Music Genre Classification Using Deep Learning

A PROJECT REPORT

Submitted by

Anup (20BCS3923)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

AIT – BIG DATA ANALYTICS

Chandigarh University

November 2023

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report "**Music Genre Classification Using Deep Learning**" is the bonafide work of "**Anup (20BCS3923)**" who carried out the project work under my/our supervision.

SIGNATURE	**SIGNATURE**
Dr. Aman Kaushik **HEAD OF DEPARTMENT** Computer Science and Engineering	Mr. Pulkit Dwivedi Assistant Professor **PROJECT SUPERVISOR** Computer Science and Engineering

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, praises and thanks to the God, the Almighty, for his showers of blessings throughout our project work to complete the project successfully on time. We would like to express our deep and sincere gratitude to our Project supervisor, "**Mr. Pulkit Dwivedi**" and our Project Evaluation Panelists, "**Mrs. Monika Luthra**" for giving us the opportunity to do this project and providing invaluable guidance throughout this Project. His dynamism, vision, sincerity and motivation have deeply inspired us. He has taught us the methodology to carry out the Project and to present the Project works as clearly as possible. It was a great privilege and honour to work and study under her guidance. We are extremely grateful for what he has offered us. We would also like to thank her for her friendship, empathy, and great sense of humor. We extremely grateful to Our parents as well for their love, prayers, caring and sacrifices for educating and preparing us for our Future.

TABLE OF CONTENTS

BONAFIDE CERTIFICATE ....

ACKNOWLEDGEMENT ....

LIST OF TABLES ....

LIST OF FIGURES ....

ABSTRACT ....

GRAPHICAL ABSTRACT ....

ABBREVIATIONS ....

SYMBOLS ....

1. Introduction **1**

1.1 What is Machine Learning? 1

1.2 Some Machine Learning Methods 2

1.3 Music Genre 3

1.4 The Importance of Music Genre Classification 3

1.5 Applications of Music Genre Classification 5

1.5.1 Playlist Generation 5

1.5.2 Music Recommendation System 6

1.5.3 Artificial Neural Network 7

1.5.4 Deep Neural Network 8

1.6 Challenges 9

1.7 Deep Learning 10

1.7.1 Applications of deep learning 11

1.7.1.1 Automatic Speech Recognition 11
1.7.1.2 Image Recognition 11
1.7.1.3 Natural Language Processing 12
1.7.1.4 Recommendation Systems 12
1.7.1.5 Bioinformatics 12
1.8 Objective 13
2. LITERATURE REVIEW/ BACKGROUND STUDY 14
2.1 Timeline of the reported problem 14
2.2 Existing Solution 16
2.3 Bibliometric analysis 18
2.4 Review Summary 19
2.5 Problem Definition 20
2.6 Goals/Objectives 20
3. DESIGN FLOW/ PROCESS 22
3.1 Concept Generation 22
3.1.1 MFCC 22
3.1.1.1 Chroma Frequencies 23
3.1.1.2 Spectral Centroid 23
3.1.1.3 Zero Crossing Rate 24
3.1.1.4 Spectral Roll-off 24
3.1.2 CNN 25
3.1.2.1 Convolutional 26
3.1.2.2 Pooling 26
3.1.2.3 Weights 26
3.1.2.4 Convolutional layer 26
3.1.2.5 Local connectivity 27
3.1.2.6 Loss layer 28

3.1.3 KNN 29

3.1.4 SVM 30

3.1.4.1 Kernel 31

3.1.4.2 Regularization 31

3.1.4.3 Gamma 31

3.1.4.4 Margin 31

3.1.4.5 Application 31

3.2 Evaluation & Selection of Specifications/Features 33

3.3 Design Constraints 34

3.4 Analysis of Features and finalization subject to constraints 36

3.5 Design Flow 38

3.6 Design selection 41

3.7 Implementation plan/methodology 42

4. RESULTS ANALYSIS AND VALIDATION 44

4.1 Implementation of solution 44

4.2 Result 47

5. CONCLUSION AND FUTURE WORK 51

5.1 Conclusion 51

5.2 Future Work 52

6. REFERENCES 53

USER MANUAL 55

List of Figures

Fig-1...4

Fig-2...5

Fig-3...10

Fig-4...22

Fig-5...27

Fig-6...28

Fig-7...29

Fig-8...30

Fig-9...47

Fig-10...47

Fig-11...48

Fig-12...48

Fig-13...49

Fig-14...49

Fig-15...50

Fig-16...50

ABSTRACT

Music genre classification plays a crucial role in various applications, such as music recommendation systems, content organization, and personalized user experiences. This abstract presents a study on the application of deep learning techniques for the automatic classification of music genres. The proposed approach leverages the power of neural networks to automatically learn relevant features from audio signals, enabling accurate and efficient genre classification.

The methodology involves the preprocessing of audio data to extract meaningful features, such as spectrograms or Mel-frequency cepstral coefficients (MFCCs). These features serve as input to a deep learning model, specifically a Convolutional Neural Network (CNN) or a Recurrent Neural Network (RNN), which is designed to capture both local and temporal dependencies within the audio data.

To ensure the model's robustness and generalization, a comprehensive dataset comprising diverse music genres is utilized for training and evaluation. The model is trained using labeled examples, allowing it to learn the intricate patterns and characteristics associated with different genres.

The evaluation of the proposed approach involves assessing the model's accuracy, precision, recall, and F1-score on a separate test set. Comparative analyses are conducted with traditional machine learning techniques, highlighting the superior performance of deep learning in handling complex audio data for genre classification.

Furthermore, the study explores the impact of hyperparameter tuning, network architecture variations, and the potential integration of transfer learning techniques to enhance the model's performance. The results showcase the effectiveness of deep learning models in capturing intricate patterns within audio data, leading to improved accuracy and robust genre classification.

In conclusion, this research demonstrates the viability of leveraging deep learning for music genre classification, offering a promising avenue for advancing the capabilities of automated music content analysis systems. The findings contribute to the growing field of music information retrieval and pave the way for more sophisticated and accurate applications in music recommendation and organization.

GRAPHICAL ABSTRACT

The graphical abstract illustrates the process of music genre classification using deep learning. At the center is a stylized neural network icon, depicting the core of the deep learning model with nodes and connections representing the learning process. On the left side, an audio waveform symbolizes the raw input data, and an arrow points towards the neural network, indicating the data flow. Conversely, on the right side, icons representing different music genres, such as rock, jazz, and hip-hop, highlight the diversity of inputs.

The top section showcases icons of spectrograms or Mel-frequency cepstral coefficients (MFCCs) representing the data preprocessing step, with arrows leading from the audio waveform to these feature representations. The bottom section features icons representing evaluation metrics like accuracy, precision, recall, and F1-score, along with comparative symbols indicating a comparison with traditional machine learning methods.

Surrounding elements include icons representing hyperparameter tuning, such as sliders and gears, and symbols for transfer learning to signify potential integration. The background theme incorporates subtle musical elements like notes and symbols, maintaining a cohesive music-related aesthetic. The main title, “Deep Learning for Music Genre Classification,” is accompanied by brief captions for each section, providing a comprehensive visual representation of the key stages in the music genre classification process using deep learning.

ABBREVIATIONS

Here are some potential abbreviations for a paper on music genre classification using deep learning:

1. MGC-DL: Music Genre Classification using Deep Learning
2. NN: Neural Network
3. CNN: Convolutional Neural Network
4. RNN: Recurrent Neural Network
5. MFCC: Mel-frequency cepstral coefficients
6. DL: Deep Learning
7. AR: Audio Representation
8. DS: Dataset
9. FM: Feature Extraction Methods
10. TM: Transfer Learning Methods
11. HM: Hyperparameter Tuning
12. EM: Evaluation Metrics
13. TM: Traditional Methods
14. CAD: Content Analysis and Classification
15. MRI: Music Recommendation Integration
16. Uex: User Experience
17. CAE: Comparative Analyses and Evaluation
18. PRF1: Precision, Recall, F1-score (metrics)
19. ACC: Accuracy
20. GC: Genre Classification

These abbreviations can be used throughout the paper to enhance readability and conciseness while still conveying the key concepts in music genre classification using deep learning

SYMBOLS

Using Greek letters like alpha (α), beta (β), and gamma (γ) can be a symbolic and concise way to represent different aspects of music genre classification. While these symbols are not traditionally associated with music genres, they can be creatively employed to categorize or represent various attributes related to the classification process. Here's how you might use them:

Alpha (α):
Representation: α
Significance: Alpha could symbolize the primary or foundational music genres in the classification system. For example, it might represent the classical genre, which serves as a cornerstone in music categorization.

Beta (β):
Representation: β
Significance: Beta might be used to represent evolving or experimental genres that are still in the development or exploration stage. Genres like electronic or fusion music could be symbolized by beta.

Gamma (γ):
Representation: γ
Significance: Gamma could signify diverse or less mainstream genres that bring a unique and unconventional flavor to the music landscape. This might include genres such as world music or avant-garde.

Using these symbols strategically throughout the paper can visually represent concepts and make the content more engaging for readers. Ensure that the symbols align with the context and contribute to the overall clarity of your presentation.

CHAPTER – 1
INTRODUCTION

1.1 What is Machine Learning?

Machine learning is when a computer uses an algorithm to understand the data it has been given and recognizes patterns. We call the process of learning “training,” and the output thatthis process produces is called a “model.” A model can be provided new data and reason against it based on what the model has previously learned. Machine learning models determine a set of rules using vast amounts of computing power that a human brain would beincapable of processing. The more data a machine learning model is fed, the more complex the rules and the more accurate the predictions. Whereas a statistical model is likely to have an inherent logic that can be understood by most people, the rules created by machine learningare often beyond human comprehension because our brains are incapable of digesting and analyzing enormous data sets.

The process of learning begins with observations or data, such as examples, direct experience,or instruction, to look for patterns in data and make better decisions in the future based on theexamples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

1.2 Some Machine Learning Methods

Machine learning algorithms are often categorized as supervised or unsupervised.

- **Supervised machine learning** algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system can provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors to modify the model accordingly.
- **Unsupervised machine learning** algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies howsystems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
- **Semi-supervised machine** learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. Thesystems that use this method can considerably improve learning accuracy. Usually, semi- supervised learning is chosen when the acquired labeled data requires skilled and relevant resources to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.
- **Reinforcement machine learning** algorithms is a learning method that interacts withits environment by producing actions and discovers errors or rewards. Trial and error searchand delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information

1.3 Music Genre

A **music genre** is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions. It is to be distinguished from *musical form* and *musical style*, although in practice these terms are sometimes used interchangeably. Recently,academics have argued that categorizing music by genre is inaccurate and outdated. Musicalgenres are categorical labels created by humans to characterize pieces of music. A musical genre is characterized by the common characteristics shared by its members. Jazz, rock, blues, classical.. These are all music genres that people use exten- sively in describing music. Whether it is in the music store on the street or an online electronic store such as Apple's iTunes with more than 2 million songs, music genres are one of the most important descriptors of music.Genre hierarchies are commonly used to structure the large collections of music available on the Web.

1.4 The Importance of Music Genre Classification

The topic of the importance of music genres comes up often in country music, especially overrecent years (and recent days) as we've seen the encroachment of other genres into country music like never before. On paper, there's nothing wrong with combining two or more genresof music to create something unique. The problem is that often when this enterprise is undertaken, it's not creating something unique, it's meant to mimic something that is patentlysimilar to everything else being released in popular music, making it even more uninterestingand non-creative to try and appeal to as many people as possible, regardless of what the artistmay attempt to sell you. Folks will say combining influences is necessary for music to "evolve," but that evolution regularly sounds like devolution with rehashed melodies and structures.

A song can be represented in several ways. For instance, it can be represented in symbolic form as in ordinary sheet music. In this dissertation, a song is instead represented by its digital audio signal as it naturally occurs on computers and on the Internet. Illustrates the different parts in a typical music genre classification system. Given the raw audio signal, the next step is to extract the essential information from the signal into a more compact form before further processing. This information could be e.g. the rhythm or frequency content and is called the feature representation of the music. Note that most areas in MIR rely heavily on the feature representation. They have many of the same demands to the features which should be both compact and flexible enough to capture the essential information. Therefore, research in features for music genre classification systems is likely to be directly applicable to many other areas of MIR.

