MUSIC GENRE CLASSIFICATION USING DEEP LEARNING

A Project Report submitted in the partial fulfilment of the

Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

K .RAVI VARMA (20471A05F6)

Under the esteemed guidance of

Shaik Rafi,M.Tech.,(Ph.D)

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERINGCOLLEGE
(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Cycle -1 NIRF rank in the band of 251-320 and an ISO 9001:2015 Certified Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2023-2024

NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "MUSIC GENRE CLASSIFICATION USING DEEP LEARNING" is a bonfide work done by K.RAVI VARMA (20471A05F6) in partial fulfilment of the requirements for the award of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2023-2024.

PROJECT GUIDE

Shaik Rafi, M.Tech., (Ph.D)
Assistant Professor

PROJECT CO-ORDINATOR

Dr. M. Sireesha, M.Tech., Ph.D
Assoc. Professor

HEAD OF THE DEPARTMENT

Dr. S. N. Tirumala Rao, M.Tech, Ph.D
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

I declare that this project work titled “MUSIC GENRE CLASSIFICATION USING DEEP LEARNING” is composed by ourselves that the work contains here is our own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualifications except as specified.

By

K.Ravi Varma (20471A05F6)

ACKNOWLEDGEMENT

I wish to express our thanks to various personalities who are responsible for the completion of the project. I am extremely thankful to our beloved chairman sri **M.V.Koteswara Rao,B.Sc.,** who took keen interest in us in every effort throughout this course. I owe out sincere gratitude to our beloved principal **Dr.M.Sreenivasa Kumar, M.Tech., Ph.D., MISTE., FIE(I).,** for showing his kind attention and valuable guidance throughout the course.

I express our deep felt gratitude towards **Dr.S.N.Tirumala Rao,M.Tech.,Ph.D.,** HOD of CSE department and also to our guide **Shaik Rafi,M.Tech.,(Ph.D),** of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

I extend our sincere thanks towards **Dr.Sireesha.M,M.Tech.,Ph.D.,** Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

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

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

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

By

K.Ravi Varma (20471A05F6)

ABSTRACT

Music genre classification is a pivotal area of research within audio technology, holding immense importance for content organization and recommendation. Audio feature extraction and Music genre classification constitute a complete recognition system. Audio feature analysis and Music genre classification together form an integrated recognition system for comprehensive music genre identification and organization. This technology is frequently utilized to accurately detect and classify various types of music genres or characteristics present in audio signals, contributing significantly to the effective organization and recommendation of music content. Our experiment was conducted with the dataset from GTZAN that is taken from Kaggle repository. Convolutional neural networks (CNN) are employed to train our model, which is subsequently utilized for the classification of music genres in audio signals.

INDEX

LIST OF FIGURES

LIST OF TABLES

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

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

INSTITUTION MISSION

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

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

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

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

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

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

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

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

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

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

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

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

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

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

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

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

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

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

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

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

2. Problem analysis: Identify, formulate, research literature, and analyzecomplex engineering problems reaching substantiated conclusions using firstprinciples of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complexengineering problems and design system components or processes that meetthe specified needs with appropriate consideration for the public health andsafety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-basedknowledge and research methods including design of experiments, analysis andinterpretation of data, and synthesis of the information to provide validconclusions.

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

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

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

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

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

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

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

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

Project Course Outcomes (CO'S):

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

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

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

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

Course Outcomes – Program Outcomes mapping

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

Course Outcomes – Program Outcome correlation

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

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

1. **Low level**

2. **Medium level**

3. **High level**

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

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop a smart website to detect the genre of the music	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified and divided into six modules	PO2, PO3
CC421.2, C2204.2,C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3,C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4,C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2,C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be overseen by our team. Our project specializes in deep learning-based music genre classification. Future updates may involve enhancing accuracy and exploring new features for improved breed identification	PO4, PO7
C32SC4.3	The physical design includes hardware components like computers, servers to manage user http requests.	PO5, PO6

1. INTRODUCTION

1.1 Introduction

Genre classification serves as a cornerstone in the comprehension of music, yet its subjectivity often poses challenges. In the era of digitalization, the necessity for automated music classification becomes increasingly apparent, offering a solution to the daunting task of navigating extensive music libraries. This study delves into the realm of automatic music genre classification, aiming to highlight the efficacy of machine learning and deep learning techniques in accurately categorizing songs based on their audio signatures. By automating this process, the potential for expedited and tailored music discovery experiences on digital platforms is heightened.

Our investigation delves into a comparative analysis between traditional machine learning paradigms, such as Support Vector Machines (SVM), and advanced deep learning architectures like Convolutional Neural Networks (CNN) and Convolutional Recurrent Neural Networks (CRNN). Through this inquiry, we seek to uncover the nuanced strengths and limitations inherent in each approach within the context of music genre classification. Furthermore, our examination extends to evaluating the performance of machine learning classifiers utilizing both short-duration (three-second) and long-duration (thirty-second) audio features.

This study not only provides valuable insights into the potential of these models but also emphasizes the critical role of feature duration in refining classification accuracy. By shedding light on the interplay between machine learning, deep learning, and music genre classification, we aim to propel discussions surrounding the optimization of digital music platforms and the enhancement of user experiences within this evolving landscape.

1.2 LITERATURE SURVEY:

The examination of music sort classification has earned critical consideration in inquire about circles, where different techniques have been investigated to progress exactness and effectiveness. A few essential considers have contributed to this field. Analysts have dug into different approaches to refine the classification handle, pointing to way better get it and categorize melodic sorts. Here are a few striking ponders within the domain of music sort

classification.

Sturm et al.[1] proposed a music sort classification strategy utilizing scanty representation, accomplishing 83.00% accuracy on the GTZAN dataset. He extricated MFCC highlights, built genre-specific word references, and sp track utilizing iotas from its coordinating word reference.This approach viably utilized the discriminative control of genre-specific timbral characteristics.

Benetos et al.[2] investigated music sort classification through non-negative tensor factorization (NTF), coming to 75.00% accuracy on GTZAN. They spoken to music recordings as include tensors and created a novel NTF calculation. Sort classification was accomplished by breaking down the tensor into components capturing genre-specific designs and employing a administered classifier. Whereas not the beat entertainer at the time, this approach illustrated the potential of NTF for class classification and cleared the way for encourage headways in this region.

Bahuleyan et al.[7] displayed a strategy utilizing Convolutional Neural Systems (CNNs) for sound classification on the Sound Set dataset, accomplishing 65.00% accuracy. This approach included preparing a profound CNN design on a enormous collection of labeled sound recordings, permitting it to memorize complex designs specifically from the sound information. Whereas not particularly centered on music sort classification, this work showcased the potential of CNNs for sound assignments and propelled encourage investigate into their application for music class recognizable proof. It's critical to note that Audio Set could be a broader dataset enveloping different sounds past music sorts, making coordinate comparison with genre- specific datasets like GTZAN less direct.

Choi et al.[3] proposed a music class classification strategy utilizing Convolutional Neural Systems (CNNs) on the Million Tune Dataset, accomplishing an precision of 75.00%. Their approach, named Navier Music CNN, included preparing a profound CNN design on spectrograms determined from music sound. This permitted the organize to naturally learn pertinent highlights for class distinguishing proof straightforwardly from the unearthly representations.

These examinations grandstand a assorted cluster of methods and techniques utilized in anticipating music class classifications. They emphasize the noteworthiness of consolidating different highlights, counting melodic components, craftsman measurements, and relevant

variables, to improve the exactness of class expectations. By investigating a run of approaches, analysts point to create strong models that can viably categorize and distinguish melodic classes based on comprehensive highlights and characteristics.

1.3 Existing System:

In the realm of music organization and recommendation, decisions are traditionally driven by human expertise, leading to subjective judgments and potential inefficiencies. However, this manual approach can be error-prone, time-consuming, and may result in excessive costs, ultimately impacting the quality of service provided to users. Automated music genre classification, powered by machine learning and deep learning techniques, offers a transformative solution. By leveraging algorithms to process large volumes of audio data and extract meaningful features, platforms can streamline the categorization and recommendation of music content. This automation not only improves accuracy and efficiency but also reduces costs, leading to enhanced user experiences and satisfaction within the digital music ecosystem.

Disadvantages:

1. Manual methods of music genre classification often fall short in generating accurate and efficient results due to subjective judgments and biases, impacting user satisfaction.

2. Traditional approaches to music classification require considerable computation time, posing challenges for timely music recommendations and increasing operational costs.

3. Maintaining music genre records manually can be arduous and error-prone, leading to inconsistencies in music libraries and difficulties in organizing content effectively.

4. Inaccurate genre classification may result in inefficient music recommendations, depriving users of personalized content and hindering user engagement and platform credibility.

1.4 Proposed System

We propose the development of a system tailored to music genre classification, leveraging Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Convolutional Recurrent Neural Networks (CRNN). This system aims to assist practitioners in accurately categorizing songs based on various audio features such as rhythm, tempo,

timbre, and instrumentation. By harnessing the capabilities of SVM, CNN, and CRNN, known for their effectiveness in handling complex data and extracting intricate patterns, this project seeks to provide a reliable framework for automated music genre prediction.

The proposed system not only ensures precise genre predictions but also enhances the efficiency of the classification process, thereby facilitating streamlined music discovery and organization. By integrating SVM for robust classification and CNN/CRNN for feature extraction from audio signals, this system aims to deliver accurate and efficient genre predictions. Through the utilization of these advanced machine learning and deep learning techniques, practitioners can benefit from a comprehensive and effective tool for music genre classification, ultimately enhancing the user experience within digital music platforms.

Advantages:

1. Generates accurate and efficient results

2. Computation time is greatly reduced

3. Reduces manual work

4. Efficient further treatment

5. Automated prediction

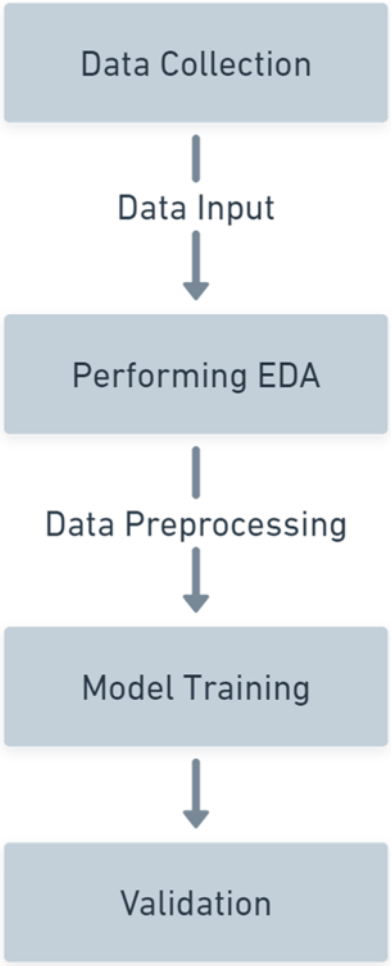


Fig 1.1 Proposed System

1.5 System Requirements

1.5.1 Hardware Requirements

- Processor : intel®core™i7-7500UCPU@2.70gh
- Cache memory : 4MB(Megabyte)
- RAM : 8 gigabyte (GB)

1.5.2 Software Requirements

- Operating System : Windows 10 Home, 64 bit Operating System
- Coding Language : Python
- Python distribution : Anaconda, Spyder

2. RELATED WORK

2.1 Machine Learning

Machine learning is one of the applications of artificial intelligence (AI) that provides computers, the ability to learn automatically and improve from experience instead of explicitly programmed. It focuses on developing computer programs that can access data and use it to learn from themselves. The main aim is to allow computers to learn automatically without human intervention and also adjust actions accordingly.

2.2 Deep Learning

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep Learning architectures such as deep neural networks, deep belief networks, graph neural networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

2.3 Some machine learning methods

Machine learning algorithms are often categorized as supervised and 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 is able to 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 in order to modify the model accordingly.

- In contrast, **unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning

studies how systems 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. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order 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 with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order 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.

2.4 Some Deep Learning Types

1. Feedforward neural network:

- This type of neural network is the very basic neural network where the flow control occurs from the input layer and goes towards the output layer.
- These kinds of networks are only having single layers or only 1 hidden layer.
- Since the data moves only in 1 direction there is no backpropagation technique in thisnetwork.
- In this network, the sum of the weights present in the input is fed into the input layer.
- These kinds of networks are used in the facial recognition algorithm using computervision.

2. Radial basis function neural networks:

- This kind of neural networks have generally more than 1 layer preferably two layers
- In this kind of networks, the relative distance from any point to the center iscalculated and the same is passed towards the next layer
- Radial basis networks are generally used in the power restoration systems to restorethe power in the shortest span of time to avoid the blackouts.

3. Multi-Layer perceptron:

- This type of network are having more than 3 layers and its used to classify the datawhich is not linear
- These kinds of networks are fully connected with every node.
- These networks are extensively used for speech recognition and other machinelearning technologies.

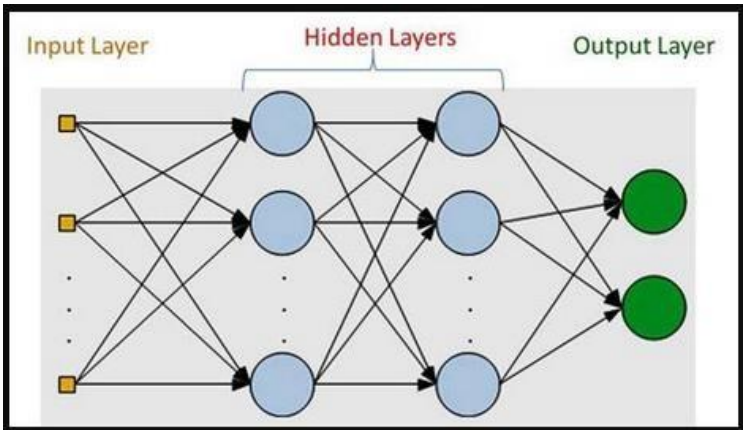


Fig 2.1 Multi-Layer Perceptron

4. Convolution Neural Network:

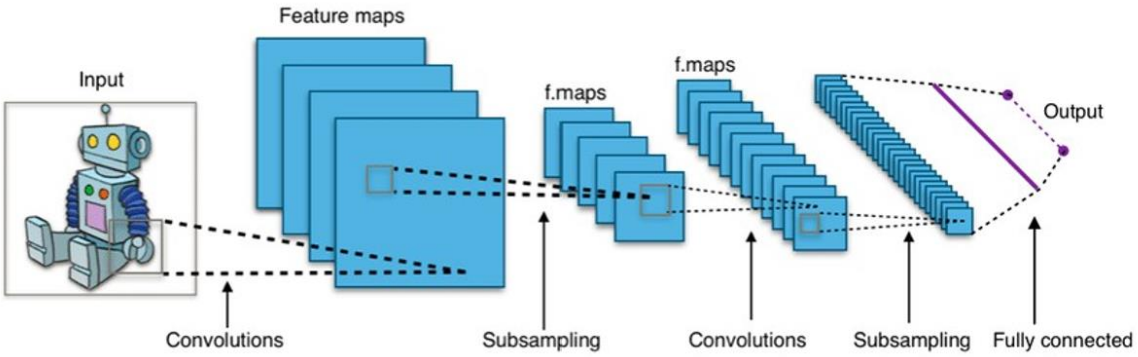


Fig 2.2 Convolutional Neural Network

- CNN is one of the variations of the multilayer perceptron.
- CNN can contain more than 1 convolution layer and since it contains a convolutionlayer the network is very deep with fewer parameters.
- CNN is very effective for image recognition and identifying different image patterns.

