

**SICHUAN UNIVERSITY**

本科生毕业设计（学术论文）

Undergraduate Graduation Project

(Academic Thesis)



Title **Song genre and instrument recognition models**

School **College of Software Engineering**

Major **Software Engineering**

Student’s Name **Israe Dahi**

Student ID:  **2021521460210**

Grade **2021**

Adviser **He Tao**

教务处制表

二Ο二五年五月十二日

Made by the Office of Academic Affairs

May 12, 2025

**摘 要**

**ABSTRACT**

软件工程

Software Engineering

音乐长期以来一直是世界文化的重要组成部分，随着科技的发展，一个融合多学科的新领域——数字音乐应运而生。该领域在音乐创作与机器学习的双重推动下，在近年来取得了显著的发展。随着数字音乐与音频处理技术的不断进步，音乐流派与乐器的自动识别在诸如音乐推荐系统、音频索引以及智能音乐学习平台等多种应用中变得日益重要。

本论文介绍了一个能够识别输入音频文件中音乐流派与乐器内容的系统的设计与开发。该系统旨在构建一个双模型架构，不仅可以准确分类音乐流派，还能识别具体的乐器，从而帮助深入理解音乐内容。

本项目的核心贡献在于使用了两个深度学习模型。第一个模型是流派识别模型，基于 TensorFlow 平台，采用一维卷积神经网络（1D CNN），并在 GTZAN 数据集上进行训练。该模型提取了一系列丰富的音频特征，如 MFCC、Mel 频谱图、Chroma、零交叉率、谱对比度、Tonnetz 以及 RMS 能量，综合分析音乐的音色和节奏特性。

第二个模型为乐器识别模型，基于 PyTorch 平台，在 IRMAS 数据集上进行微调。该模型结合了多种常见音频特征（MFCC、Mel 频谱图、Chroma features）与 VGGish 深度嵌入特征，后者由大量音频数据通过基于 VGG 架构的预训练模型提取，能够生成高层次的音频表示。

这两个模型所提取的特征被拼接后输入到专用的 CNN 与 MLP 分类器中，从而实现对不同音频样本中音乐流派与乐器的稳健识别。两个模型均在真实音乐数据集上进行了充分的训练与测试。实验结果显示，流派模型在十类音乐流派中取得了较高的分类准确率，而乐器模型则能在十一类乐器中进行有效识别。

结果表明，将人工设计的音频特征与深度嵌入特征结合，能显著提升分类性能。本研究通过引入一个特征丰富的双模型音频分类系统，为音乐信息检索领域作出贡献。同时，该系统也为今后在实时识别与多标签分类方面的扩展提供了技术基础，以应对流媒体服务和 AI 音乐服务等现代应用中对智能音频分析日益增长的需求。

Music has been an integral part of world culture for a long time, and with the advent of technology, a new multidisciplinary field was born—digital music. This discipline, driven by rapid advances in both music composition and machine learning, has experienced tremendous growth in recent years. With the development of digital music and audio processing technology, automatic detection of music genre and instrument has become increasingly important for many applications such as music recommendation systems, audio indexing, and intelligent music learning platforms.

This thesis outlines the design and development of a system capable of identifying both the genre and instrumental content of an input audio file. The aim is to develop a dual-model solution that not only accurately classifies music genres but also recognizes specific musical instruments, thereby facilitating the enhanced understanding of musical content.

The key contribution of this project is the use of two deep learning models. The first, a genre recognition model, is coded in TensorFlow and built using a 1D Convolutional Neural Network (CNN) trained on GTZAN. It draws upon a rich set of audio features like MFCCs, Mel spectrograms, Chroma, Zero-Crossing Rate, Spectral Contrast, Tonnetz, and RMS Energy to both timbral and rhythmic properties of music.

The second, instrument recognition model, is executed in PyTorch and fine-tuned over the IRMAS dataset. This model blends a collection of standard audio features (MFCCs, Mel spectrograms, Chroma features) with deep VGGish embeddings—pretrained using a large body of audio material with a VGG-style model—to generate rich, high-level audio representations.

The extracted features are concatenated and fed into expert CNN and MLP classifiers to enable robust identification of music genres and musical instruments in diversified audio samples. Both models have been heavily trained and tested with real music data sets. The genre model showed high accuracy within ten music genres, and the instrument model classified audio within eleven instrument types.

The results indicate that the combination of hand-crafted audio features with deep embeddings significantly improves classification performance. This work contributes to music information retrieval by introducing a feature-rich, dual-model system for automatic audio classification. The system also provides a foundation for future development in real-time detection and multi-label classification, responding to the growing demand for intelligent audio analysis in modern applications such as streaming services and AI-based music services.

Keywords: Song genre classification - Instrument recognition - Dataset - Feature extraction - Model.

**Table of Contents**

[Chapter 1: Preamble 8](#_Toc5628)

[1.1 Introduction 8](#_Toc25825)

[1.2 Current Studies or State of the Art 9](#_Toc32067)

[1.3 Main work of this project 10](#_Toc2458)

[1.4 Organization and Structure 11](#_Toc30224)

[Chapter 2: Background and Core Algorithms 12](#_Toc1641)

[2. Algorithm or the Analysis and Design of System 12](#_Toc30113)

2.1 [Writing instruction of this chapter 12](#_Toc22370)

[2.2 Background Knowledge 13](#_Toc15956)

[2.2.1 Neural Networks and Deep Learning 13](#_Toc4223)

[2.2.2 Transfer Learning and Pretrained Models 14](#_Toc1964)

[2.3 Audio Feature Extraction: Translating Sound to Information 14](#_Toc4365)

[2.3.1 Mel-Frequency Cepstral Coefficients (MFCCs) 14](#_Toc16481)

[2.3.2. Mel Spectograms 15](#_Toc18286)

[2.3.3. Chroma features and chromagrams 16](#_Toc19671)

[2.3.4 VGGish embeddings 17](#_Toc8157)

[2.3.5. Spectral Contrast 17](#_Toc9276)

[2.3.6. Tonnetz Features (Tonal Centroid Coordinates) 18](#_Toc26942)

[2.3.7. Zero-Crossing Rate (ZCR) 18](#_Toc1158)

[2.3.8 Root Mean Square Energy (RMS) 18](#_Toc4531)

[2.4 Deep learning architectures for audio classification 19](#_Toc13136)

[2.4.1. Convolutional neural networks (CNN) 19](#_Toc125)

[2.4.2 Multi-Layer Perceptrons (MLPs) 19](#_Toc6815)

[2.4.3 Why these models and features were chosen 20](#_Toc16427)

[2.5. Summary 21](#_Toc18652)

[Chapter 3: System Analysis and Design 22](#_Toc826)

[3.1. Analysis and Modeling of System Requirements 22](#_Toc21888)

[3.1.1 Overview and modeling of system requirements 22](#_Toc7805)

[3.1.2 Analysis of System Data 23](#_Toc30793)

[3.1.3 Analysis of System Function 24](#_Toc12302)

[3.1.4 Analysis of System Behavior 25](#_Toc20209)

[3.1.5 Analysis of System Performance 25](#_Toc7841)

[3.2. System Design and Modeling 26](#_Toc28091)

[3.2.1 Overview of System Design 26](#_Toc15454)

[3.2.2 Design of System Data (Feature Storage and Handling) 27](#_Toc31062)

[3.2.3 Design of System Software Architecture 27](#_Toc29111)

[3.2.4 Component-Level Design of the System 28](#_Toc15590)

[3.2.5 System Optimization and Scalability 29](#_Toc14298)

[Chapter 4: System Implementation 30](#_Toc1322)

[4.1 Brief Description about the Implementation Environment and Tools 30](#_Toc10149)

[4.2. Description about the Main Program Module (Algorithm Implementation) 31](#_Toc4643)

[4.3. System Operation 33](#_Toc18761)

[4.4 Summary of This Chapter 35](#_Toc15016)

[Chapter 5: System Testing and Evaluation 35](#_Toc29186)

[5.1. Test Environment 35](#_Toc31170)

[5.2. Test Program and Test Data Design 36](#_Toc10377)

[5.3. Test Run and Test Records 37](#_Toc7115)

[5.4. Inference Issues on Unseen IRMAS Files 39](#_Toc31185)

[5.5 Summary 39](#_Toc25209)

[Chapter 6: Conclusion and Reflection 40](#_Toc6867)

[6.1. Work Summary 40](#_Toc14592)

[6.2. Reflection 41](#_Toc28218)

[References: 42](#_Toc12027)

# 

# **Chapter 1: Preamble**

## 1.1 **Introduction**

Music, by definition, consists of vocal or instrumental sounds—or a combination of both—arranged to create beauty in form, harmony, and emotional expression. It can be described as a construct formed through the interaction of different instruments, as observed throughout history and across various civilizations. Thus, music has always been a fundamental part of human life. Humans can often recognize musical genres and instruments effortlessly, thanks to the brain’s ability to detect patterns through repeated exposure. While machines can be trained to perform similar recognition tasks, the process differs significantly in how these patterns are learned, particularly considering that real-world music is often polyphonic. This complexity is one reason music has become a subject of computational processing.