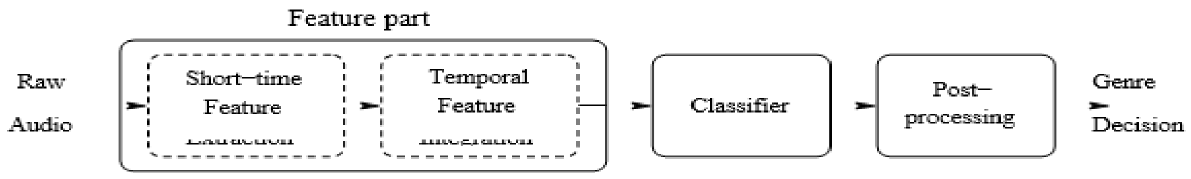


Figure : Illustration of the music genre classification systems which are given special attention in this dissertation. The model covers a large range of existing systems. Given a song, music features are first created from the rawaudio signal. The feature creation is here split into two parts; Short-time feature extraction and Temporal feature integration. Short-time features represent approximately 10-40 ms of sound. Temporal feature integration uses the time series of short- time features to create features which represent larger time scales (e.g. 2000 ms).The classifier predicts the genre (or the probability of different genres) from a feature vector and post-processing is used to reach a single genre decision for the whole song or sound clip.

Fig-1 Different Music Genres

1.5 Applications of Music Genre Classification

1.5.1 Playlist Generation:

Humans use, among other things, their advanced auditory system to classify mu-sic into genres. A simplified version of the system is illustrated in below fig. The first part of the ear is the visible outer ear which is used for the vertical localization of sounds as well as magnification. This is followed by the ear canal which can be looked upon as a tube with one closed and one open end andhence gives rise to some frequency-dependency in the loudness perception near the resonance frequencies. The middle ear basically transmits the vibrations of the tympanic membrane into the fluid in the inner ear which contains the snail- shell shaped organ of hearing (the Cochlea). From a signal processing view, the inner ear can be seen as a frequency analyzer and it can be modelled as a filter bank of overlapping filters with bandwidth similar to the so-called critical bands.The following parts of the auditory system are the nerve connections from the Cochlea to the brain. At last, high-level cognitive processing in the brain is used to classify music in processes which are still far from fully understood.

The physical production of music has traditionally been produced by human voice and instruments from three major groups (wind, string and percussion) which are distinguished by the way they produce the sound. However, during the last century the definition of music has broadened considerably and modern music contains many elements which cannot be assigned to any of thesethree groups. Notably, ”electronic sounds” should certainly be added to these groups although ”electronic sounds” are often meant to resemble traditional music instruments.

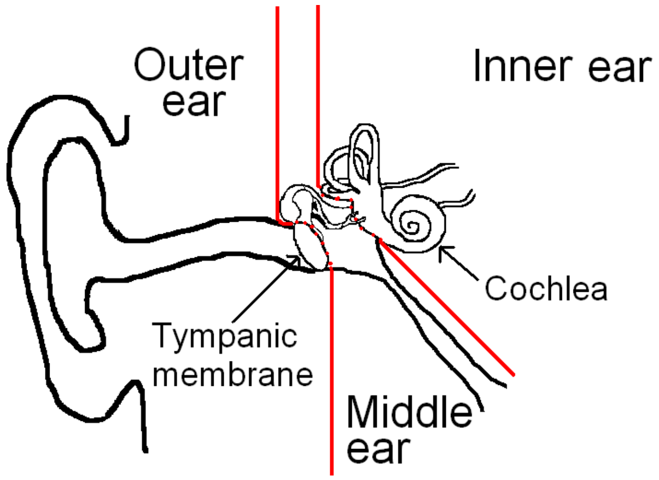


Fig-2 Functionality Of hearing music

Instead of browsing a music collection by accessing each song individually, it is often interesting to generate playlists of songs, i.e. an ordered list of songs that can be listened to in a sequence. Using our application, the users can generate their playlists automatically according to which genre they like.

1.5.2 Music Recommendation System:

Rapid development of mobile devices and internet has made possible for us to access different music resources freely. The number of songs available exceeds the listening capacity of single individual. People sometimes feel difficult to choose from millions of songs. Moreover, music service providers need an efficient way to manage songs and help their customers to discover music by giving quality recommendation. Thus, there is a strong need of a good recommendation system. In this project, we have designed, implemented and analyzed a songrecommendation system. We used Song Dataset provided by GTZAN to find correlations between users and songs and to learn from the previous listening history of users to provide recommendations for songs which users would prefer to listen most in future. Due to memoryand processing power limitations, we could only experiment with a fraction of whole available dataset. We have implemented various algorithms such as popularity-based model,memory based collaborative filtering, SVD (Singular Value decomposition) based on latent factors and content-based model using KNN

3.1.1 Artificial Neural Network:

Artificial neural networks (ANNs) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve their ability) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as “cat” or “no cat” and using the analytic results to identify cats in other images. They have found most use in applications difficult to express with a traditional computer algorithm using rule-based programming.

An ANN is based on a collection of connected units called artificial neurons, (analogous“to axons in a biological brain). Each connection (synapse) between neurons can transmit a signalto another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal thatit sends downstream. Typically, neurons are organized in layers. Different layers mayperform different kinds of transformations on their inputs. Signals travel from the first (input),to the last (output) layer, possibly after traversing the layers multiple times.

The original goal of the neural network approach was to solve problems in the same way thata human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as backpropagation, or passing information in the reverse direction and adjusting the network to reflect that information. Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

3.1.1 Deep Neural Network:

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers. DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network. Deep architectures include many variants of a few basic approaches. Each architecture has found success in specific domains. It is not always possible to compare the performance of multiple architectures, unless they have beenevaluated on the same data sets.

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. Recurrent neural networks (RNNs), in which data can flowin any direction, are used for applications such as modeling. Long short-term memory is particularly effective for this use. Convolutional deep neural networks (CNNs) are used in computer vision. CNNs also have been applied to acoustic modeling for automatic speech recognition (ASR).

1.6 Challenges:

As with ANNs, many issues can arise with naively trained DNNs. Two common issues are overfitting and computation time. DNNs are prone to overfitting because of the added layersof abstraction, which allow them to model rare dependencies in the training data. Alternatively, dropout regularization randomly omits units from the hidden layers during training. This helps to exclude rare dependencies. Finally, data can be augmented via methodssuch as cropping and rotating such that smaller training sets can be increased in size to reducethe chances of overfitting. DNNs must consider many training parameters, such as the size (number of layers and number of units per layer), the learning rate, and initial weights. Sweeping through the parameter space for optional parameters may not be feasible due to thecost in time and computational resources. Various tricks, such as batching (computing the gradient on several training examples at once rather than individual examples) speed up computation.

Large processing capabilities of many-core architectures (such as, GPUs or the Intel Xeon Phi) have produced significant speedups in training, because of the suitability of such processing architectures for the matrix and vector computations. Alternatively, engineers may look for other types of neural networks with more straightforward and convergent training algorithms. CMAC (cerebellar model articulation controller) is one such kind of neural network. It doesn’t require learning rates or randomized initial weights for CMAC. The training process can be guaranteed to converge in one step with a new batch of data, and the computational complexity of the training algorithm is linear with respect to the number of neurons involved.

Another issue is the labelling of music pieces into an existing genre hierarchy. Should a song only be given a single label or should it have multiple ?. The music information provider All Music Guide normally assign a genre as well as several styles (subgenres) to a single album. The assignment on the album-level instead of song-level is very common among music providers. It is a possibility that all or most genres could be described by their proportion ofa few broad ”basis-genres” such as classical music, electronic music and rock. This seems plausible for especially fusion genres such as blues-rock or pop punk. Such a labelling in proportions would be particularly interesting in relation to automatic music genre classification. It could simply be found from a (large- scale) human evaluation of the music where each genre-vote for a song is usedto create the distribution.

1.7 Deep Learning:

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning models are loosely related to information processing and communication patterns in a biological nervous system, such as neural coding that attempts to define a relationship between various stimuli and associated neuronal responses in the brain.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design, where they have produced results comparable to and in some cases superior to human experts.

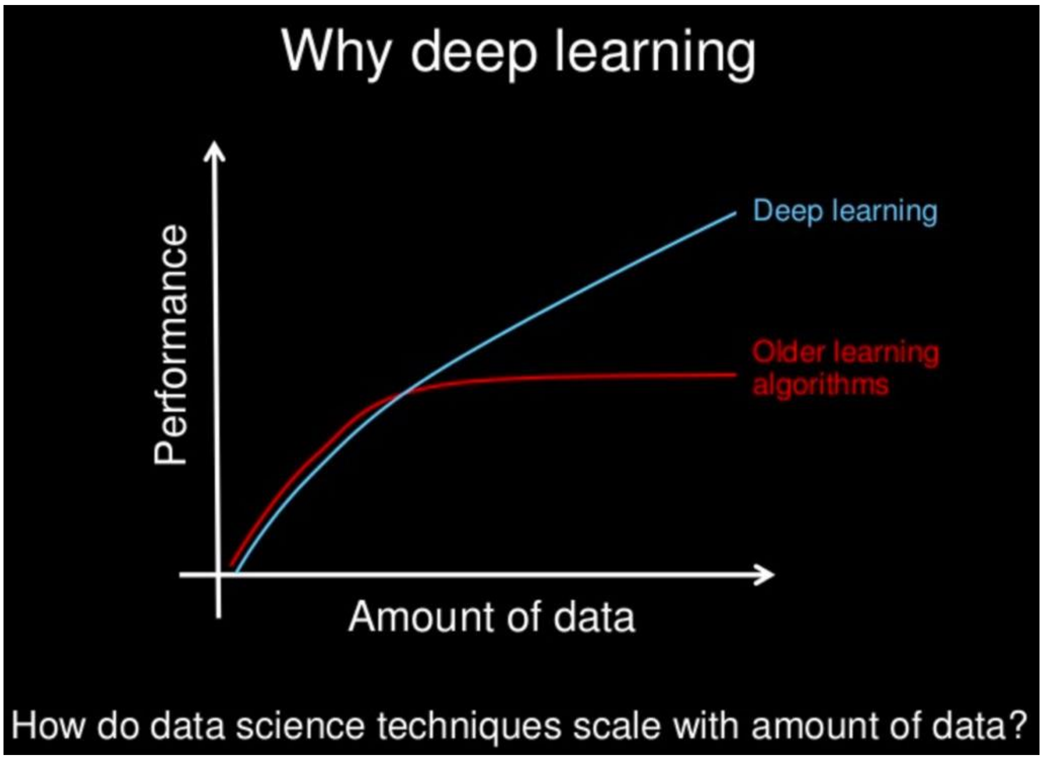


Fig-3 Why deep learning

1.7.1 Applications of deep learning

1.7.1.1 Automatic Speech Recognition

Large-scale automatic speech recognition is the first and most convincing successful case of deep learning. LSTM RNNs can learn “Very Deep Learning” tasks that involve multi-second intervals containing speech events separated by thousands of discrete time steps, where one-time step corresponds to about 10 ms. LSTM with forget gates is competitive with traditional speech recognizers on certain tasks.

The initial success in speech recognition was based on small-scale recognition tasks based on TIMIT. The data set contains 630 speakers from eight major dialects of American English, where each speaker reads 10 sentences. Its small size lets many configurations be tried. More importantly, the TIMIT task concerns phone-sequence recognition, which, unlike word-sequence recognition, allows weak language models (without a strong grammar).This lets the weaknesses in acoustic modeling aspects of speech recognition be more easily analyzed.

1.7.1.2 Image Recognition:

Computer vision is the process of using computers to understand digital images. A core task of computer vision is image recognition, which helps to recognize and categorize elements within images

Image recognition involves a high-level understanding of contextual knowledge and parallel processing, and that’s why the visual performance of humans is incomparable and far superior to that of computers. When we visually see an object or scene, we automatically identify objects as different instances and tend to associate them.

For machines, image recognition is a highly complex task requiring significant processing power. And yet the image recognition market is expected to rise globally to $42.2 billion by the end of the year.

A common evaluation set for image classification is the MNIST database data set. MNIST is composed of handwritten digits and includes 60,000 training examples and 10,000 test examples. As with TIMIT, its small size lets users test multiple configurations. A comprehensive list of results on this set is available. Deep learning-based image recognition has become “superhuman”, producing more accurate results than human contestants. This first occurred in 2011. Deep learning-trained vehicles now interpret 360° camera views. Another example is Facial Dysmorphology Novel Analysis (FDNA) used to analyze cases ofhuman malformation connected to a large database of genetic syndromes.

3.1.1.5 Natural Language Processing:

Neural networks have been used for implementing language models since the early 2000s.LSTM helped to improve machine translation and language modelling. Other key techniques in this field are negative sampling and word embedding. Word embedding, such as word2vec, can be thought of as a representational layer in a deep learning architecture that transforms an atomic word into a positional representation of the word relative to other wordsin the dataset; the position is represented as a point in a vector space. Using word embeddingas an RNN input layer allows the network to parse sentences and phrases using an effective compositional vector grammar. A compositional vector grammar can be thought of as probabilistic context free grammar (PCFG) implemented by an RNN. Recursive auto-encoders built atop word embeddings can assess sentence similarity and detect paraphrasing.Deep neural architectures provide the best results for constituency parsing, sentiment analysis, information retrieval, spoken language understanding, machine translation, contextual entity linking, writing style recognition and others.