5. Recurrent Neural Network:

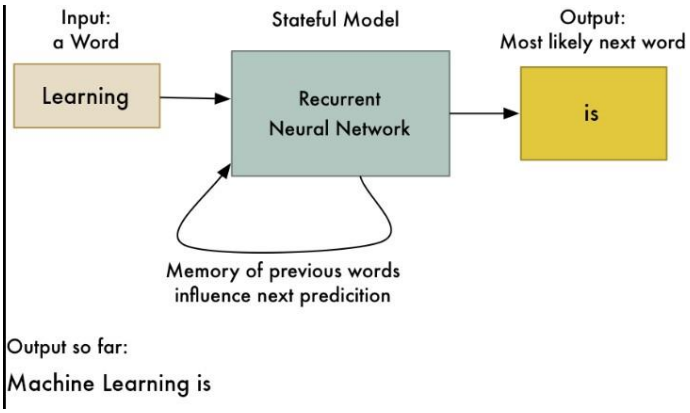


Fig 2.3 Recurrent Neural Network

2.5 Applications of machine learning and deep learning

1. Virtual Personal Assistants
2. Predictions while Commuting
3. Videos Surveillance
4. Social Media Services
5. Email Spam and Malware Filtering
6. Online Customer Support
7. Search Engine Result Refining
8. Product Recommendations
9. Online Fraud Detection

2.6 Prevalence of Music Genre Classification

The prevalence of music genre classification has skyrocketed in response to the explosion of digital music libraries and the growing demand for personalized content discovery. With millions of songs available across various platforms, manual categorization becomes impractical, necessitating the use of automated classification algorithms. These algorithms analyze the sonic characteristics of songs, such as rhythm, instrumentation, and timbre, to automatically assign them to specific genres, enabling efficient organization and navigation of vast music collections.

Furthermore, music genre classification plays a pivotal role in enhancing user experiences on streaming platforms and radio stations. By accurately categorizing songs, recommendation systems can generate personalized playlists and suggest similar artists, while radio stations can automate playlist generation and target specific audience demographics with tailored content. This prevalence underscores the importance of genre classification in modern music consumption, content curation, and audience engagement strategies.

2.7 Importance of machine learning in Music Genre Classification

The significance of machine learning in music genre classification stems from its capacity to efficiently process vast datasets containing complex audio features, surpassing human capabilities. By leveraging machine learning algorithms, such as Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Convolutional Recurrent Neural Networks (CRNN), music genre classifiers can reliably extract intricate patterns and features from audio signals. This analytical prowess enables the conversion of raw audio data into meaningful genre classifications, empowering music professionals and enthusiasts alike to explore and organize vast music libraries effortlessly.

Moreover, machine learning in music genre classification contributes to enhancing user experiences by providing personalized recommendations and playlists tailored to individual preferences. This not only enriches the enjoyment of music but also facilitates music discovery and exploration.

Furthermore, machine learning-driven genre classification has implications beyond just music organization and recommendation. It can also be applied in various music-related industries, such as radio broadcasting, music streaming services, and advertising, to optimize content delivery and engagement strategies.

In essence, machine learning in music genre classification facilitates efficient and accurate categorization of music, enabling better organization, recommendation, and utilization of musical content across various platforms and applications.

2.8 Architectural Methods for Deep Learning Algorithms

1. Back Propagation

In this algorithm, we calculate partial derivatives. In general, the gradient descent method for optimization, derivatives (gradients) are calculated at each iteration. In deep learning, functions are not simple; they are the composition of different functions. In this case, it is hard to calculate gradients, so we use approximate differentiation to calculate derivatives. The more the number of parameters, the more expensive approximate differentiation will be.

2. Stochastic Gradient Descent

In Gradient descent, the goal is to find global minima or optimum solution. But to get that, we have to consider local minima solutions (not desirable) also. If the objective function is a convex function, it is easy to find the global minima. The initial value for the function and learning rate are deciding parameters for finding global minima. This can easily be understood by considering a river from the mountain top and searching for a foothill (global minima). But in the way, there will be some ups and downs (local minima) which must be avoided. The river originating point and speed (initial value and learning rate in our case) are deciding factors to find global minima.

3. Learning Rate

The learning rate is like the speed of the river; it can reduce training time and increase performance. In general, to learn any technique/sport, in the beginning, the learning rate is relatively high than at the end when one is to master it. After the intermediate stage, the learning will be slow; the focus will be on fine-tuning. The same is applied in deep learning; too large changes are tackled by a higher learning rate and by slowly decreasing the learning rate later for fine-tuning.

4. Batch Normalization

In deep learning initial value of weight (randomly chosen) and learning rate is defined for a minibatch. In the beginning, there would be many outliers, and during backpropagation, these outliers must be compensated to compute the weights to get output. This compensation results in extra epochs. So to avoid it, we use batch normalization.

5. Drop Out

In deep learning, we generally encounter the problem of overfitting. Overfitting in large networks with several parameters makes it difficult to predict on test data. So, to avoid that, we use the dropout method, which drops random units during training by creating different

'thinned networks'. When testing these thinned networks' predictions are averaged, which helps to avoid overfitting.

2.9 Libraries of Machine Learning and Deep Learning

In the older days, people used to perform Machine Learning tasks manually by coding all the algorithms and mathematical and statistical formula. This made the process time consuming, tedious and inefficient. But in the modern days, it is become very much easy and efficient compared to the olden days by various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reason is its vast collection of libraries. Python libraries that used in Machine Learning are:

All the libraries which are generally used for deep learning are open source and few of themare as follows:

- Numpy
- Scipy
- Scikit-learn
- TensorFlow
- Keras
- PyTorch
- Pandas
- Matplotlib

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

Skikit-learn is one of the most popular Machine Learning libraries for classical Machine Learning algorithms. It is built on top of two basic Python libraries, NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit- learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with Machine Learning.

TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

Keras is a very popular Machine Learning library for Python. It is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One of the best thing about Keras is that it allows for easy and fast prototyping.

PyTorch is a popular open-source Machine Learning library for Python based on Torch, which is an open-source Machine Learning library which is implemented in C with a wrapper in Lua. It has an extensive choice of tools and libraries that supports on Computer Vision, Natural Language Processing(NLP) and many more ML programs. It allows developers to perform computations on Tensors with GPU acceleration and also helps in creating computational graphs.

Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

Matpoltlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, histogram, error charts, bar chats, etc.

2.10 Layers in Convolutional Neural Networks

Below are the Layers of convolutional neural networks:

1. **Image Input Layer:**

   The input layer gives inputs (mostly images), and normalization is carried out. Inputsize has to be mentioned here.

2. **Convolutional Layer:**

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

3. **Non-Linearity Layer:**

   Here feature maps are taken as input, and activation maps are given as output with the help of the activation function. The activation function is generally implemented as sigmoid or hyperbolic tangent functions.

4. **Rectification Layer:**

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

5. **Rectified Linear Units(ReLU):**

   ReLU combines non-linear and rectification layers on CNN. This does the

threshold operation where negative values are converted to zero. However, ReLU doesn't changethe size of the input.

6. **Pooling Layer:**

   The pooling layer is also called the downsampling layer, as this is responsible for reducing the size of activation maps. A filter and stride of the same length are applied

   to the input volume. This layer ignores less significant data; hence image recognition is done in a smaller representation. This layer reduces overfitting. Since the amount of parameters is reduced using the pooling layer, the cost is also reduced. The input is divided into rectangular pooling regions, and either maximum or average is calculated, which returns maximum or average consequently. Max Pooling is a popular one.

7. **Dropout Layer:**

   This layer randomly sets the input layer to zero with a given probability. More results in different elements are dropped after this operation. This layer also helps to reduce overfitting. It makes the network to be redundant.

8. **Fully Connected Layer:**

   Activation maps, which are the output of previous layers, is turned into a classprobability distribution in this layer. FC layer multiplies the input by a weight matrix and adds the bias vector.

9. **Output Layer:**

   FC layer is followed by softmax and classification layers. The softmax function is applied to the input. The classification layer computes the cross-entropy and loss function for classification problems.

10. **Regression Layer:**

    Half the mean squared error is computed in this layer. This layer should follow the FClayer.

2.11 Common steps for any TensorFlow based Algorithms:

The basic steps of TensorFlow algorithm are:

Step 1: Data is Imported/Generated: TensorFlow Models depends heavily on the huge amount of Data. Either you can import your own dataset or TensorFlow also comes with the collection of Type this command to check out available datasets in TensorFlow.

```
import TensorFlow as tf

import TensorFlow datasets as tendata

#This command will generate a list of datasets available in the
TensorFlowprint(tfds.list_builders())
```

Step 2: Data Normalization or Transformation: If the data is not in the appropriate forum. The Batch Normalization is the command approach used to normalize data in the TensorFlow.

Step 3: Set the Parameters of the Algorithm: For eg; the number of Iterations, Learning rate,etc.

Step 4: Set and initialize the variables and Placeholders: Variables and Placeholders are two basic programming Elements of the TensorFlow. Variables hold the state of the graph andplaceholders are used to feed the data in the graph at the later date.

Step 5: Create Model structure: What operations will be performed on the data is defined.

Step 6: Define the Loss Function: It calculates the difference between predicted values andactual values. It tells how well your model is trained basically used to evaluate the output.

Step 7: Train Model: Initialize computational graph and create an Instance of a graph. Feed data into the model with the help of placeholders and let the TensorFlow do the rest of the processing for better predictions.

Step 8: Evaluate the performance: Evaluate the model by checking with new data.

Step 9: Predict the Outcome: Also checks your model on new and unseen data.To better visualize model TensorFlow provides Tensorboard. It helps us to visualize any statistics of theneural network, debug and optimize them. You can check what happens in the code and will give you a detailed understanding of the inner working. You can fix problems very easily with the help of this tool. Tensorboard provides five types of Visualizations:

- Scalars
- Images
- Audio
- Histograms
- Graphs

3. SYSTEM ANALYSIS

3.1 Scope of the project

This project primarily focuses on predicting music genres based on various audio features. The scope of this project entails users being able to determine the likelihood of a song belonging to a particular genre based on specific attributes. The aim is to enable users to assess the genre classification of songs and ascertain whether they accurately represent the intended genre or not.

3.2 Analysis

The dataset utilized in this project is the GTZAN dataset, obtained from the GTZAN repository. This dataset comprises 60 features utilized for music genre classification. Details regarding the specific attributes are outlined in the table below

Table-1.Dataset Description

Column Name	Description
filename	Name of the audio file
length	Duration of the audio file (in seconds)
chroma_stft_mean	Mean of the Chroma Short-Time Fourier Transform
chroma_stft_var	Variance of the Chroma Short-Time Fourier Transform
rms_mean	Root Mean Square value of the audio signal
rms_var	Variance of the Root Mean Square value
spectral_centroid_ mean	Mean of the Spectral Centroid
spectral_centroid_ var	Variance of the Spectral Centroid
spectral_bandwidth_mean	Mean of the Spectral Bandwidth
spectral_bandwidth_var	Variance of the Spectral Bandwidth
rolloff_mean	Mean of the Spectral Roll-off
rolloff_var	Variance of the Spectral Roll-off
zero_crossing_rate_mean	Mean of the Zero-Crossing Rate
zero_crossing_rate_var	Variance of the Zero-Crossing Rate
harmony_mean	Mean of the harmony feature
harmony_var	Variance of the harmony feature

perceptr_mean	Mean of the perceptual entropy
perceptr_var	Variance of the perceptual entropy
tempo	Tempo of the audio file
mfcc1_mean - mfcc20_var	Mean and Variance of Mel-Frequency Cepstral Coefficients 1 to 20
label	Genre label of the audio file

filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	...	mfcc19_mean	mfcc19_var	mfcc20_mean	mfcc20_var	label
pop.00025.8.wav	66149	0.335017	0.086521	0.170618	...	-7.469169	62.932865	-6.115783	27.100780	pop
country.00080.5.wav	66149	0.418494	0.079523	0.146645	...	-0.394299	33.864754	-4.276742	29.910364	country
country.00096.0.wav	66149	0.296502	0.087063	0.166526	...	6.430115	48.474289	4.764173	30.933424	country
jazz.00055.5.wav	66149	0.337066	0.085436	0.077174	...	0.533961	36.287605	6.028970	79.830788	jazz
hiphop.00000.3.wav	66149	0.537404	0.069471	0.127345	...	-7.540948	41.904297	1.986606	17.988197	hiphop
disco.00024.2.wav	66149	0.355544	0.081845	0.140602	...	2.575610	33.358250	4.583081	48.475796	disco
disco.00022.9.wav	66149	0.471858	0.086836	0.136385	...	-0.264330	35.182247	-0.077982	38.113560	disco
pop.00081.8.wav	66149	0.361000	0.080877	0.102693	...	-1.979990	25.965775	-0.874761	33.864761	pop
jazz.00013.8.wav	66149	0.350711	0.090876	0.061404	...	-4.432117	42.089867	1.360967	55.540588	jazz
pop.00022.8.wav	66149	0.392248	0.091105	0.239263	...	-1.281341	54.172432	0.123123	63.891014	pop

Fig 3.1 Dataset

3.3 Data Pre-processing

Compelling pre-processing procedures are necessarily to planning input information for a music sort classification show, particularly given the significant measure of our dataset. We utilize a assorted set of pre-processing strategies to handle challenges such as exceptions and clamor inborn in music information. Different strategies, counting name encoding and coding strategies like PowerTransformer, StandardScaler, and MinMaxScaler, are connected to make a comprehensive technique for dealing with both categorical and numerical highlights.