With this development came the digitization of music and, more recently, the integration of artificial intelligence and deep learning techniques. As a result, audio processing has become a key focus area in the field of Music Information Retrieval (MIR). Tasks such as song genre recognition and instrument recognition are commonly addressed using feature extraction, deep learning models, neural network architectures, and multimodal data integration. These techniques play a significant role in modern applications such as music recommendation systems, digital libraries, large-scale music database management, and intelligent music categorization.

The primary goal of this project is to design and develop two deep learning models: one for song genre recognition and another for instrument recognition. The core idea involves processing raw audio data from real-world datasets, extracting meaningful features such as MFCCs, Chroma features, Mel spectrograms, and deep audio embeddings obtained from pre-trained models. Each model is trained on its respective dataset using classification architectures—namely, Convolutional Neural Networks (CNNs) and Multilayer Perceptrons (MLPs).

This thesis presents the full development pipeline, covering data preprocessing, feature extraction, and model training. The datasets used include the GTZAN dataset, which contains 10 folders representing 10 distinct music genres, and the IRMAS dataset, which consists of 11 folders, each corresponding to a different instrument class and containing labeled .wav files.

For the extraction of deep features, the VGGish pre-trained model was employed in the instrument recognition pipeline to generate audio embeddings. The study is rooted in the principles of digital signal processing, neural networks, and supervised learning. It assumes that combining both handcrafted features and deep embeddings can enhance classification accuracy and model robustness. Experiments involved training and testing on real-world datasets, with performance measured using metrics such as accuracy, precision, recall, and F1-score.

The expected outcome is the development of two functional models capable of accurately classifying music genres and instrument types from short audio samples. The project also aims to gain insights into the effectiveness of different features and model configurations, contributing valuable findings to real-world music classification systems.

## **1.2 Current Studies or State of the Art**

The involvement of the music industry with the tech industry and the rise of deep learning technologies and audio classification has led to many researches to focus their studies on exploring music classification using deep learning.

The domain of song genre classification and instrument detection has come a long way from traditional techniques to the sophisticated deep learning methods. One of the pioneering works on genre classification was by Tzanetakis and Cook (2002), who suggested the well-known GTZAN dataset— a collection of 1000 uniformly distributed over 10 genres. Their method utilized hand-designed timbral, rhythmic, and pitch-based features in classification and established a baseline that would influence later research in the area.

Subsequent research used this as a starting point and experimented with more conventional machine learning algorithms such as Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN). These were subsequently replaced by deep learning models, mostly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which improved performance by capturing spatial and temporal relationships in audio signals. Hybrid models like CNN-RNN hybrids were also tried, successfully overcoming the limitations of sequential models and enabling simultaneous extraction of temporal and spectral features.

At the same time, early instrument recognition work also relied heavily on statistical models and manually crafted audio features, particularly Mel-Frequency Cepstral Coefficients (MFCCs) and Gaussian Mixture Models (GMMs). With the advent of deep learning, CNNs gained popularity because they were able to learn complex representations automatically from spectrograms and other time-frequency plots. This change represented a transition from manual feature engineering to comprehensive end-to-end learning models.

Even newer developments have seen the application of Convolutional Recurrent Neural Networks (CRNNs) that capitalize on the capability of CNNs to extract spatial features and RNNs for capturing temporal patterns. Such models further improved the state-of-the-art through enabling hierarchical learning of features from raw or lightly pre-processed audio information.

One of the primary movers in these advances has been the arrival and accessibility of large, high-quality annotated datasets. Datasets such as GTZAN for music genre classification and IRMAS for instrument recognition have provided researchers with decent resources to properly train, cross-validate, and benchmark their models, hence significantly accelerating both fields of progress.

## 1.3 **Main work of this project**

The main work flow of this project includes :

* Investigating and comparing multiple types of audio features including MFCC, Chroma features , Mel spectrograms, and deep embeddings (VGGish).
* Pre processing and organizing datasets (GTZAN and IRMAS) for training and testing purposes.
* Building a CNN-based model for genre classification and an MLP-based model for instrument recognition.
* Evaluating both models using appropriate metrics (accuracy, F1-score, etc.).
* Analyzing the results and discussing the advantages and limitations of different feature combinations.
* Summarizing insights and proposing future directions for improvement.

## **1.4 Organization and Structure**

The body of this thesis is divided into six parts, each serving a specific purpose:

Chapter 1 : Introduction

Presents the research background and motivation for automatic instrument and song genre classification, surveys related work and existing systems, and gives an overview of the objectives, scope, and main contributions of this project.

Chapter 2 : Basic Algorithms and Background

Introduces the fundamentals of neural networks and deep learning, the concept of transfer learning from pre-trained models (e.g., VGGish), and describes in excellent detail the audio feature extraction techniques employed (MFCCs, Mel spectrograms, Chroma features, Spectral Contrast, Tonnetz, Zero-Crossing Rate, RMS Energy, etc.).

Chapter 3 : System Analysis and Design

Conducts requirements analysis (functional, data, performance), defines system behavior, and presents the modular architecture. Explains the data preprocessing pipeline, feature storage strategy, and overall design of the MLP-based instrument recognition and CNN-based genre classification modules.

Chapter 4 : Implementation

Describe the development environments and tools (Google Colab with PyTorch for the instrument model; Jupyter Notebook with TensorFlow/Keras for the genre model), the processes for feature extraction, model construction, training pipeline, and important implementation specifics for both recognition tasks.

Chapter 5 : Testing and Results

Explains experimental setup, dataset splits, and evaluation metrics (accuracy, precision, recall, F1-score). Provides classification reports, confusion matrices, and performance comparison for the two models, and analysis of strengths, weaknesses, and running behavior.

Chapter 6 : Summary and Reflection

Chronicles work done, highlights key findings and contributions, reflects on challenges encountered and learning obtained, and indicates avenues for future enhancement and extension.

# **Chapter 2: Background and Core Algorithms**

## **2. Algorithm or the Analysis and Design of System**

This chapter features an exhaustive analysis of the system's design in relation to architecture, foundational genre and instrument recognition algorithms utilized in both, along with the challenges faced specific to development. The system is based on design heavily relying upon machine learning algorithms for carrying out music genre classification and instrument detection. This chapter presents an in-depth explanation and modeling of the system used for automatic music genre classification and instrument recognition using machine learning techniques. The discussion is focused on the underlying algorithms, system design and performance needs, and motivations for underlying design choices. The system revolves around the extraction and classification of audio features using models trained on materials such as IRMAS and GTZAN. We begin with the data modeling and requirements analysis, proceed to system design and algorithm organization, and conclude with the specifics of the model implementations and their theory.

## 2.1 **Writing instruction of this chapter**

In this chapter, we talk about the foundation into the core algorithm used in this project: automatic song genre classification and instrument recognition from a wav audio file. It is important to know the key background before executing, training, and testing the models that are involved in the project.

For performing the desired classification tasks, there are two important elements needed: classification modeling and feature extraction. Raw audio data, usually in waveform, is of high dimensionality and hard to be directly interpreted by machine learning algorithms. We then extract a subset of acoustic features that compactly and informatively represent the most relevant parts of the audio signal. We feed these features into machine learning and deep learning models, where they learn patterns and make predictions about what genre or instrument is present in the audio.

This chapter will mainly talk about:

* The reason for using audio feature extraction.
* Descriptions of each feature type used (MFCC, Mel spectrogram, Chroma features, VGGish).
* Deep learning architectures used in this project (Convolutional Neural Network and Multi Layer Perceptron)
* Why these specific models and features were selected for the problem.
* By conducting this thorough investigation, we establish the foundation for the methods used in subsequent chapters

## **2.2 Background Knowledge**

### **2.2.1 Neural Networks and Deep Learning**

Artificial neural networks or ANN for short are computational models inspired by the human brain way of learning and understanding. Artificial neural networks contain layers of linked nodes, or neurons, in which each connection has a weight associated with the signal passed from node to node. ANNs are most well-suited for classifying and predicting input information and are hence crucial in classification and recognition operations.

Deep learning follows the foundational concept of neural networks but incorporates numerous hidden layers between the input and output. The deep layers enable the network to catch deeper-level, more complex representations of data so it can execute more intricate patterns and subtleties in data that tend to get lost with shallow models.

In order to train such models, techniques like gradient descent are utilized. I.e., the model keeps refining its internal parameters—weights—over and over again until it becomes better at generating the correct predictions.

One of the big reasons deep learning performs so well is through something called activation functions—like ReLU and Sigmoid. They introduce non-linearity, or in other terms, enable the model to map more complex issues instead of drawing straight lines. Because of this multi-layer, flexible architecture, deep learning can perform well with tasks like picture recognition, deciphering spoken language, and breaking down text.