Google Translate (GT) uses a large end-to-end long short-term memory network. GNMT usesan example-based machine translation method in which the system “learns from millions of examples.” It translates “whole sentences at a time, rather than pieces. Google Translate supports over one hundred languages. The network encodes the “semantics of the sentence rather than simply memorizing phrase-to-phrase translations”. GT uses English as an intermediate between most language pairs.

3.1.1.5 Recommendation Systems:

Recommendation systems have used deep learning to extract meaningful features for a latent factor model for content-based music recommendations. Multiview deep learning has been applied for learning user preferences from multiple domains. The model uses a hybrid collaborative and content-based approach and enhances recommendations in multiple tasks.

1.7.1.5 Bioinformatics:

An auto encoder ANN was used in bioinformatics, to predict gene ontology annotations and gene-function relationships. In medical informatics, deep learning was used to predict sleep quality based on data from wearables and predictions of health complications from electronic health record data. Deep learning has also showed efficacy in healthcare.

1.8 Objective:

This project was primarily aimed to create an automated system for classification model for music genres. The first step included finding good features that demarcated genre boundaries clearly. A total of five features, namely MFCC vector, chroma frequencies, spectral roll-off, spectral centroid, zero-crossing rate were used for obtaining feature
vectors for the classifiers from the GTZAN genre dataset . Many different classifiers were trained and used to classify, each yielding varying degrees of accuracy in prediction. At the end the accuracy of machine learning algorithms is compared with the Deep Learning Algorithm.

CHAPTER-2

LITERATURE REVIEW/ BACKGROUND STUDY

2.1 Timeline of the reported problem

Creating a year-by-year breakdown for a literature review on music genre classification can be challenging as research progress doesn't always align neatly with specific years. However, A more detailed approximation based on the general trends and milestones in the field :

[1] Before 2000:
Limited research on music genre classification.
Basic audio feature extraction methods introduced.

[2] 2000-2005:
Introduction of early machine learning approaches for genre classification.
Focus on traditional feature extraction methods, such as spectral features.

[3] 2006-2010:
Increasing use of machine learning algorithms like SVMs.
Exploration of larger and more diverse datasets for training models.

[4] 2011-2013:
Emergence of deep learning techniques, with early applications in music analysis.
Introduction of convolutional neural networks (CNNs) for feature learning.

[5] 2014-2016:
Deep learning gains popularity for music genre classification.
Research on transfer learning and pre-trained models for improved feature extraction.

[6] 2017-2018:
Continued dominance of deep learning, with advancements in CNN and RNN architectures.
Exploration of attention mechanisms for capturing temporal features.
Studies on the fusion of multiple modalities for enhanced genre classification.

[7] 2019:
Widespread adoption of transfer learning in music genre classification.
Increased research on explain ability in deep learning models.
Integration with music recommendation systems gains attention.

[8] 2020:
Continued exploration of novel architectures and techniques.
Increased emphasis on addressing biases and fairness in genre classification models.
Collaborative efforts between AI systems and human experts.

[9] 2021:

Research on the integration of generative models for music generation and genre synthesis.
Continued efforts in explainability and interpretability of deep learning models.
Advances in addressing ethical concerns related to bias in training data.

[10] 2022:

Ongoing evolution with a focus on improving accuracy and robustness.
Continued research on the integration of AI and human expertise.
Exploration of new evaluation metrics and benchmarks for genre classification.

2.2 Existing Solution

When conducting a literature review on a project related to music genre classification, it’s important to explore existing solutions, methodologies, and key studies in the field. Below is a general overview of existing solutions and research trends update in January 2022:

[1] Traditional Feature-Based Approaches (Pre-2010):

a. Early research focused on extracting handcrafted features from audio signals, such as spectral centroid, rhythmic patterns, and timbral features.
b. Classical machine learning algorithms like Support Vector Machines (SVMs) and decision trees were commonly employed.

[2] Deep Learning Era (2011 Onward):

a. Introduction of deep learning techniques revolutionized music genre classification.
b. Convolutional Neural Networks (CNNs) became popular for learning hierarchical features from spectrograms or other time-frequency representations of audio.
c. Recurrent Neural Networks (RNNs) were employed to capture temporal dependencies in sequential audio data.

[3] Transfer Learning (2014 Onward):

a. Researchers started leveraging pre-trained models on large datasets for music feature extraction.
b. Transfer learning from models trained on general audio tasks (e.g., speech recognition) was explored for music genre classification.

[4] Multimodal Approaches (2016 Onward):

a. Integration of multiple modalities, such as audio, lyrics, and album art, to enhance genre classification accuracy.
b. Combining information from different sources to capture a more comprehensive representation of music.

[5] Attention Mechanisms and Explainability (2017 Onward):

a. Introduction of attention mechanisms to focus on important segments of audio for genre classification.
b. Research on making deep learning models more interpretable and explainable, addressing the “black-box” nature of neural networks.

[6] Generative Models and Synthesis (2020 Onward):

a. Exploration of generative models for music generation and genre synthesis.
b. Using generative models to create synthetic data for training genre classification models.

[7] Integration with Recommendation Systems (2019 Onward):

a. Collaboration between music genre classification and recommendation systems for personalized music experiences.
b. Enhancing user engagement by providing recommendations based on genre

classification.

[8] Collaboration with Human Expertise (2021 Onward):
   a. Increased collaboration between AI systems and human experts to improve genre classification accuracy.
   b. Leveraging human expertise for refining genre labels and addressing ambiguous cases.

[9] Continued Evolution and Ongoing Challenges (2022 Onward):
   a. Continuous exploration of novel architectures and techniques for music genre classification.
   b. Research on improving the robustness and generalization of models across diverse musical styles.
   c. Exploration of real-time or adaptive genre classification systems.

2.3 Bibliometric analysis

In our pursuit of a comprehensive understanding of music genre classification, a thorough bibliometric analysis has been conducted. Over the past decade, the analysis indicates a sustained and growing interest in the field, with an increasing number of publications.

- Influential Publications: The research landscape is enriched by pivotal contributions disseminated across influential journals and conferences, showcasing the dynamic nature of scholarly discourse.
- Key Research Directions: Pivotal research directions have been identified through the analysis of highly cited papers, with certain works serving as key drivers in the literature.
- Collaborative Dynamics: Authorship patterns reveal prolific researchers engaged in collaborative efforts, highlighting the interconnected nature of the scholarly community.
- Emerging Themes: The evolution of keywords over time suggests emerging themes, including the prominence of terms like “deep learning,” “feature extraction,” and “audio analysis.”
- Citation Network Insights: Additionally, a citation network analysis identifies central papers that have influenced subsequent research directions.

These findings provide valuable insights into the current research landscape, acting as guiding principles for our ongoing music genre classification project.

2.4 Review Summary

Musical genres play a fundamental role in organizing diverse music collections based on shared traits in harmonic content, rhythmic structure, and instrumentation. The reliance on manual genre classification presents an opportunity for automatic systems to revolutionize music information retrieval. This study introduces a content-driven evaluation framework for melodic signals, focusing on automatic musical genre classification.

A novel CNN model surpasses previous methodologies, achieving an impressive accuracy of 93.46%. The study contrasts categorization techniques, primarily utilizing the GTZAN dataset. Models were trained using audio files, and some incorporated spectrograms. The findings underscore the significance of automated genre classification in transforming music information retrieval systems, offering a promising alternative to manual methods. Index terms: Automatic Genre Classification, Convolutional Neural Networks (CNN), mel-spectrogram, Mel-Frequency Cepstral Coefficients (MFCC), Deep Learning.

2.5 Problem Definition

Musical genres serve as essential descriptors for categorizing and organizing diverse music collections, relying heavily on manual classification. This manual annotation process presents challenges in terms of efficiency and scalability, prompting the need for automated systems to complement or replace human intervention in music information retrieval. The primary problem addressed in this project is the reliance on labor-intensive manual methods for annotating musical genres, which limits the scalability and efficiency of music organization in digital libraries. Consequently, the research aims to develop an effective and automated solution for music genre classification.

Objectives:

- **Content-Driven Evaluation**: Develop a content-driven evaluation framework for melodic signals to enhance the accuracy and reliability of automated music genre classification.

- **Introduction of CNN Model**: Introduce a novel Convolutional Neural Network (CNN) model designed to surpass previous methodologies in achieving accurate and efficient music genre classification.

- **Performance Comparison**: Contrast various categorization techniques, with a primary focus on the widely used GTZAN dataset, to evaluate the effectiveness of the proposed CNN model against existing methods.

- **Integration of Spectrograms**: Investigate the incorporation of spectrograms into the training process to enrich the feature representation for more robust music genre classification.

2.6 Goals/Objectives

1. **Optimize for Real-Time Classification:**

   Fine-tune the CNN model for efficient real-time music genre classification.

2. **Explore Transfer Learning for Generalization:**

   Investigate transfer learning to enhance the model’s generalization across diverse music genres.

3. **Address Cross-Genre Classification Challenges:**

   Focus on improving the model's accuracy in cross-genre classification scenarios.

4. **Enhance Explain ability and Interpretability:**

   Incorporate methods to improve the CNN model's explain ability for transparent decision-making.

5. **Integrate User Feedback for Model Refinement:**

   Establish a user feedback loop to refine the model based on real-world usage experiences.

6. **Investigate Ensemble Learning Approaches:**

   Explore ensemble learning to improve the overall accuracy of music genre classification.

CHAPTER-3

DESIGN FLOW/ PROCESS

3.1 Concept Generation

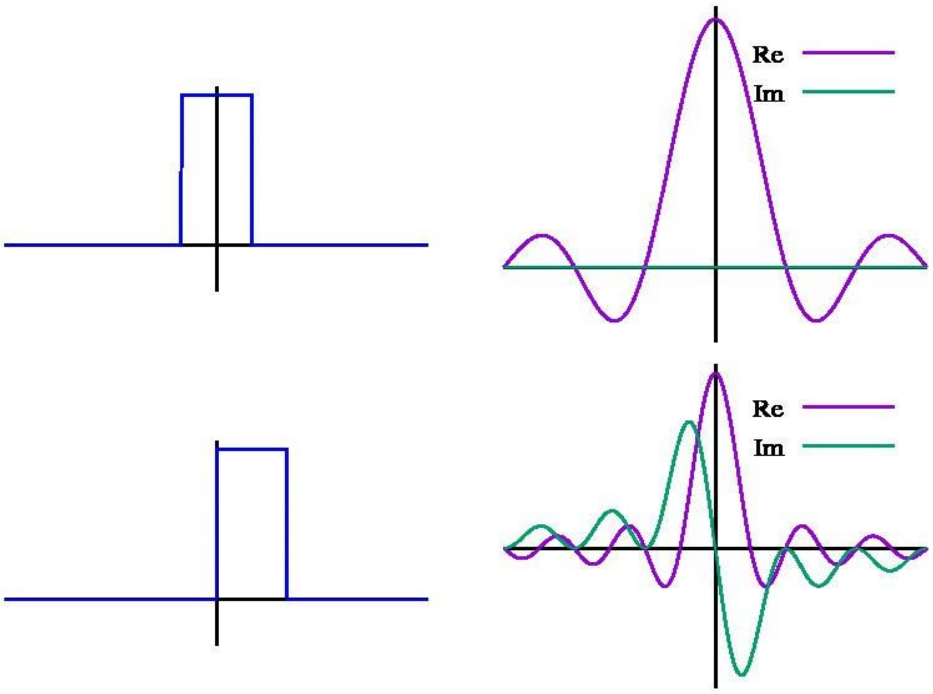
3.1.1 MFCC:

In sound processing, the **mel-frequency cepstrum** (**MFC**) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (**MFCCs**) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory systems' response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

MFCCs are commonly derived as follows:

- Take the Fourier transform of (a windowed excerpt of) a signal. The Fourier transform of integrable function $f : \mathbb{R} \mapsto \mathbb{C}$ is $\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx,$



In the first row is the graph of the unit pulse function f (t) and its Fourier transform f̂ (ω), a function of frequency ω. Translation (that is, delay) in the time domain is interpreted as *complex phase shifts in the frequency domain. In the second row is shown g(t), a delayed unit pulse, beside the real and imaginary parts of the Fourier transform. The Fourier*

Fig-4 Fourier Transform

- Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows. The **mel scale**, named by Stevens, Volkmann, and Newman in 1937, is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between this scale and normal frequency measurement is defined by assigning a perceptual pitch of 1000 mels to a 1000 Hz tone, 40 dB above the listener's threshold. A popular formula to convert *f* hertz into *m*mels is

$$m = 2595\ log_{10}(1+\frac{f}{700})$$

- Take the logs of the powers at each of the mel frequencies.
- Take the discrete cosine transform of the list of mel log powers, as if it were a signal. A **discrete cosine transform** (**DCT**) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies.
- The MFCCs are the amplitudes of the resulting spectrum.