The basis behind utilizing these pre-processing strategies lies in their capacity to improve the vigor and execution of the music sort classification demonstrate. Name encoding guarantees that categorical highlights are fittingly changed over into numerical arrange, encouraging the compatibility of information with machine learning calculations. In the interim, procedures like PowerTransformer, StandardScaler, and MinMaxScaler play a vital part in normalizing and scaling numerical highlights, tending to issues related to changing scales and disseminations.

Particular consideration is coordinated towards refining key highlights such as 'chroma_stft,'

'spectral_centroid,' and 'tempo.' Methods like PowerTransformer with Yeo-Johnson, StandardScaler, and MinMaxScaler are deliberately connected to guarantee compelling normalization and scaling for these highlights. This fastidious approach points to relieve potential challenges like predisposition and overfitting, contributing to the by and large vigor of the show. The consolidation of a ColumnTransformer advance upgrades the pre-processing pipeline by adeptly isolating categorical and numerical highlights. This isolation empowers custom fitted pre-processing for each sort of include, optimizing the planning of information for consequent stages within the classification demonstrate. In substance, the intensive and comprehensive pre-processing methodologies utilized in this ponder are adapted towards ensuring that the input information is well-conditioned, minimizing commotion and exceptions, and maximizing the model's capacity to memorize important designs from the music class dataset.

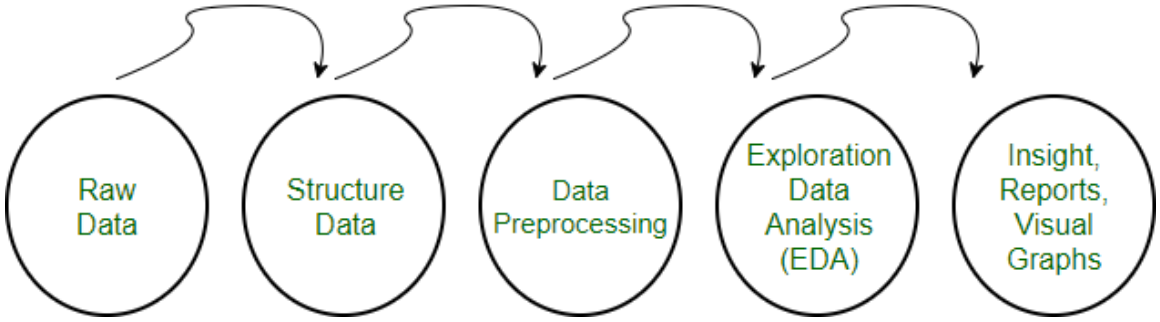


Fig 3.2 Data Pre-processing

3.4 Yeo-Johnson :

I have applied the Yeo-Johnson transformation to normalize skewed distributions, thereby converting them into more symmetrical, approximately normal distributions. This transformation aims to enhance the performance of classification models by improving the distribution of the data, facilitating more accurate predictions and analysis.

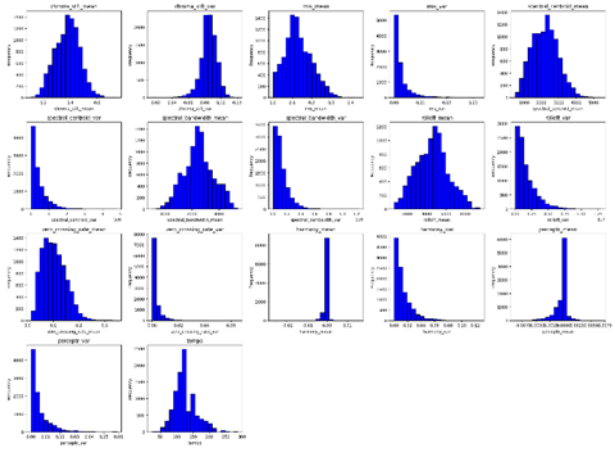


Fig 3.3 Class distribution before Yeo-Johnson

The figure depicts the skewed distribution of the data, which can result in improper classification. Therefore, the Yeo-Johnson technique is applied to transform the skewed distribution into a normal distribution.

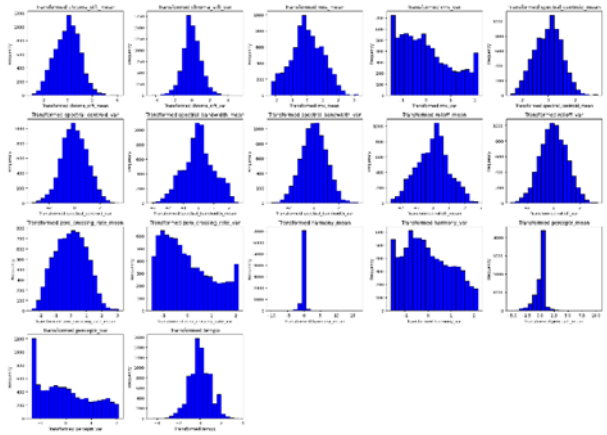


Fig 3.4 Class distribution after Yeo-Johnson

The figure depicts the normal distribution of the data after applying the Yeo-Johnson.

3.5 Feature Selection:

Under the feature selection aspect, all features within the GTZAN dataset were included without any exclusions. No columns were removed during preprocessing, ensuring that the entirety of the dataset was utilized for analysis. This comprehensive approach aimed to capture the full spectrum of information contained within the dataset, allowing for a thorough exploration of all potential predictors for music genre classification. By considering all

features, the analysis sought to maximize the predictive power of the model and facilitate a comprehensive understanding of the relationships between various attributes and music genres.

3.6 Classification:

In this project, several classification algorithms have been explored to determine the most effective approach for music genre classification. The choice of algorithm depends on the characteristics of the dataset and the specific requirements of the task. For instance, if the dataset exhibits complex patterns and relationships, more sophisticated algorithms may be necessary.

The algorithms employed in this project include Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Convolutional Recurrent Neural Network (CRNN). These algorithms are applied to two different feature sets extracted from the songs: the 3-second feature set and the 30-second feature set.

SUPPORT VECTOR MACHINE:

Support Vector Machine is a supervised machine learning algorithm can be used for both classification and regression techniques. Mostly it is used for the classification problems

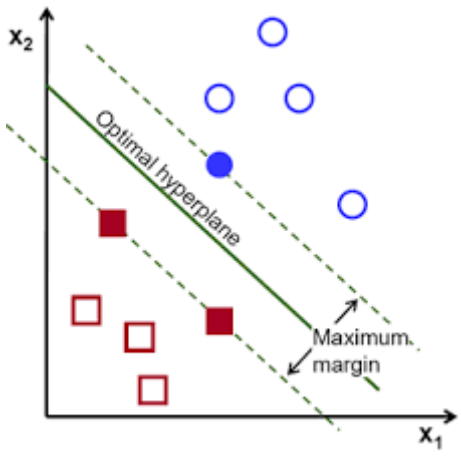


Fig 3.5 Support Vector Machine

Figure3.6.2 describes about the support vector machine. It plots each record as a point in n-dimensional space with value of each feature being value of particular coordinate. Where n is number of features. Then classification is performed by finding hyper-plane which differentiates the two classes.

We utilized the Support Vector Machines (SVM) algorithm as the cornerstone of our machine-learning model, implementing it through the Scikit-Learn library. To fine-tune the SVM model, an exhaustive search for optimal hyperparameters was conducted, resulting in the selection of the following values:

'C': 42

'gamma': 1.52

'kernel': 'poly'

These specific hyperparameter choices were determined to deliver optimal performance within our experimental setup. In the subsequent sections, we will delve into comprehensive descriptions of our SVM models, covering architectural details, hyperparameter tuning specifics, and intricacies of the training process.

CNN (Convolutional Neural Network):

CNNs are commonly used for image classification tasks, but they can also be adapted for audio classification, as they can effectively learn hierarchical representations of audio features. CNNs are particularly suited for extracting features from spectrograms or other time-frequency representations of audio signals.

The Convolutional Neural Network (CNN) architecture in this research was constructed using Keras. The CNN built here has an input layer and five convolutional blocks, with each convolutional block consisting of the following:

- Convolutional Layer:

  Mirrored padding

  1x1 stride

  3x3 filter

- Activation Function:

  Rectified Linear Activation (ReLU)

- Maximum Pooling Layer:

  2x2 stride

  Window size

- Dropout Regularization:

  Probability of 0.2 for dropout

- Dense Layers:

  Four dense layers with 512, 256, 128, and 64 units respectively

  Each followed by ReLU activation and 0.2 dropout

- Output Layer:

  10 units with softmax activation for multiclass classification

- Training Configuration:

  Loss Function: Sparse Categorical Crossentropy

  Optimizer: Adam

  Training Epochs: 200

  Batch Size: 32

  Validation Split: 20%

CRNN (Convolutional Recurrent Neural Network):

CRNN combines the strengths of CNNs and Recurrent Neural Networks (RNNs) by incorporating both convolutional and recurrent layers. This architecture is well-suited for sequential data like audio signals, as it can capture both temporal dependencies and spatial features.

The Convolutional Recurrent Neural Network (CRNN) architecture in this research was constructed using Keras. The CRNN model built here has an input layer and five repeating blocks, with each CRNN block consisting of the following:

- Convolutional Layers:

  Conv1D layer with 64 filters, 3 kernel size, and ReLU activation

  MaxPooling1D layer with a pool size of 2

- Additional Convolutional Layers:

  Conv1D layer with 128 filters, 3 kernel size, and ReLU activation

  MaxPooling1D layer with a pool size of 2

- Dense Layers:

  Four dense layers with 512, 256, 128, and 64 units respectively

  Each followed by ReLU activation and 0.2 dropout

- Output Layer:

  10 units with softmax activation for multiclass classification

- Compilation and Training Configuration:

  Optimizer: Adam

  Loss Function: Sparse Categorical Crossentropy

  Training Epochs: 200

  Batch Size: 32

  Validation Split: 20%

K-FOLD CROSS VALIDATION: Cross Validation is a resampling process used to evaluate machine learning models[19].

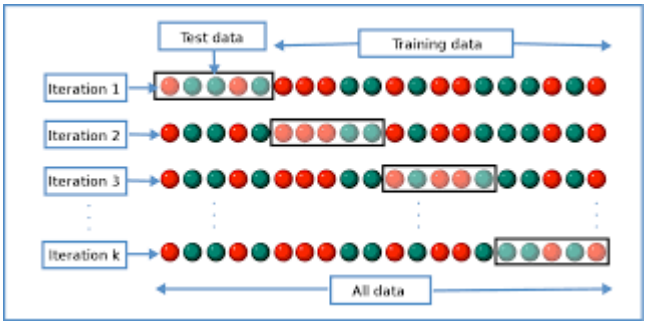


Fig 3.6 K-Fold Cross Validation

Figure shows how cross validation process is done. The process has a single parameter k. Where k is number of groups that a given data sample is to be split into. When k value is chosen, it is used in place of k, such as k=5 as 5-fold cross validation. In the first iteration (k-1) data samples are taken as training data and kth data is taken as test data and in the next iteration kth data sample which is taken as test data is replaced by one of data sample in (k-1) and vice versa. The process continues until k number of iterations which covers all data samples.

3.7 CONFUSION MATRIX

Performance Evaluation of classification algorithm is calculated by using confusion matrix[20]. Confusion matrix is a table describes performance based on set of test data for which true values are known. Performance is calculated by considering actual and predicted class. A confusion matrix is a table that is often used to describe the performance of a classification model(or "classifier") on a set of test data for which true values are known.

A true positive (tp) is a result where the model predicts the positive class correctly.Similarly, a true negative (tn) is an outcome where the model correctly predicts the negative class.

A false positive (fp) is an outcome where the model incorrectly predicts the positive class.And a false negative (fn) is an outcome where the model incorrectly predicts the negative class.

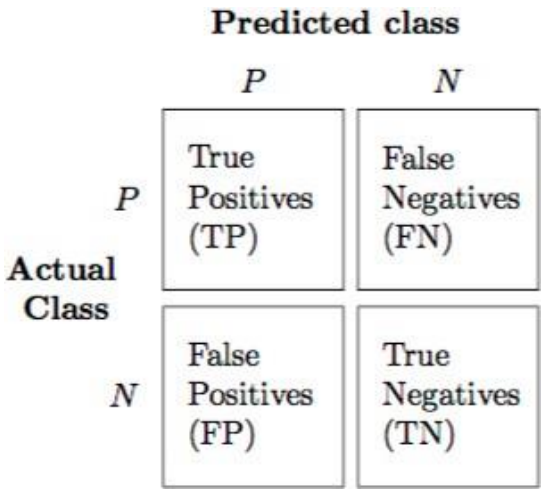


Fig 3.7 Confusion Matrix

Sensitivity or Recall or hit rate or true positive rate (TPR)

It is the proportion of individuals who actually have the disease were identified as having the disease.

TPR = tp / (tp + fn)

Specificity, selectivity or true negative rate (TNR)

It is the proportion of individuals who actually do not have the disease were identified as not having the disease.

TNR = tn / (tn + fp) =1-FPR

Precision or positive predictive value (PPV)

If the test result is positive what is the probability that the patient actually has the disease.
PPV = tp / (tp + fp)

Negative predictive value (NPV)

If the test result is negative what is the probability that the patient does not have disease.
NPV = tn / (tn + fn)

Miss rate or false negative rate (FNR)

It is the proportion of the individuals with a known positive condition for which the test result is negative.

FNR = fn / (fp + tn)

Fall-out or false positive rate (FPR)

It is the proportion of all the people who do not have the disease who will be identified as having the disease.

FPR = fp/ (fp + tn)

False discovery rate (FDR)

It is the proportion of all the people identified as having the disease who do not have the disease.

FDR = fp / fp + tp

False omission rate (FOR)

It is the proportion of the individuals with a negative test result for which the true condition is positive.

FOR = fn / (fn + tn)

Accuracy

The accuracy reflects the total proportion of individuals that are correctly classified.
Accuracy = (tp + tn) / (tp + tn + fp + fn)

F1 score

It is the harmonic mean of precision and sensitivity
F1 = 2tp / (2tp+ fp + fn)

4.DESIGN ANALYSIS

In this project the dataset utilized is GTZAN dataset which is taken from kaggle repository. The null values are replaced with mean and all features are scaled in first step. The transformations are applied to the dataset like standard scaler and yeo-johnson and normalizationa in the second step. The CNN model is trained to predict the genre of the music.