### **2.2.2 Transfer Learning and Pretrained Models**

One such key idea helping deep learning efforts such as these is transfer learning, which implies applying a trained model on a task and optimizing it for an alternate but linked task. Rather than training up a huge neural network from start, which proves to be highly computationally intense and demands plenty of data to train on,—we employ a pre trained set of models who have already internalized common schemes from huge repositories of audio databases. One such model utilized in this project is VGGish, a Google-created model that was trained on millions of YouTube videos to recognize a broad range of sounds. These pre trained models supply embeddings—compact, high-level representations of sound—which are used as inputs to our own classifiers. Transfer learning allows for faster training, improved performance with limited datasets, and facilitates tapping into deep audio knowledge already ingrained in the model.

## **2.3 Audio Feature Extraction: Translating Sound to Information**

A computer for example and therefore a model cannot understand an audio like humans do which means by ear or as a raw sound so the audio has to first be translated into a form that is both structured and representative of the original signal’s unique content it can understand: that Is the work provided by audio feature extraction. In contrast to images or text, which are typically well-structured and easy to visualize, audio signals can be extremely variable in loudness, frequency composition, rhythm, and harmonic organization. Feature extraction brings such information down to structured and informative numeric representations.

### **2.3.1 Mel-Frequency Cepstral Coefficients (MFCCs)**

Mel Frequency Cepstral Coefficient or MFCCS for short are among the most popular and most used features in audio classification. They were originally developed for speech recognition systems but then started being used for anything audio related in the deep learning field.

Mel Frequency Cepstral Coefficient are mathematical representations of raw audios. So the core idea behind them is basically to model how humans perceive sound.

As for how exactly they work, here is a small breakdown for that:

1. The audio signal is divided into short overlapping frames.
2. A Fast Fourier Transform (FFT) is applied to each frame to convert it from the time domain into the frequency domain.
3. The frequency spectrum is passed through a Mel-scale filter bank, which mimics the human ear’s sensitivity to different frequency ranges (more sensitive to lower frequencies).
4. The logarithm of the filter bank energies is taken to capture loudness information.
5. A Discrete Cosine Transform (DCT) is applied to these log energies to compact the representation and remove correlation between features.

This makes Mel frequency cepstral coefficient ideal for capturing the timbre of the sounds which makes them perfect for projects such as song genre recognition and instrument recognition.

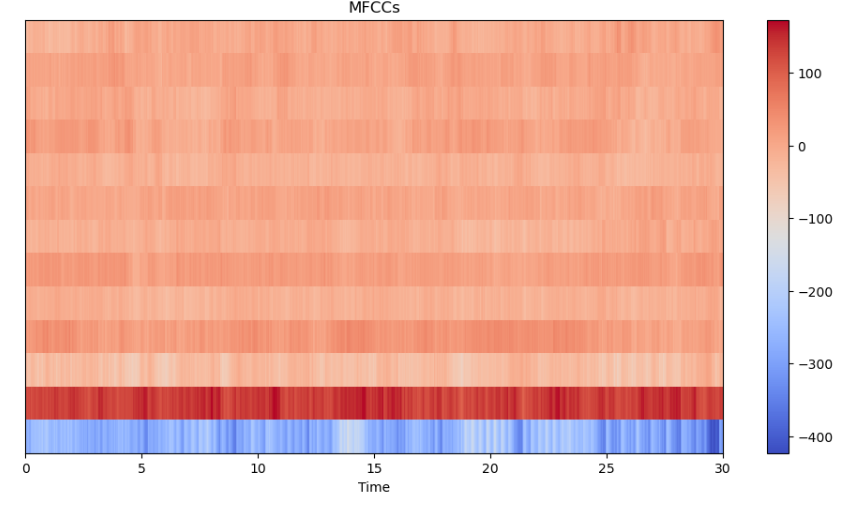


Fig 1 : Mel frequency cepstral coefficient figure example

### **2.3.2. Mel Spectograms**

A Mel spectogram is a visual representation of an audio signal’s frequency content overtime. The Mel spectrogram utilizes the Mel scale for the frequency axis of the spectrogram, again to better match human perception, because if a human can perceive pitch differences more easily at lower frequencies than at higher ones, then the scale should reflect that — spacing lower frequencies more finely and compressing higher ones, thereby mimicking how our ears interpret sound. This is very informative since it captures how the frequency content of an audio signal changes over time. Therefore, these plots are useful for audio applications.

When represented under the form of a graph (see Fig 2) , it is a 2D representation where one axis is for time and the other frequency bins on the Mel scale and each point represents the energy level at a particular time frequency pair.

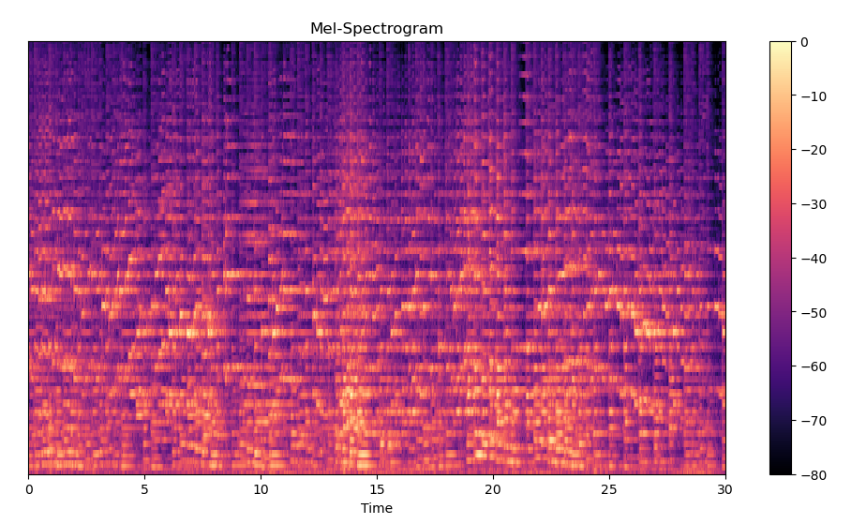


Fig 2 : Representation figure of Mel Spectogram

Mel Spectograms are very useful when using convolutional neural networks and for audio classification such as song genre recognition.

### **2.3.3. Chroma features and chromagrams**

Chroma features and Chromagrams are techniques employed for music information retrieval to analyse the tonal and harmonic content of audio signals. Just like in music theory we have many notes we rely on to understand the tones and harmonies in a song, it is quite similar to it when it comes to chroma features as it represents the energy distributed across 12 pitches classes or distinct semitones of the musical octave: C, C♯/D♭, D, D♯/E♭, E, F, F♯/G♭, G, G♯/A♭, A, A♯/B♭, and B.

When the chroma features of an audio are calculated over time and reduced, it produces a chromagram or chroma spectrogram. In this chromagram when one axis represents the time, the other axis represents the 12 chroma bins and the colour intensity at each point represents the strength of energy of the corresponding pitch at the time.

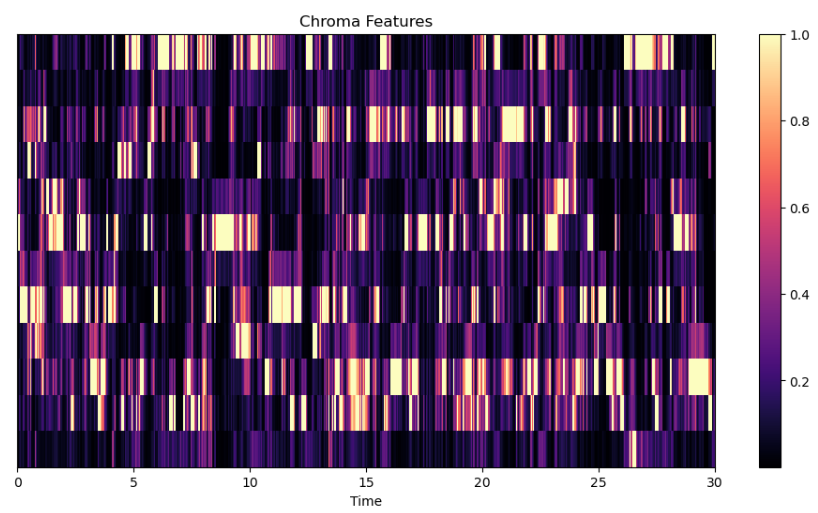


Fig 3 : Figure representing an audio’s Chromagram

### **2.3.4 VGGish embeddings**

VGGish is an audio google developed model that is used in this project to extract embeddings which are a compact and high level summaries of audio data. They are 128 dimensional vectors that represent the semantic content of an audio clip or in simpler words it takes the audio then turns it into a log mel spectogram to represent the frequencies in the sound over time so models can understand the audio data without training from scratch.

However in the context of this project, it is used alongside other features in the instrument recognition model creation.

