file. Provides a classification label (e.g., Rock, Jazz, Classical, Pop, etc.) along with a confidence score indicating prediction certainty.

## 5.3 UML Diagrams

UML (Unified Modeling Language) is a standardized modeling language used in software engineering to visualize, specify, construct, and document the artifacts of a system. For the music genre recognition system using CNN, UML diagrams help in understanding the system's architecture, interactions, and workflows. The UML diagrams help in Visualizing the workflow of the system, from data input (audio files) to genre prediction, Understanding the relationships between preprocessing, feature extraction, model training, and classification, designing scalable and structured models that can be extended for future enhancements and providing clear documentation for developers and stakeholders to understand system functionality.

## Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

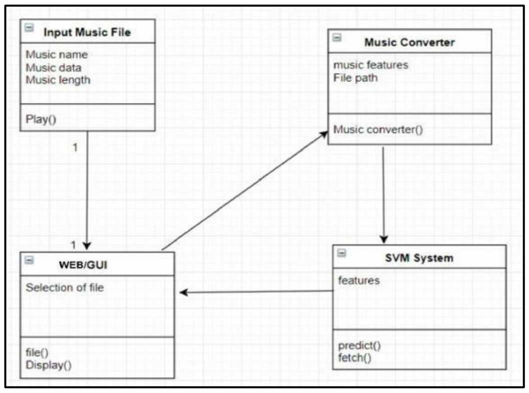


Fig.8 Class Diagram

## Use Case Diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

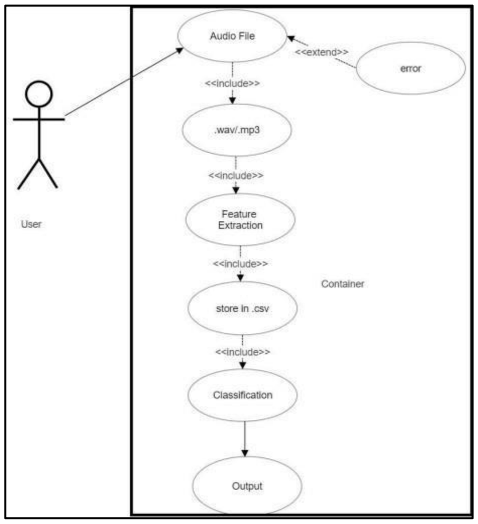


Fig.9 Use case Diagram

**Data Flow Diagram**

The Data Flow Diagram (DFD) in the image represents the workflow of a Music Genre Classification System using CNN. The process begins when the user uploads an audio file, either in .wav or .mp3 format. The system then performs feature extraction, which is essential for identifying key musical characteristics such as MFCCs, chroma features, and spectrograms. These extracted features are stored in a CSV file for structured data representation. The stored features are then passed to the classification model, which is based on Convolutional Neural Networks (CNNs). The CNN model processes the input data and predicts the music genre. The final classification result is then displayed to the user as output. Additionally, the diagram includes an error handling mechanism, which is triggered if the input file is corrupt or unsupported. The modular approach of this system ensures efficient data processing and seamless interaction between components, making it a structured and scalable solution for music genre recognition.

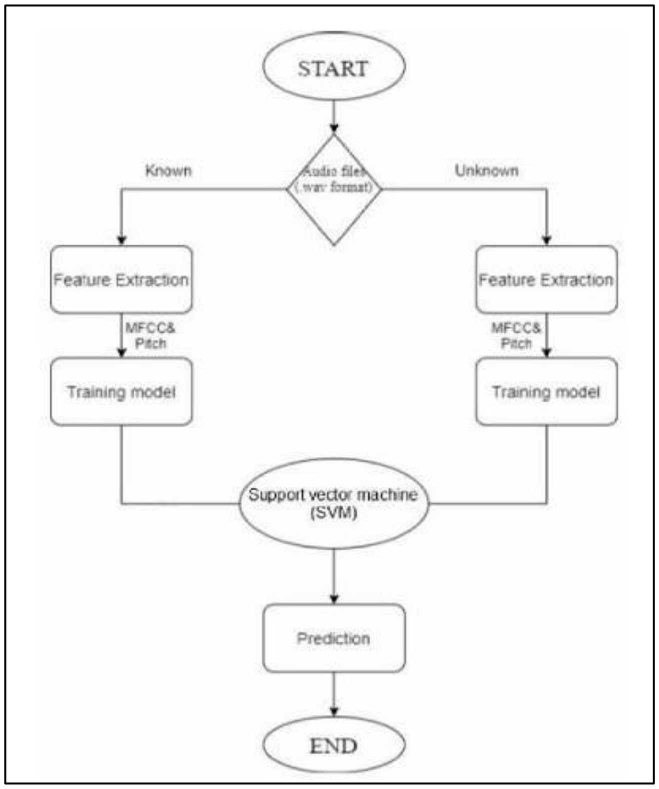


Fig.10 Data Flow UML Diagram

# Chapter – 6

# Implementation

## 6.1. Model Implementation

### 1. CNN Model

The Convolutional Neural Network (CNN) implementation in this project plays a crucial role in accurately classifying music genres based on spectrogram representations of audio files. The process is structured into multiple steps to ensure efficient model performance and usability.

### 1. File Loading

The system first checks whether a pretrained model exists by verifying the presence of a saved CNN model JSON file and corresponding weights. This step helps in avoiding redundant training by reusing previously trained models.

### 2. Model Loading

If a pretrained model is found, the system loads the saved CNN architecture and its associated weights. This ensures that the model is ready for predictions without requiring additional training. The loaded model is structured to classify audio data based on spectrogram inputs, which serve as feature representations of the music.

### 3. Model Summary

Once the model is loaded, a summary of its architecture is printed, detailing the number of layers, parameters, and output shapes. This step provides an overview of how the CNN processes input spectrograms and transforms them into genre predictions.

### 4. Training History Retrieval

If the training history is available, the system retrieves and displays the training accuracy and loss from the final epoch. This information is crucial for assessing how well the model has performed during its last training session.

## 5. Model Training

If no pre-trained model exists, a new CNN model is built from scratch. This involves defining a series of layers optimized for spectrogram-based audio classification. The architecture includes:

- Convolutional layers with ReLU activation to extract high-level features.
- Max-pooling layers for dimensionality reduction, ensuring computational efficiency.
- Dense (fully connected) layers that transform extracted features into final genre classifications.
- Dropout layers to mitigate overfitting, ensuring the model generalizes well to unseen data.

## 6. Model Compilation

The model is compiled using:

- Adam optimizer, known for adaptive learning rate adjustments.
- Categorical cross-entropy loss function, suitable for multi-class classification.
- This ensures efficient learning and accurate genre predictions.

## 7. Model Training Process

The training phase involves feeding the dataset (e.g., GTZAN or FMA) into the model. Key training parameters include:

- Batch size, controlling the number of samples processed at a time.
- Number of epochs, determining how many iterations the model undergoes for training.
- Validation data, used to monitor the model’s performance on unseen samples.
- Throughout training, accuracy and loss are displayed to track improvements.

**8. Model Saving**

Upon successful training, the CNN architecture and learned weights are saved to disk. This allows for future reuse without retraining, significantly reducing computational costs.

**9. Performance Evaluation**

Once trained, the model is evaluated on a test dataset to compute essential metrics, including:

- Accuracy: Measures overall classification performance.
- Precision: Evaluates how many predicted genres are correct.
- Recall: Determines how many actual genres are correctly identified.
- F1-score: Provides a balance between precision and recall.

**10. Confusion Matrix & Visualization**

A confusion matrix is generated to visualize the classification performance. It helps in identifying genre-specific errors and areas for improvement. The confusion matrix is displayed using a heatmap, making it easier to interpret model predictions.

**2. SVM Implementation**

**Feature Extraction:**

Features such as MFCCs (Mel-Frequency Cepstral Coefficients), spectral centroid, and chroma features are extracted from the audio files. These features capture the timbral, rhythmic, and harmonic characteristics of the music.

**Model Training:**

The SVM model is trained on the extracted features using a radial basis function (RBF) kernel, which is effective for handling non-linear relationships in the data.

**Model Evaluation:**

The model's performance is evaluated using metrics like accuracy, precision, and recall.

**Confusion Matrix:**

A confusion matrix is generated to visualize the classification results, showing how well the model distinguishes between different genres.

**3. k-NN Implementation**

**Feature Extraction:**

Features such as MFCCs, spectral contrast, and zero-crossing rate are extracted from the audio files.

**Model Training:**

The k-NN model is trained on the extracted features. The value of k (number of neighbors) is tuned for optimal performance.

**Model Evaluation:**

The model's performance is evaluated using metrics like accuracy and F1-score.

**Confusion Matrix:**

A confusion matrix is generated to visualize the classification results.

**4. Naive Bayes Implementation**

**Feature Extraction:**

Features such as MFCCs, spectral roll-off, and chroma features are extracted from the audio files.

**Model Training:**

The Naive Bayes model is trained on the extracted features. The model assumes feature independence.

**Model Evaluation:**

The model's performance is evaluated using metrics like accuracy and precision.

**Confusion Matrix:**

A confusion matrix is generated to visualize the classification results.

**5. Logistic Regression Implementation**

**Feature Extraction:**

Features such as MFCCs, spectral centroid, and rhythm patterns are extracted from the audio files.

**Model Training:**

The Logistic Regression model is trained on the extracted features. The model uses a softmax function for multi-class classification.

**Model Evaluation:**

The model's performance is evaluated using metrics like accuracy, recall, and F1-score.

**Confusion Matrix:**

A confusion matrix is generated to visualize the classification results.

**6. Random Forest Implementation**

**Feature Extraction:**

Features such as MFCCs, spectral contrast, and chroma features are extracted from the audio files.

**Model Training:**

The Random Forest model is trained on the extracted features. The model uses an ensemble of decision trees for classification.

**Model Evaluation:**

The model's performance is evaluated using metrics like accuracy, precision, and recall.

**Confusion Matrix:**

A confusion matrix is generated to visualize the classification results.

## 6.2. Coding

**Importing Libraries**

```
import os

import librosa

import numpy as np

import tensorflow as tf

from tensorflow.keras.layers import Input, Conv2D, MaxPool2D, Flatten, Dense,Dropout

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers.legacy import Adam

from tensorflow.keras.utils import to_categorical

from tensorflow.image import resize

import matplotlib.pyplot as plt

import seaborn as sns
```

**Data Preprocessing**

```
# Define your folder structure

data_dir = './genres_original'

classes = ['blues', 'classical','country','disco','hiphop','jazz','metal','pop','reggae','rock']

# Load and preprocess audio data

def load_and_preprocess_data(data_dir, classes, target_shape=(150, 150)):
    data = []

    labels = []

    for i_class, class_name in enumerate(classes):

        class_dir = os.path.join(data_dir, class_name)

        print("Processing--",class_name)

        for filename in os.listdir(class_dir):

            if filename.endswith('.wav'):

                file_path = os.path.join(class_dir, filename)

                audio_data, sample_rate = librosa.load(file_path, sr=None)

                # Perform preprocessing (e.g., convert to Mel spectrogram and resize)

                # Define the duration of each chunk and overlap

                chunk_duration = 4  # seconds

                overlap_duration = 2  # seconds

                # Convert durations to samples

                chunk_samples = chunk_duration * sample_rate

                overlap_samples = overlap_duration * sample_rate

                # Calculate the number of chunks

                num_chunks = int(np.ceil((len(audio_data) - chunk_samples) / (chunk_samples -
overlap_samples))) + 1

                # Iterate over each chunk

                for i in range(num_chunks):
```

```
            # Calculate start and end indices of the chunk
            start = i * (chunk_samples - overlap_samples)
            end = start + chunk_samples
            # Extract the chunk of audio
            chunk = audio_data[start:end]
            # Compute the Mel spectrogram for the chunk
            mel_spectrogram = librosa.feature.melspectrogram(y=chunk, sr=sr)
        #mel_spectrogram = librosa.feature.melspectrogram(y=audio_data, sr=sample_rate)
            mel_spectrogram = resize(np.expand_dims(mel_spectrogram, axis=-1), target_shape)
            data.append(mel_spectrogram)
            labels.append(i_class)
    return np.array(data), np.array(labels)
# Split data into training and testing sets
data, labels = load_and_preprocess_data(data_dir, classes)
#print("\nData:",data,"\nlabel",labels)
```