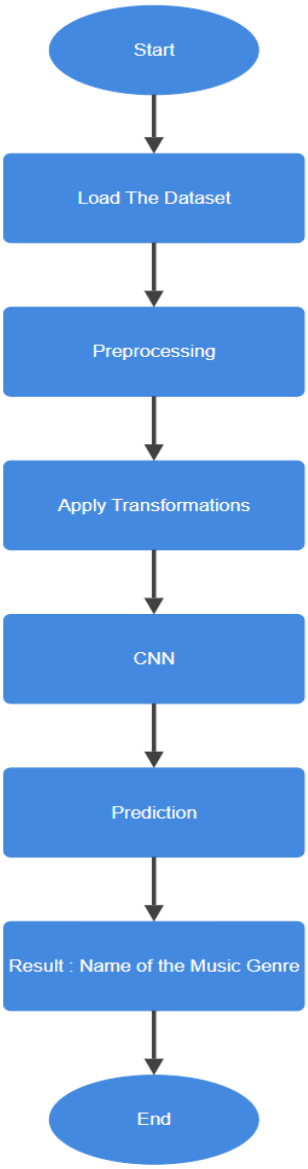


Fig:4.1 Data Flow Diagram

5. IMPLEMENTATION CODE

5.1 Deep Learning Model Creation Code:

Import Libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

Import Dataset

```
df = pd.read_csv("features_3_sec.csv")

df.head()

df.shape

df.columns

df.dtypes

duplicated_rows = df[df.duplicated()]

duplicated_rows

null_values = df.isnull().sum()

null_values

df.describe()

plt.figure(figsize=(12, 6))

sns.countplot(x='label', data=df)

plt.title('Genre Distribution')
```

```
plt.show()

import librosa

import librosa.display

import IPython

#audio file

audio = "genres_original/classical/classical.00035.wav"

#Load & decode the audio as a time series, where sr represents the sampling rate

data , sr = librosa.load(audio)

print(type(data))

print(type(sr))

data

sr

IPython.display.Audio(data, rate=sr)

plt.figure(figsize=(12,4))

librosa.display.waveshow(data, color = "green")

plt.show()

stft = librosa.stft(data)

stft_db = librosa.amplitude_to_db(abs(stft))

plt.figure(figsize=(12,4))

librosa.display.specshow(stft, sr=sr, x_axis='time', y_axis='hz')

plt.colorbar()

stft = librosa.stft(data)
```

```
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize=(12,4))
librosa.display.specshow(stft_db, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

Plotting of Data

```
columns_to_plot = ['chroma_stft_mean', 'chroma_stft_var', 'rms_mean',
                   'rms_var', 'spectral_centroid_mean', 'spectral_centroid_var',
                   'spectral_bandwidth_mean', 'spectral_bandwidth_var', 'rolloff_mean',
                   'rolloff_var', 'zero_crossing_rate_mean', 'zero_crossing_rate_var',
                   'harmony_mean', 'harmony_var', 'perceptr_mean', 'perceptr_var', 'tempo']
# Set the figure size
plt.figure(figsize=(20, 40))
# Loop through each column and plot a histogram
for column in columns_to_plot:
    plt.subplot(11, 5, columns_to_plot.index(column) + 1)
    plt.hist(df[column], bins=20, color='blue', edgecolor='black')
    plt.title(column)
    plt.xlabel(column)
    plt.ylabel('Frequency')
# Adjust layout
plt.tight_layout()
```

```
# Show the plot
plt.show()
plt.figure(figsize=(12, 8))
sns.boxplot(data=df)
plt.title('Boxplot for Each Column')
plt.show()
columns_to_plot = ['chroma_stft_mean', 'chroma_stft_var', 'rms_mean',
                   'rms_var', 'spectral_centroid_mean', 'spectral_centroid_var',
                   'spectral_bandwidth_mean', 'spectral_bandwidth_var', 'rolloff_mean',
                   'rolloff_var', 'zero_crossing_rate_mean', 'zero_crossing_rate_var',
                   'harmony_mean', 'harmony_var', 'perceptr_mean', 'perceptr_var', 'tempo']
plt.figure(figsize=(20, 40))
for i, column in enumerate(columns_to_plot, start=1):
    plt.subplot(11, 5, i)
    plt.boxplot(df[column])
    plt.title(column)
    plt.ylabel('Values')
plt.tight_layout()
plt.show()
X = df.iloc[:, 2:-1]
y = df['label']
X
```

```
y
```

Splitting The Data

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

y_labeled = label_encoder.fit_transform(y)

y_labeled

label_mapping=dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)))

print("Label Mapping:")

for label, numerical_value in label_mapping.items():

    print(f"{label}: {numerical_value}")



from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y_labeled, test_size=0.2,
random_state=42)

X.columns
```

Applying Column Transformer

```
from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import PowerTransformer, StandardScaler, MinMaxScaler

yeo_johnson_cols = ['chroma_stft_mean', 'chroma_stft_var', 'rms_mean',

                    'rms_var', 'spectral_centroid_mean', 'spectral_centroid_var',

                    'spectral_bandwidth_mean', 'spectral_bandwidth_var', 'rolloff_mean',

                    'rolloff_var', 'zero_crossing_rate_mean', 'zero_crossing_rate_var',
```

```
                'harmony_mean', 'harmony_var', 'perceptr_mean', 'perceptr_var', 'tempo']
    standard_scaling_cols = list(X.columns)
    normalization_cols = list(X.columns)
    preprocessor = ColumnTransformer(
        transformers=[
            ('yeo_johnson',PowerTransformer(method='yeo-johnson',standardize=True),
yeo_johnson_cols),
            ('standard_scaling', StandardScaler(), standard_scaling_cols),
            ('normalization', MinMaxScaler(), normalization_cols)
        ],
        remainder='passthrough'  # Include columns not specified in transformers
    )
    x_train_transformed = preprocessor.fit_transform(x_train)
    x_test_transformed = preprocessor.transform(x_test)
    x_train_transformed
    x_test_transformed
```

Plotting of Data Ater Applying Column Transformer

```
    # Plot the histograms for transformed data for specific columns
    plt.figure(figsize=(20, 40))
    for i, column in enumerate(columns_to_plot, start=1):
        plt.subplot(11, 5, i)
        plt.hist(x_train_transformed[:, columns_to_plot.index(column)], bins=20, color='blue',
edgecolor='black')
```

```
        plt.title(f'Transformed {column}')
        plt.xlabel(f'Transformed {column}')
        plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
    # Boxplots for transformed data for specific columns
    plt.figure(figsize=(20, 40))
    for i, column in enumerate(columns_to_plot, start=1):
        plt.subplot(11, 5, i)
        plt.boxplot(x_train_transformed[:, columns_to_plot.index(column)])
        plt.title(f'Transformed {column}')
        plt.ylabel('Values')
    plt.tight_layout()
    plt.show()
```

CNN Model

```
    import tensorflow as tf
    from sklearn.metrics import accuracy_score
    from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import PowerTransformer, StandardScaler, MinMaxScaler
    from sklearn.model_selection import train_test_split
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(512,activation="relu",
```

```
input_shape=(x_train_transformed.shape[1],)),
        tf.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(10, activation="softmax")
    ])

    # Print the model summary
    print(model.summary())
    # Assuming you have a trainModel function that compiles and fits the model
    def trainModel(model, epochs, optimizer):
        model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
        model.fit(x_train_transformed,y_train,epochs=epochs,validation_split=0.2,
batch_size=32)
```

```
    return model
# Train the model
model_history = trainModel(model=model, epochs=200, optimizer='adam')
plt.plot(model_history.history.history['accuracy'], label='Training Accuracy')
plt.plot(model_history.history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy vs. Epoch')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
y_probabilities = model.predict(x_test_transformed)
# Get the predicted classes
y_pred = tf.argmax(y_probabilities, axis=1).numpy()
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on the test set: {accuracy * 100:.2f}%")
y_pred
y_test
from sklearn import metrics
from sklearn.metrics import ConfusionMatrixDisplay


confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
classes = sorted(set(y_test))
```

```
cm_display=ConfusionMatrixDisplay(confusion_matrix=confusion_matrix,
display_labels=classes)

cm_display.plot()

plt.show()

from sklearn.metrics import classification_report

# Generate the classification report

classification_rep=classification_report(y_test,y_pred,
target_names=label_encoder.classes_, output_dict=True)

# Access the accuracy for each class

class_accuracies = {class_name: report['precision'] for class_name, report in
classification_rep.items() if class_name in label_encoder.classes_}

# Print the class-wise accuracies

for class_name, accuracy in class_accuracies.items():

    print(f"Accuracy for {class_name}: {accuracy * 100:.2f}%")
```

K-FOLd cross Validation

```
from sklearn.model_selection import StratifiedKFold

from sklearn.metrics import accuracy_score

import numpy as np

# Perform k-fold cross-validation

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# List to store validation accuracies for each fold

fold_accuracies = []

for fold, (train_index, val_index) in enumerate(kfold.split(X, y_labeled), 1):
```

```
    print(f"Fold {fold}")


    # Split data into train and validation sets
    x_train, x_val = X.iloc[train_index], X.iloc[val_index]
    y_train, y_val = y_labeled[train_index], y_labeled[val_index]


    # Apply feature transformations
    x_train_transformed = preprocessor.fit_transform(x_train)
    x_val_transformed = preprocessor.transform(x_val)
    # Define your model
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(512,activation="relu",
input_shape=(x_train_transformed.shape[1],)),
        tf.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(10, activation="softmax")
        ])


        # Compile and fit the model
        model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
        model.fit(x_train_transformed,y_train,epochs=200,
validation_data=(x_val_transformed, y_val), batch_size=32, verbose=0)
        # Evaluate the model on the validation set
        val_loss, val_accuracy = model.evaluate(x_val_transformed, y_val, verbose=0)
        fold_accuracies.append(val_accuracy)
        print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")
    # Calculate and print the average accuracy
    average_accuracy = np.mean(fold_accuracies)
    print(f"\nAverage Accuracy: {average_accuracy * 100:.2f}%")
```

Saving Model and Preprocessing

```
    import pickle
    with open('label_encoder.pkl', 'wb') as le_file:
        pickle.dump(label_encoder, le_file)
    # Save ColumnTransformer
    with open('column_transformer.pkl', 'wb') as ct_file:
```

```
        pickle.dump(preprocessor, ct_file)

    model.save('cnn_model.h5')
```

CRNN Model

```
    crnn_model_transformed = tf.keras.models.Sequential([

        tf.keras.layers.Reshape((x_train_transformed.shape[1],1),
input_shape=(x_train_transformed.shape[1],)),


        tf.keras.layers.Conv1D(64, 3, activation='relu'),

        tf.keras.layers.MaxPooling1D(2),


        tf.keras.layers.Conv1D(128, 3, activation='relu'),

        tf.keras.layers.MaxPooling1D(2),


        tf.keras.layers.LSTM(64, return_sequences=True),


        tf.keras.layers.Flatten(),


        tf.keras.layers.Dense(512, activation="relu"),

        tf.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(10, activation="softmax")
    ])

    # Compile and train the CRNN model on transformed data
    crnn_model_transformed.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    crnn_model_transformed.fit(x_train_transformed,y_train,epochs=200,
validation_split=0.2, batch_size=32)
    # Evaluate the CRNN model on the transformed test set
    crnn_accuracy_transformed=crnn_model_transformed.evaluate(x_test_transformed,
y_test)[1]
    print(f"CRNN Accuracy on the transformed test set: {crnn_accuracy_transformed *
100:.2f}%")
    y_pred_crnn = crnn_model_transformed.predict(x_test_transformed)
    # If you want the predicted class labels, you can use argmax
    predicted_labels = np.argmax(y_pred_crnn, axis=1)
```

```
    accuracy = metrics.accuracy_score(y_test, predicted_labels)
    print(f'Accuracy: {accuracy * 100:.2f}%')
    confusion_matrix = metrics.confusion_matrix(y_test, predicted_labels)
    classes = sorted(set(y_test))
    # Plot the confusion matrix
    cm_display=ConfusionMatrixDisplay(confusion_matrix=confusion_matrix,
display_labels=classes)
    # Display the confusion matrix plot
    cm_display.plot()
    plt.show()
    crnn_model_transformed = tf.keras.models.Sequential([
        tf.keras.layers.Reshape((x_train_transformed.shape[1],1),
input_shape=(x_train_transformed.shape[1],)),

        tf.keras.layers.Conv1D(128, 3, activation='relu'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.MaxPooling1D(2),

        tf.keras.layers.Conv1D(256, 3, activation='relu'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.MaxPooling1D(2),

        tf.keras.layers.LSTM(128, return_sequences=True),
```

```
        tf.keras.layers.LSTM(64, return_sequences=True),

        tf.keras.layers.Flatten(),

        tf.keras.layers.Dense(512,activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.5),

        tf.keras.layers.Dense(256,activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.5),

        tf.keras.layers.Dense(128,activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.5),

        tf.keras.layers.Dense(64,activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.5),
```

```
        tf.keras.layers.Dense(10, activation="softmax")
    ])
    # Compile and train the CRNN model on transformed data

crnn_model_transformed.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                               loss='sparse_categorical_crossentropy',
                               metrics=['accuracy'])
    crnn_model_transformed.fit(x_train_transformed,y_train,epochs=300, validation_split=0.2, batch_size=64)
    # Evaluate the CRNN model on the transformed test set
    crnn_accuracy_transformed=crnn_model_transformed.evaluate(x_test_transformed, y_test)[1]
    print(f"CRNN Accuracy on the transformed test set: {crnn_accuracy_transformed * 100:.2f}%")
```

5.2 Flask Code to Connect FrontEnd:

```
from flask import Flask, render_template, request, jsonify, flash
import os
import librosa
import numpy as np
import joblib
from music_data_generator import extract_features
from keras.models import load_model
```

```
import tensorflow as tf

from flask_sqlalchemy import SQLAlchemy

from datetime import datetime


app = Flask(__name__)

app.secret_key = '123456789'


#DATABASE

app.config['SQLALCHEMY_DATABASE_URI'] =
'mssql+pyodbc://Ravi\\SQLEXPRESS/music_contact_us?trusted_connection=yes&driver=ODBC+Driver+
17+for+SQL+Server'

db = SQLAlchemy(app)

# Define the Contact model

class Contact(db.Model):

    __tablename__ = 'contacts'  # Specify the correct table name


    id = db.Column(db.Integer, primary_key=True)

    name = db.Column(db.String(50), nullable=False)

    email = db.Column(db.String(50), nullable=False)

    message = db.Column(db.String(500), nullable=False)

    entry_time = db.Column(db.DateTime, default=datetime.utcnow)




# Load the saved CNN model
```

```
model = load_model('cnn_model.h5')

# Load the saved LabelEncoder and ColumnTransformer

label_encoder = joblib.load('label_encoder.pkl')

column_transformer = joblib.load('column_transformer.pkl')

def allowed_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() == 'wav'

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

def upload_file():

    prediction = None

    if request.method == 'POST':

        file = request.files['file']

        if file and allowed_file(file.filename):

            # Load the file directly into memory using librosa

            y, sr = librosa.load(file, sr=None)

            # Extract features

            features = extract_features(y, sr)

            # Apply the saved ColumnTransformer

            transformed_features = column_transformer.transform(features)

            # Make the prediction using the loaded model

            predicted_class = model.predict(transformed_features)

            y_pred = tf.argmax(predicted_class, axis=1).numpy()

            class_label = label_encoder.inverse_transform(y_pred)
```

```
            # Display the predicted class on the webpage
            prediction = (class_label[0]).capitalize()
    return render_template('upload.html', prediction=prediction)
@app.route('/contact', methods=['GET', 'POST'])
def contact():
    return render_template('contact_us.html')
@app.route('/submit_form', methods=['POST'])
def submit_form():
    flash('Form submitted successfully!', 'success')
    data = {
        'name': request.form['name'],
        'email': request.form['email'],
        'message': request.form['message']
    }
    print("Received data:", data)
    new_contact = Contact(name=data['name'], email=data['email'], message=data['message'])
    db.session.add(new_contact)
    db.session.commit()
    return render_template('contact_us.html')
@app.route('/about', methods=['GET', 'POST'])
def about():
    return render_template('about.html')
```

```
@app.route('/help', methods=['GET', 'POST'])
def help():
    return render_template('help.html')
if __name__ == '__main__':
    app.run(debug=True)
```

6. RESULT ANALYSIS

The model is built using our own CNN with the following layers: Input Layer, Dense layer, and flatten the layer with activation function called ‘softmax’. Finally the model has given accuracy of 93.0% for the 3_Sec_feature_dataset.