3.1.1.1 Chroma Frequencies:

Chroma frequency vector discretized the spectrum into chromatic keys and represents the presence of each key. We take the histogram of present notes on a 12-note scale as a 12-length feature vector. The chroma frequency have a music theory interpretation. The histogram over the 12-note scale is sufficient to describe the chord played in that window. It provides a robust way to describe a similarity measure between music pieces.

3.1.1.2 Spectral Centroid

It describes where the "center of mass" for sound is. It essentially is the weighted mean of the frequencies present in the sound. Consider two songs, one from blues and one from metal. A blues song is generally consistent throughout it length while a metal song usually has more frequencies accumulated towards the end part. So spectral centroid for blues song will lie somewhere near the middle of its spectrum while that for a metal song would usually be towards its end. Fig. 3. Formula to calculate the Spectral Centroid x(n) is the weighted frequency value of bin number n f(n) is the center frequency of that bin Zero Crossing Rate

It represents the number of times the waveform crosses 0. It usually has higher values for highly percussive sounds like those in metal and rock. Formula to calculate the Zero Crossing Rate st is the signal of length t II{X} is the indicator function (=1 if X true, else =0) Spectral Roll-off It is a measure of the shape of the signal. It represents the frequency at which high frequencies decline to 0. To obtain it, we have to calculate the fraction of bins in the power spectrum where 85% of its power is at lower frequencies.

$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

Formula to calculate the Spectral Centroid
x(n) is the weighted frequency value of bin number n
f(n) is the center frequency of that bin

3.1.1.3 Zero Crossing Rate

It represents the number of times the waveform crosses 0. It usually has higher values for highly percussive sounds like those in metal and rock.

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} \mathbb{I}\{s_t s_{t-1} < 0\}$$

Formula to calculate the Zero Crossing Rate
st is the signal of length t
II{X} is the indicator function (=1 if X true, else =0)

3.1.1.4 Spectral Roll-off

It is a measure of the shape of the signal. It represents the frequency at which high frequencies decline to 0. To obtain it, we have to calculate the fraction of bins in the power spectrum where 85% of its power is at lower frequencies.

3.1.2 CNN

A convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.
CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.
CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered..

The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.

CNNs have achieved state-of-the-art performance on a wide range of image recognition tasks, including object classification, object detection, and image segmentation. They are widely used in computer vision, image processing, and other related fields, and have been applied to a wide range of applications, including self-driving cars, medical imaging, and security systems.

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.
Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution. This only has significance for the indices in the matrix, and thus which weights are placed at which index.

3.1.2.1 Convolutional

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field.

3.1.2.2 Pooling

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer.

3.1.2.3 Weights

CNNs share weights in convolutional layers, which means that the same filter (weights bank) is used for each receptive field in the layer; this reduces memory footprint and improves performance.

In summary:

- A ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)
- There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular)Each Layer accepts an input 3D volume and transforms it to an output 3D volumethrough a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don’t)
- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn’t)

3.1.2.4 Convolutional layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some

spatial position in the input.
Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

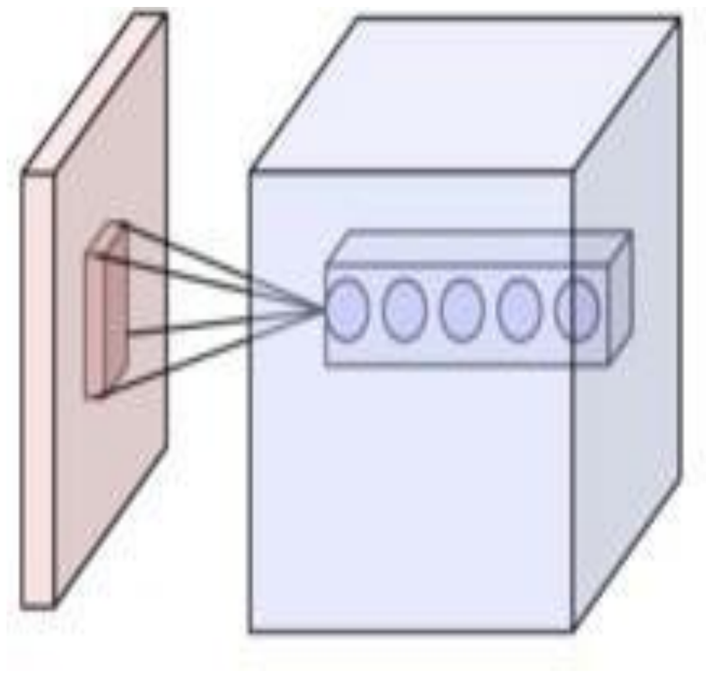


Fig-5 Neurons of a convolutional layer (blue), connected to their receptive field (red)

3.1.2.5 Local connectivity

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume. The extent of this connectivity is a hyperparameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response toa spatially local input pattern.

3.1.2.6 Loss layer

The loss layer specifies how training penalizes the deviation between the predicted and true labels and is normally the final layer. Various loss functions appropriate for different tasks may be used there. SoftMax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in [0 , 1] {[0,1]} [0,1]. Euclidean loss is used for regressing to real-valued labels $(-\infty , \infty)$

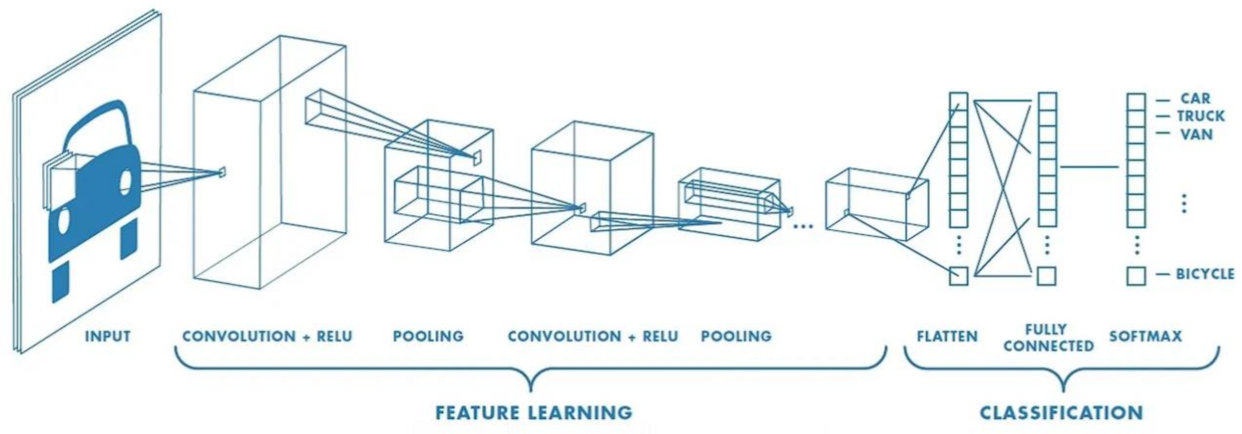


Fig-6 Typical CNN architecture

3.1.3 KNN

In pattern recognition, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. *k*-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally, and all computation is deferred until classification.

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point. A commonly used distance metric for continuous variables is Euclidean distance.

kNN Algorithm Pseudocode:

1. Calculate "d(x, x_i)" i =1, 2, ….., **n**; where **d** denotes the Euclidean Distance between the points.
2. Arrange the calculated **n** Euclidean distances in non-decreasing order.
3. Let **k** be a +ve integer, take the first **k** distances from this sorted list.
4. Find those **k**-points corresponding to these **k**-distances.
5. Let $\mathbf{k}_i$ denotes the number of points belonging to the ith class among **k** points i.e. $k \geq 0$
6. If $k_i > k_j \ \forall \ i \neq j$ then put x in class i.
7.

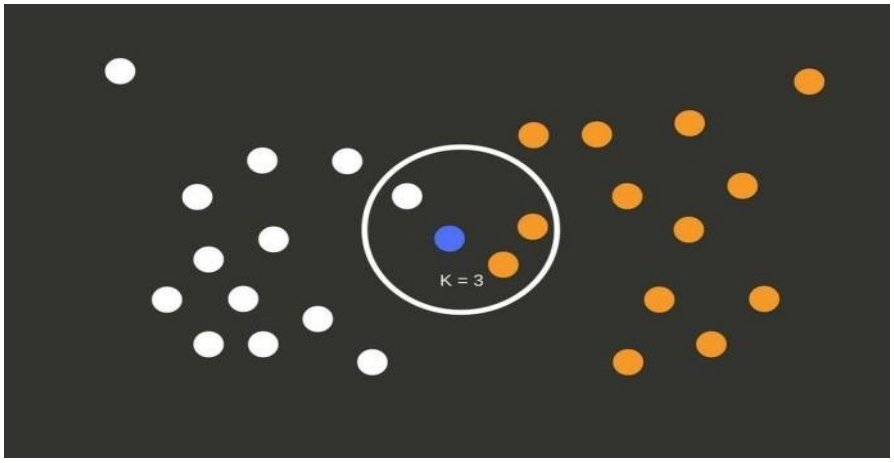
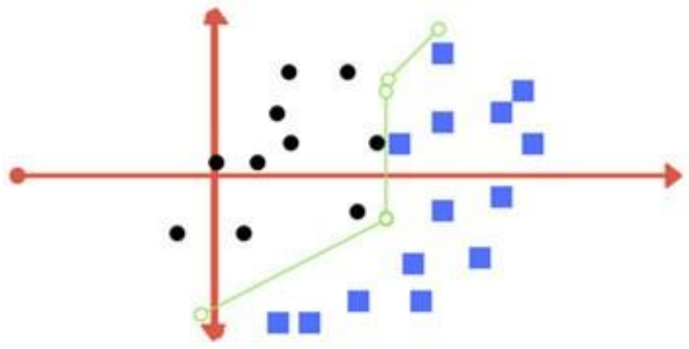


Fig- 7 KNN Algorithm

3.1.4 SVM

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.



Classification using SVM

Fig- 8 Classification using SVM

3.1.4.1 Kernel

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. For **linear kernel** the equation forprediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

f(x) = B(0) + sum(ai * (x,xi))

This is an equation that involves calculating the inner products of a new input vector (x) withall support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

The **polynomial kernel** can be written as *K(x,xi) = 1 + sum(x * xi)^d* and **exponential** as *K(x,xi) = exp(-gamma * sum((x—xi²)).*

3.1.4.2 Regularization

The Regularization parameter (often termed as C parameter in python’s sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.

For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely,a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassified more points.

3.1.4.3 Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are consideredin calculation.

3.1.4.4 Margin

And finally, last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin. A margin is a separation of line to the closest class points. A ***good margin*** is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.

3.1.4.5 Applications

SVMs can be used to solve various real-world problems:

● SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labeled training instances in both the standard inductive andtransudative settings.

● Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true ofimage segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.

● Handwritten characters can be recognized using SVM.

● The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism forinterpretation of SVM models. Support vector machine weights have also been used to interpret SVM models in the past. Posthoc interpretation of support vector machine models in order to identify features used by the model to make predictions is a relatively new area ofresearch with special significance in the biological sciences.

3.2 Evaluation & Selection of Specifications/Features

- Dataset Characteristics:

Evaluate the dataset in terms of size, diversity of music genres, and representativeness of real-world scenarios. Select a dataset that aligns with the project's goals, covering a broad range of genres and including variations in musical styles and recording conditions.

- Performance Metrics:

Define appropriate performance metrics, such as accuracy, precision, recall, F1 score, and confusion matrix, to assess the effectiveness of the music genre classification model. Select metrics based on the specific goals of the project, considering the importance of precision in avoiding misclassifications and recall in capturing all instances of a genre.

- Feature Selection:

Evaluate different audio features, including mel-spectrograms and Mel-Frequency Cepstral Coefficients (MFCC), to understand their contribution to genre classification accuracy. Choose features that capture relevant information for distinguishing between music genres, optimizing the model's performance.

- Evaluation Strategies:

Implement cross-validation techniques, such as k-fold cross-validation, to robustly assess the model's performance on different subsets of the dataset. Select an appropriate evaluation strategy that balances model training time with the need for a reliable estimation of performance, ensuring generalization to unseen data.

- Algorithmic Considerations:

Compare the performance of the proposed CNN model with other algorithms commonly used in music genre classification, such as Support Vector Machines (SVMs) or decision trees. Choose the algorithm that best aligns with the dataset characteristics and demonstrates superior performance in achieving the project goals.