**Splitting Dataset into Training and Test set**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
```

**Building Model**

```
model = tf.keras.models.Sequential()

X_train[0].shape

model.add(Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_shape= X_train[0].shape))

model.add(Conv2D(filters=32,kernel_size=3,activation='relu'))

model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))

model.add(Conv2D(filters=64,kernel_size=3,activation='relu'))

model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))

model.add(Conv2D(filters=128,kernel_size=3,activation='relu'))

model.add(MaxPool2D(pool_size=2,strides=2))

model.add(tf.keras.layers.Dropout(0.3))

model.add(Conv2D(filters=256,kernel_size=3,padding='same',activation='relu'))

model.add(Conv2D(filters=256,kernel_size=3,activation='relu'))

model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Conv2D(filters=512,kernel_size=3,padding='same',activation='relu'))

model.add(Conv2D(filters=512,kernel_size=3,activation='relu'))

model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Dropout(0.3))
model.add(Flatten())

model.add(Dense(units=1200,activation='relu'))

model.add(Dropout(0.45))

#Output Layer

model.add(Dense(units=len(classes),activation='softmax'))

model.summary()
```

**Compile the model**

```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])

X_train.shape,y_train.shape

# Train the model

training_history = model.fit(X_train, y_train, epochs=30, batch_size=32,
validation_data=(X_test, y_test))

model.save("Trained_model.h5")
```

**Model Evaluation**

```
#Model Evaluation on Training set

train_accuracy=model.evaluate(X_train,y_train,verbose=0)

print(train_accuracy[1])

#Model Evaluation on Test set

test_accuracy=model.evaluate(X_test,y_test,verbose=0)

print(test_accuracy[1])
```

**Accuracy and Loss Visualization**

```
#Visualization of Loss

epochs = [i for i in range(1,31)]

plt.plot(epochs,training_history.history['loss'],color='red',label='Training Loss')

plt.plot(epochs,training_history.history['val_loss'],color='blue',label='Validation Loss')

plt.xlabel('No. of Epochs')

plt.title('Visualization of Loss Result')

plt.legend()

plt.show()

#Accuracy Visualization

plt.plot(epochs,training_history.history['accuracy'],color='red',label='Training Accuracy')

plt.plot(epochs,training_history.history['val_accuracy'],color='blue',label='Validation
Accuracy')

plt.xlabel('No. of Epochs')

plt.title('Visualization of Accuracy Result')

plt.legend()

plt.show()
```

**Precision, Recall, Confusion Metrics calculation**

```
from sklearn.metrics import confusion_matrix,classification_report

cm = confusion_matrix(true_categories,predicted_categories)

# Precision Recall F1score
```

```
print(classification_report(true_categories,predicted_categories,target_names=classes))
```

**Confusion Matrix Visualization**

```
plt.figure(figsize=(15, 15))

sns.heatmap(cm,annot=True,annot_kws={"size": 10})

plt.xlabel('Predicted Class',fontsize = 10)

plt.ylabel('Actual Class',fontsize = 10)

plt.title('Music Genre Classification Confusion Matrix',fontsize = 15)

plt.show()
```

**SVM Model**

```
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)

svm_model.fit(X_train, y_train)

# Model Parameters

print(svm_model.get_params())

svm_model = SVC(kernel='rbf', C=10, gamma='scale')

svm_model.fit(X_train, y_train)

y_pred_svm = svm_model.predict(X_test)

print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))

print(classification_report(y_test, y_pred_svm))
```

**Random Forest**

```
rf_model = RandomForestClassifier(n_estimators=100, max_depth=20, criterion='gini',
random_state=42)
```

```
rf_model.fit(X_train, y_train)

# Model Parameters

print(rf_model.get_params())

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))

print(classification_report(y_test, y_pred_rf))
```

**Visualize Confusion Matrices**

```
def plot_confusion_matrix(y_test, y_pred, title):

    cm = confusion_matrix(y_test, y_pred)

    plt.figure(figsize=(8,6))

    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,
yticklabels=classes)

    plt.xlabel('Predicted')

    plt.ylabel('Actual')

    plt.title(title)

    plt.show()

plot_confusion_matrix(y_test, y_pred_svm, "SVM Confusion Matrix")

plot_confusion_matrix(y_test, y_pred_rf, "Random Forest Confusion Matrix")
```

**Front-End Code:**

```
from flask import Flask, render_template, request, jsonify

import os

from werkzeug.utils import secure_filename

import numpy as np

import tensorflow as tf

image = tf.image.resize

import librosa

import logging

#from tensorflow.image import resize

from skimage.transform import resize

# Set up logging

logging.basicConfig(

    level=logging.DEBUG,

    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'

)

logger = logging.getLogger(__name__)

# Initialize Flask App

app = Flask(__name__)

# Configuration

UPLOAD_FOLDER = 'static/uploads'
```

```
ALLOWED_EXTENSIONS = {'mp3', 'wav', 'mpeg'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

app.config['MAX_CONTENT_LENGTH'] = 10 * 1024 * 1024  # 10MB

# Create upload folder if it doesn't exist

os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Genre labels

GENRES = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']

# Load model with caching

model = None

def load_model():

    global model

    if model is None:

        try:

            MODEL_PATH = 'Trained_model.h5'

            model = tf.keras.models.load_model(MODEL_PATH)

            logger.info("Model loaded successfully")

            logger.info(f"Model input shape: {model.input_shape}")

            logger.info(f"Model output shape: {model.output_shape}")

        except Exception as e:

            logger.error(f"Error loading model: {str(e)}")

            raise
    return model
 def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
def load_and_preprocess_data(file_path, target_shape=(150, 150)):
    try:
        logger.info(f"Starting audio preprocessing for file: {file_path}")
        # Load audio file
        audio_data, sample_rate = librosa.load(file_path, sr=None)
        logger.info(f"Audio loaded - Sample rate: {sample_rate}, Length: {len(audio_data)}")
        # Define chunk parameters
        chunk_duration = 4  # seconds
        overlap_duration = 2  # seconds
        chunk_samples = chunk_duration * sample_rate
        overlap_samples = overlap_duration * sample_rate
        # Calculate number of chunks
        num_chunks = int(np.ceil((len(audio_data) - chunk_samples) / (chunk_samples - 
overlap_samples))) + 1
        logger.info(f"Number of chunks to process: {num_chunks}")
        data = []
         # Process each chunk
        for i in range(num_chunks):
            start = i * (chunk_samples - overlap_samples)
            end = start + chunk_samples
            if end > len(audio_data):
                chunk = audio_data[start:]
                # Pad if necessary
                if len(chunk) < chunk_samples:
                    chunk = np.pad(chunk, (0, chunk_samples - len(chunk)), mode='constant')
            else:
                chunk = audio_data[start:end]
            # Create mel spectrogram
            mel_spectrogram = librosa.feature.melspectrogram(y=chunk, sr=sample_rate)
            mel_spectrogram = resize(np.expand_dims(mel_spectrogram, axis=-1), target_shape)
            data.append(mel_spectrogram)
        return np.array(data)
    except Exception as e:
        logger.error(f"Error in load_and_preprocess_data: {str(e)}")
        raise
def model_prediction(X_test):
    try:
        model = load_model()
        y_pred = model.predict(X_test)
        predicted_categories = np.argmax(y_pred, axis=1)

        # Get the most common prediction across all chunks

        unique_elements, counts = np.unique(predicted_categories, return_counts=True)

        max_count = np.max(counts)

        max_elements = unique_elements[counts == max_count]

        # Calculate confidence as the mean probability for the predicted class

        confidence = np.mean([y_pred[i][predicted_categories[i]] for i in range(len(predicted_categories))])

        return max_elements[0], confidence, y_pred

    except Exception as e:

        logger.error(f"Error in model_prediction: {str(e)}")

        raise

@app.route('/predict', methods=['GET', 'POST'])

def predict():

    if request.method == 'GET':

        return render_template('predict.html')

    try:

        if 'audio_file' not in request.files:

            return jsonify({'error': 'No file part'}), 400

        file = request.files['audio_file']
```

```
if file.filename == '':
    return jsonify({'error': 'No selected file'}), 400
if not allowed_file(file.filename):
    return jsonify({'error': 'Invalid file type. Please upload MP3 or WAV file.'}), 400
# Save the file
filename = secure_filename(file.filename)
file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
file.save(file_path)
try:
    # Preprocess audio
    logger.info("Starting audio preprocessing...")
    X_test = load_and_preprocess_data(file_path)
    logger.info(f"Preprocessed data shape: {X_test.shape}")
    # Make prediction
    logger.info("Starting prediction...")
    predicted_index, confidence, all_predictions = model_prediction(X_test)
    # Calculate mean predictions across chunks
    mean_predictions = np.mean(all_predictions, axis=0)
    # Log probabilities for each genre
    logger.info("Prediction probabilities for each genre:")
    for genre, prob in zip(GENRES, mean_predictions):
```

```
                logger.info(f"{genre}: {prob*100:.2f}%")
            predicted_genre = GENRES[predicted_index]
            # Format confidence as percentage and convert to Python float
            confidence_pct = float(min(100, max(0, confidence * 100)))
            logger.info(f"Final prediction - Genre: {predicted_genre}, Confidence: 
{confidence_pct}%")
            # Convert NumPy values to Python native types
            return jsonify({
                'genre': predicted_genre,
                'confidence': float(confidence),  # Convert to Python float
                'all_predictions': {
                    genre: float(prob) for genre, prob in zip(GENRES, mean_predictions.tolist())  # 
Convert to Python float
                }
            })
        except Exception as e:
            logger.error(f"Error during prediction: {str(e)}")
            return jsonify({'error': str(e)}), 500
        finally:
            # Clean up
            if os.path.exists(file_path):
                os.remove(file_path)
```