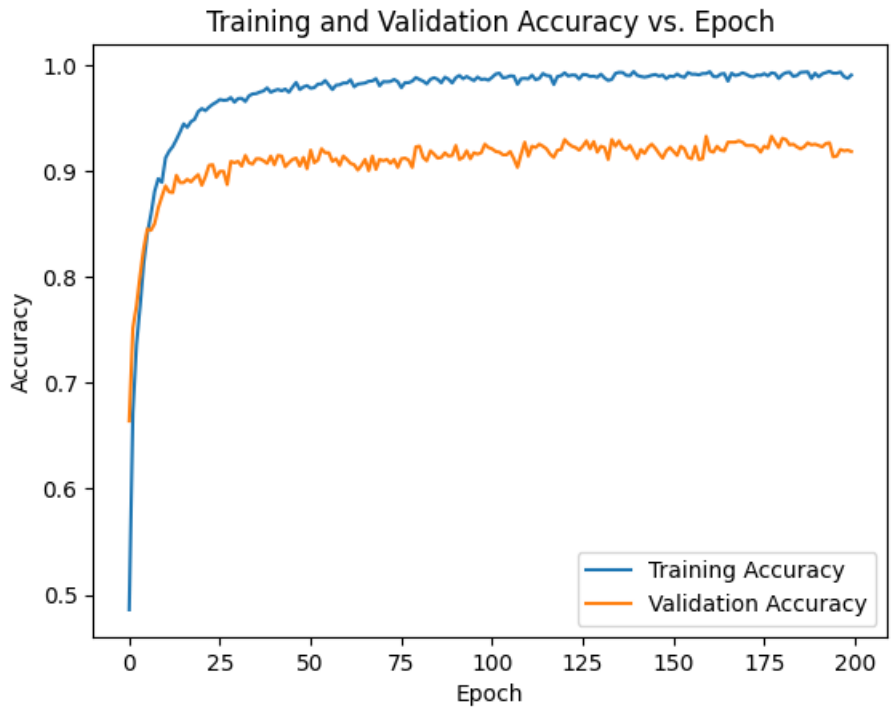


Fig 6.1 Training and Validation Accuracy vs Epoch graph

Figure 6.1 shows the comparison of Training and Validation Accuracy vs Epoch graph and the training accuracy is 97% and validation accuracy is 93% for the 3_sec_feature dataset.

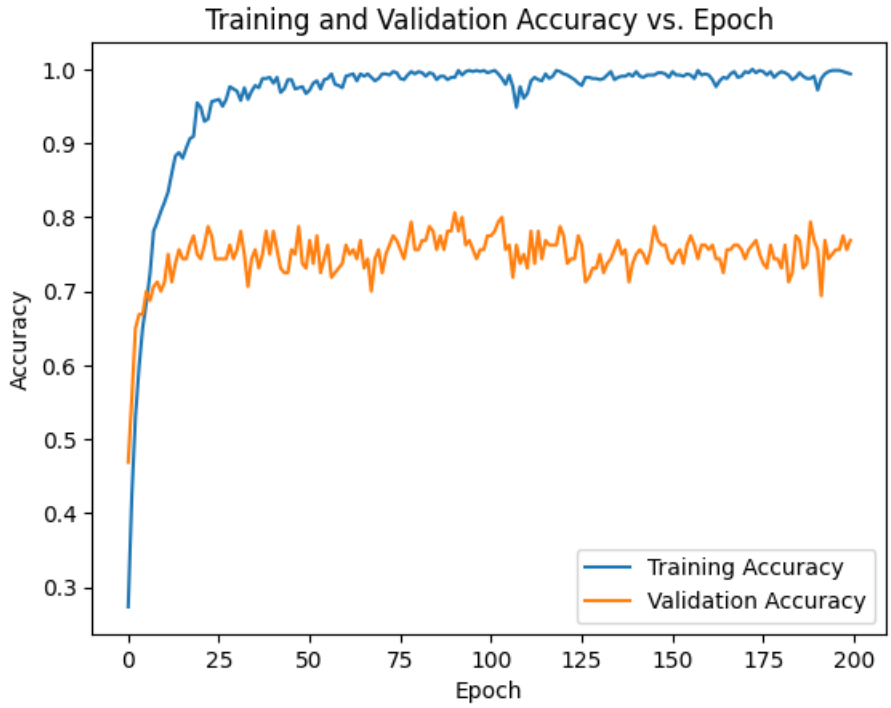
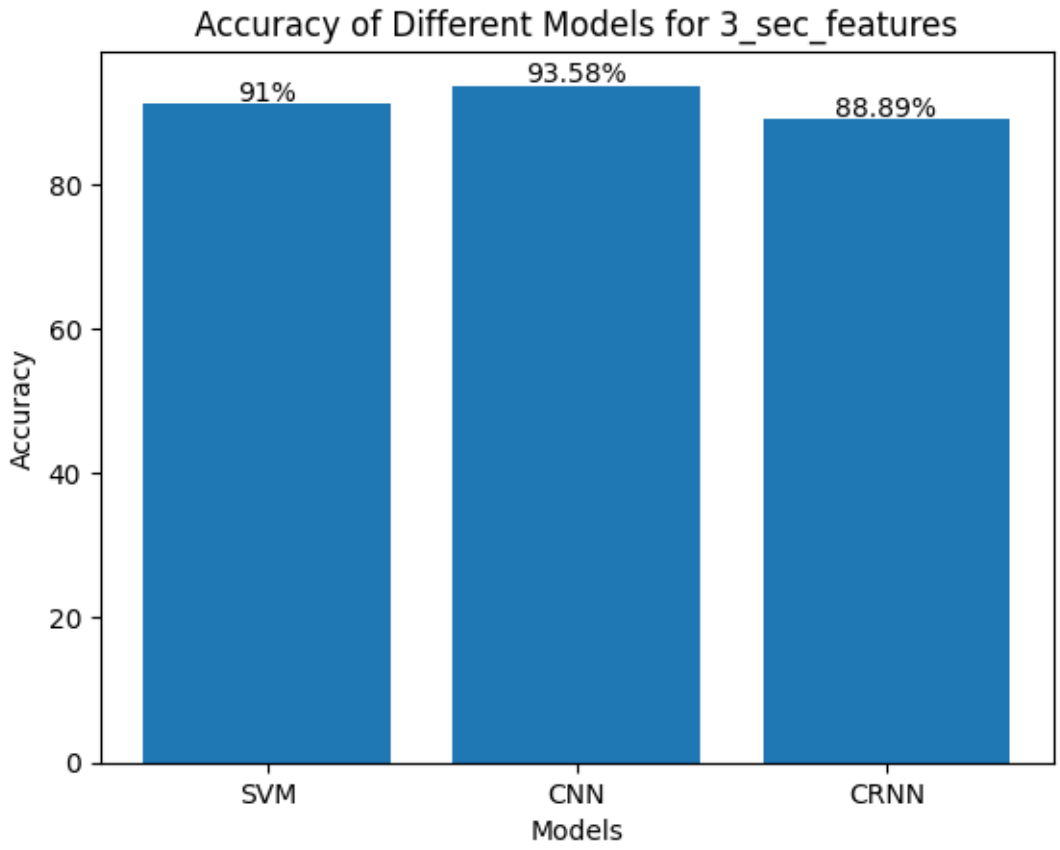


Fig 6.2 Training and Validation Accuracy vs Epoch graph

Figure 6.2 shows the comparison of Training and Validation Accuracy vs Epoch graph and the training accuracy is 97% and validation accuracy is 75% for the 30_sec_feature dataset.



.

Fig 6.3 Comparison of accuracy of different models for 3_sec_feature

Figure 6.3 shows the Comparison of accuracy of different models for 3_sec_feature and the highest accuracy is attained by CNN which is 93.58%.

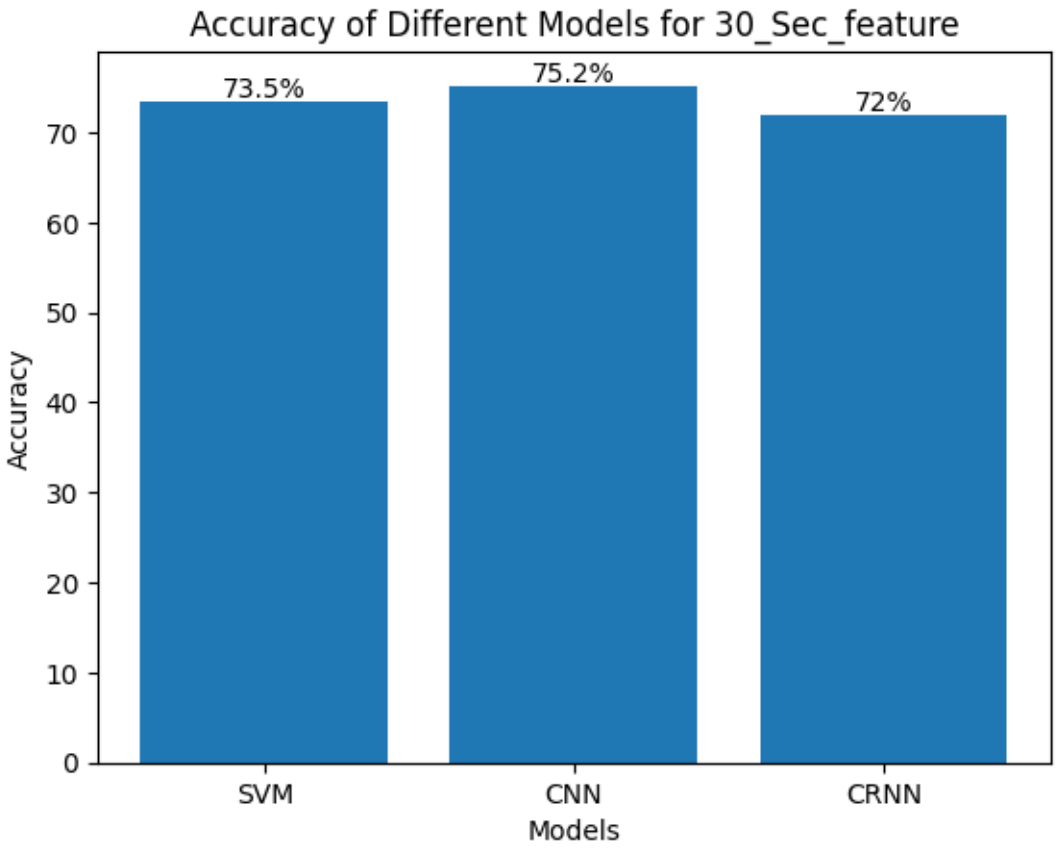


Fig 6.4 Comparison of accuracy of different models for 30_sec_feature

Figure 6.4 shows the Comparison of accuracy of different models for 30_sec_feature and the highest accuracy is attained by CNN which is 75.2%.