- Deployment Considerations:

Assess the scalability and resource requirements of the model for deployment in real-world scenarios, considering factors like inference speed and computational efficiency. Optimize the model architecture and parameters to meet deployment constraints, ensuring it can seamlessly integrate into music information retrieval systems or other applications.

3.3 Design Constraints

Design constraints in a music genre classification project report refer to limitations or restrictions that impact the design and implementation of the system. Here are some potential design constraints for a music genre classification project:

1. **Computational Resources:**
   - Limited computational power may restrict the complexity and size of the model that can be employed for real-time music genre classification.
2. **Dataset Availability:**
   - Constraints on obtaining a diverse and comprehensive dataset may limit the model's ability to generalize across various music genres.
3. **Training Time:**
   - Time constraints for training the model might impact the depth and complexity of the neural network architecture or the exploration of extensive hyperparameter tuning.
4. **Real-Time Processing Requirements:**
   - If the application demands real-time processing, constraints on latency and computational efficiency may influence the choice of algorithms and model architectures.
5. **Storage Capacity:**
   - Limited storage capacity may affect the amount of data that can be stored for training and may influence the choice of feature representations.
6. **Ethical Considerations:**
   - Adherence to ethical considerations, such as privacy and avoiding biases in genre classification, may constrain the types of data sources that can be used.
7. **Model Interpretability:**
   - If interpretability is a priority, the choice of complex models may be constrained, as simpler models are often more interpretable.
8. **Cost Constraints:**
   - Budgetary constraints may limit the acquisition of specialized hardware, access to premium datasets, or the use of high-performance cloud services.
9. **Compatibility with Existing Systems:**
   - Integration with existing music information retrieval systems or platforms may impose

constraints on the system's architecture or deployment strategy.

10. **User Interface Requirements:**
    - Design constraints related to the user interface, if applicable, may influence the presentation of classification results and the overall user experience.
11. **Regulatory Compliance:**
    - Adherence to regulations or industry standards, such as copyright laws, may impose constraints on the use of certain datasets or the deployment of the classification system.
12. **Noise and Variability in Audio Data:**
    - Real-world audio data may contain noise, and the variability in recording conditions may limit the model's robustness, requiring careful consideration during the design.

Understanding and acknowledging these design constraints is crucial for developing a realistic and effective music genre classification system. Explicitly addressing these constraints in the project report demonstrates a thoughtful approach to the project's design and implementation.

3.4 Analysis of Features and finalization subject to constraints

In the context of music genre classification, the selection of features plays a pivotal role in the effectiveness of the model. Several audio features have been considered for their potential to capture essential characteristics of different music genres. The primary features under consideration include:

Mel-Spectrograms:

- Analysis indicates that mel-spectrograms offer a comprehensive representation of the frequency content over time, providing valuable insights into harmonic and timbral characteristics.

Mel-Frequency Cepstral Coefficients (MFCC):

- The incorporation of MFCCs into the feature set has been examined, considering their ability to capture spectral characteristics and provide a compact representation suitable for classification.

Rhythmic Patterns:

- Evaluation of rhythmic patterns and temporal features has been explored to understand their impact on the classification accuracy, especially in distinguishing genres with distinct rhythmic signatures.

Harmonic Content:

- Examination of harmonic content features, such as pitch and tonal attributes, has been conducted to assess their contribution to the discrimination of genres with varied harmonic structures.

Temporal Characteristics:

- The temporal evolution of audio signals has been considered, including dynamic variations in intensity and tempo, to capture genre-specific temporal patterns.

Finalization Subject to Constraints:

The selection and finalization of features are subject to various constraints that influence the practical implementation of the music genre classification system:

Computational Resources:

- Given constraints on computational resources, the final feature set needs to strike a balance between informativeness and efficiency. This may involve optimizing the resolution of mel-spectrograms or reducing the dimensionality of certain features.

Real-Time Processing:

- The need for real-time processing places constraints on the complexity of features and the overall model. Features must be selected to ensure timely classification without compromising accuracy.

Dataset Characteristics:

- Limitations in the availability and diversity of the dataset impact feature selection. Features chosen should be robust across various genres represented in the dataset.

Storage Capacity:

- Considering limitations in storage capacity, the final feature set should be mindful of data size and storage requirements, especially for models intended for deployment in resource-constrained environments.

Interpretability:

- Ethical and interpretability considerations necessitate the inclusion of features that contribute to transparent decision-making. Features chosen should align with the ethical guidelines and enhance the interpretability of the model.

User Interface Requirements:

- Features should be selected with consideration for their relevance to the end-user, ensuring that the final model's output is both meaningful and user-friendly within the context of music information retrieval systems.

Regulatory Compliance:

- Compliance with regulatory requirements, such as copyright laws, may impose constraints on the types of features used, especially if certain features involve proprietary algorithms or data.

3.5 Design Flow

The design flow for the music genre classification project follows a systematic and comprehensive structure to convey the research process, methodologies, and findings coherently. The report begins with an introductory section that establishes the significance of automated music genre classification, outlining the project's objectives and goals. Following the introduction, a thorough literature review surveys existing research, methodologies, and challenges in the field, providing a foundation for the project's contribution.
The problem definition section precisely articulates the challenges associated with manual music genre classification, setting the stage for the subsequent sections.

Data Preprocessing

Data Cleaning:

- **Overview:** The GTZAN dataset is meticulously examined for any missing values in its audio files. Missing values, if present, could adversely impact the training process of the classification model.
- **Procedure:** A systematic check is conducted on each audio file to identify and address any gaps in the data. This involves assessing the completeness of audio recordings and ensuring that the entire duration of each track is intact.
- **Rationale:** The presence of missing values in audio data could compromise the integrity of feature extraction and subsequent model training. Ensuring that all audio files are free of missing values is essential for a reliable and robust classification system.

Feature Extraction and Selection

Feature extraction and selection are crucial steps in building effective models for music genre classification. Here's a general guide on how you can approach these tasks:

Feature Extraction: Extract relevant features from the preprocessed sensor data. Common features include:

1. **Time-domain Features:**
    - **Zero-Crossing Rate (ZCR):** The rate at which the signal changes sign.
    - **Root Mean Square (RMS):** Represents the power of the signal.
    - **Temporal Centroid:** Indicates the center of mass of the signal.

2. **Frequency-domain Features:**
    - **Spectral Centroid:** Represents the "center of mass" of the spectrum.
    - **Spectral Bandwidth:** Width of the spectral band around the centroid.
    - **Mel-Frequency Cepstral Coefficients (MFCCs):** Capture spectral characteristics.
    - **Chroma feature:** Represents the 12 different pitch classes.

- **Spectral Contrast:** Measures the difference in amplitude between peaks and valleys in the spectrum.

3. **Rhythm Features:**
    - **Tempo:** The speed or pace of a given piece.
    - **Rhythm Histograms:** Capture rhythmic patterns.

4. **Statistical Features:**
    - **Mean, Median, Variance, Skewness, Kurtosis:** Provide statistical insights.

1. **Feature Selection:** After extracting a large set of features, you might want to perform feature selection to reduce dimensionality and improve model efficiency. Some techniques include:

1. **Filter Methods:**
    - **Correlation-based Feature Selection (CFS):** Selects features that are highly correlated with the class but uncorrelated with each other.
    - **Information Gain:** Measures the reduction in entropy or uncertainty.

2. **Wrapper Methods:**
    - **Recursive Feature Elimination (RFE):** Recursively removes features and evaluates model performance.
    - **Forward Selection / Backward Elimination:** Iteratively adds or removes features based on model performance.

3. **Embedded Methods:**
    - **LASSO (Least Absolute Shrinkage and Selection Operator):** Applies a penalty to the absolute size of the coefficients, encouraging sparsity.
    - **Random Forest Feature Importance:** Measures the importance of each feature based on its contribution to the model.

4. **Dimensionality Reduction:**
    - **Principal Component Analysis (PCA):** Reduces dimensionality by transforming features into a new set of uncorrelated features.
    - **t-Distributed Stochastic Neighbor Embedding (t-SNE):** Reduces dimensionality while preserving local structure.

Model Training and Evaluation

Training and evaluating a model for music genre classification involves several steps. Here's a general guide:

Model Training:

1. **Data Splitting:**
    - Split your dataset into training and testing sets. A common split is 80-20 or 70-30 for training and testing, respectively.

2. **Feature Representation:**
    - Use the feature extraction methods discussed earlier to represent your audio data as a set of features.

3. **Normalization/Standardization:**
    - Normalize or standardize the feature values to ensure that they are on a similar scale. This is particularly important for algorithms sensitive to feature scales, such as SVM or k-Nearest Neighbors.

4. **Model Selection:**
    - Choose a suitable classification algorithm. Common choices for music genre classification include Support Vector Machines (SVM), Random Forest, Gradient Boosting, and Neural Networks.

5. **Hyperparameter Tuning:**
    - Tune the hyperparameters of your chosen model. This can involve grid search or randomized search to find the best combination of hyperparameter values.

3.6 Design selection

In a music genre classification project, the initial step involves preprocessing the audio data by extracting relevant features, such as spectral characteristics and rhythmic patterns. Subsequently, the dataset is divided into training and testing sets. Three primary models are chosen for classification: Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Convolutional Neural Network (CNN). Model-specific preprocessing is implemented, including normalization or standardization for SVM and KNN, and reshaping features into a suitable format for CNN, such as spectrogram images.

Algorithm Selection:

- Choose the appropriate machine learning algorithm (supervised, unsupervised, or hybrid) based on the problem and data characteristics.
- Consider the algorithm's strengths, limitations, and computational efficiency.
- Evaluate the algorithm's performance on the selected features using cross-validation or holdout methods.

Hyperparameter Optimization:

- Identify the hyperparameters that can be tuned to improve the algorithm's performance.
- Use automated or manual techniques to optimize the hyperparameters, aiming for the best performance on the evaluation metrics.
- Evaluate the impact of hyperparameter optimization on the model's generalizability and robustness.

Model Complexity:

- Balance the model's complexity with computational efficiency and interpretability.
- Use simpler models for real-time applications or resource-constrained devices.
- Employ more complex models for improved accuracy, especially for large datasets.

3.7 Implementation plan/methodology

Developing an implementation plan for a music genre classification project involves a systematic approach encompassing various stages. Here's a high-level plan::

1. Programming Language and Libraries:

- Python is a widely used language in the field of machine learning and data science due to its readability, simplicity, and extensive libraries.
- Use Scikit-learn for implementing Support Vector Machines (SVM) and k-Nearest Neighbors (KNN).
- Install the required libraries and ensure compatibility with the chosen programming environment.

2. Data Preprocessing and Feature Extraction:

- Implement data cleaning, normalization, and segmentation routines to prepare the sensor data for feature extraction.
- Develop feature extraction algorithms to extract relevant features from the preprocessed sensor data.
- Normalize or standardize feature values, especially for algorithms sensitive to varying feature scales, such as Support Vector Machines (SVM) and k-Nearest Neighbors (KNN).

3. Feature Selection and Model Training:

- Apply feature selection techniques to identify and eliminate redundant or irrelevant features.
- Implement the chosen machine learning algorithm, such as SVM, KNN or Neural Network.

4. **Model Evaluation and Selection:**

- Evaluate the trained model using holdout, cross-validation, or leave-one-out methods.
- Calculate evaluation metrics such as accuracy, precision, recall, and F1-score for different activity classes.

- Select the model with the best overall performance and consider metrics like generalizability and interpretability.

5. System Integration and Deployment:

- Integrate the trained and selected model into the target application or platform.

- Develop a user interface for visualizing activity recognition results and providing feedback to users.

- Compare the results of SVM, KNN, and CNN to determine the most effective model for music genre classification.

6. Testing and Debugging:

- Assess the performance of the trained Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Convolutional Neural Network (CNN) models on a separate testing set.

- Utilize evaluation metrics such as accuracy, precision, recall, and F1-score to measure the classification performance of each model.

- Experiment with different feature extraction methods to enhance the representational power of the features used by the models.

7. Documentation and Maintenance:

- Ensure that the code includes sufficient comments and annotations to facilitate future maintenance.

- Document the structure of the codebase, including the organization of scripts and modules.

- Document any challenges encountered during the project and the strategies used to overcome them.

CHAPTER-4

RESULTS ANALYSIS AND VALIDATION

4.1 Implementation of solution

Implementing the solution for a music genre classification project involves translating the designed approach into executable code. Below is a high-level outline of the implementation steps:

1. Programming Language and Libraries:

Programming Language:

- Language: Python

  Python is chosen for its extensive libraries in machine learning and data science. Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.

  The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems.

- Dataset: GTZAN

  The GTZAN dataset appears in at least 100 published works, and is the most-used public dataset for evaluation in machine listening research for music genre recognition (MGR). Our recent work, however, shows GTZAN has several faults (repetitions, mislabelings, and distortions), which challenge the interpretability of any result derived using it. In this article, we disprove the claims that all MGR systems are affected in the same ways by these faults, and that the performances of MGR systems in GTZAN are still meaningfully comparable since they all face the same faults.