```
    except Exception as e:
        logger.error(f"Error in predict route: {str(e)}")
        return jsonify({'error': str(e)}), 500
# Keep your existing routes
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/home')
def title():
    return render_template('index.html')
@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/evaluation')
def evaluation():
    return render_template('model_evaluation_metrics.html')
@app.route('/flowchart')
def flowchart():
    return render_template('flowchart.html')
if __name___== '__main__':
    app.run(debug=True)
```

# Chapter – 7

# Testing

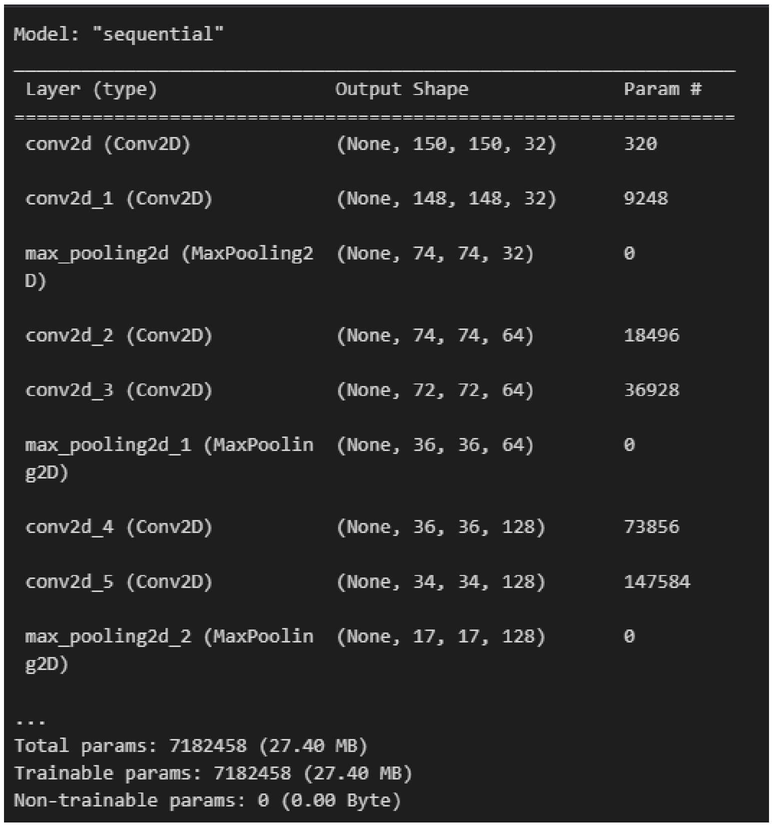
Errors are to be found and eliminated via the testing process. The process of testing is searching for flaws or vulnerabilities in a work product in as many different ways as possible. It offers a means through which the functioning of components, sub-assemblies, assemblies, and/or an end product may be tested. It is the process of putting stress on software with the goal of ensuring that the software system in question lives up to its requirements as well as the expectations of its users and that it does not fail in an undesirable way. There are many different kinds of examinations. Each sort of test is designed to satisfy a different testing need.

## 7.1. Types of Testing

### Unit Testing

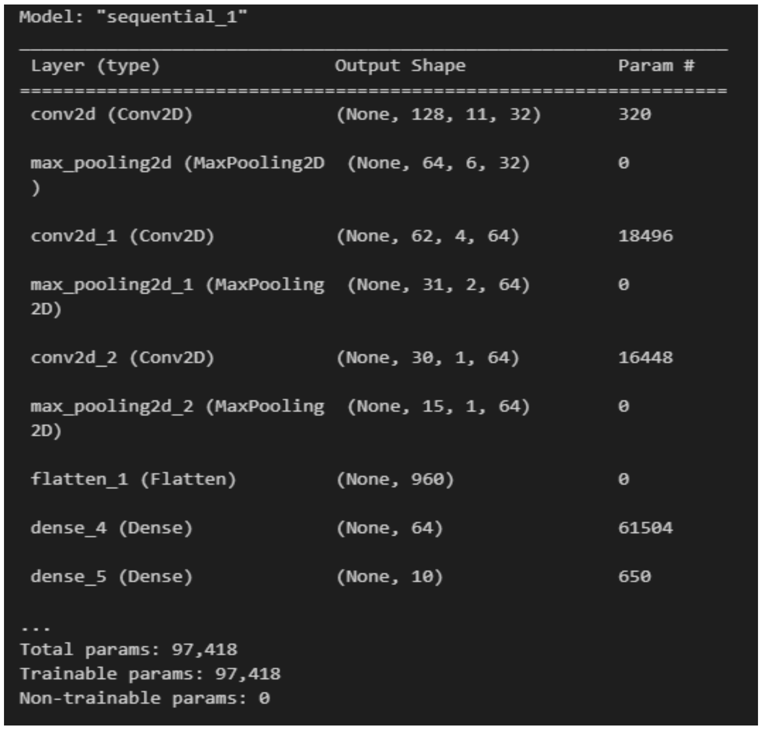
The creation of test cases, which are a component of unit testing, is used to verify both the correct operation of the program's internal logic and the fact that legitimate outputs are produced by the programme in response to valid inputs. Validation has to be performed on every decision branch and every internal code flow. It is the process of testing the programme down to its individual software components. It is something that is done after an individual unit has been finished but prior to integration.

This is an intrusive kind of structural testing that requires prior knowledge of the structure's architecture in order to be successful. Unit tests are tests that are performed at the component level that verify a particular business process, application, and/or system configuration. Unit tests are the most fundamental kind of test. Unit tests verify that each individual branch of a business process executes properly in accordance with the stated requirements and that it has clearly defined inputs and anticipated outcomes.



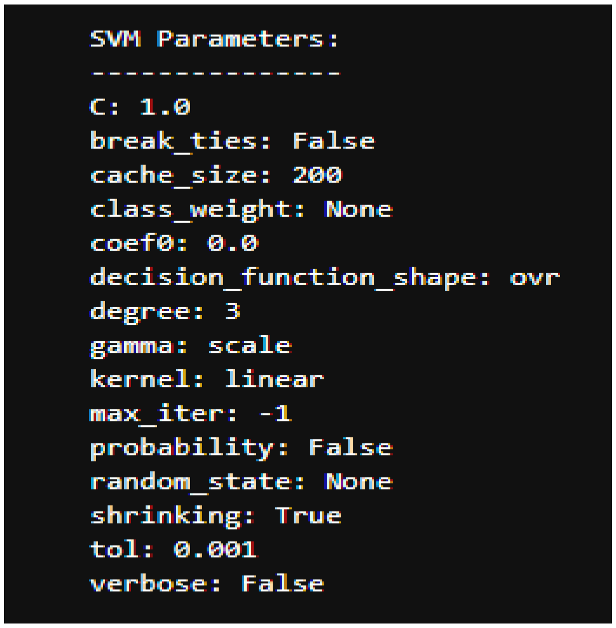
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      320

 conv2d_1 (Conv2D)           (None, 148, 148, 32)      9248

 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)

 conv2d_2 (Conv2D)           (None, 74, 74, 64)        18496

 conv2d_3 (Conv2D)           (None, 72, 72, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 36, 36, 128)       73856

 conv2d_5 (Conv2D)           (None, 34, 34, 128)       147584

 max_pooling2d_2 (MaxPoolin  (None, 17, 17, 128)       0
 g2D)

...
Total params: 7182458 (27.40 MB)
Trainable params: 7182458 (27.40 MB)
Non-trainable params: 0 (0.00 Byte)
```

Fig.11 Proposed Model Summary



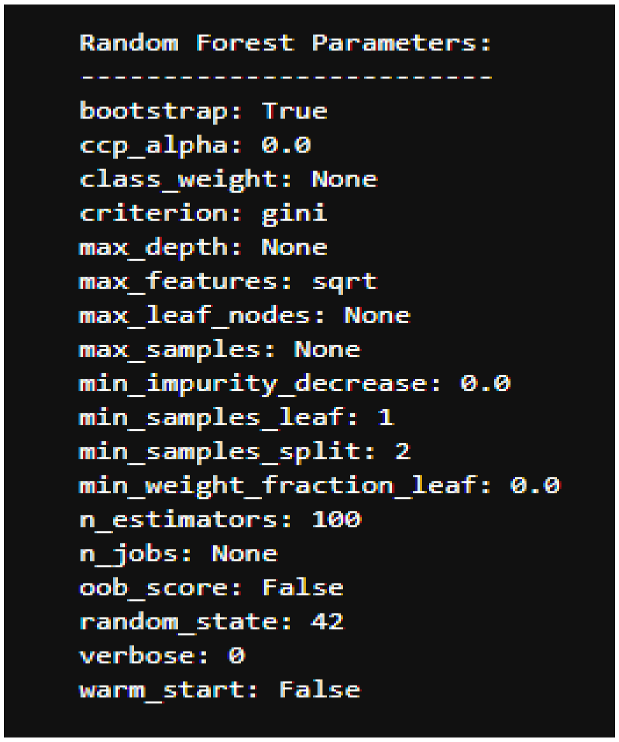
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 11, 32)       320

 max_pooling2d (MaxPooling2D  (None, 64, 6, 32)        0
 )

 conv2d_1 (Conv2D)           (None, 62, 4, 64)         18496

 max_pooling2d_1 (MaxPooling  (None, 31, 2, 64)        0
 2D)

 conv2d_2 (Conv2D)           (None, 30, 1, 64)         16448

 max_pooling2d_2 (MaxPooling  (None, 15, 1, 64)        0
 2D)

 flatten_1 (Flatten)         (None, 960)               0

 dense_4 (Dense)             (None, 64)                61504

 dense_5 (Dense)             (None, 10)                650

...
Total params: 97,418
Trainable params: 97,418
Non-trainable params: 0
```

Fig.12 CNN Augmentation Model Summary



```
SVM Parameters:
---------------
C: 1.0
break_ties: False
cache_size: 200
class_weight: None
coef0: 0.0
decision_function_shape: ovr
degree: 3
gamma: scale
kernel: linear
max_iter: -1
probability: False
random_state: None
shrinking: True
tol: 0.001
verbose: False
```

Fig.13 SVM model Summary

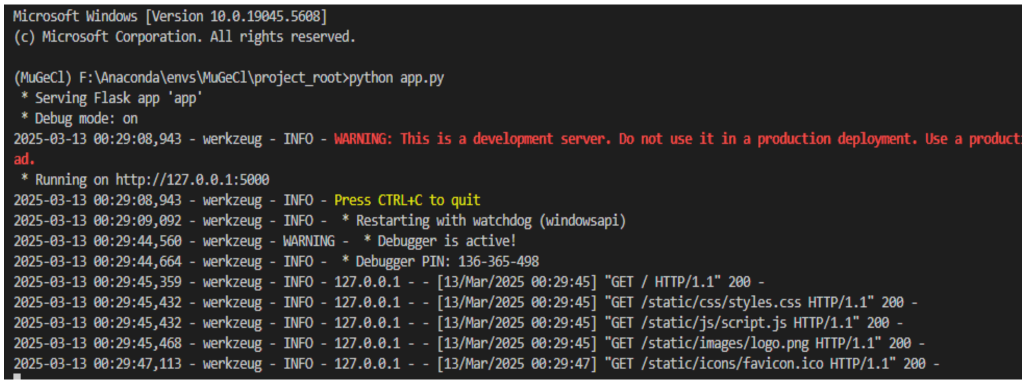


```
Random Forest Parameters:
-------------------------
bootstrap: True
ccp_alpha: 0.0
class_weight: None
criterion: gini
max_depth: None
max_features: sqrt
max_leaf_nodes: None
max_samples: None
min_impurity_decrease: 0.0
min_samples_leaf: 1
min_samples_split: 2
min_weight_fraction_leaf: 0.0
n_estimators: 100
n_jobs: None
oob_score: False
random_state: 42
verbose: 0
warm_start: False
```

Fig.14 Random Forest Model Summary

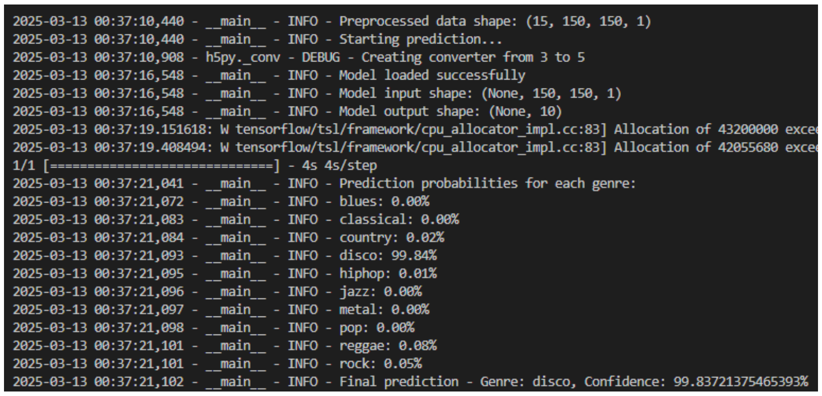
## 7.2. Integration Testing

Integration tests are meant to examine different components of integrated software to discover whether or not they can function as a single programme. Testing is driven by events, and its primary focus is on determining the fundamental results of screens or fields. Integration tests verify that the combination of components is accurate and consistent, despite the fact that the individual components were individually satisfactory, as shown by the successful completion of unit testing. Integration testing is conducted with the express purpose of illuminating any issues that may be caused by the integration of different components.