7. SCREEN SHOTS

HOME PAGE:

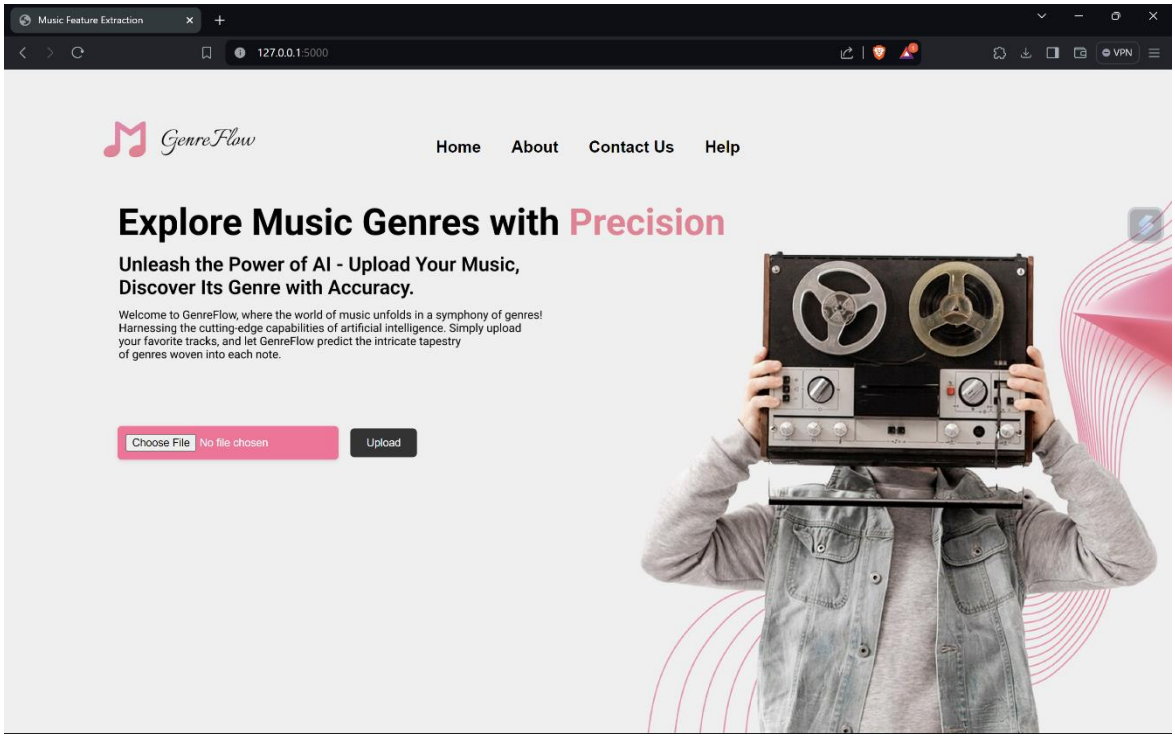


Fig 7.1 Output screen for Home page

MUSIC GENRE PREDICTION PAGE:

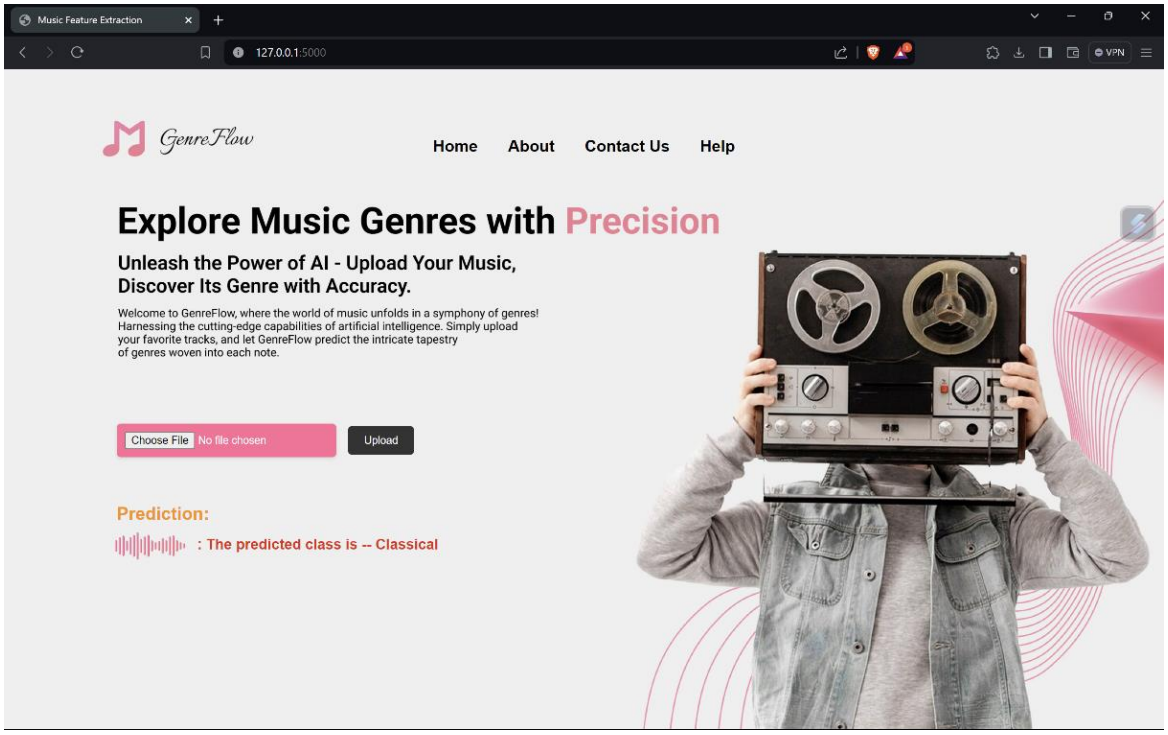
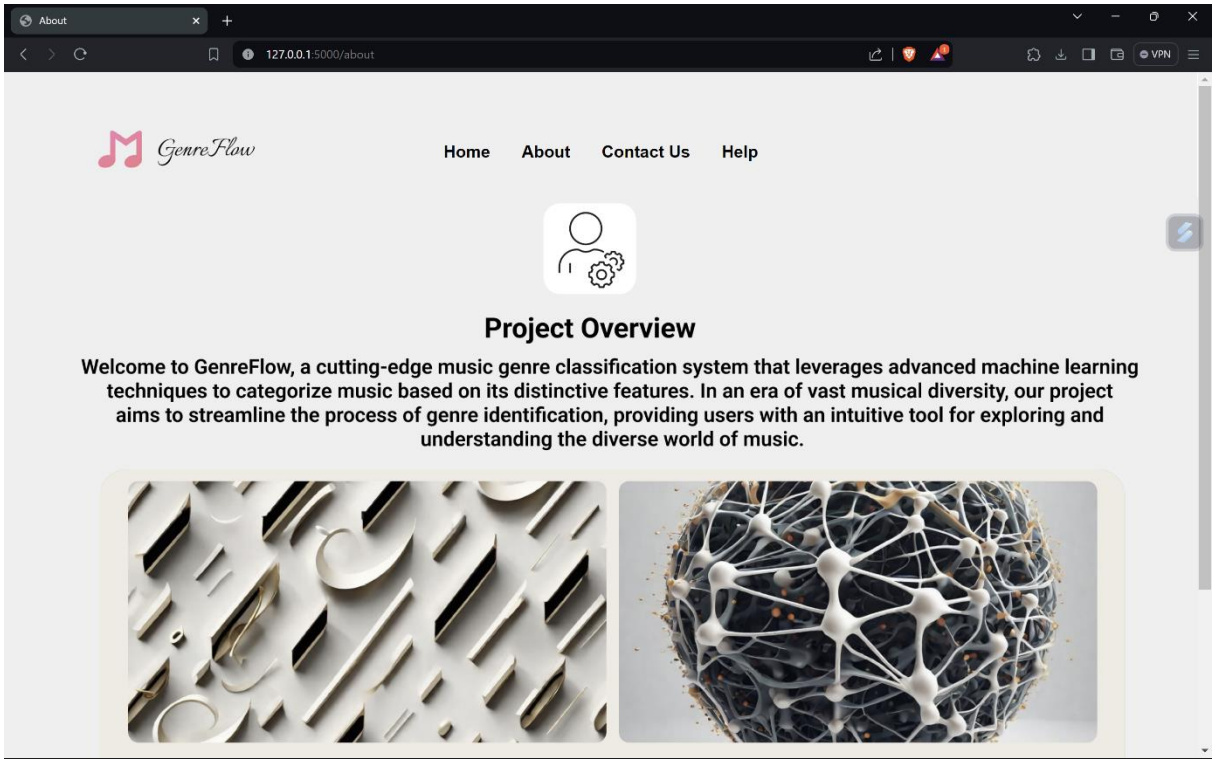


Fig 7.2 Output screen for Music Genre Prediction

ABOUT PAGE:



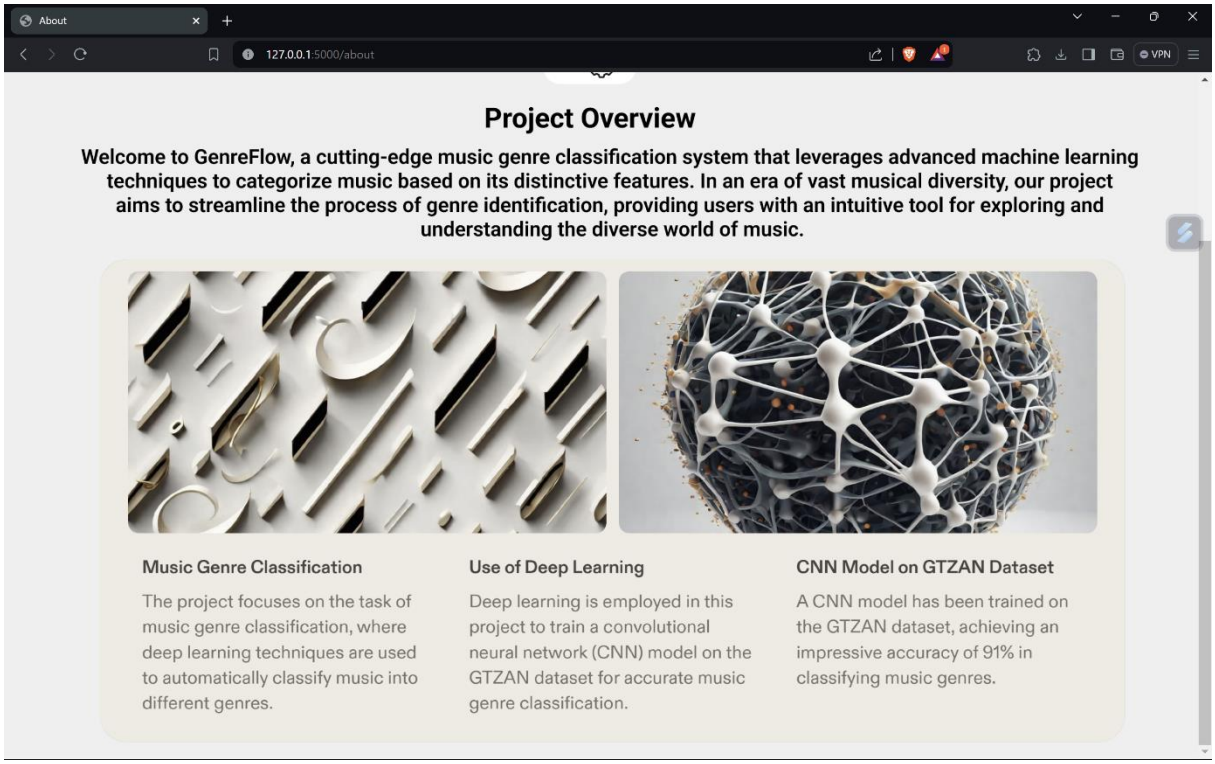


Fig 7.3 Output screen for About page

CONTACT_US PAGE:

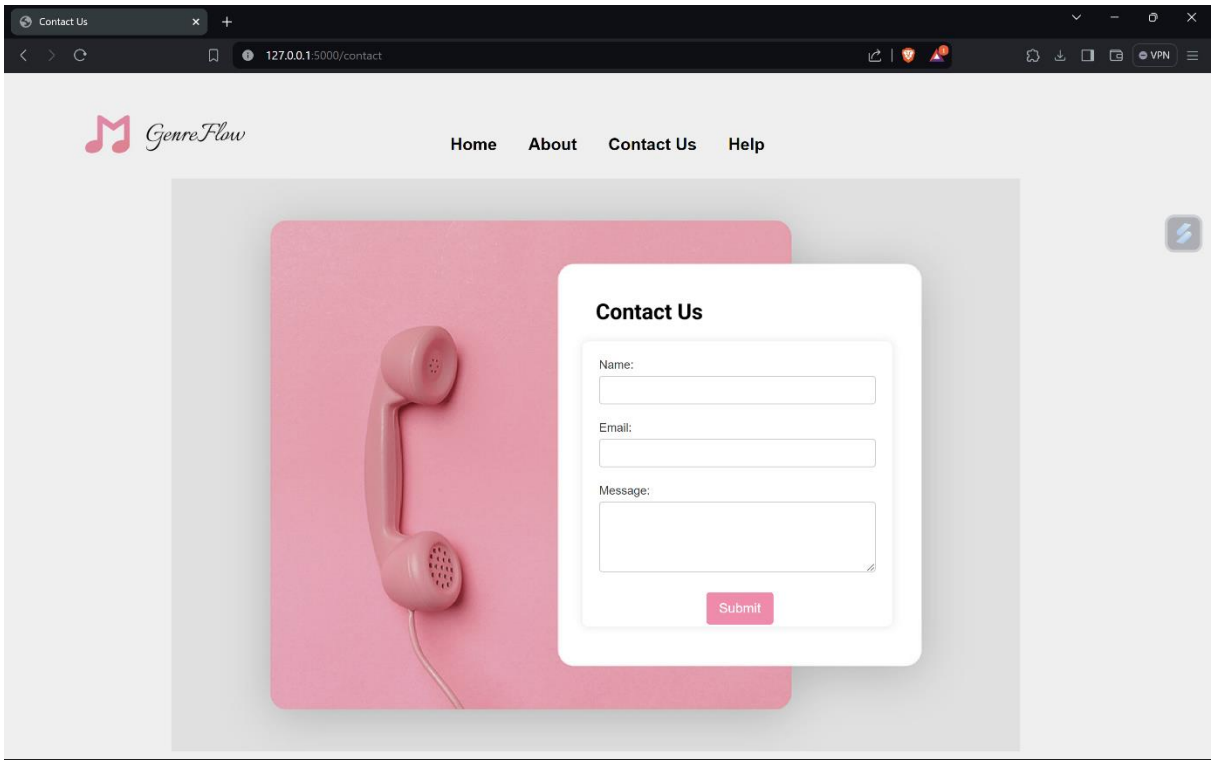
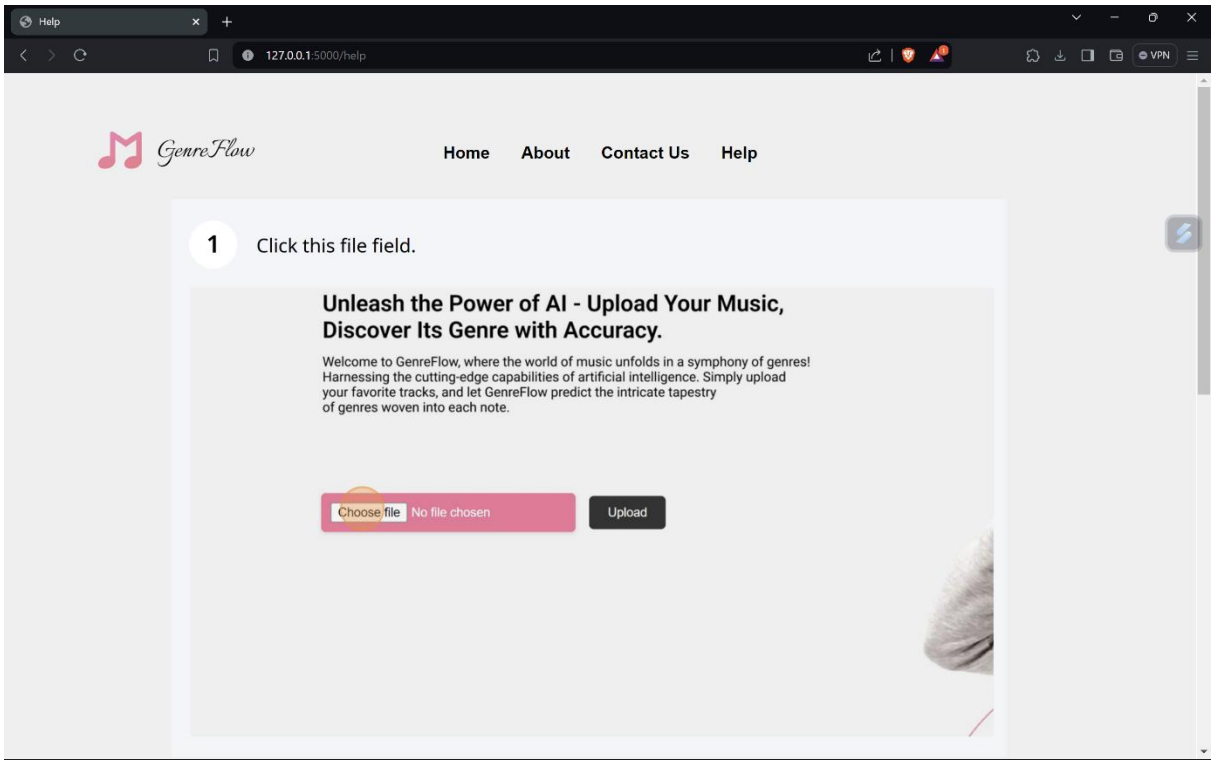