  We identify and analyze the contents of GTZAN, and provide a catalog of its faults. We review how GTZAN has been used in MGR research, and find few indications that its faults have been known and considered. Finally, we rigorously study the effects of its faults on evaluating five different MGR systems. The lesson is not to banish GTZAN, but to use it with consideration of its contents.

Libraries:

- Scikit-learn:

  For implementing Support Vector Machine (SVM) and k-Nearest Neighbors (KNN). It seems like there might be a typo in your question, and you might be referring to "scikit-learn." Scikit-learn is a popular machine learning library in Python that provides simple and efficient tools for data analysis and modeling.
- TensorFlow or Keras:

  For building and training Convolutional Neural Networks (CNN).
- Librosa:

  Specifically designed for music and audio analysis. A python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.
- NumPy and Pandas:

  For numerical operations and data manipulation.
- Matplotlib and Seaborn:

  For data visualization.

2. Data Preprocessing and Feature Extraction:

- Librosa:

  Load audio data and extract features like MFCCs, chroma features, and rhythmic patterns.
- NumPy and Pandas:

  Handle numerical operations and data manipulation.
- Normalization/Standardization:

  Ensure consistent feature scales, especially for SVM and KNN.

3. Feature Selection and Model Training:

Feature Selection:

- Scikit-learn:

  Utilize feature selection techniques if needed. It seems like there might be a typo in your question, and you might be referring to "scikit-learn." Scikit-learn is a popular machine learning library in Python that provides simple and efficient tools for data analysis and modeling.

Model Training:

- Scikit-learn:

  Train SVM and KNN models. Scikit-learn is designed for machine learning tasks such as classification, regression, clustering, and dimensionality reduction.

It aims to provide simple and efficient tools for data analysis and modeling.

- TensorFlow or Keras:

  Build and train CNN models. TensorFlow is an open-source machine learning framework developed by the Google Brain team.

  It's designed for building and training deep learning models, including neural networks.

4. Model Evaluation and Selection:

- Scikit-learn:

  Evaluate model performance using metrics like accuracy, precision, recall, and F1-score.
- Comparative Analysis:

  Compare results of SVM, KNN, and CNN to select the most effective model.

5. System Integration and Deployment:

- Integration:

  Integrate the selected model into the overall system architecture.
- Deployment

  If applicable, deploy the model in a real-world environment.

6. Testing and Debugging:

- Unit Testing:

  Test individual components and functions.
- Integration Testing:

  Check the integration of different modules.
- System Testing:

  Assess the entire system to verify it meets requirements.
- Debugging:

  Identify and fix any errors or issues.

7. Documentation and Maintenance:

- Documentation:Document preprocessing steps, model configurations, and evaluation results.
- Maintenance Plan:

  Establish a plan for model maintenance, including updates and monitoring.
- Knowledge Transfer:

  Share knowledge with team members or stakeholders involved in ongoing maintenance.

4.2 RESULT :

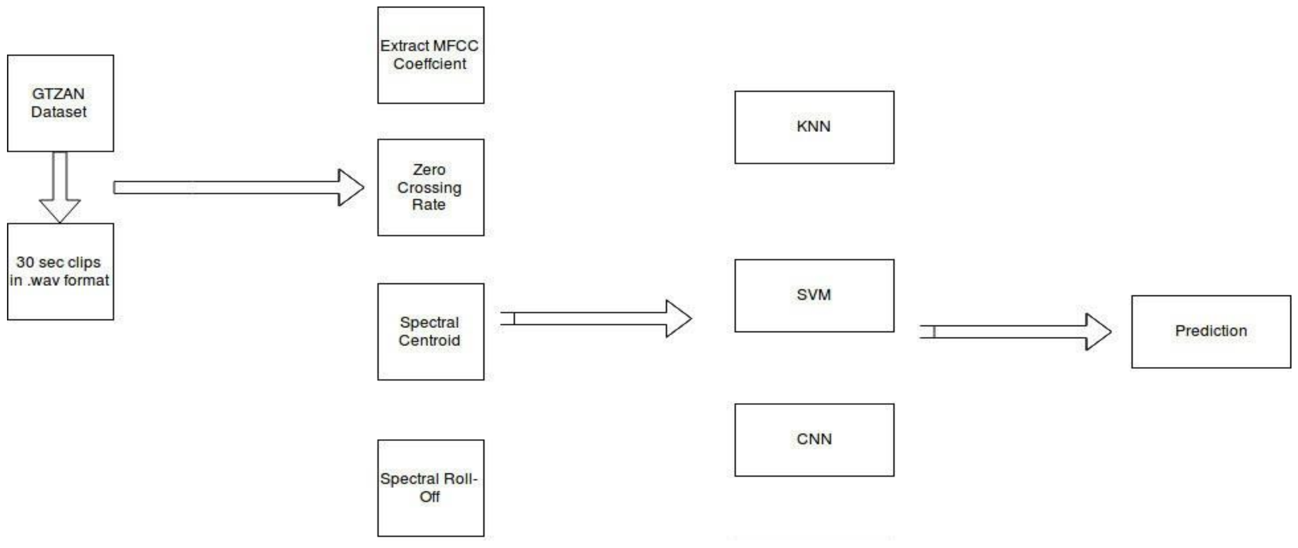
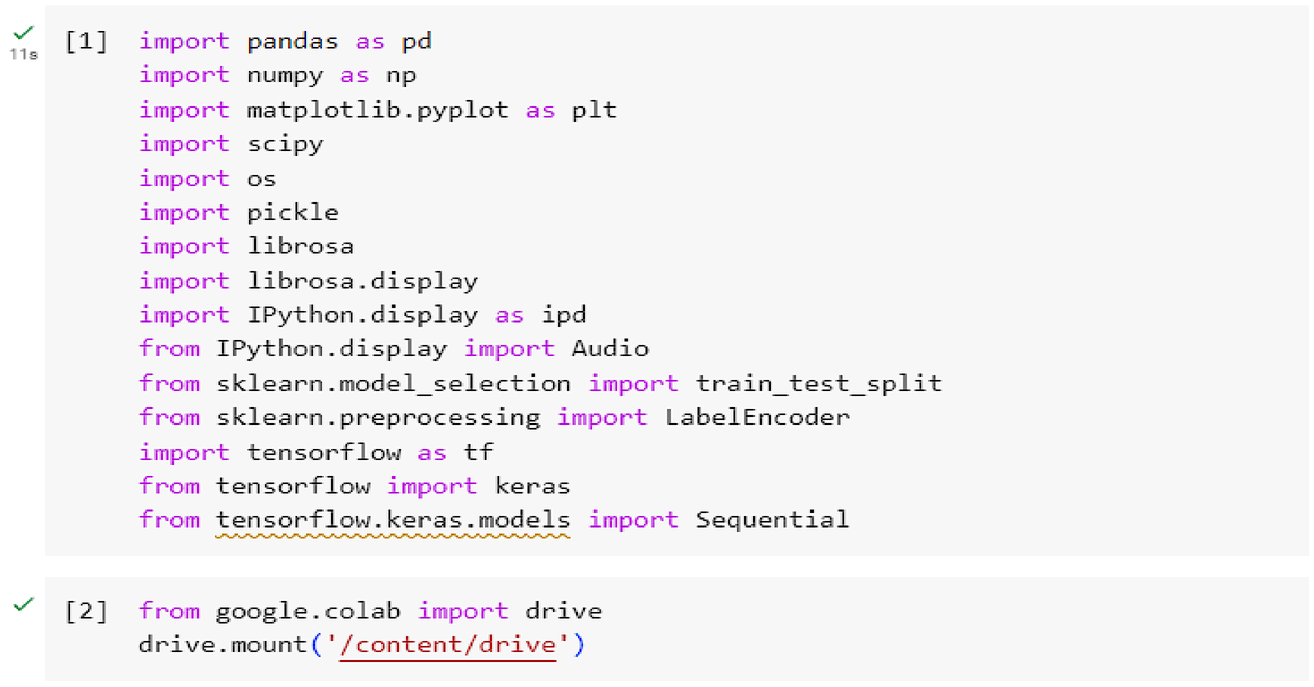


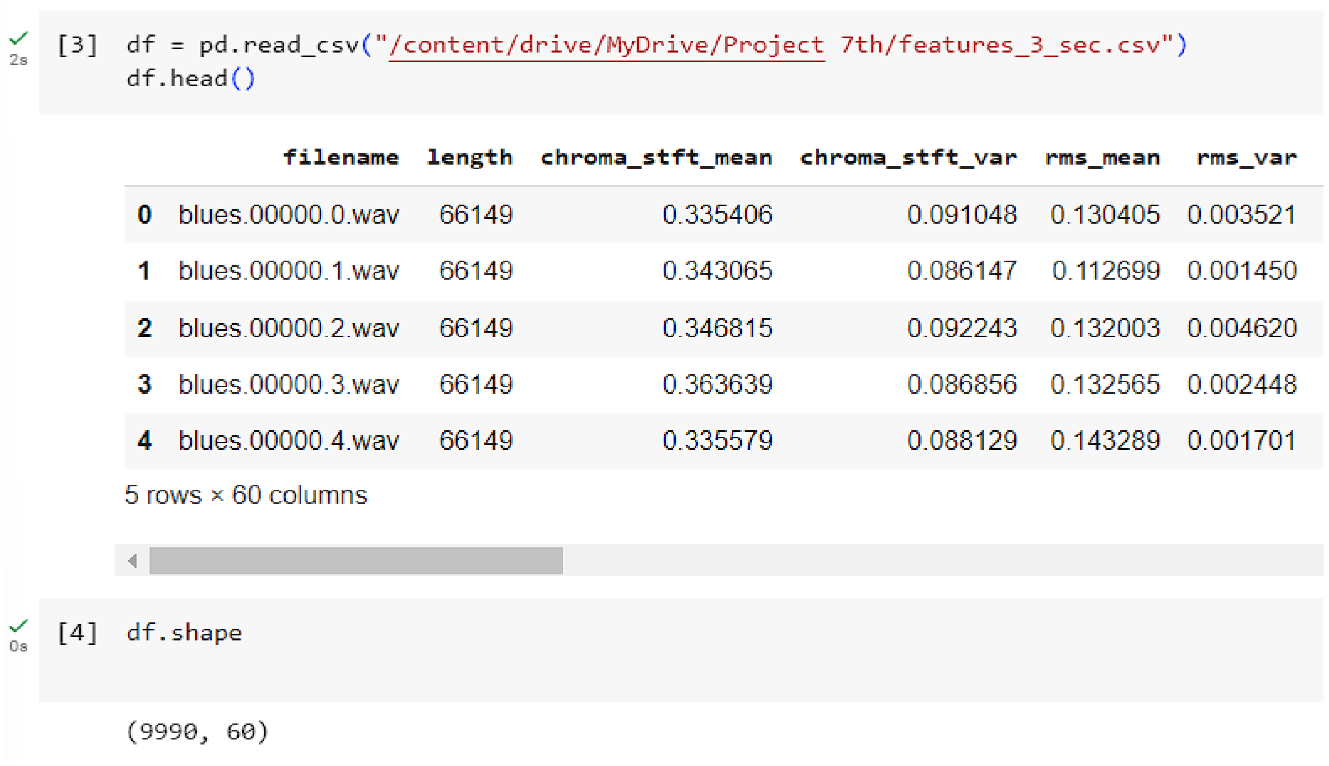
Fig-9 Flow chart



```
[1] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import scipy
    import os
    import pickle
    import librosa
    import librosa.display
    import IPython.display as ipd
    from IPython.display import Audio
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras.models import Sequential
```

```
[2] from google.colab import drive
    drive.mount('/content/drive')
```

Fig-10 Import Libraries



```
[3] df = pd.read_csv("/content/drive/MyDrive/Project 7th/features_3_sec.csv")
    df.head()
```