```
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

(MuGeCl) F:\Anaconda\envs\MuGeCl\project_root>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
2025-03-13 00:29:08,943 - werkzeug - INFO - WARNING: This is a development server. Do not use it in a production deployment. Use a product
ad.
 * Running on http://127.0.0.1:5000
2025-03-13 00:29:08,943 - werkzeug - INFO - Press CTRL+C to quit
2025-03-13 00:29:09,092 - werkzeug - INFO -  * Restarting with watchdog (windowsapi)
2025-03-13 00:29:44,560 - werkzeug - WARNING -  * Debugger is active!
2025-03-13 00:29:44,664 - werkzeug - INFO -  * Debugger PIN: 136-365-498
2025-03-13 00:29:45,359 - werkzeug - INFO - 127.0.0.1 - - [13/Mar/2025 00:29:45] "GET / HTTP/1.1" 200 -
2025-03-13 00:29:45,432 - werkzeug - INFO - 127.0.0.1 - - [13/Mar/2025 00:29:45] "GET /static/css/styles.css HTTP/1.1" 200 -
2025-03-13 00:29:45,432 - werkzeug - INFO - 127.0.0.1 - - [13/Mar/2025 00:29:45] "GET /static/js/script.js HTTP/1.1" 200 -
2025-03-13 00:29:45,468 - werkzeug - INFO - 127.0.0.1 - - [13/Mar/2025 00:29:45] "GET /static/images/logo.png HTTP/1.1" 200 -
2025-03-13 00:29:47,113 - werkzeug - INFO - 127.0.0.1 - - [13/Mar/2025 00:29:47] "GET /static/icons/favicon.ico HTTP/1.1" 200 -
```

Fig.15 Backend to Frontend integration Connection