Fig 7.4 Output screen for Contact_us page

HELP PAGE:



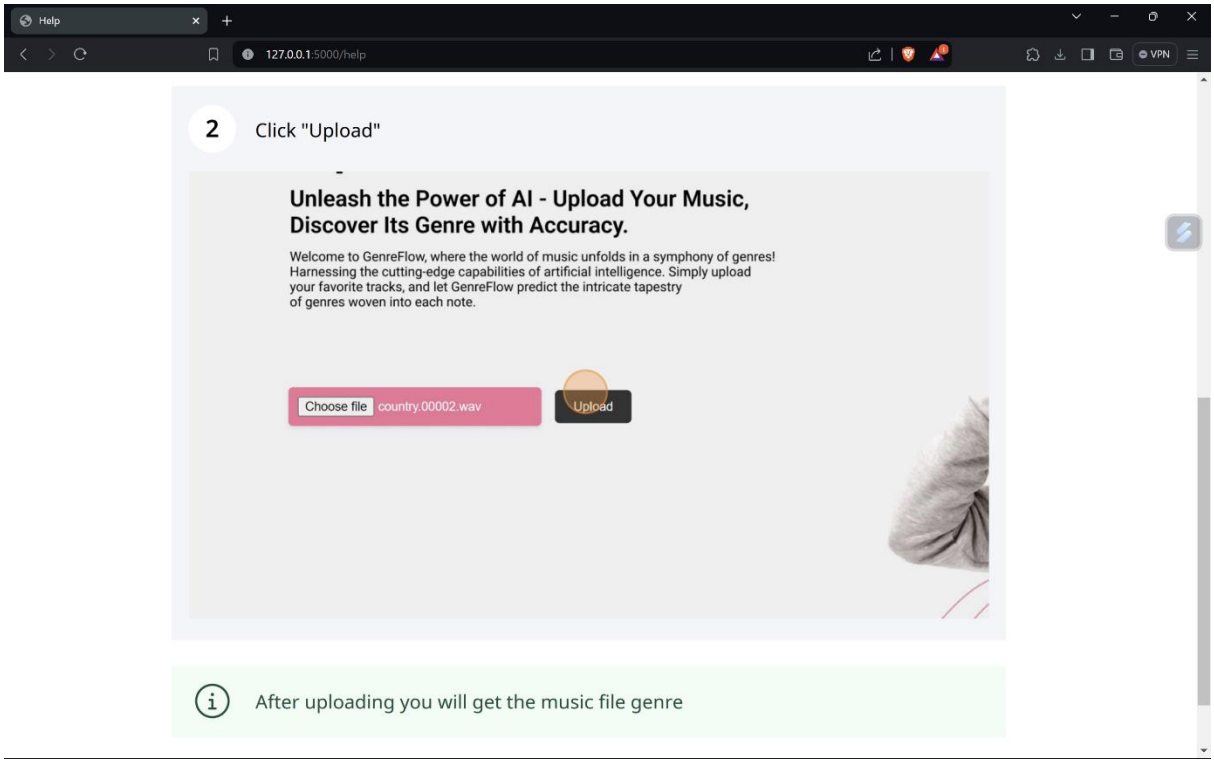


Fig 7.5 Output screen for Help page

8. CONCLUSION

In conclusion, our investigation into music genre classification using Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Convolutional Recurrent Neural Networks (CRNN) has provided valuable insights into the effectiveness of different techniques in this domain. The results highlight the robustness of SVM, achieving a commendable accuracy of 91%, while also showcasing the superior performance of deep learning models. CNN, in particular, demonstrated an impressive accuracy of 93%, emphasizing its strong feature extraction capabilities. Moreover, the study underscores the importance of capturing both spatial and temporal data in music genre classification tasks. CNN's success further reinforces the significance of leveraging deep learning techniques for such tasks, as they excel in extracting intricate patterns and features from complex data. Additionally, the findings emphasize the critical roles of meticulous feature engineering, model tuning, and data curation in achieving high classification accuracies. These aspects are pivotal in optimizing the performance of machine learning and deep learning models for music genre classification.

9. FUTURE SCOPE

In the future, music genre classification holds promise for further advancements through the exploration of hybrid approaches combining traditional machine learning and deep learning techniques. Additionally, there is potential for the development of specialized deep learning architectures tailored for music data, alongside the integration of multimodal information for enhanced classification accuracy. Further research into transfer learning methods and interdisciplinary collaborations are also avenues for accelerating progress in this field. By embracing these opportunities, researchers can continue to refine classification algorithms and expand the capabilities of music genre classification systems.

10. REFERENCES

[1] L. Sturm, “On music sort classification by means of compressive sampling,” in 2013 IEEE Universal Conference on Mixed media and Expo(ICME), 2013, pp. 1–6.

[2] E. Benetos and C. Kotropoulos, “A tensor-based approach for programmed music sort classification,” in 2008 16th European Flag Preparing Conference, 2008, pp. 1–4.

[3] K. Choi, G. Fazekas, and M. Sandler, “Explaining profound convolutional neural systems on music classification,” 2016.

[4] Music Genre Classification: A review of Deep-Learning and Traditional Machine-Learning Approaches. (2021, April 21).

[5] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, “Aggregate highlights and adaboost for music classification,” Machine Learning, vol. 65, pp. 473–484, 12 2006

[6] I. Fujinaga, “Adaptive optical music recognition,” Ph.D. paper, McGill College, CAN, 1997, aAINQ29937.

[7] H.Bahuleyan, “Music sort classification utilizing machine learning techniques,” 2018.

[8] D. S. Lau and R. Ajoodha, “Music class classification:A comparative consider between deep-learning and conventional machine learning approaches,” in 6th Worldwide Congress on Data and Communication Innovation (6th ICICT). Springer, 2021, pp. 1–8.

[9] J. H. Lee and J. S. Downie, “Survey of music data needs, employments, and looking for practices: preparatory findings.” in ISMIR, vol. 2004. Citeseer, 2004, p. 5th.

[10] A. Lerch, An Presentation to Sound Substance Examination: Applications in Flag Handling and Music Informatics. Wiley Online Library, 10 2012.

[11] T. Li, M. Ogihara, and Q. Li, “A comparative think about on content- based music class classification,” in Procedures of the 26th Yearly Universal ACM SIGIR Conference on Inquire about and Improvement in Informaion Recovery, ser. SIGIR '03. Unused York, NY, USA: Affiliation for Computing Apparatus, 2003,p.282– [Online].Available:https://doi.org/10.1145/860435.860487

[12] T. Lidy, A. Rauber, A. Pertusa, and J. Iñesta, “Combining sound and typical descriptors for music classification from audio,” 2007.

[13] C. McKay, R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga, “Ace: A system for optimizing music classification,” in ISMIR, 2005.

[14] C. McKay and I. Fujinaga, “Musical class classification: Is it worth seeking after and how can it be improved?” in ISMIR, 2006.

[15] T. Nkambule and R. Ajoodha, “Classification of music by class utilizing probabilistic graphical models and profound learning models,” in 6th Worldwide Congress on Data and Communication Innovation (6th ICICT). Springer, 2021, pp. 1–6.

[16] R. Ajoodha, R. Klein, and B. Rosman, “Single-labelled music class classification utilizing content-based features,” in 2015 Design Acknowledgment Affiliation of South Africa and Mechanical autonomy and Mechatronics Universal Conference (PRASA- RobMech), 2015, pp. 66–71.

[17] R. Ajoodha, R. Klein, and M. Jakovljevic, “Using measurable models and developmental calculations in algorithmic music composition,” in Reference book of Data Science and Innovation, Third Version. IGI Worldwide, 2015, pp. 6050–6062.

[18] A. Olteanu. Gtzan dataset - music sort classification. [Online]. Available: https://www.kaggle.com/andradaolteanu/gtzan-dataset- music-genre-classification.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Pretten hofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, “Scikit- learn: Machine learning in python,” 2018

Music Genre Classification using Deep Learning

Shaik Rafi[1], K.Ravi Varma[2], M.Kodanda Rami Reddy[3], Shaik Madire Ashaduddeen [4]

[1] Professor, [2, 3 & 4] Student

[1] shaikrafinrt@gmail.com, [2] 57varma@gmail.com, [3] kodandaramireddymirla@gmail.com, [4] ashaduddeen2230@gmail.com

Department of Computer Science and Engineering,

Narasaraopeta Engineering College, Narasaraopet, Andhra Pradesh, India

***Abstract*—Music genre classification is a pivotal area of research within audio technology, holding immense importance for content organization and recommendation. Audio feature extraction and Music genre classification constitute a complete recognition system. Audio feature analysis and Music genre classification together form an integrated recognition system for comprehensive music genre identification and organization. This technology is frequently utilized to accurately detect and classify various types of music genres or characteristics present in audio signals, contributing significantly to the effective organization and recommendation of music content. Our experiment was conducted with the dataset from GTZAN that is taken from Kaggle repository. Convolutional neural networks (CNN) are employed to train our model, which is subsequently utilized for the classification of music genres in audio signals.**

Keywords— Music, Genre, Deep Learning, Support Vector Machine, CNN, CRNN

I. INTRODUCTION

Genre plays a crucial role in distinguishing between music pieces, though it can be influenced by personal biases. Despite diverse interpretations of genre on a global scale, the rise of digital platforms highlights the potential advantages of implementing automated music classification. This research delves into the world of automatic music genre classification, seeking to showcase the power of machine learning and deep learning techniques in accurately categorizing songs based on audio signals[1]. This automation has the potential to significantly decrease search times within the vast music databases commonly found on digital platforms[6].

Our study delves into the comparison of traditional machine-learning models, such as Support Vector Machines (SVM), and advanced deep-learning models like Convolutional Neural Networks (CNN) and Convolutional Recurrent Neural Networks (CRNN). By conducting this analysis, we aim to uncover the strengths and limitations of each method in the realm of music genre classification. Furthermore, we expand our investigation to evaluate the effectiveness of machine-learning classifiers using both three-second and thirty-second duration features[3].

This study adds valuable insight into the potential of these models and the importance of feature duration, further fueling the conversation on utilizing machine learning and deep learning to achieve more efficient and precise music genre classification[2]. This has the power to benefit all those within the ever-evolving world of digital music platforms.

To this approach of making the research to held the performance of a music works in deep learning according to the dataset

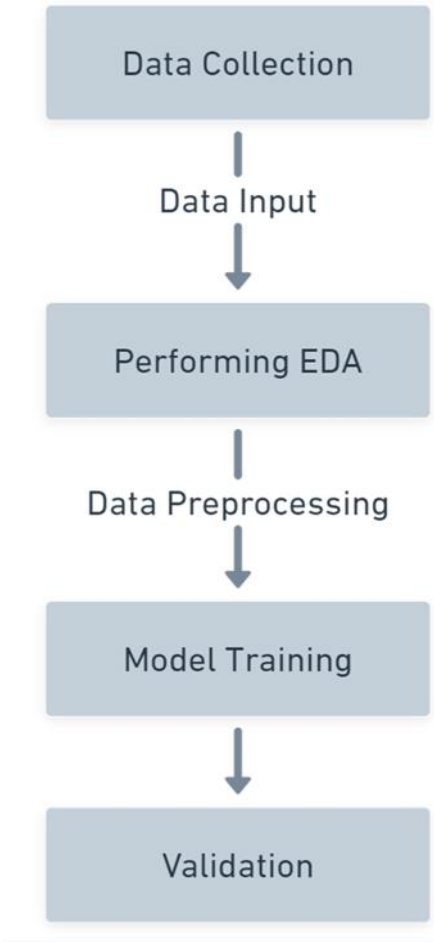


Fig-1 Steps involved in a Model

II. LITERATURE SURVEY

The examination of music sort classification has earned critical consideration in inquire about circles, where different techniques have been investigated to progress exactness and effectiveness. A few essential considers have contributed to this field. Analysts have dug into different approaches to refine the classification handle, pointing to way better get it and categorize melodic sorts.

Here are a few striking ponders within the domain of music sort classification:

Sturm et al.[1] proposed a music sort classification strategy utilizing scanty representation, accomplishing 83.00% accuracy on the GTZAN dataset. He extricated MFCC highlights, built genre-specific word references, and sp track utilizing iotas from its coordinating word reference.

This approach viably utilized the discriminative control of genre-specific timbral characteristics. Benetos et al.[2] investigated music sort classification through non-negative tensor factorization (NTF), coming to 75.00% accuracy on GTZAN. They spoken to music recordings as include tensors and created a novel NTF calculation. Sort classification was accomplished by breaking down the tensor into components capturing genre-specific designs and employing a administered classifier. Whereas not the beat entertainer at the time, this approach illustrated the potential of NTF for class classification and cleared the way for encourage headways in this region.

Bahuleyan et al.[7] displayed a strategy utilizing Convolutional Neural Systems (CNNs) for sound classification on the Sound Set dataset, accomplishing 65.00% accuracy. This approach included preparing a profound CNN design on a enormous collection of labeled sound recordings, permitting it to memorize complex designs specifically from the sound information. Whereas not particularly centered on music sort classification, this work showcased the potential of CNNs for sound assignments and propelled encourage investigate into their application for music class recognizable proof. It's critical to note that Audio Set could be a broader dataset enveloping different sounds past music sorts, making coordinate comparison with genre-specific datasets like GTZAN less direct.