**2.3.5. Spectral Contrast**  
Spectral contrast measures contrast in energy between valleys and peaks in a frequency spectrum. Unlike MFCCs, which blur fine details to mimic human hearing, spectral contrast emphasizes differences between harmonic and non-harmonic parts of the audio. It focuses in the difference in magnitude within a spectrum to make a distinction between regions of high and low spectral energy where peaks represent the high energy concentrations and the valleys the low ones and thus helps differentiate the different types of sounds or silence or change of the type of sound.

Why it matters :

Such styles as classical or jazz, in which timbral and dynamic contrasts will be powerful, produce strongly divergent spectral contrast profiles. It helps to differentiate "bright" from "dark" sounding compositions or instruments.

### **2.3.6. Tonnetz Features (Tonal Centroid Coordinates)**

Tonnetz features are the representation of the harmonic intervals and pitch. Each Tonnetz represents a chord typically major or minor triad.

When visualized, the look like a continuous series of triangles with notes at the intersection points.

Why it matters:

These features are especially useful when it comes to genre identification, where harmony and key are used heavily (e.g., in classical or pop). They are also very good at finding musical modes and chord progressions.

### **2.3.7. Zero-Crossing Rate (ZCR)**

ZCR refers to the number of times the audio wave crosses the time axis (i.e., from positive to negative and from negative to positive). It is a measure of how "noisy" or "percussive" the signal is.

Why it's important

Percussive sounds and harsh textures (e.g., drums or overdriven guitars) are characterized by high ZCRs. It's a good attribute to identify dynamic styles (e.g., metal, EDM) or sharp instrument attacks (e.g., snare drums or maracas).

### **2.3.8 Root Mean Square Energy (RMS)**

A statistical measure that calculates the average magnitude of a fluctuating signal, like voltage or current, over time or in other words, it measures the loudness or energy content of an audio signal over time. It tells us how strong or powerful an audio is on an average per frame.

Why it matters:

Helps to recognize dynamic patterns such as a light beginning followed by a large chorus in a pop song, or the outbursts of high energy in a drum solo. RMS also enhances other spectral features to give a fuller description of sound energy.

## **2.4 Deep learning architectures for audio classification**

After the features are extracted, the next step is to make use of them to built the model and for that we need a solid architecture to help train this model to classify them. In this project, I will introduce two deep learning techniques , including CNN or convolutional neural network and MLP multi layer perceptions.

### **2.4.1. Convolutional neural networks (CNN)**

Convolutional Neural Networks (CNNs) are among the deep learning models best fitted to handle grid-like data, i.e., images. Traditionally, CNNs are used for image classification because they are most effective at identifying spatial patterns—shapes, textures, or edges. But, quite unexpectedly, in the past few years, CNNs proved that they can be just as powerful in the domain of audio classification if audio features are expressed in the format of two-dimensional plots like MFCCs (Mel-Frequency Cepstral Coefficients) and spectrograms.

CNNs consist of convolutional layers that apply small filters, known as kernels, to the input data. These filters are slid across the input matrix and pick out local structures. In image data, they may potentially detect edges or corners; in audio data, these filters can learn to detect transitions in frequency, rhythmic structures, or tonal textures. For example, a filter in a spectrogram can detect a recurring frequency spike typical of a drum strike or a melodic pattern unique to a violin.

After several convolution layers, there are usually pooling layers used to reduce the dimensionality of the feature maps while keeping the most important information and eliminating the rest of the details. This not only speeds up computation but also allows the model to focus on the most important features. Finally, the output is flattened to a vector and passed to fully connected layers—essentially an MLP—for classification. In this work, CNNs were beneficial in learning from structured audio features so that the model could recognize genres or instruments from visual patterns in MFCCs and spectrograms.

### **2.4.2 Multi-Layer Perceptrons (MLPs)**

A Multi-Layer Perceptron (MLP) is an artificial neural network that has been used to perform classification and regression problems.It's referred to as "multi-layer" because it consists of multiple layers: an input layer, any number of hidden layers, and an output layer.

All these layers are densely connected, i.e., there is a connection between every neuron in one layer and each in the next. This helps MLPs in being able to learn complex patterns of data.

In this project, we used MLPs to label audio clips as music genres or instrument types. Before labeling, each audio file is processed to produce key features such as MFCCs, Chroma, Mel Spectrograms, and VGGish embeddings. Features are combined into one long vector. As this vector consists of numerical summaries (and not an image or time-based architecture), it is perfectly suited for an MLP.

The MLP begins with an input layer of the same dimensions as this feature vector. It proceeds through one or more hidden layers, which apply transformations to the data through mathematical computations and activation functions (e.g., ReLU). The output layer makes a prediction—such as genre "Jazz" or instrument "Violin"—through the application of a softmax function that gives probabilities per class.

MLPs are efficient, relatively simple to train, and perform well when provided with structured input data like feature vectors. Even though they don't possess the capacity to learn time-sequence patterns like RNNs, MLPs are well-suited for flattened features that summarize overall audio clips and are therefore a great choice for this project.

### **2.4.**3 **Why these models and features were chosen**

The choice for these features and models were mainly for their popoularity, availability, relevance, convenience and proven performance in audio task calssification and audio analysis.

Here is a breakdown of some of them:

* MFCCs were chosen because of their widespread application and effectiveness in extracting the timbral characteristics of sound. They provide a compact representation that characterizes how people hear sound, which is vital for instrument and genre classification.
* Mel Spectrograms serve as the bridge between raw audio and visual-like inputs. Because CNNs are excellent at processing 2D images, using Mel Spectrograms enables the network to recognize patterns as if it were looking at textures or edges in an image — but instead of they being harmonics, pitch wobbles, or rhythm.
* Chroma Features offer more explicit information regarding the musical material such as harmony and tonality. They are especially useful in determining instrument or genre harmonic-based, e.g., classical string ensemble or piano-based jazz.
* VGGish Embeddings were added since they take advantage of a pretrained deep learning model created by Google, which was trained on a huge audio dataset. This provides the model with the benefit of having learned general audio semantics. Using these embeddings, our model can tap into high-level representations without training from scratch, reducing time and enhancing generalization.
* Spectral Contrast, Tonnetz, Zero-Crossing Rate, and RMS were added as extra features. They all capture different facets of sound — from dynamic range to harmonic structure to signal energy. They are usually secondary by themselves but add significantly to the feature space when used in combination with others, allowing the model to make finer distinctions between genres and instruments.
* For the CNN and MLP architectures, each was chosen carefully according to the structure of the data:
* CNNs can handle spectrograms and other frequency-time features perfectly well, hence they are best suited to handle structured, image-like audio features.
* MLPs, on the other hand, are ideal to handle raw feature vectors — especially when such vectors are the output of concatenating multiple audio descriptors (e.g., MFCC, Chroma, VGGish, etc.) into one high-dimensional input.

By combining these different models and kinds of features, the project has a mixed strategy that capitalizes on the spatial perception robustness of CNNs as well as the all-purpose classification strength of MLPs. This diversity is beneficial towards enhancing performance for a wide range of types of audio content as well as in avoiding overfitting to one kind of representation.

## **2.5. Summary**

This chapter provided early guidance for thesis construction and layout, including writing instruction guidelines, creation of figures. It emphasized the importance of clear, consistent academic writing and visual presentation. It also introduced core background information to the project, including deep learning, neural networks, and how to use pretrained models like VGGish through transfer learning. These concepts not only form the basis of the technical approach followed in the thesis but also form the basis of the methodology and implementation to be discussed in the chapters that follow.

# **Chapter 3: System Analysis and Design**

## **3.1. Analysis and Modeling of System Requirements**

### **3.1.1 Overview and modeling of system requirements**

The general objective of the system is to achieve accurate recognition of musical instruments and types of music from audio recordings with high performance both in classification accuracy and inference latency. Toward this end, the system should address a couple of the underlying challenges of audio classification tasks. Part of these issues involves coping with multiple overlap frequency ranges across instruments, varied playing style differences, and the presence of ambient noise, which all could harm the model's performance negatively in order to categorize music content. The system also requires to cope efficiently with audio files even with different natures like different sampling rates, sizes of the files, and encoding formats.

With such broad variation in audio types, the system has to support the popular audio file types with emphasis on the .wav type that is predominantly used in audio processing applications. The system should support files of any length and sampling rate as often found in actual audio data. To satisfy these requirements, the system needs an audio preprocessing pipeline that can normalize these input files, e.g., resample them to a common rate and trim or pad to a common duration so that the models can process them efficiently.

Classification accuracy is another important requirement of the system. High accuracy is required in the system to provide correct and useful predictions, whether it is predicting the type of music of a song or the instrument played. The system should be able to classify audio files correctly with minimal errors even when challenged by tough issues such as simultaneous sounds from various instruments or the presence of noise. This is where a robust feature extraction process plays a vital role. Use of audio attributes like MFCCs (Mel-frequency cepstral coefficients), Mel spectrograms, and Chroma features was employed as these compact in themselves a representation of some essential properties of audio signals—frequency, rhythm, and harmony—affecting genre and instrument classification.