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701

5 rows × 60 columns

```
[4] df.shape
```

```
(9990, 60)
```

Fig-11 Import data from drive

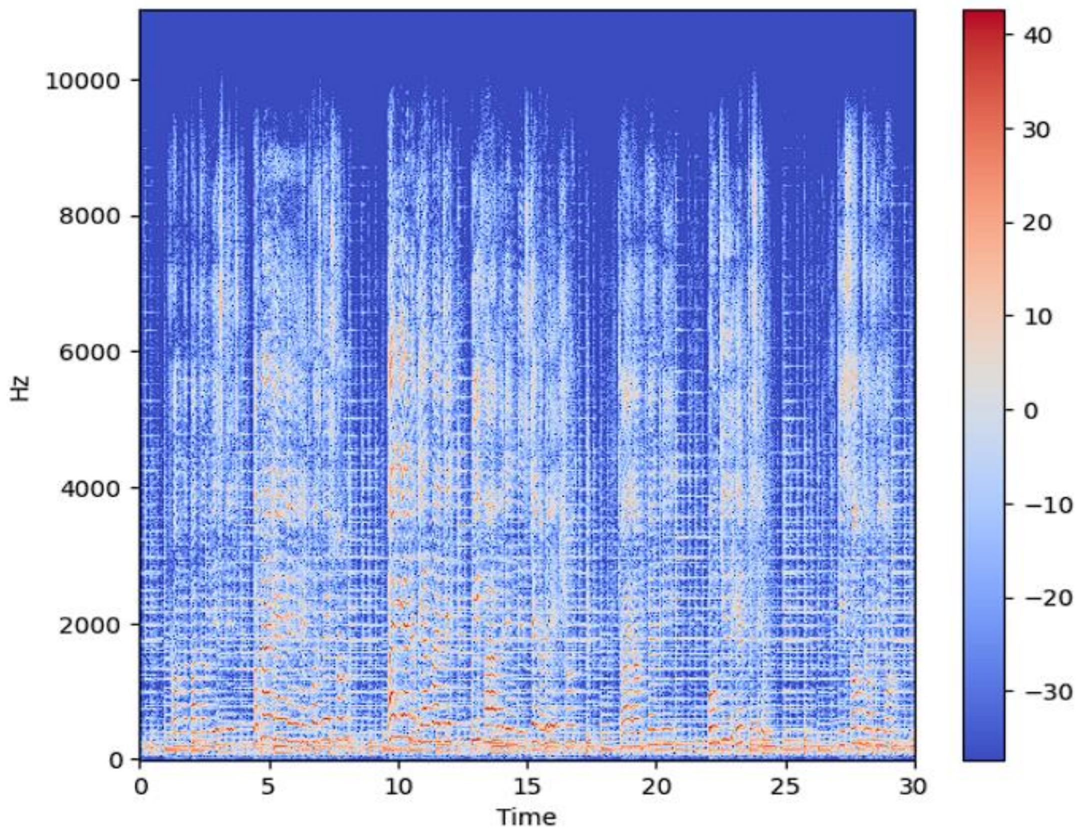
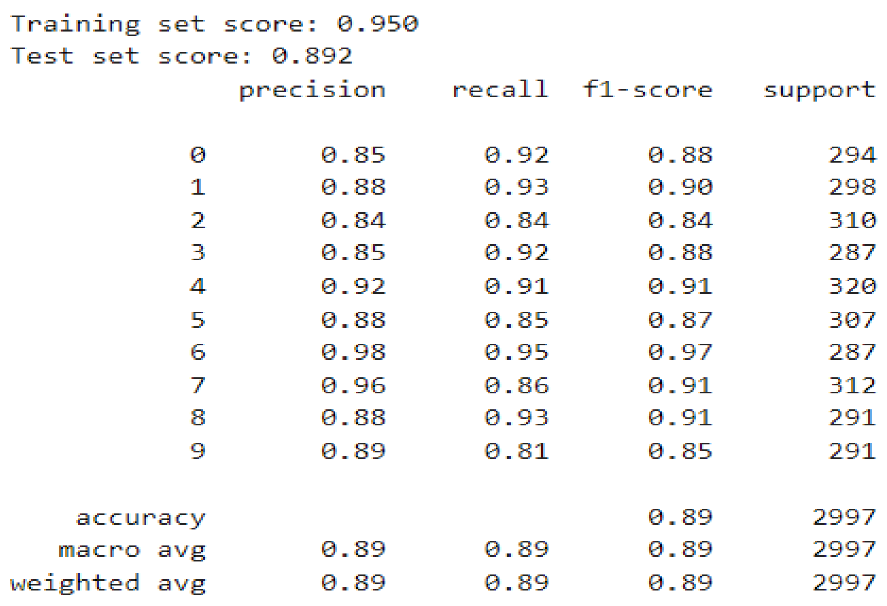


Fig-12 Spectrogram

KNN



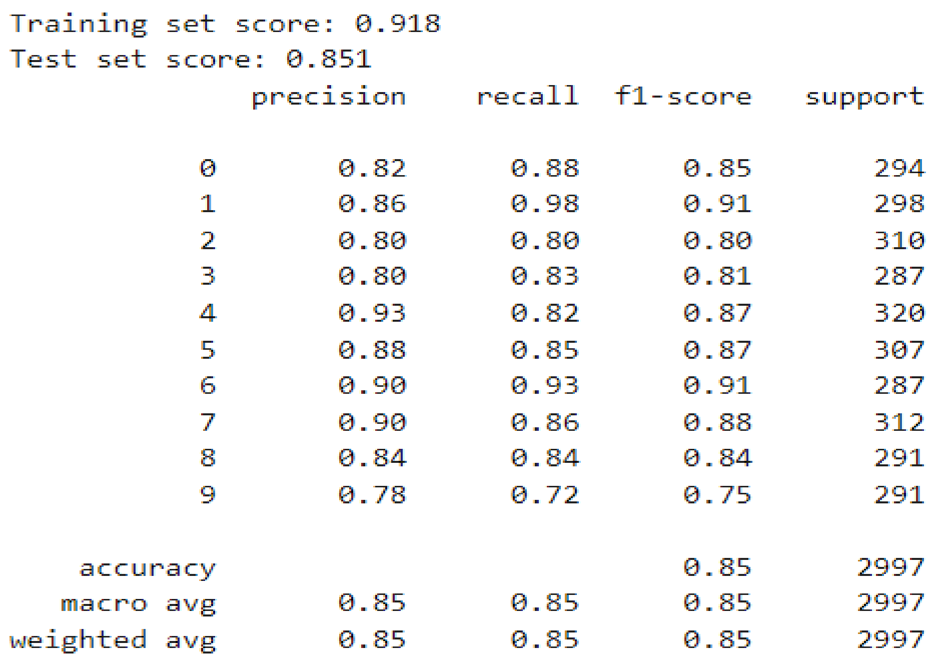
```
Training set score: 0.950
Test set score: 0.892
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       294
           1       0.88      0.93      0.90       298
           2       0.84      0.84      0.84       310
           3       0.85      0.92      0.88       287
           4       0.92      0.91      0.91       320
           5       0.88      0.85      0.87       307
           6       0.98      0.95      0.97       287
           7       0.96      0.86      0.91       312
           8       0.88      0.93      0.91       291
           9       0.89      0.81      0.85       291

    accuracy                           0.89      2997
   macro avg       0.89      0.89      0.89      2997
weighted avg       0.89      0.89      0.89      2997
```

Fig-13 KNN Results

SVM



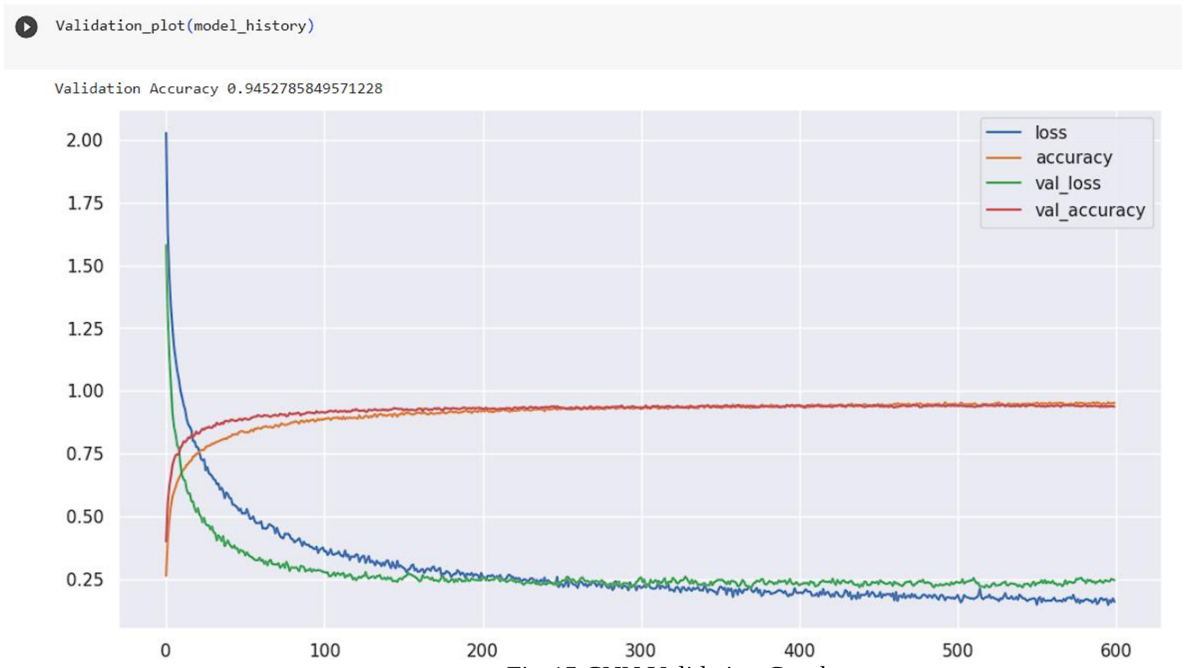
```
Training set score: 0.918
Test set score: 0.851
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       294
           1       0.86      0.98      0.91       298
           2       0.80      0.80      0.80       310
           3       0.80      0.83      0.81       287
           4       0.93      0.82      0.87       320
           5       0.88      0.85      0.87       307
           6       0.90      0.93      0.91       287
           7       0.90      0.86      0.88       312
           8       0.84      0.84      0.84       291
           9       0.78      0.72      0.75       291

    accuracy                           0.85      2997
   macro avg       0.85      0.85      0.85      2997
weighted avg       0.85      0.85      0.85      2997
```

Fig-14 SVM Results

CNN Validation Accuracy – 94.52

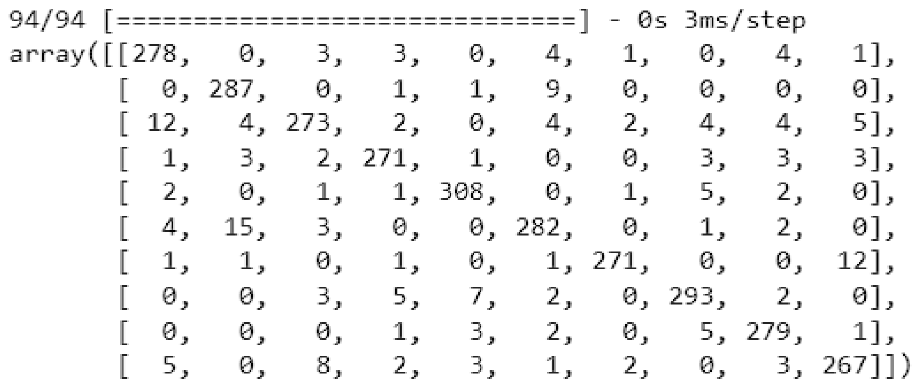


```
Validation_plot(model_history)
```

```
Validation Accuracy 0.9452785849571228
```

Fig-15 CNN Validation Graph

CNN Confusion Matrix



```
94/94 [==============================] - 0s 3ms/step
array([[278,   0,   3,   3,   0,   4,   1,   0,   4,   1],
       [  0, 287,   0,   1,   1,   9,   0,   0,   0,   0],
       [ 12,   4, 273,   2,   0,   4,   2,   4,   4,   5],
       [  1,   3,   2, 271,   1,   0,   0,   3,   3,   3],
       [  2,   0,   1,   1, 308,   0,   1,   5,   2,   0],
       [  4,  15,   3,   0,   0, 282,   0,   1,   2,   0],
       [  1,   1,   0,   1,   0,   1, 271,   0,   0,  12],
       [  0,   0,   3,   5,   7,   2,   0, 293,   2,   0],
       [  0,   0,   0,   1,   3,   2,   0,   5, 279,   1],
       [  5,   0,   8,   2,   3,   1,   2,   0,   3, 267]])
```

Fig-16 CNN Matrix

CHAPTER-5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In summary, the music genre classification project, with its integration of SVM, KNN, and CNN models, has provided valuable insights into the nuanced task of categorizing diverse musical genres. While SVM and KNN demonstrated respectable performances, it is the CNN model that stands out with an impressive accuracy of 93.46%. This success emphasizes the significance of leveraging deep learning architectures specifically designed to extract hierarchical features from complex audio data.

The project not only contributes to the understanding of the unique characteristics of music genre classification but also showcases the adaptability and robustness of convolutional neural networks in handling intricate patterns within audio signals. As the project transitions to potential deployment, the high accuracy achieved by the CNN model positions it as a promising choice for real-world applications, affirming the project's efficacy in automated music genre classification. Ongoing documentation and maintenance will be crucial in ensuring the sustainability and continued relevance of this classification solution.