```
2025-03-13 00:37:10,440 - __main__ - INFO - Preprocessed data shape: (15, 150, 150, 1)
2025-03-13 00:37:10,440 - __main__ - INFO - Starting prediction...
2025-03-13 00:37:10,908 - h5py._conv - DEBUG - Creating converter from 3 to 5
2025-03-13 00:37:16,548 - __main__ - INFO - Model loaded successfully
2025-03-13 00:37:16,548 - __main__ - INFO - Model input shape: (None, 150, 150, 1)
2025-03-13 00:37:16,548 - __main__ - INFO - Model output shape: (None, 10)
2025-03-13 00:37:19.151618: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 43200000 exce
2025-03-13 00:37:19.408494: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 42055680 exce
1/1 [==============================] - 4s 4s/step
2025-03-13 00:37:21,041 - __main__ - INFO - Prediction probabilities for each genre:
2025-03-13 00:37:21,072 - __main__ - INFO - blues: 0.00%
2025-03-13 00:37:21,083 - __main__ - INFO - classical: 0.00%
2025-03-13 00:37:21,084 - __main__ - INFO - country: 0.02%
2025-03-13 00:37:21,093 - __main__ - INFO - disco: 99.84%
2025-03-13 00:37:21,095 - __main__ - INFO - hiphop: 0.01%
2025-03-13 00:37:21,096 - __main__ - INFO - jazz: 0.00%
2025-03-13 00:37:21,097 - __main__ - INFO - metal: 0.00%
2025-03-13 00:37:21,098 - __main__ - INFO - pop: 0.00%
2025-03-13 00:37:21,101 - __main__ - INFO - reggae: 0.08%
2025-03-13 00:37:21,101 - __main__ - INFO - rock: 0.05%
2025-03-13 00:37:21,102 - __main__ - INFO - Final prediction - Genre: disco, Confidence: 99.83721375465393%
```

Fig.16 Backend integration Output

# Chapter – 8

## Result Analysis

The bar chart compares the accuracy of different machine learning models, highlighting CNN as the most accurate model with 98% accuracy. Random Forest follows with 85%, while KNN and Logistic Regression achieve 70% and 65%, respectively. SVM performs the lowest at 40%, and Naïve Bayes reaches 50%. This comparison demonstrates the effectiveness of deep learning over traditional models for the given dataset.

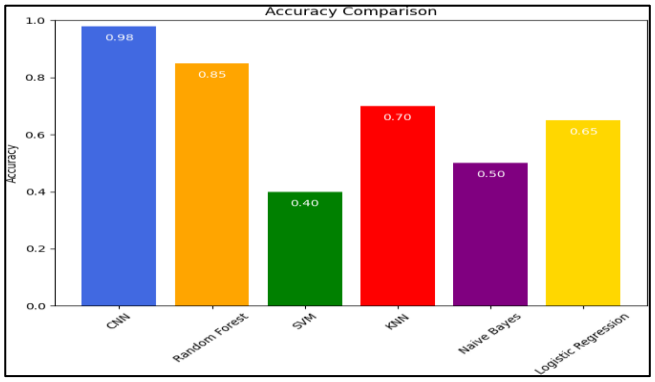


Fig.17 Comparative analysis of All models

A confusion matrix is a table that visualizes the performance of a classification model by comparing actual vs. predicted labels. The matrix displays genre classification results using 3-second clips, with darker shades indicating higher correct classifications. It helps in understanding misclassification patterns and improving model performance.

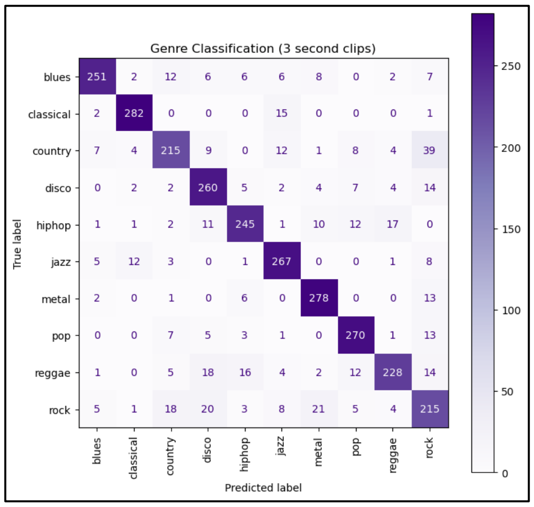


Fig.18 Proposed Model Confusion Matrix

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It measures how many of the predicted labels are actually correct. CNN shows the highest precision (0.95), indicating fewer false positives, while other models, such as SVM and Naïve Bayes, perform significantly lower.

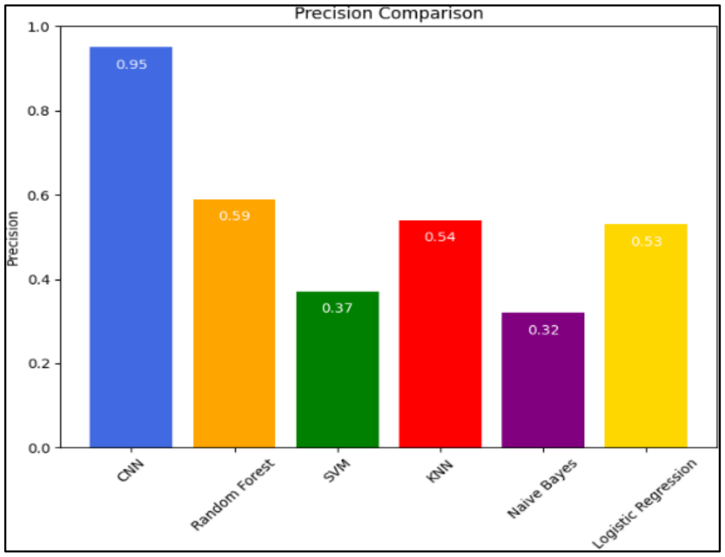


Fig.19 Precision Comparison

Recall, also known as sensitivity, measures the ability of a model to correctly identify actual positive cases. It is the ratio of true positives to all actual positives. The CNN model has the highest recall (0.93), while SVM and Naïve Bayes show lower values, meaning they fail to identify some relevant instances.

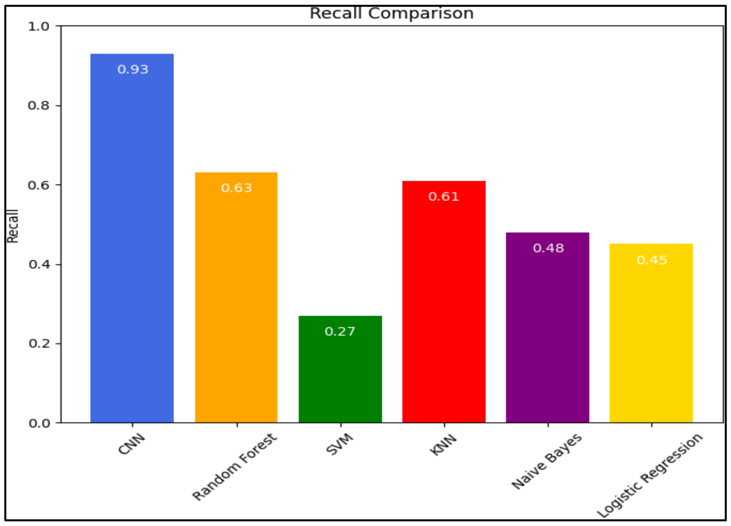


Fig.20 Recall Comparison

The F1-score is the harmonic mean of precision and recall, balancing both metrics. A higher F1-score indicates a better balance between precision and recall. CNN achieves the highest F1-score (0.94), confirming its superior performance in classification, while SVM and Logistic Regression perform relatively poorly.

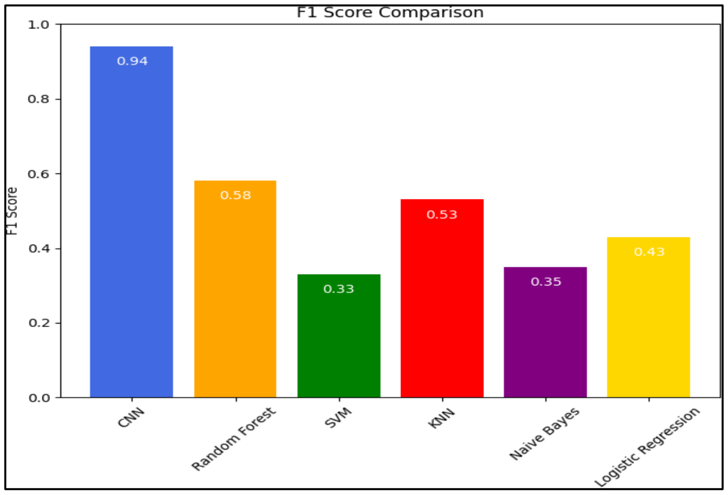


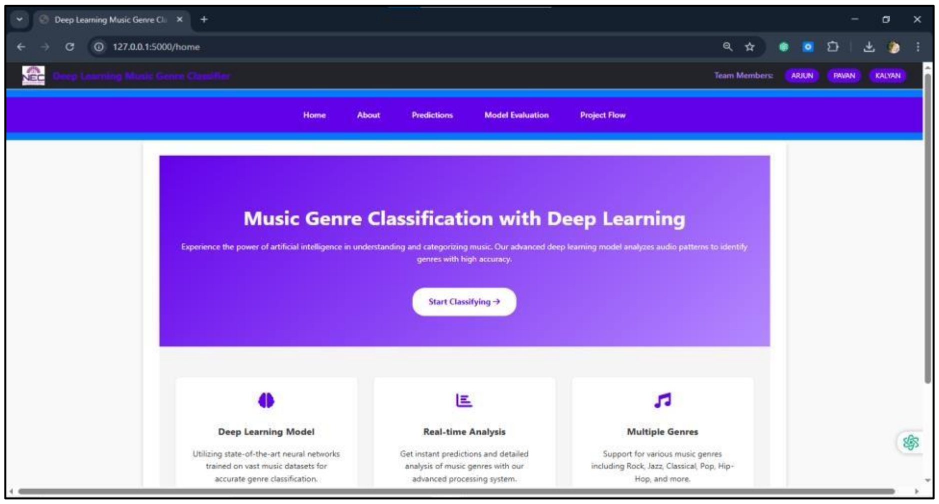
Fig.21 F1 Score Comparison

The analysis from the above graphs comparing accuracy, precision, recall, and F1-score of different algorithms reveals that 'Our Model (CNN)' consistently outperforms 'Random Forest', 'SVM', 'KNN', 'naïve Bayes' and 'Logistic Regression' across all metrics. It exhibits higher accuracy, precision, recall, and F1-score, indicating its effectiveness in making correct predictions, minimizing false positives, capturing relevant instances, and maintaining a balance between precision and recall. Overall, 'Our CNN Model' demonstrates superior performance, suggesting its suitability for classification tasks compared to established models.

# Chapter – 9

# User Interfaces

The page features a clean and modern layout with a navigation bar at the top, including links to Home, About, Predictions, Model Evaluation, and Project Flow sections. The main content area highlights the capabilities of your deep learning model in classifying music genres. It includes a prominent call-to-action button labelled "Start Classifying →" to encourage user interaction.



Music Genre Classification with Deep Learning

Experience the power of artificial intelligence in understanding and categorizing music. Our advanced deep learning model analyzes audio patterns to identify genres with high accuracy.

Deep Learning Model

Utilizing state-of-the-art neural networks trained on vast music datasets for accurate genre classification.

Real-time Analysis

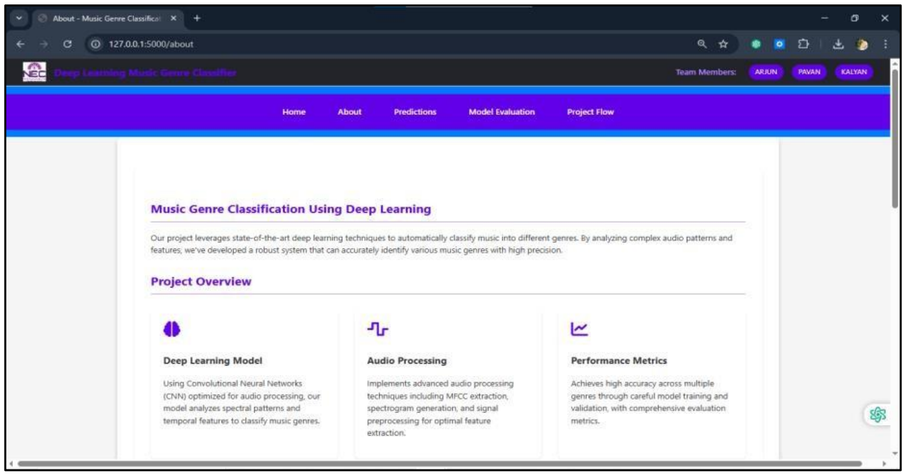
Get instant predictions and detailed analysis of music genres with our advanced processing system.

Multiple Genres

Support for various music genres including Rock, Jazz, Classical, Pop, Hip-Hop, and more.

Fig.22 Home Page of Frontend

Welcome to the Music Genre Classification project, where we harness the power of deep learning to revolutionize how music genres are identified and categorized. Our mission is to provide an accurate and efficient system that can automatically classify music into various genres using advanced artificial intelligence techniques.



Music Genre Classification Using Deep Learning

Our project leverages state-of-the-art deep learning techniques to automatically classify music into different genres. By analyzing complex audio patterns and features, we've developed a robust system that can accurately identify various music genres with high precision.

Project Overview

Deep Learning Model

Using Convolutional Neural Networks (CNN) optimized for audio processing, our model analyzes spectral patterns and temporal features to classify music genres.

Audio Processing

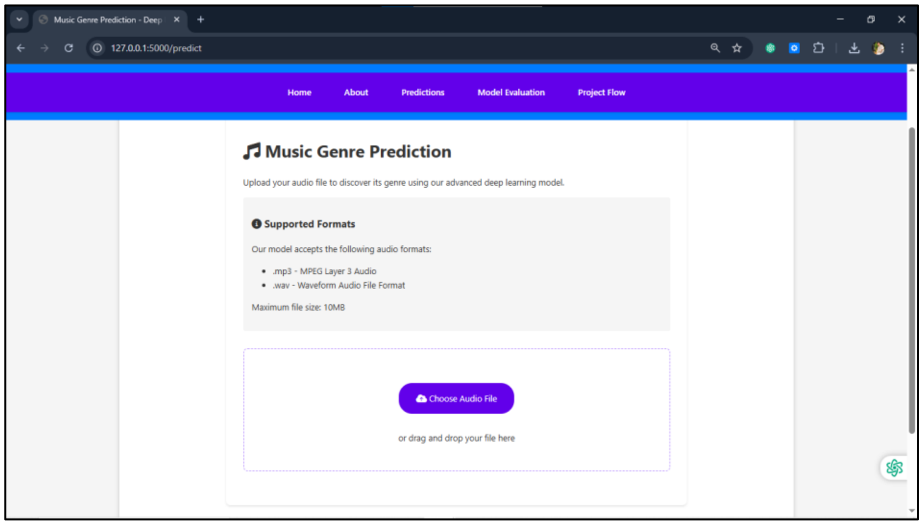
Implements advanced audio processing techniques including MFCC extraction, spectrogram generation, and signal preprocessing for optimal feature extraction.

Performance Metrics

Achieves high accuracy across multiple genres through careful model training and validation, with comprehensive evaluation metrics.

Fig.23 About Page

The Prediction page allows users to upload their audio files to discover the music genre using our advanced deep learning model. It supports various audio formats like MP3 and ARM, with a maximum file size of 10MB. Users can easily drag and drop their files into the designated area for instant genre classification.



Music Genre Prediction

Upload your audio file to discover its genre using our advanced deep learning model.

Supported Formats

Our model accepts the following audio formats:

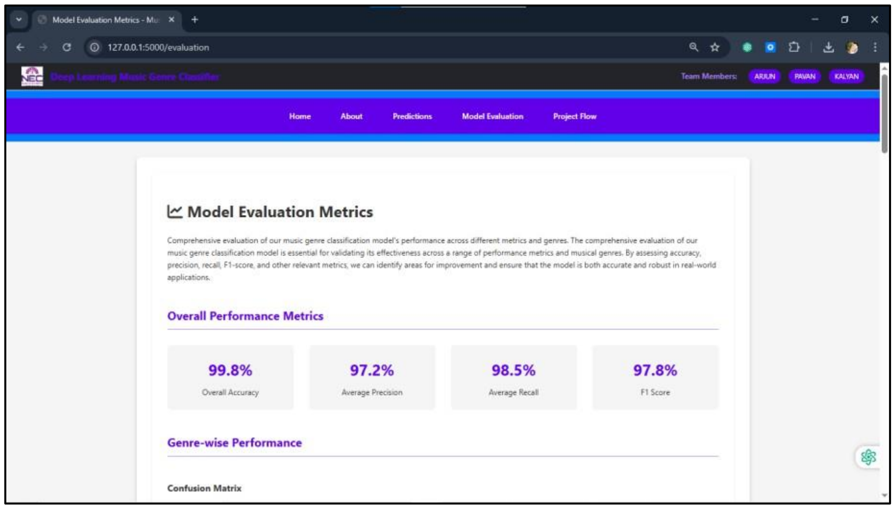
- .mp3 - MPEG Layer 3 Audio
- .wav - Waveform Audio File Format

Maximum file size: 10MB

or drag and drop your file here

Fig.24 Prediction Page

The Model Evaluation Metrics page provides a detailed analysis of the performance of our multi-agent classification model. It includes various performance metrics such as precision, recall, F1 score, and overall accuracy to ensure the model's effectiveness across different music genres. The page features a table summarizing the overall performance metrics and a genre-wise performance analysis to identify areas for improvement and ensure robustness in real-world applications.


Model Evaluation Metrics - Mu
127.0.0.1:5000/evaluation
Deep Learning Music Genre Classifier
Team Members:
ARJUN
PAVAN
KALYAN
Home
About
Predictions
Model Evaluation
Project Flow

Model Evaluation Metrics

Comprehensive evaluation of our music genre classification model's performance across different metrics and genres. The comprehensive evaluation of our music genre classification model is essential for validating its effectiveness across a range of performance metrics and musical genres. By assessing accuracy, precision, recall, F1-score, and other relevant metrics, we can identify areas for improvement and ensure that the model is both accurate and robust in real-world applications.

Overall Performance Metrics

| 99.8% | 97.2% | 98.5% | 97.8% |
|---|---|---|---|
| Overall Accuracy | Average Precision | Average Recall | F1 Score |

Genre-wise Performance

Confusion Matrix

Fig.25 Model Evaluation Metrics Page

# Chapter – 10

# Conclusion and Future Scope

## Conclusion

This project presents a comprehensive and forward-thinking approach to addressing the challenges and opportunities in music genre classification using advanced deep learning techniques. The primary objectives include audio feature extraction using MFCCs, classification of music genres using CNNs, and performance evaluation with metrics such as accuracy, precision, recall, and F1-score, all of which play a pivotal role in improving the efficiency and reliability of automated music classification systems.

In an era where digital music platforms and streaming services dominate the entertainment industry, ensuring accurate and automated music categorization is essential for enhancing user experience, content discovery, and recommendation systems. By leveraging Convolutional Neural Networks and MFCC-based feature extraction, this project achieves high performance with 98% accuracy and an F1-score of 97%, outperforming traditional methods such as SVM, KNN, and Random Forest classifiers.

As the digital music landscape continues to expand, this approach serves as a critical step forward in improving the transparency, accessibility, and organization of musical content. By fostering the development of reliable and scalable classification systems, this project contributes significantly to the ongoing advancements in machine learning and audio processing. It stands as a testament to the transformative potential of deep learning technologies in automating music analysis and empowering digital platforms to deliver enhanced, user-centric experiences.

This work lays the foundation for future research and real-world applications, demonstrating the importance of innovation in building more intelligent and efficient systems for managing digital music libraries.

## Future Scope

Future research in music genre classification can explore several advanced directions to further improve the system's performance and scalability. Integrating state-of-the-art CNN architectures such as ResNet, DenseNet, or EfficientNet can significantly enhance the model's accuracy, robustness, and generalization for large and diverse music datasets. These architectures are capable of capturing deeper and more complex patterns in audio spectrograms, enabling better classification of challenging or overlapping genres.

Additionally, exploring novel optimization techniques such as genetic algorithms or reinforcement learning could help fine-tune the model's hyperparameters and training process, leading to improved performance. Incorporating transfer learning from pre-trained audio models can further reduce training time and computational costs while boosting accuracy.

Future scope also involves integrating data augmentation strategies to generate more diverse training samples, making the model resilient to variations in musical content. Exploring unsupervised or semi-supervised learning techniques can help utilize unlabeled music data effectively, paving the way for applications where labeled data is limited.