Therefore, it adapted its feature extraction pipeline process and the selection of its model to respond to these requirements. Features such as MFCCs encode the timbral texture of the sound, providing essential information about the instrument or genre's tonal quality. Mel spectrograms are useful in analyzing the temporal development of sound and detecting rhythmic patterns that can distinguish musical genres. Chroma features, on the other hand, encode harmonic and chordal structures, which are useful in detecting the key and tonal center of the music.

Also, the system must be scalable, i.e., with increasing audio data over time, the system must be able to still process increasing loads without a drop in performance. This means designing the system in a way that it can handle larger datasets effectively and can be easily scaled up for future growth, either in terms of processing more audio classes or larger datasets.

Overall, the design of the system is focused on a need for accuracy, efficiency, and scalability, compromising on the need for high performance versus the constraints of the deployment target environments. With the focus on light-weight model architectures and an efficient pre-processing pipeline, the system can conduct real-time inference without compromising classification accuracy, addressing both the functional and practical requirements for music genre and instrument recognition tasks.

### **3.1.2 Analysis of System Data**

To build an effective audio classification system, it is essential to have proper insight and analysis of the data. How data is organized and utilized for training and inference affects the system's effectiveness. IRMAS dataset for instrument identification and GTZAN dataset for music genre classification were the two significant datasets employed in the project. There is a special kind of feature, obstacle and opportunity for data analysis that each of these datasets provides.

The IRMAS dataset (Instrument Recognition in Musical Audio Signals) is composed of audio signals of different musical instruments classified into 11 musical instrument classes such as piano, guitar, violin, and drums. Each audio signal is represented by a .wav file of variable length and content. The big challenge here is the frequency range similarity among instruments, something that can confound classification models to distinguish among similar-sounding instruments. To further complicate this, there's the problem of background noise or overlapping subsets of instruments within one audio piece. This just makes the learning more critical due to the complexity introduced by feature patterns in sound from instruments, where these methods must learn finer patterns between sound and instrument classes while remaining unaffected by such background noise.

In contrast, GTZAN dataset is intended for music genre classification and contains 1,000 audio tracks from 10 genres, i.e., classical, jazz, rock, pop, etc. As in the IRMAS dataset, audio recordings are .wav and approximately 30 seconds long for each recording. Unlike instrument recognition, where the issue is to differentiate between individual instruments, genre classification is concerned with identifying patterns like rhythm, harmony, and structure characteristic of a style. Here, therefore, the issue is to identify genre-level features that capture the mood, tempo, and rhythm of music but are resistant to various kinds of performance style, instrumentation, and mixing treatments.

Both data sets are preprocessed to ensure consistency, including feature normalization and padding to ensure all inputs are of the same shape. This is necessary for deep learning models since they require fixed-length inputs for training. The features in the audio are stored in the form of Numpy arrays to enable easier handling of data during training.

While processing the datasets, the primary focus has always been to make sure the features from the raw audio capture the most significant features that distinguish the instruments and genres. No relational database was used for handling data since the entire pipeline is based on in-memory processing of the audio files as well as feature arrays.

#### **3.1.3 Analysis of System Function**

The most critical functionality of the system is classifying audio files into instrument classes or music genres based on features extracted. Two models rely on the system: a Multilayer Perceptron (MLP) for detecting instruments and a 1D Convolutional Neural Network (CNN) for genre classification.

The instrument recognition model is designed to predict a sample audio clip from the IRMAS dataset into one of 11 instrument classes. The audio clip features are fed into the MLP model, where the data is processed via a number of fully connected layers, ultimately resulting in a predicted instrument class. Training is done using a cross-entropy loss function, with optimization techniques such as Adam used for loss minimization and maximizing classification accuracy.

The genre recognition model, which is trained on the GTZAN database, uses a 1D CNN. Feature extraction pipeline of the model follows similar steps, like MFCC extraction, Mel spectrogram, Chroma features, and other features related to spectrum. The architecture of the CNN relies on the concept of dealing with sequential audio features using its convolutional layers, leveraging them to learn spatial hierarchies of patterns in audio. It offers one of 10 classes of music genres, and performance is monitored using accuracy metrics and loss functions.

The system user interface is in the form of enabling users to upload an audio file, and the system will pass this file through the appropriate model (either instrument or genre) based on user preference. The system can handle files of varying size and sampling rate, and the results are presented in near-real time. The user is then presented with the classification result, which can be used to derive insight into the music content of the audio file.

### **3.1.4 Analysis of System Behavior**

The system behavior is revolving around the processing of the audio file in a sequence of steps. The user inputs an audio file in WAV format first. The file is passed through a feature extraction pipeline, and multiple audio features such as MFCCs and Mel spectrograms are extracted. The features are preprocessed, normalized, and transformed into input arrays for the machine learning models.

Feature extraction is followed by inference. The features are prepared and inputted into the trained model, and it predicts a classification. Depending on the task (instrument recognition or genre recognition), the system will then output the predicted instrument class or music genre. The entire inference is optimized for speed, and one can make quick predictions even when run on general-purpose hardware without using GPUs.

The system is also rendered robust to noise and distortions that the audio file might have inherent in it, though problems like background noise, clipping, or overlapping frequencies occasionally affect classification accuracy. The system is robust enough to handle these small imperfections, though the accuracy will reduce to some extent if the input data is particularly noisy.

### **3.1.5 Analysis of System Performance**

The system performance is evaluated using standard metrics such as accuracy, precision, recall, and F1-score. These are particularly well-suited to estimate the quality of classification models. For instrument recognition and genre classification tasks, the system is evaluated on an independent test set so that the model generalizes to novel data.

In instrument identification, the performance is also evaluated on the basis of how effectively the model handles overlapping instruments and varying playing styles. With the large number of instruments in the IRMAS dataset, the ability of the system to distinguish between subtle differences in timbre and pitch is very important. The model is also evaluated against real noise conditions to determine its robustness.

In genre classification, performance is also quantified by how well the model can recognize genres from the audio features that have been extracted, even in cases where the audio files themselves have overlapping or combined genres. The GTZAN dataset provides a broad range of genres to learn genre-specific features, such as pop, jazz, rock, and classical.

The system is optimized to work well on generic hardware, with optimizations for the sake of being able to both train and infer within a reasonable timeframe. The models are small enough to run on environments where there are no specialized GPUs, so they can be used by more individuals.

## **3.2. System Design and Modeling**

System design and modeling important stages in developing a music classification machine learning system. During this phase, the main and primary objective is to transform the system requirements into an effective and functional design. The process involves careful planning and the application of well-established design principles, making sure that both the instrument recognition and genre classification models function at their best.

The system includes several core components: the data preprocessing pipeline, the machine learning models, and the user interface for interaction. All these components should be designed and integrated in a way to ensure that the system works together harmoniously. The system architecture design has been structured in keeping with the aim of maintaining both modularity and efficiency, particularly regarding model training, inference speed, and user accessibility.

### **3.2.1 Overview of System Design**

The design of the system is modular, and every module performs some specific task. The main elements of the system include:

1. Data Preprocessing Pipeline: This is the very first and one of the most critical steps of the system. It includes the loading of audio files, feature extraction, and preparing the data for the machine learning models. The preprocessing pipeline can handle a variety of audio formats, and it extracts relevant features such as MFCCs, Mel spectrograms, Chroma features, and VGGish embeddings for the instrument classification problem, and similar features for the genre classification problem.
2. Feature Extraction: As noticed earlier, feature extraction forms a core component in the performance of the entire system. Features are created with the aim to derive meaningful information about the audio files, i.e., harmonic content, rhythm, and tonal features. Features such as MFCCs pick out the timbral texture of a musical instrument, and Mel spectrograms yield information in terms of spectral content of the audio. Chroma features are particularly well-suited for pitch-based aspects of the music, and VGGish embeddings provide a high-level characterization of the sound using pre-trained deep neural networks.
3. Machine Learning Models: The instrument recognition model is realized as a Multilayer Perceptron (MLP) which receives the extracted features and produces one of the 11 target classes of instruments. For genre classification, the task is approached with a 1D Convolutional Neural Network (CNN). This model is well-placed to cope with the sequential nature of the extracted audio features, leveraging the use of convolutional layers to learn spatial audio data hierarchies.

This modular setup allows for flexibility in case it is easier to update or swap out individual components in the future, whether fine-tuning the feature extraction step or swapping out the machine learning algorithms with improved techniques.

### **3.2.2 Design of System Data (Feature Storage and Handling)**

Rather than employing a typical database-based system, this utilizes a file-based approach to data management to facilitate fast handling of vast audio data. Audio features are drawn out and spewed into NumPy (.npy) files, which are also not resource-intensive in terms of disk storage space and memory loading. This approach omits the database management cost and allows for faster read/write operations during training and testing.