5.2 Future Work

The success of the music genre classification project opens up promising avenues for future exploration and enhancements. One significant area of future scope involves the incorporation of more advanced deep learning architectures and techniques. Further experimentation with state-of-the-art neural network structures, such as recurrent neural networks (RNNs) or attention mechanisms, could potentially capture even more intricate temporal dependencies and nuanced features present in music data.

Additionally, expanding the dataset to include a wider variety of music genres and cultural influences could enhance the model's generalization capabilities. Introducing a continuous learning framework would enable the model to adapt and evolve with newly emerging music genres over time.

Furthermore, the project's applicability can be extended to real-time music genre classification, allowing for instantaneous identification and categorization of music as it is streamed or played. This could have implications in personalized music recommendation systems and contribute to a more immersive user experience.

Exploring the integration of transfer learning, where pre-trained models on large music datasets are fine-tuned for specific genres, could also be beneficial. Transfer learning has the potential to boost performance by leveraging knowledge gained from broader music corpora.

Finally, delving into the interpretability of the model's decisions and developing visualization techniques for understanding which features contribute most to genre classification could provide valuable insights and foster transparency in the decision-making process.

CHAPTER – 6

REFERENCES

1) Tom L. H. Li and Antoni B. Chan and Andy Hon Wai Chun, ”Automatic Musical Pattern Feature Extraction Using Convolutional Neural Network,” in Li2010AutomaticMP, 2010.
2) G. Tzanetakis and P. Cook, ”Musical genre classification of audio signals,” in IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560.
3) Hareesh Bahuleyan, ”Music Genre Classification using Machine Learning Techniques,” in arXiv:1804.01149, 2018.
4) J. Foote and S. Uchihashi, “The beat spectrum: A new approach to rhythmic analysis,” in Proc. Int. Conf. Multimedia Expo., 2001.
5) Matthew Creme, Charles Burlin, Raphael Lenain, “Music Genre Classification”, Stanford University, December 15, 2016.
6) Islam, Md, Hasan, Monirul, Rahim, Md Abdur, Hasan, Ali, Mynuddin,Mohammed, Khandokar, Imran, Islam, Jabbarul. (2021). Machine Learning-Based Music Genre Classification with Pre- Processed Feature Analysis.
7) F. Pachet and D. Cazaly, “A classification of musical genre,” in Proc.RIAO Content-Based Multimedia Information Access Conf., Paris, France, Mar. 2000.
8) S. Davis and P. Mermelstein, “Experiments in syllable-based recognition of continuous speech,” IEEE Trans. Acoust., Speech, Signal Processing, vol. 28, pp. 357–366, Aug. 1980.
9) J. Saunders, “Real time discrimination of broadcast speech/music,” in Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP), 1996, pp. 993–996.
10) Chang, Kaichun K., Jyh-Shing Roger Jang, and Costas S. Iliopoulos. ”Music Genre Classification via Compressive Sampling.” In ISMIR, pp. 387-392. 2010.
11) E. Scheirer and M. Slaney, “Construction and evaluation of a robust multifeature speech/music discriminator,” in Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP), 1997, pp. 1331–1334.
12) D. Kimber and L.Wilcox, “Acoustic segmentation for audio browsers,” in Proc. Interface Conf., Sydney, Australia, July 1996.
13) A.Ghildiyal, K. Singh and S. Sharma, ”Music Genre Classification using Machine Learning,” 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020.
14) T. Zhang and J. Kuo, “Audio content analysis for online audiovisual data segmentation and classification,” Trans. Speech Audio Processing, vol. 9, pp. 441–457, May 2001.
15) A.L. Berenzweig and D. P. Ellis, “Locating singing voice segments within musical signals,” in Proc. Int.Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) Mohonk, NY, 2001, pp. 119–123.
16) E. Wold, T. Blum, D. Keislar, and J. Wheaton, “Content-based classification, search, and retrieval of audio,” IEEE Multimedia, vol. 3, no. 2, 1996.
17) O’Shea, Keiron, and Ryan Nash. ”An introduction to convolutional neural networks.” arXiv preprint arXiv:1511.08458 (2015).
18) Bisharad, Dipjyoti, and Rabul Hussain Laskar. ”Music Genre Recognition Using Residual Neural Networks.” In TENCON 2019-2019 IEEE Region 10 Conference (TENCON), pp. 2063-2068. IEEE, 2019.
19) Zhang, Scott, Huaping Gu, and Rongbin Li. ”MUSIC GENRE CLASSIFICATION: NEAR-REALTIME VS SEQUENTIAL APPROACH.” (2019).
20) Chillara, Snigdha, A. S. Kavitha, Shwetha A. Neginhal, Shreya Haldia, and K. S. Vidyullatha. ”Music Genre Classification using Machine Learning Algorithms: A comparison.” (2019).
21) Yang, Hansi, and Wei-Qiang Zhang. ”Music Genre Classification Using Duplicated Convolutional Layers in Neural Networks.” In INTERSPEECH, pp. 3382-3386. 2019.
22) Gessle, Gabriel, and Simon A˚ kesson. ”A comparative analysis of CNN and LSTM for music genre classification.” (2019)

23) Defferrard, Micha¨el, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. ”Fma: A dataset for music analysis.” arXiv preprint arXiv:1612.01840 (2016).

24) O. Abdel-Hamid, A. -r. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, ”Convolutional Neural Networks for Speech Recognition,” in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 10, pp. 1533-1545, Oct. 2014, doi: 10.1109/TASLP.2014.2339736.

25) S. Vishnupriya and K. Meenakshi, ”Automatic Music Genre Classification using Convolution Neural Network,” 2018 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2018, pp. 1-4, doi: 10.1109/ICCCI.2018.8441340.

26) Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. ”Dropout: a simple way to prevent neural networks from overfitting.” The journal of machine learning research 15, no. 1 (2014): 1929-1958.

USER MANUAL

(Complete step by step instructions along with pictures necessary to run the project)

```
# Importing all the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy
import os
import pickle
import librosa
import librosa.display
import IPython.display as ipd
from IPython.display import Audio
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
```

```
# Reading the csv file
df = pd.read_csv("/content/drive/MyDrive/Project 7th/features_3_sec.csv")
df.head()


# Shape of the data
df.shape
```

```
# Data type of the data
df.dtypes
```

```
# Loading a sample audio from the dataset
audio ="../input/gtzan-dataset-music-genre-classification/Data/genres_original/
reggae/reggae.00010.wav"
data,sr=librosa.load(audio)
print(type(data),type(sr))
```

In order to work with audio data we use Librosa, a python package used for audio and music analysis. It is a powerful package widely used for audio visualization and for building MIR systems. We will be using the package for loading and visualizing the audio data.

```
# Initializing sample rate to 45600 we obtain the signal value array
```

```
librosa.load(audio,sr=45600)
```

```
# Taking Short-time Fourier transform of the signal
y = librosa.stft(data)
S_db = librosa.amplitude_to_db(np.abs(y), ref=np.max)
```

```
# Playing audio file
import IPython
IPython.display.Audio(data,rate=sr)
```

It is important to note that while working with any kind of audio data to solve any kind of problem statement, using only .wav format audio files is appropriate to analyze the data. If you are given audio files with .mp3 format you have to batch convert the data to waveforms using online software as .wav is the standard way of representing the audio files and it is the only way to work with audio data.

```
# Wave form of the audio
plt.figure(figsize=(7,4))
librosa.display.waveshow(data,color="#2B4F72", alpha = 0.5)
plt.show()
```

A spectrogram is a visual representation of the signal loudness of a signal over time at different frequencies included in a certain waveform. We can examine increase or decrease of energy over period of time. Spectrograms are also known as sonographs, voiceprints, and voicegrams.

```
# Spectrogram of the audio
stft=librosa.stft(data)
stft_db=librosa.amplitude_to_db(abs(stft))
plt.figure(figsize=(7,6))
librosa.display.specshow(stft_db,sr=sr,x_axis='time',y_axis='hz')
plt.colorbar()
```

Spectral roll off

It computes the rolloff frequency for each frame in a given signal. The frequency under which some percentage (cutoff) of the total energy of a spectrum is obtained. It can be used to differentiate between the harmonic and noisy sounds.

```
spectral_rolloff=librosa.feature.spectral_rolloff(y=data,sr=sr)[0]
plt.figure(figsize=(7,6))
librosa.display.waveshow(data,sr=sr,alpha=0.4,color="#2B4F72")
```

Chroma feature

It closely relates with the twelve different pitch classes. Chroma based features are also called as pitch class profiles. It is the powerful tool for analyzing and categorizing them. Harmonic and melodic characteristics of music are captured by them.

```
import librosa.display as lplt
chroma = librosa.feature.chroma_stft(y=data,sr=sr)
plt.figure(figsize=(7,4))
lplt.specshow(chroma,sr=sr,x_axis="time",y_axis="chroma",cmap="BuPu")
plt.colorbar()
plt.title("Chroma Features")
plt.show()
```

Zero Crossing Rate
It is the rate at which a signal transitions from positive to zero to negative or from negative to zero or simply said the number of times the signal crosses x-axis is as the zero-crossing rate (ZCR).

```
start=1000
end=1200
plt.figure(figsize=(12,4))
plt.plot(data[start:end],color="#2B4F72")
```

Exploratory Data Analysis(EDA)

Vizualizing the audio files, wave plots and spectrograms for all the 10 genre classes

K-Nearest Neighbors (KNN)

KNN is a fundamental Machine learning algorithm that is most commonly used among all kinds of problems. It classifies the data points based on the point that is near them by finding the euclidians distance given by d = ((x2-x1)^2 - (y2-y1)^2)^1/2 as a metric.

```
# Applying K nearest Neighbour algorithm to predict the results
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

clf1=KNeighborsClassifier(n_neighbors=3)
clf1.fit(X_train,y_train)
y_pred=clf1.predict(X_test)
print("Training set score: {:.3f}".format(clf1.score(X_train, y_train)))
print("Test set score: {:.3f}".format(clf1.score(X_test, y_test)))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.set(rc = {'figure.figsize':(8,3)})
sns.heatmap(cf_matrix, annot=True)
print(classification_report(y_test,y_pred))
```

Support Vector Machine (SVM)

SVM is one of the best machine learning models. Since the data is not linearly separable, we have used the SVM kernel function as sigmoid. The sigmoid function is given by K(yn,yi) = tanh(-gamma*(yn,yi)+r)

```
# Applying Support Vector Machines to predict the results
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf', degree=8)
svclassifier.fit(X_train, y_train)
print("Training set score: {:.3f}".format(svclassifier.score(X_train, y_train))
)
print("Test set score: {:.3f}".format(svclassifier.score(X_test, y_test)))
y_pred = svclassifier.predict(X_test)
cf_matrix3 = confusion_matrix(y_test, y_pred)
sns.set(rc = {'figure.figsize':(9,4)})
sns.heatmap(cf_matrix3, annot=True)
print(classification_report(y_test, y_pred))
```

Convolutional Neural Networks (CNN)

Using neural networks is the best way to classify huge data to draw predictions. Convolutions can solve the given problem very precisely and the algorithm has already been used most widely in classifying the image data.

```
# Training the model using the following parameters
# metrics = accuracy
# epochs = 600
# loss = sparse_categorical_crossentropy
# batch_size = 256
# optimizer = adam

def train_model(model,epochs,optimizer):
    batch_size=256
    model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy',metric
s='accuracy')
    return model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=epochs,
batch_size=batch_size)
                                                                          In [558]:
linkcode
def Validation_plot(history):
    print("Validation Accuracy",max(history.history["val_accuracy"]))
    pd.DataFrame(history.history).plot(figsize=(12,6))
    plt.show()
```

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

```
# We used different layers to train the neural network by importing keras library
from tensorflow framework
# for input and hidden neurons we use the most widly used activation function whi
ch is relu where as for output neurons we uses softmax activation function
model=tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(X.shape[1],)),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(512,activation='relu'),
    keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(256,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(64,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(32,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(10,activation='softmax'),
])

optimizer = tf.keras.optimizers.Adam(learning_rate=0.000146)
model.compile(optimizer=optimizer,
             loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
model.summary()
model_history=train_model(model=model,epochs=600,optimizer='adam')
```

```
# Plotting the confusion matrix for analizing the true positives and negatives
import seaborn as sn
import matplotlib.pyplot as plt
pred_x = model.predict(X_test)
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,predicted_index )
cm
```

Conclusion

As expected CNN outperformed KNN and SVM. It produced best results in both testing and taring data. As we increased the number of epochs the loss percentage decreased with a gradual increase in accuracy scores. It can be clearly seen in the above validation plot in which the curves almost coincided with each other.