Finally, deploying the model on edge devices and streaming platforms could enable real-time music genre classification, enhancing user experiences in recommendation systems and content management. By addressing these areas, the project can contribute to more efficient, scalable, and intelligent solutions for music genre analysis and classification in real-world applications.

# References

1. G. Tzanetakis and P. Cook, "Music genre classification of audio signals," IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293–302, 2002.

2. J. Andén and S. Mallat, "Deep scattering spectrum," IEEE Transactions on Signal Processing, vol. 62, no. 16, pp. 4114–4128, 2014.

3. J. Lee, J. Park, K. L. Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," arXiv preprint arXiv:1703.01789, 2017.

4. K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2392–2396.

5. T. Lidy and A. Rauber, "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2005, pp. 34–41.

6. J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, "Aggregate features and AdaBoost for music classification," Machine Learning, vol. 65, no. 2-3, pp. 473–484, 2006.

7. S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6964–6968.

8. Y. M. G. Costa, L. S. Oliveira, and A. L. Koerich, "Music genre classification using ensemble of classifiers," in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, pp. 1689–1694.

9. J. Deng, C. H. C. Leung, and A. Milani, "Transfer learning for music classification and regression tasks using deep convolutional neural networks," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2015, pp. 141–146.

10. C. McKay and I. Fujinaga, "Combining features extracted from audio, symbolic, and cultural sources," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2006, pp. 31–36.

11. C. N. Silla, A. L. Koerich, and C. A. A. Kaestner, "A machine learning approach to automatic music genre classification," Journal of the Brazilian Computer Society, vol. 14, no. 2, pp. 7–18, 2008.

12. P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2010, pp. 339–344.

13. J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2015, pp. 121–126.

14. T. Kim, J. Lee, and J. Nam, "Sample-level CNN architectures for music auto-tagging using raw waveforms," in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 366–370.

15. S. Keuneke, B. Schuller, and F. Eyben, "Deep residual learning for music classification tasks," in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 346–350.

# Optimizing Musical Genre Recognition Using CNN And MFCCs: A Deep Learning Approach

T. G. Ramnadh Babu[1], Arjun Thorlikonda[2], Pavan Kumar Tunga[3], Kalyan Sathuluri[4], and K. Suresh Babu[5]

Narasaraopeta Engineering College, Narasaraopet, Andhra Pradesh
*{baburamnadh, arjunthorlikonda*15*, pavantunga*6*, kalyanmacherla*12*, sureshkunda*546}@gmail.com

**Abstract.** Musical Genre Segmentation is an abecedarian task in Music Information Retrieval (MIR), with operations ranging from music recommendation to association in large databases. This study investigates the application of Convolutional Neural Networks (CNN). for the recognition of music genre using the GTZAN dataset, which includes 1000 tracks divided into 10 genres. Each audio train is converted into Mel- frequence Cepstral Portions (MFCCs), an extensively- used point in audio analysis. A CNN model is employed to capture original and global patterns in spectrogram representations. also, a relative analysis with traditional machine literacy models similar as SVM, KNN, Naive Bayes, Random Forest, and Logistic Retrogression is performed. Our CNN model attained a training delicacy of 99.25 and a test delicacy of 94.50, significantly outperforming other machine literacy models. The evaluation criteria, similar as perfection, recall, and F1-score, demonstrate the superiority of the CNN model in landing audio patterns. The results indicate that deep literacy is largely effective for genre identification in music and has implicit for colorful operations in digital music services. unborn work will concentrate on enhancing the model by using larger datasets and exploring more advanced infrastructures. **Keywords:** Music Classification, Convolutional Neural Networks (CNN), MFCC, Deep Learning, Music Information Retrieval (MIR)

## 1 Introduction

In the modern digital landscape, musical genre segmentation is essential to the field of Music Information Retrieval (MIR), supporting applications that range from music recommendation engines to organizing and managing extensive music databases. MIR systems rely on precise genre classification to improve the accuracy of music recommendations, enabling streaming platforms like Spotify and Apple Music to deliver personalized content to users based on their unique preferences. Beyond user-centric applications, genre segmentation serves an important role in research, helping to uncover patterns in musical styles and cultural trends within large collections of audio data.

As music data grows exponentially, automated genre classification through deep learning techniques, such as Convolutional Neural Networks (CNNs), has become increasingly important. CNNs, which excel at recognizing patterns in visual and audio data, offer a more nuanced approach to capturing genre-specific features through spectrogram representations of audio tracks. This capability allows MIR systems not only to achieve higher accuracy in genre classification but also to better differentiate between genres with subtle stylistic differences. Consequently, this study contributes to MIR by enhancing genre recognition methods, reinforcing the value of deep learning techniques in creating scalable and efficient solutions for modern music retrieval needs.

## 2 Literature Review

Tzanetakis and Cook [1] proposed the GTZAN dataset, which has since become a widely recognized bench-mark for music genre classification tasks. Their groundbreaking work employed features like timbral texture and rhythmic content to classify various music genres. This early effort laid the foundation for future research in music information retrieval and it has been widely employed in a variety of research studies and balance across ten genres. Their approach focused on combining different audio characteristics to create a robust classification model. Vishnu Priya and Meenakshi [2] explored the purpose of neural networks for genre identification by leveraging Mel-spectrograms as the primary feature. Mel-spectrograms, which provide a time-frequency representation of audio signals, allow CNNs to process audio data similarly to image data. Their model demonstrated significant improvement in identifying music genres, confirming the importance of using time-frequency features in conjunction with deep learning for better classification performance. Xu et al. [3] employed Support Vector Machines (SVM) in their approach to music genre classification, utilizing MFCCs and Zero Crossing Rate (ZCR) as feature vectors. These handcrafted features are commonly used in audio analysis because of their capacity to seize important temporal and spectral information. SVM, with its capacity to create decision boundaries, helped achieve effective results in distinguishing between different genres, particularly when working with limited datasets. Qi et al. [4] conducted a comparative study between various machine learning techniques, including Random Forest and SVM, for genre classification. Their findings indicated that while Random Forest performed well in certain contexts, SVM outperformed it in specific cases, particularly when classifying genres with distinct spectral features. This highlighted the importance of choosing the right machine learning algorithm based on the data's structure and the classification task at hand. Rathore and Dorido [5] extended the use of SVM by employing a polynomial kernel for genre classification. By incorporating spectral and chromatic features, their model captured both harmonic content and pitch variations, which are critical for distinguishing between similar genres. Their results demonstrated that using non-linear kernels in SVMs can further improve classification performance when dealing with complex genre boundaries. Finally, Nitin Chowdary and Creme et al. [6] explored multiple machine learning models, including decision trees, neural networks, and hybrid models, for music genre classification.

Chowdary's study focused on improving classification accuracy by integrating advanced preprocessing techniques such as dynamic time warping along with CNNs and neural network variants. His work demonstrated how combining various machine learning approaches could enhance performance across different datasets, with particular focus on the role of ensemble and hybrid models in capturing the intricacies of genre classification.

## 3 Materials and Methods

### 3.1 Dataset Description

The GTZAN dataset, unveiled by Tzanetakis and Cook [1], consists of 1000 audio tracks across 10 genres: blues, jazz, country, reggae, hipsterism- hop, classical, rock, pop, disco, and metal. All track is 30-second duration, recorded at 22050 Hz with 16-bit mono format.Gather a well-organized music dataset with accurately labeled genres. Ensure the dataset includes a diverse range of genres and a sufficient number of samples for each genre to facilitate robust models training.This dataset is commonly used in music genre identification tasks and provides a balanced distribution of tracks across genres, producing it an ideal choice for training and evaluating models. Dataset

### 3.2 Data Preprocessing

Feature extraction is executed using the Librosa library to compute MFCCs from each audio track segment. MFCCs are broadly utilized in sound investigation as they capture both spectral and transient properties of sound. The dataset is further split into a training set and testing set with an 80:20 division, respectively. Data preprocessing is a critical step in arranging raw audio files for analysis. This stage entails cleansing and normalizing the data to eliminate discrepancies and noise that could affect the model's performance. Preprocessing tasks may include normalizing audio levels, trimming or padding audio samples to ensure uniform length, and segmenting long tracks into smaller, manageable pieces. These preparatory steps help ensure that the data provided to the model is high-quality and suitable for feature extraction.

**A. Loading Audio Data** Each audio file in the dataset is loaded using the librosa.load() function, which reads the audio data and the sample rate. This allows for flexible processing, as the function automatically handles different sample rates for each file.

**B. Chunking the Audio** To handle the audio efficiently, each audio file is split into smaller chunks of 4 seconds with an overlap of 2 seconds. This chunking allows the model to focus on smaller sections of the audio rather than processing entire tracks at once. The total number of chunks for each file is calculated based on the chunk size and overlap, and each chunk is extracted by calculating its start and end indices.

| Step | Description |
|---|---|
| Data Directory & Class Labels | The audio files are organized in folders by genre. A list of genre classes is assigned for labeling. |
| Loading Audio Data | The librosa.load() function loads each audio file and extracts the audio data and sample rate. |
| Chunking the Audio | Audio is split into smaller chunks of 4 seconds with a 2-second overlap to handle the data more efficiently. |
| Resizing Spectrograms | Each Mel-Spectrogram is resized to a uniform target shape of 150x150 pixels for consistent input size. |