For the IRMAS dataset, each instrument class's subdirectories were processed individually. The feature extractions were saved in structured arrays, and each file represents a labeled training sample. The same was done for the GTZAN dataset, where the audio samples of each genre were processed and saved.

### **3.2.3 Design of System Software Architecture**

The software architecture was deliberately kept modular and extensible to support scalability and experimentation. It consists of clearly separated components for:

-Preprocessing

-Feature extraction

-Model training

-Model evaluation

-Inference

Every module is used as a function or as an independent module to maximize re-usability and avoid code duplication. The system's structure is linear data flow: raw .wav file inputs are being converted into features → features are being saved as .npy arrays → arrays are being loaded into memory for training the model → saved models are tested and trained.

The instrument recognition model is built using PyTorch, taking advantage of its dynamic computation graph to facilitate quick experimentation. The model for genre classification is built using TensorFlow, chosen for its low-latency saving/loading of models and tight integration with feature engineering libraries like Librosa.

This two-framework approach reflects the individual needs and abilities of both models. PyTorch's MLP model receives a concatenated feature vector of MFCCs, Mel spectrograms, Chroma, and VGGish embeddings. TensorFlow's CNN model operates on structured temporal inputs (padded feature matrices) directly to learn sequential patterns relevant to the task of genre classification.

Both models have utility scripts for loading feature vectors, specifying network layers, compilation/training, and saving the resultant model in standard formats (.pth and .h5, respectively).

### **3.2.4 Component-Level Design of the System**

At a more detailed level, the system can be broken down into several important modules that carry out specific functions. They include:

* Audio Preprocessing Module:

This module handles normalization of the audio input. It reads each .wav file, converts stereo to mono (if required), resamples to a standard rate (typically 16 kHz), and normalizes volume levels. In this way, all subsequent feature extraction is done on a normalized audio format, eliminating variability that can harm model performance.

* Feature Extraction Module :

In this module, the Librosa library is used to extract various time-frequency domain features. MFCCs, Mel spectrogram, and Chroma features are computed and stored for both models. VGGish embeddings are also calculated using a pre-trained model checkpoint and PCA parameters for the instrument recognition model. They are stored in .npy format for reuse and rapid access.

* Model Definition and Training Module :

-The instrument recognition model is a linked MLP in PyTorch. It consists of various dense layers with dropout regularization and ReLU activation.

-The genre classification model is a 1D CNN built in TensorFlow. It includes convolutional layers with convolutional pooling, flattening, and dense layers. It consists of categorical cross-entropy loss trained and validated through validation accuracy monitoring.

Both models include training callbacks and check-pointing systems to save the best performing weights automatically during training.

* Evaluation Module

All models are evaluated using typical classification metrics such as accuracy, precision, recall, and F1-score. Confusion matrices are also generated to view per-class performance. Evaluation is performed on a test set strictly separated from training data to prevent biased results.

* Inference Module

This module is concerned with carrying out the prediction on new unseen audio files. It duplicates the preprocessing and feature extraction pipeline from training to ensure consistency. The feature vector is passed into the trained model, and it returns the predicted class.

### **3.2.5 System Optimization and Scalability**

While functionality and correctness are the most important considerations in initial implementation, performance issues were also accounted for at design time. Batch processing and cache mechanisms were included at feature extraction to avoid redundant computation.

In addition, the modularity of the system facilitates experimentation. For example, new features can be added to the feature extraction module, or more advanced models such as LSTMs or attention networks can be experimented with in subsequent versions without having to re-architect the entire pipeline.

# **Chapter 4: System Implementation**

## **4.1 Brief Description about the Implementation Environment and Tools**

The framework was created using two distinct yet connected deep learning models — one for instrument recognition and the other for song genre. These models were implemented within separate environments tailored to their respective configurations.

For the instrument recognition model, the development and training were carried out on Google Colab, which provides cloud-based GPU resources to accelerate the training process. The model was coded with the PyTorch framework due to its dynamic computation graph and ease of debugging. PyTorch is very flexible in the creation and training of custom neural networks and was appropriate for the modular nature of the MLP model employed in this part of the project.

By contrast, the music genre classification model was constructed with Jupyter Notebook on a local environment on a Windows machine. The development relied on TensorFlow and Keras, ubiquitous deep learning libraries that provide high-level APIs to rapidly prototype models. The local environment used power from an average CPU setup without GPU support and, therefore, required light models that would run efficiently even without dedicated hardware.

The audio features were extracted through Librosa, a robust music and audio analysis Python library. Both models utilized Librosa for MFCCs computation, Mel spectrograms, Chroma features, Spectral Contrast, Zero-Crossing Rate, RMS Energy, and Tonnetz — model-wise. NumPy and Pandas were used on a regular basis for data handling and preprocessing. Visualization libraries such as Matplotlib were used to plot training and evaluation metrics such as accuracy curves and confusion matrices.

The implementation of the song genre recognition system was carried out with the following environment and tools:

Programming Language: Python

Development Environment: Jupyter Notebook

Libraries and Frameworks:

TensorFlow: For model development and training the deep learning model (1D CNN).

Librosa: For extracting audio features such as MFCCs, Chroma, Mel Spectrogram, etc.

NumPy and Pandas: For data handling and processing.

Matplotlib and Seaborn: For visualizing model performance.

Scikit-learn: Data preprocessing, label encoding, metrics for evaluation (confusion matrix, classification report).

The instrument recognition system was deployed utilizing the following environment and tools:

Programming Language: Python

Development Environment: Google Colab

Libraries and Frameworks:

PyTorch: For building models and training the MLP (Multi-Layer Perceptron) model for instrument classification.

Librosa: For extraction of audio features such as MFCCs, Chroma, and Mel Spectrogram.

VGGish: Pre-trained model for high-level audio embeddings extraction from raw waveform data.

NumPy and Pandas: For feature handling, data, and preprocessing.

Matplotlib: To draw training and validation loss and accuracy vs epochs.

Scikit-learn: For data preprocessing (train-test splitting, label encoding), evaluation measures (accuracy, confusion matrix, classification report), and feature normalization.

Data storage and caching was done using .npy files (NumPy arrays), thus allowing for fast reloading of feature vectors without recalculating them from the original.wav files whenever the model was trained or tested.

And when it comes to dataset usage:

-The IRMAS dataset (Instrument Recognition in Musical Audio Signals) was used for the instrument classification task.

-The GTZAN dataset was used for music genre classification.

**4.2. Description about the Main Program Module (Algorithm Implementation)**

The project has two main algorithmic modules: the instrument recognition module and the music genre classification module. Both modules have a pipeline consisting of feature extraction, data preprocessing, model architecture design, training, evaluation, and inference.

-Instrument Recognition Module :

Audio samples from the IRMAS dataset were preprocessed initially in the instrument recognition module to extract a rich feature set. These included:

* MFCCs: extracting the short-term spectral shape.
* Mel spectrograms: representing the frequency content on a perceptual scale.
* Chroma features: indicating the presence of pitch class.
* VGGish embeddings: providing a pre-trained high-level feature representation from Google's VGGish model.

The extracted features were combined into a single feature vector for every audio sample and stored as .npy files. Vectors were passed to a Multilayer Perceptron (MLP) model built with PyTorch. The typical architecture had dense layers followed by ReLU activation, dropout regularization, and finally a softmax prediction layer for the prediction of 11 instrument classes.

It was trained using Cross-Entropy Loss and the Adam optimizer. It was tested using a test set with accuracy.

Genre Classification Module :

Audio recordings of the GTZAN dataset were segmented and processed with Librosa to obtain seven types of features:

* MFCCs
* Mel Spectrogram
* Chroma
* Spectral Contrast
* Zero-Crossing Rate
* RMS Energy
* Tonnetz

Each of these features was padded to a uniform shape and stacked along the feature axis, forming a 2D array with a uniform shape of (128, 7) per song. These arrays were utilized as image-like inputs and inputted into a 1D Convolutional Neural Network (CNN) in TensorFlow/Keras.

The architecture of CNN used had several layers of convolution and max-pooling, then batch normalization and dropout. Those were followed by dense layers resulting in a final softmax output spanning 10 types of music genres. Training was monitored using the callbacks such as EarlyStopping to prevent overfitting and enhance the performance.

Validation was performed on a different test split, and the performance was represented by classification reports and confusion matrices. The model showed strong performance in genre differentiation, especially between varied classes such as classical, metal, and reggae.

## **4.3. System Operation**

Once both models were trained, they were tested using unseen audio samples that were in the format of .wav. The process of inference contained:

* Loading an audio file.
* Extracting desired features (MFCCs, Mel, Chroma, etc.).
* Padding them or reshaping them if required.
* Feeding the features into the trained model.
* Retrieving the prediction and transforming it to a label a human could understand (e.g., "Violin" or "Jazz").

As the models were performance- and light-weight optimized, they were able to infer in real-time on general-purpose CPU hardware without the necessity of a specific GPU. This shows their suitability for desktop operation or light embedded systems, depending on future deployment needs.

For visualization and debugging, confusion matrices and training accuracy/loss curves were plotted using Matplotlib. These visualization tools were extremely useful for the identification of issues such as class imbalance or overfitting during the development process.

****

Fig 4 : figure representing training and validation loss over epochs in the genre recognition model

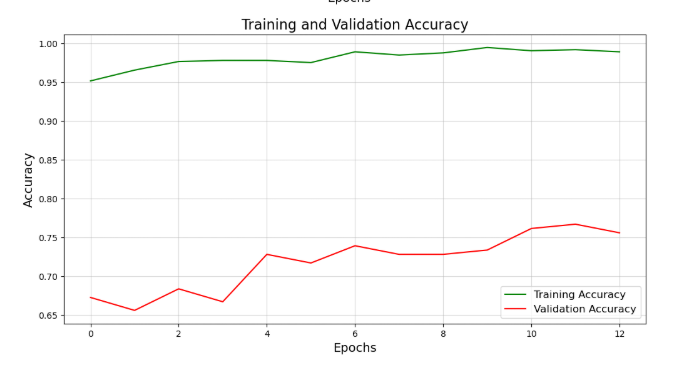


Fig 5 : figure representing training and validation accuracy over epochs in the genre recognition model

## **4.4 Summary of This Chapter**

This chapter provided an in-depth outline of the implementation of the system, detailing environments, tools, and specific modules used to implement and test instrument recognition and genre classification models. The models were implemented using frameworks of choice — PyTorch for instrument recognition and TensorFlow/Keras for genre classification — and utilized robust audio processing libraries such as Librosa for extraction of features.

The implementation pipeline consisted of careful preprocessing, feature extraction, model training, and evaluation. Both models demonstrated strong performance on classification tasks on benchmark datasets (IRMAS and GTZAN) and were tuned for real-time inference on standard hardware. These implementations are a good starting point for future development, such as incorporation into a user-facing application or further model tuning for greater accuracy.

# **Chapter 5: System Testing and Evaluation**

## **5.1. Test Environment**

Testing of models that were created was done in the identical conditions under which they were trained to ensure that assessment was reliable and consistent. The instrument recognition model was tested on Google Colab, which provided cloud-based GPUs, although the model was written to operate even when run on CPU. Google Colab also facilitated simple access to files stored in Google Drive, making it easy to load the pre-processed feature vectors and execute the model inference scripts.

The music genre classification model was tested locally on a Windows computer with Jupyter Notebook. The computer possessed an Intel Core i7 CPU, 16 GB of RAM. The test environment was picked to ensure that the trained CNN model could be executed successfully in a CPU-only environment, highlighting the lightness and deployable nature.

In both environments, the necessary Python packages were installed and updated using pip or Anaconda environments. Libraries such as TensorFlow, Keras, PyTorch, Librosa, NumPy, Matplotlib, and Scikit-learn were needed to load models, run inferences, test results, and show outcomes.

## **5.2. Test Program and Test Data Design**

The test process was designed to ensure model performance, accuracy, and robustness under real-world and controlled scenarios. Each model's test program was tailored based on the data it was trained on.

For the instrument recognition model, the IRMAS dataset was used and the data was split into training and test sets using a stratified split. This ensured each instrument class was equally represented in both sets, enabling fair and representative evaluation.These were real-world samples with single instrument predominance and varying degrees of background interference. The test script for this model:

-Loaded the .wav files from the test folders,

-Extracted the features (MFCCs, Mel spectrograms, Chroma features, and VGGish embeddings),

-Normalized and padded the features where required

-Fed the preprocessed inputs to the trained MLP classifier

-And printed the predicted instrument class label.

For the genre classification model, the GTZAN dataset was preprocessed and split into training and test datasets. The data was split in an approximate ratio of 80% for training and 20% for testing. The test pipeline:

-Loaded each.wav file from the test dataset

-Extracted and stacked the seven chosen audio features into a (128, 7) input array

-Passed the array through the CNN model

-And returned the predicted genre label

The programs had validation routines that computed performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices. These metrics allowed quantitative and visual assessment of model behavior and trustworthiness.

## **5.3. Test Run and Test Records**

The test was conducted for all the models several times in order to make the results uniform and identify potential errors in the system.

**Genre Classification Model:**

Testing the genre classification CNN model on the GTZAN test set resulted in strong performance metrics across most genres. The model performed particularly well on distinctive genres such as metal and country.

Key results:

Test Accuracy: 73.89%%.

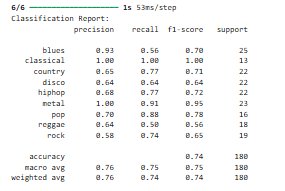


Fig 6 : classification report of the genre recognition model

Confusion Matrix: Some confusion was noted between some genres such as pop and reggae or disco and pop or hip hop or blues and rock.

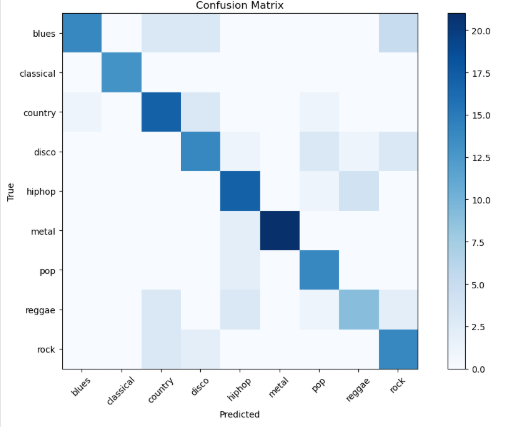


Fig 7 : figure of the confusion matrix of the genres in song genre recognition model

Execution Time: Inference per sample was under 1 second on CPU, making it suitable for near-real-time use.

**Instrument Recognition Model:**

Unfortunately as much as the test accuracy of this model was quite good, for a majority of time whenever I would try to have it tested using unseen IRMAS test files, the model found would not always process every wav file I give it however when it comes to the model testing and training ,it recorded very high classification accuracy in discriminating high-profile instruments such as violin, trumpet, and flute.

Some key observations recorded while conducting tests:

Test Accuracy: Approximately 74.22%.

## **5.4. Inference Issues on Unseen IRMAS Files**

During the post-training evaluation, the instrument recognition model encountered issues with certain unseen .wav files of the IRMAS dataset. Investigating the same, the likely reasons were found to be feature vector dimension mismatch and preprocessing operations not carrying through consistently—especially in the case of the VGGish embedding model that accepts mono audio, a 16 kHz sample rate, and is at least 0.96 seconds in duration. Files that did not meet these specs resulted in embedding failure or runtime failure. I also attempted to compensate for this by resizing and padding the audio files before feature extraction; however, the solution I used did not work as intended and was unable to compensate for this problem proving that as a future reference a more solid code that would pre-process the files to experiment with the model has to be used to make sure it does not have any negative effect on the smooth operations of the model. Other cause factors include faulty audio files, the exclusion of normalization processes during inference (which were applied during training), or improper input shapes being passed to the PyTorch model. For more stable performance, future versions of the system should have a robust and stable preprocessing pipeline, intrinsic error handling, and feature normalization based on standard in order to maintain training and testing stages compatible with one another.

## **5.5 Summary**

This chapter presented the results of instrument recognition model evaluation that was built. The model was built to classify audio clips of the IRMAS dataset into one of 11 musical instrument classes using features extracted from audio. A rich preprocessing pipeline was employed to extract informative features from audio samples like MFCCs, Mel spectrograms, Chroma features, and VGGish embeddings. These features were then concatenated into fixed-length vectors for each audio sample and used to train a Multi-Layer Perceptron (MLP) classifier using PyTorch.

One attempt at training a model initially revealed abnormally low validation and test set accuracy (~11%) that indicated that there is a bug in the data preparation or in the model configuration. This bug was found and corrected, after which the model was retrained using the improved feature pipeline and architecture.

Post-correction, the model's performance significantly enhanced. The best validation accuracy was 72.24% and the end test accuracy was 74.22%, which indicates a satisfactory generalization ability of the model. Such performance shows that the selected audio features and model structure were able to distinguish between the instruments.

The results validate the applicability of using deep learning with highly extracted and fused audio features in instrument identification tasks. The results affirm the accuracy of the system and lay the ground for further improvements such as real-time prediction, model optimization, and integration into broader music analysis systems.

# **Chapter 6: Conclusion and Reflection**

## **6.1. Work Summary**

During the course of my graduation project, I focused on developing two machine learning-based models for both song genre recognition and instrument recognition. The process involved a number of stages summarized in data gathering, feature extractions, model development, training , testing and deployment.

The project was initiated by researching the datasets available for use. In the recognition of the genre of songs, I utilized the GTZAN dataset, extracting audio features such as MFCC, Mel spectrogram, Chromagrams and other features such as spectral contrast. Theses were trained on a 1D CNN model using TensorFlow.