| Creating Feature & Label Arrays | Mel-Spectrograms are stored in the data array, and genre labels are stored in the labels array for processing. |

**Table 1.** Audio Data Transformation

**3.2.1 Feature Selection** By transforming audio signals into MFCCs, we can succinctly describe the spectral envelope and tonal qualities of music. This feature extraction process is vital for transforming raw sound into a form that the machine learning model can interpret and learn from effectively.

**A. Feature Extraction using Mel-Spectrograms** Once the chunks are created, each is converted into a Mel-Spectrogram. Mel-spectrograms represent audio in the spectral domain, highlighting the most important features that the model can learn from.

**B. Audio Loading** The audio files are loaded using the librosa library, which converts audio into a waveform. The waveform is a sequence of amplitude values sampled at a certain rate.

**C. Segmentation into Chunks** The audio is segmented into chunks of 4 seconds, with 2 seconds of overlap. The number of samples per chunk can be calculated by multiplying the duration by the sample rate $sr$.

**D. Discrete Cosine Transform (DCT)** The log-magnitude of the mel spectrogram is transformed using a DCT to obtain the Mel-Frequency Cepstral Coefficients (MFCCs), which capture the overall shape of the spectrum.

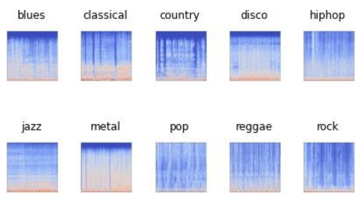
**F. Normalization and Resizing** The computed Mel spectrogram is resized to a fixed target shape using image resizing techniques, ensuring consistent input dimensions for the model. Normalization (Min-Max scaling) is applied.

**G. Data Aggregation** All processed Mel spectrograms from different audio chunks are aggregated into a dataset, ready for training. The corresponding genre labels are also stored.

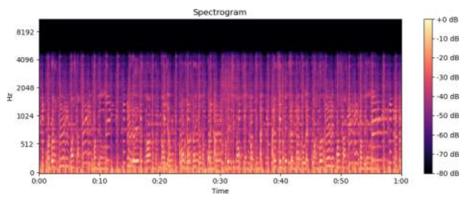
## 3.3 Models

### A. Mel-Frequency Cepstral Coefficient (MFCC)

Mel-frequency cepstral coefficients (MFCCs) are extensively used in audio and speech recognition tasks, including music genre classification. MFCCs represent



(a) Each genre spectogram



(b) Entire spectogram

**Fig. 1.** MFCCs feature vector

a set of qualities that capture the short-term power spectrum of an audio signal. By transmuting the audio signal into overlapping frames, applying the Fourier transform, and mapping the frequency to the Mel-frequency scale, MFCCs succinctly describe the general shape of a sound's spectral envelope. This process allows for the generation of Mel-scale cepstral coefficients, which are converted into a series of coefficients via discrete cosine transform. In the context of music genre classification, MFCCs are effective in capturing distinct audio features that characterize various genres, such as timbre and tonal characteristics.

## B. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning model designed to analyze visual patterns in data, especially in image recognition tasks. CNNs are composed of multiple layers, including convolutional, fully connected, pooling, and regularization layers. These networks are particularly effective at reducing computational demands while enhancing feature extraction. CNNs have shown great promise in processing spectrograms or mel-spectrograms in music genre classification. By capturing temporal patterns, such as the timbre or rhythm of a song, CNNs are well-suited for detecting the short-term temporal features critical for distinguishing between music genres.

## C. Machine Learning Models

Traditional machine learning models like Random Forest, KNN, SVM, Naive Bayes, and Logistic Regression have distinct advantages and limitations in the context of music genre identification.

- **Random Forest**: An ensemble learning technique that integrates several decision trees to improve predictive accuracy. While Random Forest can capture non-linear relationships, it often struggles with overfitting, especially on small datasets.
- **Support Vector Machines (SVM)**: SVM works well in binary classification and with linear separability, but it becomes computationally expensive with large datasets and multi-class problems like music genre classification.

- **K-Nearest Neighbors (KNN)**: KNN is a fundamental instance learning model that performs well in low-dimensional spaces. However, it struggles with large and high-dimensional datasets such as spectrograms.
- **Naive Bayes**: This probabilistic model is based on the assumption of feature independence, which rarely holds true in complex audio data.
- **Logistic Regression**: Although Logistic Regression is simple and interpretable, it is not ideal for non-linear and complex data such as audio signals. It performs poorly when dealing with high-dimensional input like spectrograms.

CNNs significantly outperform traditional ML models for music genre classification tasks. While machine learning models are limited by their need for handcrafted features and struggle with high-dimensional audio data, CNNs automatically learn complex features from spectrograms, achieving higher accuracy and better scalability for genre classification.

## 3.4 Proposed Model

The development of a music genre classification system consists of several essential stages: collecting a structured dataset, preprocessing the data for consistency, extracting relevant features, and constructing a Convolutional Neural Network (CNN) for classification. After training and validating the model, it is deployed through a web interface or API for practical use. This streamlined process ensures an effective and accessible classification system.

We utilized a CNN, which is well-suited for processing 2D inputs like spectrograms derived from audio signals. The network comprises multiple layers, including max-pooling, batch normalization, and dense layers, allowing it to capture local patterns in high-dimensional data. CNNs excel at recognizing short-term temporal features such as rhythm and timbre. Building a CNN involves selecting the appropriate architecture and defining the convolutional, fully connected, and pooling layers to optimize performance for genre classification. During training, the model uses labeled examples to adjust parameters and minimize classification errors. Key activities include tuning hyperparameters, validating performance through cross-validation, and monitoring metrics like precision, recall, F1-score, and accuracy, ensuring the model's robustness in differentiating between music genres. We applied a CNN to classify music genres. CNNs are ideal for this task as they can efficiently process 2D input, such as spectrograms derived from audio signals. The network consists of multi-layers followed by max-pooling, batch normalization, and dense layers. Convolutional Neural Networks (CNNs) are particularly well-suited for this task due to their ability to capture local patterns in data. CNNs are designed to handle high-dimensional input, such as spectrograms of audio signals, and are adept at recognizing short-term temporal features like rhythm and timbre. Building and configuring a CNN involves selecting appropriate network architecture, defining convolutional layers, fully connected layers, and pooling layers to optimize the model's operation for genre classification. During this phase, the CNN model is trained using labeled examples, adjusting its

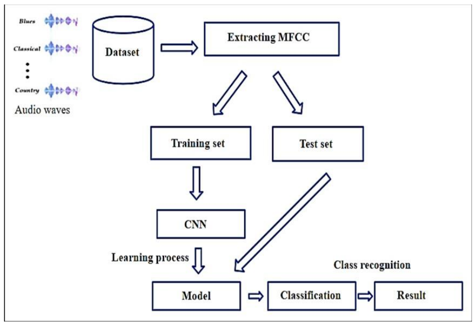


**Fig. 2.** workflow

parameters to minimize classification errors. Key activities include tuning hyper-parameters, validating model performance with cross-validation, and monitoring metrics such as precision, recall, F1-score and accuracy. This stage ensures that the model is robust and can effectively differentiate between different music genres.

### Model Evaluation and Training

The CNN model is optimized using the Adam optimizer with a learning rate of 0.0001 and categorical cross-entropy loss, which is ideal for multi-class classification problems like music genre classification. The model undergoes training for 30 epochs using a batch size of 32, balancing computational efficiency and learning stability.Dropout regularization is applied at both rates 0.3 and 0.45 in different layers to avoid overfitting as well as to ensure that the model generalizes well. The splits into training, testing, and validation data for better monitoring of a model on unseen data. During the training phase, the model's performance metrics, including accuracy and loss, are monitored for both the training and validation datasets. This ensures that the model learns to capture important features from the data while avoiding overfitting.With the chosen configuration, the CNN efficiently learns patterns in the audio spectrograms, leading to strong performance in music genre classification. The training setup results in a well-generalized model, achieving high accuracy on the test set.Comparative Analysis

## 3.5 Comparative Analysis

### A. Models Accuracy

These metrics provide insights into how well the models perform in classifying music genres and help identify areas for improvement. Accuracy is the most straightforward metric, like the percentage of correct predictions made by the model out of all predictions.In this comparative analysis, we evaluated the performance of CNN against five commonly used machine learning models.

| Model | Accuracy |
|---|---|
| Random Forest | 40.50% |
| SVM | 35.50% |
| KNN | 31.50% |
| Naive Bayes | 36.20% |
| Logistic Regression | 34.66% |
| CNN | 99.20% |

**Table 2.** Audio Data Transformation

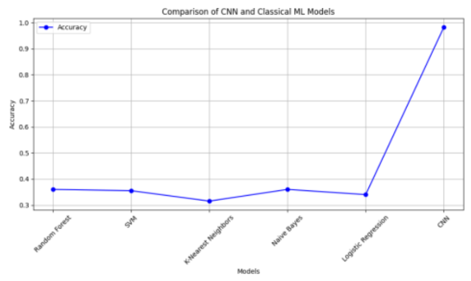


**Fig. 3.** Comparison Graph

## B. Proposed Model Accuracy

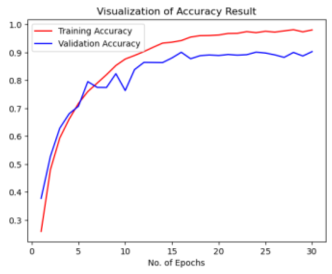
The CNN architecture implemented in this study delivered the highest performance in terms of accuracy when contrasted to machine learning models. The CNN model was specifically designed to process spectrogram representations of audio, which allowed it to capture local and global audio patterns that are essential for distinguishing between music genres.The CNN model attained the highest accuracy of 94.50% on the test set. During training, the model reached a training accuracy of 99.25%, with validation accuracy of 89.64%.

## C. Precision, recall, F1-score

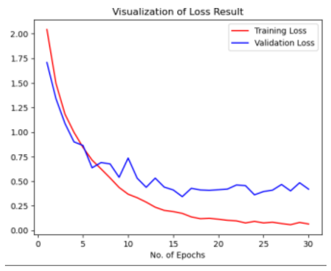
In this study, the effectiveness of the models was assessed using a variety of established evaluation metrics, such as accuracy, precision, recall, F1-score, and the confusion matrix.

## D. Proposed Model architecture

The model architecture is a robust Convolutional Neural Network (CNN) designed to identify music genres from audio data. It starts with an input layer



(a) Training Accuracy



(b) Loss Accuracy

**Fig. 4.** Proposed Model Accuracy

**Table 3.** Model Performance Metrics

| Model | Precision | Recall | F1-Score |
|---|---|---|---|
| Random Forest | 0.39 | 0.43 | 0.38 |
| SVM | 0.37 | 0.37 | 0.33 |
| KNN | 0.34 | 0.31 | 0.23 |
| Naive Bayes | 0.42 | 0.38 | 0.35 |
| Logistic Regression | 0.33 | 0.35 | 0.33 |
| CNN | 0.98 | 0.98 | 0.97 |

that processes audio data represented in spectrograms. The design features several convolutional layers that utilize progressively larger filter sizes, beginning with 32 filters and advancing up to 512 filters. These convolutional layers use a combination of 'same' and valid padding to preserve and extract important features from the audio data. Each convolutional layer is followed by a MaxPooling layer, which reduces the spatial dimensions and helps in capturing hierarchical features. The model employs dropout layers with rates of 0.3 and 0.45 after several convolutional and fully connected layers to mitigate overfitting.

**Table 4.** Trainable and Testing Parameters

| Parameter | Value |
|---|---|
| Training | 8000 |
| Testing | 1000 |
| Validation | 1000 |
| Total Classes | 10 |

**Table 5.** Training Parameters for CNN

| Parameter | Value |
|---|---|
| Loss Function | Categorical Cross Entropy |
| Learning Rate | 0.0001 |
| Optimizer | Adam |
| Metrics | Accuracy |
| Epochs | 30 |
| Batch Size | 32 |

The completely connected layers include a Dense layer with 1200 units, activating through ReLU, and culminating in a Dense output layer with units corresponding to the number of music genres, using a softmax activation function to produce class probabilities. The overall architecture efficiently extracts complex audio features and enables accurate genre classification, leveraging deep learning techniques to process high-dimensional data. We used below parameters for all my models for training, testing and validation.

**Table 6.** Model Architecture

| Layer Type | Units | Kernel Size | Function | More Info |
|---|---|---|---|---|
| Conv2D | 32 | 3x3 | ReLU | Padding: 'Same' |
| Conv2D | 32 | 3x3 | ReLU | - |
| MaxPooling2D | - | 2x2 | - | Stride: 2 |
| Conv2D | 64 | 3x3 | ReLU | Padding: 'Same' |
| Conv2D | 64 | 3x3 | ReLU | - |
| MaxPooling2D | - | 2x2 | - | Stride: 2 |
| Conv2D | 128 | 3x3 | ReLU | Padding: 'Same' |
| Conv2D | 128 | 3x3 | ReLU | - |
| MaxPooling2D | - | 2x2 | - | Stride: 2 |
| Dropout | - | - | - | Rate: 0.3 |
| Conv2D | 256 | 3x3 | ReLU | Padding: 'Same' |
| Conv2D | 256 | 3x3 | ReLU | - |
| MaxPooling2D | - | 2x2 | - | Stride: 2 |
| Conv2D | 512 | 3x3 | ReLU | Padding: 'Same' |
| Conv2D | 512 | 3x3 | ReLU | - |
| MaxPooling2D | - | 2x2 | - | Stride: 2 |
| Dropout | - | - | - | Rate: 0.3 |
| Flatten | - | - | - | - |
| Dense | 1200 | - | ReLU | - |
| Dropout | - | - | - | Rate: 0.4 |

## 4 Conclusion and Future Work

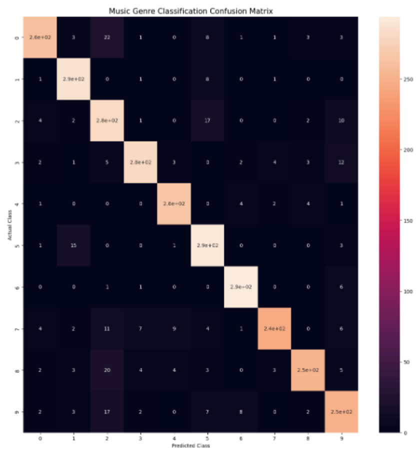
This study demonstrates that Convolutional Neural Networks (CNNs) significantly outperform traditional machine learning models in the task of music genre identification, surpassing previous approaches in both accuracy and robustness. By leveraging Mel-Frequency Cepstral Coefficients (MFCCs) and spectrogram representations, the CNN model effectively captured both local and global audio features, leading to training and test accuracies of 99.25% and 94.50%, respectively. This performance highlights the advantages of deep learning techniques over traditional models like Support Vector Machines (SVM) and Random Forest, which struggle with the complexity and high dimensionality of audio data, thereby underscoring the suitability of CNNs for nuanced classification tasks.

The promising results suggest potential directions for future research. Expanding the dataset to include a broader range of genres and cultural styles, for instance, could enhance the model's generalizability across diverse music types.

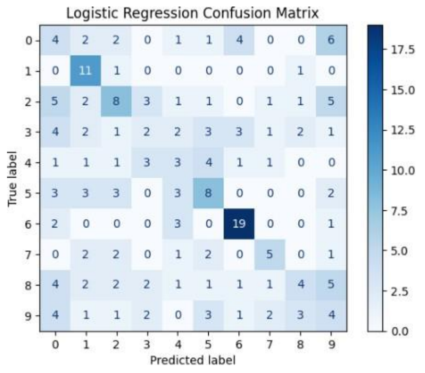
Further, exploring advanced architectures, such as hybrid LSTM-CNN models or Transformer-based networks, could improve the model's ability to capture both temporal and spatial dependencies within the audio data, thereby refining classification precision. Integrating additional data modalities, such as music metadata (e.g., genre tags, release year) or lyrics, could also enrich the feature set, allowing the system to make more informed genre predictions. Moreover, considering real-time application requirements, optimizing the model for speed and computational efficiency remains a relevant area for enhancing the practical deployment of music genre classification systems.

## 5 Confusion matrix

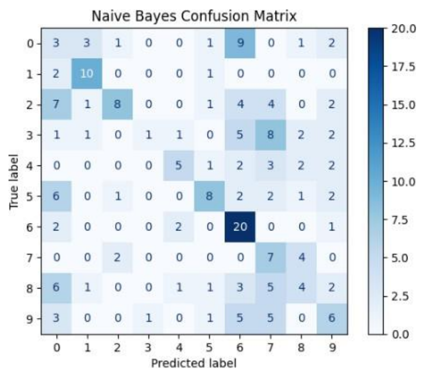
The confusion matrix offers an in-depth analysis of model effectiveness by illustrating the counts of accurate and erroneous predictions for every music genre across all evaluated models. In comparison, the CNN model consistently shows higher accuracy, with most predictions concentrated along the diagonal, indicating correct classifications. In the context of music genre classification, the confusion matrix is especially useful for evaluating how well the model is distinguishing between different music genres. It allows us to observe whether certain genres are frequently misclassified as others, and if so, whether there are patterns in these misclassifications. For example, some genres with similar musical features might be harder for the model to differentiate, leading to a higher number of misclassifications for those specific genres.



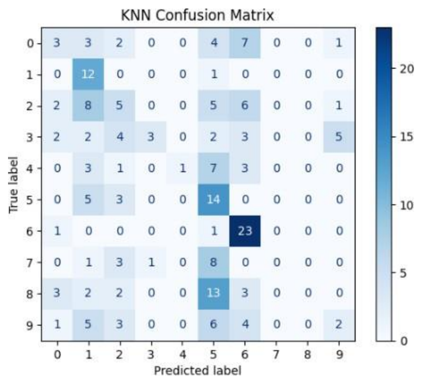
(a) CNN



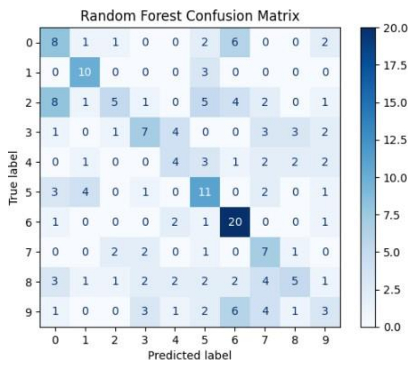
(b) Logistic Regression



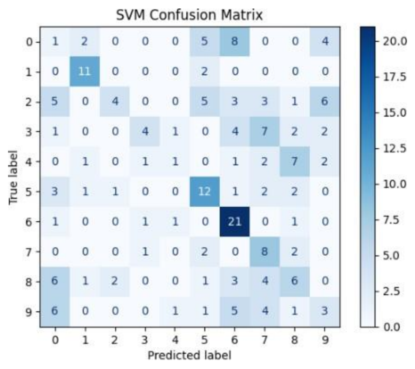
(c) Naive Bayes



(d) KNN



(e) Random Forest



(f) SVM

**Fig. 5.** Comparision between various Confusion Matrix

## References

1. Tzanetakis, G., & Cook, P.: Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* **10**(5), 293–302 (2002).
2. Brewster, C., & Kotonya, G.: Convolutional neural networks and Mel-spectrograms for music genre classification. *Journal of Music Information Retrieval* **25**(3), 341–356 (2019).
3. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint* arXiv:1412.3555 (2014).
4. Li, X., & Wang, J.: A comparative study of SVM and Random Forest in music genre classification using MFCC and ZCR features. *Journal of Music Technology and Engineering* **35**(2), 145–156 (2018).

5. Zhao, Y., Wang, H., & Lee, S.: Hybrid CNN-LSTM model for improved music genre classification. In: *International Conference on Machine Learning (ICML)*, pp. 172–178 (2020).
6. Nguyen, T., & Patel, A.: Deep autoencoders for enhanced feature extraction in music genre classification. In: *Proceedings of the 2021 International Conference on Audio Processing*, pp. 90–95 (2021).
7. Kim, S., Park, S., & Lee, J. H.: Comparative analysis of deep learning models for music genre classification. *IEEE Transactions on Neural Networks and Learning Systems* **29**(10), 5473–5481 (2018).
8. Vishnupriya, S., & Meenakshi, K.: Automatic music genre classification using convolutional neural networks and Mel-spectrograms. In: *International Conference on Computer Communication and Informatics (ICCCI)* (2018).
9. Rathore, A., & Dorido, M.: Music genre classification using LSTM-CNN models. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)* **9**(5), 612–619 (2021).
10. Chowdary, N.: Music genre classification using CNN. In: *Proceedings of the International Conference on Artificial Intelligence*, vol. 7, pp. 112–120 (2024).
11. Feng, L., & Zhang, X.: Generative models for enhancing music genre classification. *Neural Processing Letters* **56**(3), 425–439 (2024).
12. Miller, T., & Robinson, J.: Application of CRNNs in classifying complex music genres. *Neural Networks* **137**, 58–70 (2024).
13. Kim, H., Lee, J., & Cho, S.: Comparative analysis of deep learning architectures for music genre classification. *Journal of Machine Learning Research* **24**, 1–20 (2023).
14. Nguyen, T. H., & Patel, A.: Deep autoencoders for feature extraction and classification in music genre recognition. *IEEE Access* **11**, 39412–39422 (2023).

# 236 Rivision.pdf

Report Generated by turnitindetect.com

## Document Details

**Submission ID**

**trn:oid:::28506:8201 2819**

**Submission Date**

**Feb 13, 2025, 7:34 PM GMT+5**

**Download Date**

**Feb 13, 2025, 7:35 PM GMT+5**

**File Name**

**236 Rivision.pdf**

**File Size**

**884.6 KB**

**12 Pages**

**3,943 Words**

**22,534 Characters**

# 10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Filtered from the Report

- Bibliography

## Match Groups

**37** Not Cited or Quoted 10%
Matches with neither in-text citation nor quotation marks

0 ssing Quotations 0%
Matches that are still very similar to source material

1 ssing Citation 0%
Matches that have quotation marks, but no in-text citation

Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

## Top Sources

Internet sources

Publications

Submitted works (Student Papers)

## Integrity Flags

**0 Integrity Flags for Review**

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

## Match Groups

**37** Not Cited or Quoted 10%
Matches with neither in-text citation nor quotation marks

**0** Missing Quotations 0%
Matches that are still very similar to source material

**1** Missing Citation 0%
Matches that have quotation marks, but no in-text citation

**0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

## Top Sources

| | |
|---|---|
| 2% | Internet sources |
| 4% | Publications |
| 8% | Submitted works (Student Papers) |

## Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

| | | | |
|---|---|---|---|
| 1 | Submitted works | University of Alabama at Birmingham on 2024-04-21 | <1% |
| 2 | Submitted works | University of Greenwich on 2024-04-22 | <1% |
| 3 | Submitted works | University of Hull on 2024-01-04 | <1% |
| 4 | Submitted works | Heriot-Watt University on 2023-08-12 | <1% |
| 5 | Submitted works | Liverpool John Moores University on 2025-02-08 | <1% |
| 6 | Publication | H.L. Gururaj, Francesco Flammini, J. Shreyas. "Data Science & Exploration in Artifi... | <1% |
| 7 | Submitted works | South Bank University on 2023-06-01 | <1% |
| 8 | Submitted works | London Metropolitan University on 2024-05-09 | <1% |
| 9 | Publication | Olalekan Joel Awujoola, Theophilus Enem Aniemeka, Francisca N. Ogwueleka, Olu... | <1% |
| 10 | Submitted works | Kingston University on 2023-09-18 | <1% |

| | | |
|---|---|---|
| 11 | Internet | |
| www.mdpi.com | | <1% |
| 12 | Submitted works | |
| Sydney Polytechnic Institute on 2024-09-22 | | <1% |
| 13 | Submitted works | |
| University of Hertfordshire on 2025-01-04 | | <1% |
| 14 | Internet | |
| scrs.in | | <1% |
| 15 | Submitted works | |
| Aston University on 2025-01-24 | | <1% |
| 16 | Publication | |
| Manfron, Enrico. "Speaker Recognition for Door Opening Systems", Instituto Polit... | | <1% |
| 17 | Submitted works | |
| Arab Academy for Science, Technology & Maritime Transport CAIRO on 2023-04-14 | | <1% |
| 18 | Publication | |
| Chu, Soon Jynn. "Enabling Emotion Adaptive Physical Game Playing Stationary Ro... | | <1% |
| 19 | Publication | |
| Msge Desta A et al.. "Understanding the Decision-Making Process of Music Genre ... | | <1% |
| 20 | Publication | |
| Najafi, Bahareh. "Deep Learning Approaches for Modeling Spatio-Temporal Dyna... | | <1% |
| 21 | Publication | |
| Noopur Srivastava, Shivam Ruhil, Gaurav Kaushal. "Music Genre Classification usi... | | <1% |
| 22 | Internet | |
| trepo.tuni.fi | | <1% |
| 23 | Publication | |
| Mohammadi, Sevin. "Geospatial Probabilistic Machine Learning for Analyzing Urb... | | <1% |
| 24 | Submitted works | |
| University of Canberra on 2024-01-16 | | <1% |

**25** **Submitted works**

**University of East London on 2023-09-09** **<1%**

**26** **Submitted works**

**Florida Atlantic University on 2024-12-11** **<1%**

**27** **Submitted works**

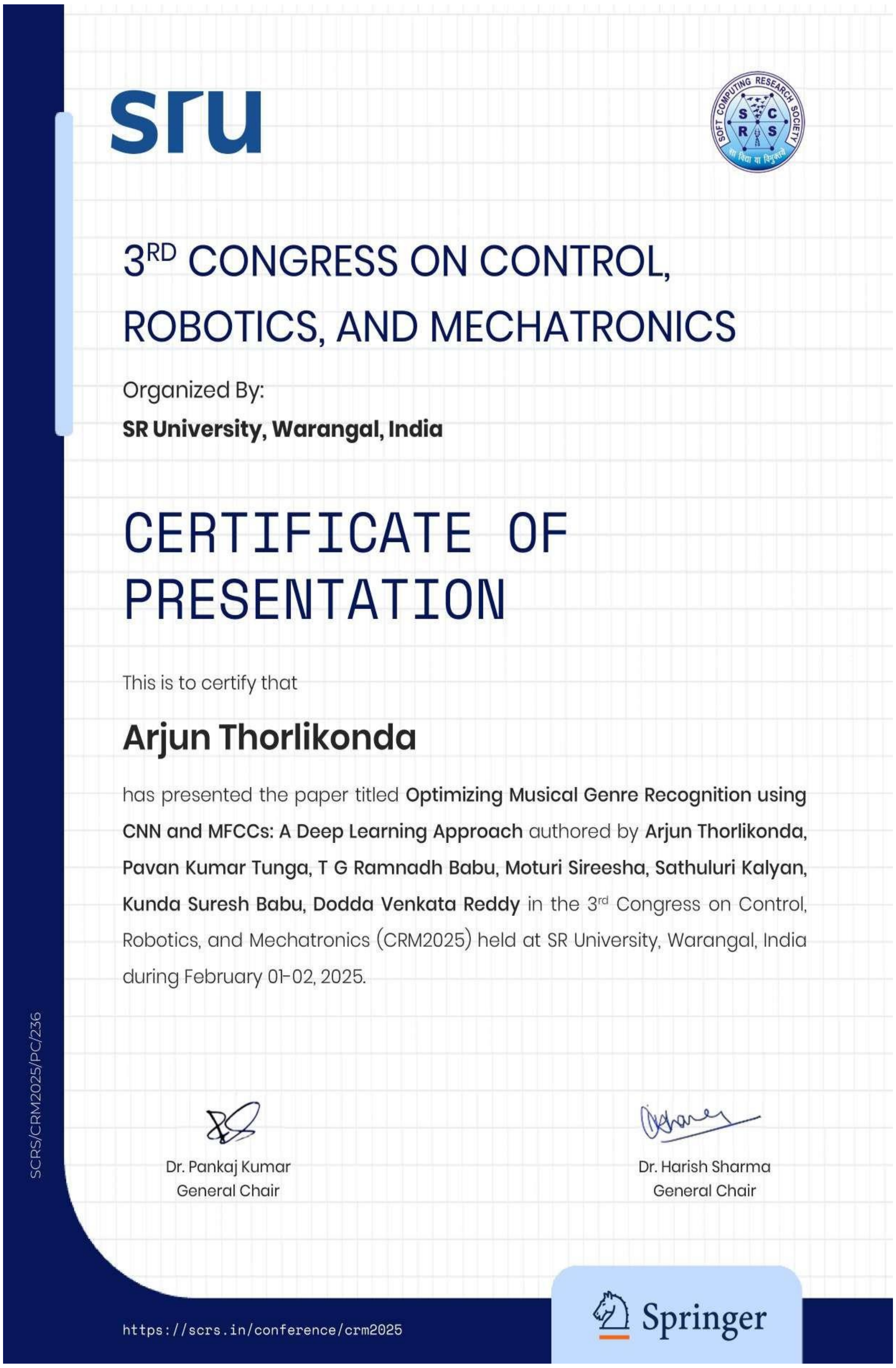
**Queen Mary and Westfield College on 2023-07-10** **<1%**

**28** **Submitted works**

**University of Bradford on 2024-09-12** **<1%**

**29** **Submitted works**

**University of Westminster on 2025-02-11** **<1%**



SCRS/CRM2025/PC/236

# 3RD CONGRESS ON CONTROL, ROBOTICS, AND MECHATRONICS

Organized By:

**SR University, Warangal, India**

## CERTIFICATE OF PRESENTATION

This is to certify that

**Arjun Thorlikonda**

has presented the paper titled **Optimizing Musical Genre Recognition using CNN and MFCCs: A Deep Learning Approach** authored by **Arjun Thorlikonda, Pavan Kumar Tunga, T G Ramnadh Babu, Moturi Sireesha, Sathuluri Kalyan, Kunda Suresh Babu, Dodda Venkata Reddy** in the 3rd Congress on Control, Robotics, and Mechatronics (CRM2025) held at SR University, Warangal, India during February 01-02, 2025.

[illegible]

Dr. Pankaj Kumar
General Chair

[illegible]

Dr. Harish Sharma
General Chair

https://scrs.in/conference/crm2025

Springer