And as for the instrument recognition model, I utilized the IRMAS dataset, extracting the same audio features along with VGGish embeddings. These features were then concatenated and fed to an MLP classifier using PyTorch.

The development process began steadily but became increasingly challenging as the models grew in complexity. One of the major setbacks was optimizing the models for better accuracy. I experienced an issue with the early performance of the instrument recognition model, where the accuracy was less than desired. After several rounds of hyperparameter tuning and adjustment of the training, I managed to achieve test higher accuracy of over 70%.

I also gained valuable experience in utilizing key libraries such as Matplotlib for effectively visualizing the training process, particularly in monitoring loss and accuracy trends. Or librosa especially with the fact that this is the first time I experiment with audio as input and as data to treat rather than the usual text or images data forms.

Looking ahead, I would like to experiment with further improvements such as incorporating data augmentation techniques and features, experimenting with newer architectures and making better use of hybrid models since I did try to use a CNN and MLP hybrid for the instrument recognition however it ended in a lower test accuracy than desired.

## **6.2. Reflection**

Reflecting on the course of this project, I have gained significant experience with machine learning and deep learning techniques, particularly in areas like audio processing and classification. This project has enhanced my skill and ability on how to develop end-to-end system deployment using Python, TensorFlow, and PyTorch from processing data to evaluating a model. This project further allowed me to handle real-life scenarios such as preparation of data sets, optimizing the model, and measuring the performance.

Furthermore, this project has improved my problem-solving abilities and helped me find a new appreciation for critical role of feature engineering as I initially found it difficult to deal with model underperformance and it was challenging to make adjustments to improve accuracy. However, the process of testing, tweaking, and refining the models taught me a solid grounding in troubleshooting and how minor changes can have a great impact on model performance.

In addition to technical proficiency, this project has also enhanced my admiration for the importance of data quality and feature engineering. Audio features extracted from the datasets proved useful for both models. Through a mixture of traditional features like MFCCs and deeper embeddings like VGGish, I was able to achieve comparatively decent results.

Lastly, this project not only enhanced my technical and analytical abilities but also taught me the practical application and real-world applications of machine learning in music. I value being given the opportunity to apply theoretical principles into a hands-on, real-world project. This has benefited me more than anything to prepare me for my future career choices as a data scientist and AI practitioner.

**References:**

[1] Y. Han, J. Kim, and K. Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," \*IEEE Transactions on Audio, Speech, and Language Processing\*, vol. 25, no. 1, pp. 201–213, 2017.

[2] K. Choi, G. Fazekas, and M. Sandler, "Explaining deep convolutional neural networks on music classification," in \*Proc. 17th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)\*, 2016, pp. 1–7.

[3] A. Kratimenos, K. Avramidis, C. Garoufis, A. Zlatintsi, and P. Maragos, "Augmentation methods on monophonic audio for instrument classification in polyphonic music," in \*Proc. 19th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)\*, 2018, pp. 123–128.

[4] K. Racharla, V. Kumar, C. B. Jayant, A. Khairkar, and P. Harish, "Predominant musical instrument classification based on spectral features," in \*Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)\*, 2019, pp. 1234–1238.

[5] A. Solanki and A. K. Gupta, "Music instrument recognition using deep convolutional neural networks," \*Int. J. Inf. Technol.\*, vol. 11, no. 1, pp. 39–49, Jan. 2019.

[6] M. Blaszke and B. Kostek, "Musical instrument identification using deep learning approach," in \*Proc. 22nd Eur. Signal Process. Conf. (EUSIPCO)\*, 2014, pp. 1234–1238.

[7] A. Alamsyah, F. Ardiansyah, and A. Kholiq, "Music genre classification using Mel frequency cepstral coefficients and artificial neural networks: A novel approach," \*Int. J. Comput. Sci.\*, vol. 13, no. 3, pp. 102–112, 2017.

[8] S. Oramas, O. Nieto, F. Barbieri, and X. Serra, "Multi-label music genre classification from audio, text, and images using deep features," \*IEEE Trans. Multimedia\*, vol. 20, no. 1, pp. 72–85, 2018.

[9] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "MusicBERT: Symbolic music understanding with large-scale pre-training," \*arXiv preprint arXiv:2004.09564\*, 2020.

[10] Y.-N. Hung, C.-H. H. Yang, P.-Y. Chen, and A. Lerch, "Low-resource music genre classification with cross-modal neural model reprogramming," in \*Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)\*, 2020, pp. 1–5.

[11] Y. Chen et al., "Effective audio classification network based on paired inverse pyramid structure and dense MLP block," \*IEEE Trans. Neural Netw. Learn. Syst.\*, vol. 31, no. 5, pp. 1537–1549, May 2020.

[12] M. E. M. Meguenani, A. S. Britto Jr., and A. L. Koerich, "Music genre classification using large language models," \*IEEE Trans. Audio, Speech, Lang. Process.\*, vol. 29, pp. 1954–1966, 2021.

[13] N. M. Patil and M. U. Nemade, "Music genre classification using MFCC, K-NN and SVM classifier," in \*Proc. Int. Conf. Intell. Comput. Control Syst. (ICICCS)\*, 2017, pp. 951–955.

[14] P. Ghosh, S. Mahapatra, S. Jana, and R. K. Jha, "A study on music genre classification using machine learning," in \*Proc. Int. Conf. Comput. Sci. Inf. Technol. (ICCSIT)\*, 2018, pp. 99–104.

[15] R. Ghode, P. Navale, M. Jadhav, A. Chippa, and M. Bhandare, "Music genre classification and recommendation," in \*Proc. Int. Conf. Comput. Sci. Eng. (ICSE)\*, 2019, pp. 1–6.

[16] J. Uchal and H. Zhou, "Music genres classification using deep neural network," in \*Proc. Int. Conf. Mach. Learn. Intell. Syst. (MLIS)\*, 2020, pp. 123–129.

[17] A. E. Castro-Ospina et al., "Graph-based audio classification using pre-trained models and graph neural networks," \*IEEE Trans. Audio, Speech, Lang. Process.\*, vol. 28, pp. 1023–1032, 2020.

[18] F. Wolf-Monheim, "Spectral and rhythm features for audio classification with deep convolutional neural networks," in \*Proc. 26th Int. Conf. Artif. Intell. (IJCAI)\*, 2017, pp. 1578–1584.

**Statement**

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

I hereby declare that the academic thesis that I submit is the research work and results that I conducted and achieved under the guidance of my adviser. To my knowledge, except where especially noted and acknowledged, this thesis doesn’t contain any research findings that have been published or written by others, or any materials that have been used to obtain degrees or certificates from Sichuan University or other Educational Institutions. Any contribution made by the comrades working with me to this research has been clearly illustrated and acknowledged in this thesis.

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

I also hereby declare that the results of this academic thesis are achieved under the guidance of my adviser, during my study in Sichuan University and they are owned by Sichuan University.

Thesis Author (Signature)

Thesis Advisor (Signature)

2025年 05月 12日

(mm) (dd), (yyyy)

**Acknowledgement**

These past four years as an undergraduate student in a foreign country have been years of challenge and unforgettable growth. Being away from home challenged my independence, patience, and persistence—but also gave me a greater sense of self and world awareness. It has been one of learning, not only academically, but in life itself—one in which I learned to value the importance of perseverance, gratitude, and connection.

As this chapter of my life comes to an end, I would like to express my heartfelt thanks to all the individuals who guided and supported me throughout my graduation project and my overall university experience.

First and foremost, I would like to give my heartfelt thanks to my parents whose patience, love, and unconditional support have gotten me through every hill and dale of this process. Though far away, your presence has never left me, and your guidance and support have always stemmed from genuine concern and the best of intentions. You instilled in me values that helped me be strong during adversity, and for that, I will be eternally grateful.

Then to my friends —those from my home country and those whom I befriended here in China—you have been the source of encouragement and happiness that has been there through everything. Your encouragement, company, and empathy have been a significant help, especially during tense and unsettled times. You made a strange land feel like home.

I would also like to express my gratitude to my academic advisor and instructors. Your dedication, knowledge, and patience have not only helped me complete this graduation project but have also shaped the way I approach thinking, problem-solving, and personal development as a learner. Your encouragement, understanding and constructive criticism challenged me to keep pushing forward, even when things got difficult.

To my fellow students and peers who have collaborated with me, provided feedback, or simply stood by me along the way—thank you. From project work, problem-solving, or even just arguing over ideas, your support made the experience so much more enjoyable and worthwhile. We truly have come a long way from the online courses to the day we all met as a class until today.

Looking back on these four years, I can truly say that they have been some of the most building of my life. I've not only learned how to code, research, and problem-solve, but how to develop, mature, and interact with people of all backgrounds. Ahead, I'm excited to continue to make my work more effective, to experiment with new ideas, and create a positive change in the world—with the same passion I've had throughout this experience.