III. PROPOSED SYSTEM

Our Model is Proposed based on certain criteria as follows.

- Dataset Analysis
- Preprocessing Techniques
- Model Creation
- Result and Analysis

A. Dataset Analysis

The dataset utilized in this venture was sourced from Kaggle, a stage facilitating differing datasets from the web. Particularly, the GTZAN dataset, famous for its part in music sort classification inquire about, shapes the establishment of our examination. The GTZAN dataset comprises 1000 melodic selections, each enduring thirty seconds. fastidiously categorized dataset envelops 10 unmistakable classes, with 100 pieces distributed to each class. In expansion to the initial dataset, a partitioned ponder was conducted to improve preparing comprehensiveness. The dataset was reproduced and subdivided into 10,000 passages, each enduring three seconds. Whereas this methodology expanded the accessible preparing information, it presented a certain degree of lopsidedness among the classes. A few classes finished up containing slightly more or less than the required 1000 pieces.

The estimate and composition of the dataset are vital for understanding its scope. The first GTZAN dataset comprises of 1000 rows, equitably dispersed over 10 classes. The amplified dataset, made by imitating and subdividing the initial, contains 10,000 rows[7]. and professionals working with the GTZAN dataset within the context of music class classification thinks about

In the following table, the columns of the GTZAN dataset are presented, and all columns except 'file name' and 'length' are utilized in the training process.

B. Preprocessing Techniques

Compelling pre-processing procedures are necessarily to planning input information for a music sort classification show, particularly given the significant measure of our dataset. We utilize a assorted set of pre-processing strategies to handle challenges such as exceptions and clamor inborn in music information. Different strategies, counting name encoding and coding strategies like Power Transformer, StandardAero, and Min-Max Scaler, are connected to make a comprehensive technique for dealing with both categorical and numerical highlights.

The basis behind utilizing these pre-processing strategies lies in their capacity to improve the vigor and execution of the music sort classification demonstrate. Name encoding guarantees that categorical highlights are fittingly changed over into numerical arrange, encouraging the compatibility of information with machine learning calculations. In the interim, procedures like PowerTransformer, StandardScaler, and MinMaxScaler play a vital part in normalizing and scaling numerical highlights, tending to issues related to changing scales and disseminations.

Particular consideration is coordinated towards refining key highlights such as 'chroma_stft,' 'spectral_centroid,' and 'tempo.' Methods like PowerTransformer with Yeo-Johnson, StandardScaler, and MinMaxScaler are deliberately connected to guarantee compelling normalization and scaling for these highlights. This fastidious approach points to relieve potential challenges like predisposition and overfitting, contributing to the by and large vigor of the show. The consolidation of a ColumnTransformer advance upgrades the pre-processing pipeline by adeptly isolating categorical and numerical highlights. This isolation empowers custom fitted pre-processing for each sort of include, optimizing the planning of information for consequent stages within the classification demonstrate.

In substance, the intensive and comprehensive pre-processing methodologies utilized in this ponder are adapted towards ensuring that the input information is well- conditioned, minimizing commotion and exceptions, and maximizing the model's capacity to memorize important designs from the music class dataset.

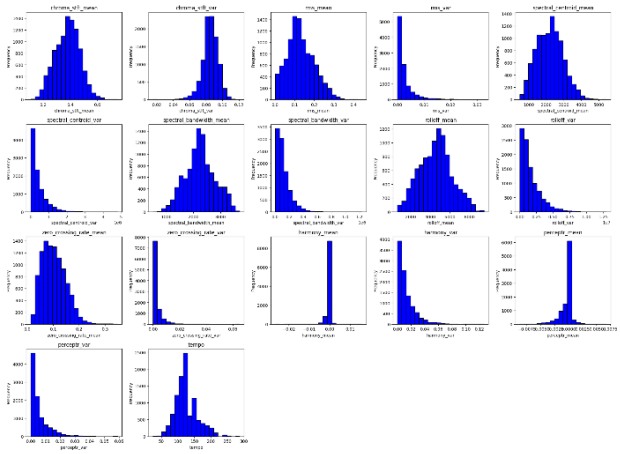


Fig-2 Distribution before Preprocessing

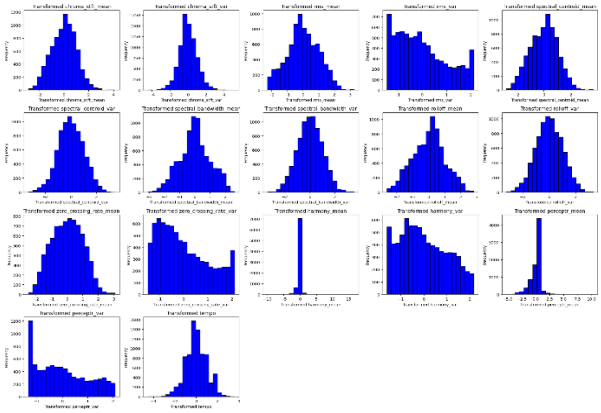


Fig-3 Distribution After Preprocessing

In Figure 2, the distribution of the dataset prior to preprocessing is visually represented, providing a snapshot of the initial data distribution. Notably, the distribution exhibits skewness, indicating an uneven spread across different classes or categories.

However, in Figure 3, we observe the distribution after the application of preprocessing techniques. The transformation is evident, with the skewed distribution evolving into a more normalized form. This shift towards a normal distribution is a significant achievement.

C. Model Creation

In the quest for precise music genre classification, three distinct models were developed: Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Convolutional Recurrent Neural Network (CRNN). Each model was carefully designed to exploit specific strengths in handling diverse audio features. SVM, a traditional machine learning algorithm, excels in capturing intricate decision boundaries. The CNN model utilizes convolutional.

1) SVM

We utilized the Support Vector Machines (SVM) algorithm as the cornerstone of our machine-learning model, implementing it through the Scikit-Learn library. To fine- tune the SVM model, an exhaustive search for optimal hyperparameters was conducted, resulting in the selection of the following values:

'C': 42

'gamma': 1.52

'kernel': 'poly'

These specific hyperparameter choices were determined to deliver optimal performance within our experimental setup. In the subsequent sections, we will delve into comprehensive descriptions of our SVM models, covering architectural details, hyperparameter tuning specifics, and intricacies of the training process.

2) CNN

The Convolutional Neural Network (CNN) architecture in this research was constructed using Keras. The CNN built here has an input layer and five convolutional blocks, with each convolutional block consisting of the following:

- Dense Layers:

Four dense layers with 512, 256, 128, and 64 units respectively

Each followed by ReLU activation and 0.2 dropout

- Training Configuration:

Optimizer: Adam

Training Epochs: 200

3) CRNN

The Convolutional Recurrent Neural Network (CRNN) architecture in this research was constructed using Keras. The CRNN model built here has an input layer and five repeating blocks, with each CRNN block consisting of the following:

- Convolutional Layers:

Conv1D layer with 64 filters, 3 kernel size, and ReLU activation

MaxPooling1D layer with a pool size of 2

- Dense Layers:

Four dense layers with 512, 256, 128, and 64 units respectively

Each followed by ReLU activation and 0.2 dropout

- Compilation and Training Configuration:

Optimizer: Adam

Training Epochs: 200

IV. RESULT AND ANALYSIS

In this section, we delve into the outcomes and insights gained from the three distinct models employed in this research: Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Convolutional Recurrent Neural Network (CRNN). Each model brings its unique approach to the task of classifying music into ten GTZAN genres. The ensuing analysis sheds light on their respective performances, revealing patterns, strengths, and areas for improvement.

The below figure-4 presents the confusion matrix attained when classifying music into ten GTZAN genres using Convolutional Neural Network model of this research. The model seems to perform well in general, with high correct classifications for most genres.

However for Classes Country (Class 2) and Jazz (Class 5) have relatively lower correct classifications compared to other genres.

The model often confuses Jazz (Class 5) with Blues (Class 0) and Classical (Class 1)

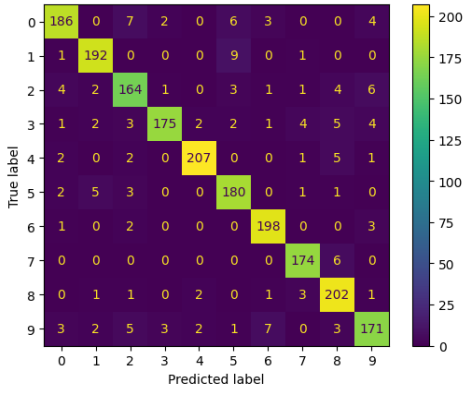


Fig 4 Confusion Matrix of CNN

The below figure-5 presents the confusion matrix attained when classifying music into ten GTZAN genres using Convolutional Recurrent Neural Network model of this research.

However the model misclassifies some of genres like metal (Class 6) ,Country (Class 2) ,Jazz (Class 5)

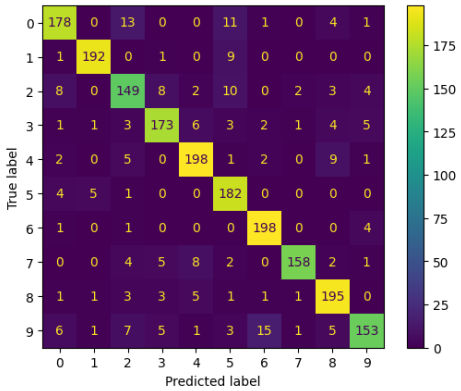


Fig 5 Confusion Matrix of CRNN

The Label Mapping: blues: 0,classical: 1,country: 2, disco: 3, hiphop: 4,jazz: 5,metal: 6, pop: 7, reggae: 8, rock: 9

Accuracy serves as a widely employed metric for assessing the efficacy of a machine learning algorithm. It quantifies the ratio of accurately classified instances relative to the entirety of instances within the test dataset.

TABLE I. ACCURACY OF EXISTING SYSTEM[4]

Classifier	**Epochs**	**Accuracy**
CNN (3-Sec Features)	50	72.4%
CNN (Spectrograms)	120	66.5%
CNN (30-Sec Features)	30	53.5%

TABLE II. ACCURACY OF DIFFERENT ALGORITHMS FOR 30_SEC_FEATURES

Classifier	**Epochs**	**Accuracy**	**Training Time(S)**
Support Vector Machines	-	73.5%	1
Convolutional Neural Network (CNN)	200	75.2%	27.5
Convolutional Recurrent Neural Network (CRNN)	200	72.0%	95.4

TABLE III. ACCURACY OF DIFFERENT ALGORITHMS FOR 3_SEC_FEATURES

Classifier	**Epochs**	**Accuracy**	**Training Time(S)**
Support Vector Machines	-	91.00%	2
Convolutional Neural Network (CNN)	200	93.58%	369
Convolutional Recurrent Neural Network (CRNN)	200	88.89%	841

In comparing the performance of our proposed models with those presented in the research paper titled "Music Genre Classification: A review of Deep-Learning and Traditional Machine-Learning Approaches" (published on April 21, 2021)[4], notable distinctions emerge across various temporal feature lengths. For the 30-second feature duration, our Support Vector Machine (SVM) model achieved an accuracy of 73.5%, slightly trailing behind the counterpart in the referenced paper, which reported 75.4%. Interestingly, our Convolutional Neural Network (CNN) model outperformed the alternative, exhibiting an accuracy of 75.2% compared to the referenced paper's CNN accuracy of 53.50%. This discrepancy underscores the efficacy of our CNN architecture in capturing temporal patterns for longer feature durations.

Shifting focus to the 3-second feature duration, our SVM model exhibited a substantial accuracy advantage with 91.0%, surpassing the accuracy of 80.8% reported in the referenced paper. Moreover, our CNN model demonstrated a significant improvement, achieving an accuracy of 93.5%, in contrast to the 72.40% accuracy recorded in the referenced paper. These results suggest that our models excel in capturing short-term temporal features, showcasing superior performance in both SVM and CNN architectures. These findings contribute valuable insights to the field, emphasizing the significance of our proposed approaches in achieving heightened accuracy for temporal feature extraction.

V. CONCLUSION AND FUTURE SCOPE

This investigation into music genre classification using SVM, CNN, and CRNN models has yielded valuable insights. The robust SVM demonstrated commendable accuracy at 91%, while the deep learning models outperformed, with CNN achieving an impressive 93% and CRNN reaching 78%. This underscores the significance of capturing both spatial and temporal data, with CNN showcasing notable feature extraction capabilities. The study reaffirms the dominance of deep learning, particularly CNNs, in this field, highlighting the critical roles of meticulous feature engineering, model tuning, and data curation for success. Looking ahead, the future of music genre classification lies in the exploration of optimal combinations of traditional and deep learning techniques to leverage their complementary strengths

REFERENCES

[1] L. Sturm, “On music sort classification by means of compressive sampling,” in 2013 IEEE Universal Conference on Mixed media and Expo(ICME), 2013, pp. 1–6.

[2] E. Benetos and C. Kotropoulos, “A tensor-based approach for programmed music sort classification,” in 2008 16th European Flag Preparing Conference, 2008, pp. 1–4.

[3] K. Choi, G. Fazekas, and M. Sandler, “Explaining profound convolutional neural systems on music classification,” 2016.

[4] Music Genre Classification: A review of Deep-Learning and Traditional Machine-Learning Approaches. (2021, April 21). IEEE Conference Publication | IEEE Xplore.https://ieeexplore.ieee.org/abstract/document/9422487

[5] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, “Aggregate highlights and adaboost for music classification,” Machine Learning, vol. 65, pp. 473–484, 12 2006

[6] I. Fujinaga, “Adaptive optical music recognition,” Ph.D. paper, McGill College, CAN, 1997, aAINQ29937.

[7] H.Bahuleyan, “Music sort classification utilizing machine learning techniques,” 2018.

[8] D. S. Lau and R. Ajoodha, “Music class classification:A comparative consider between deep-learning and conventional machine learning approaches,” in 6th Worldwide Congress on Data and Communication Innovation (6th ICICT). Springer, 2021, pp. 1–8.

[9] J. H. Lee and J. S. Downie, “Survey of music data needs, employments, and looking for practices: preparatory findings.” in ISMIR, vol. 2004. Citeseer, 2004, p. 5th.

[10] A. Lerch, An Presentation to Sound Substance Examination: Applications in Flag Handling and Music Informatics. Wiley Online Library, 10 2012.

[11] T. Li, M. Ogihara, and Q. Li, “A comparative think about on content- based music class classification,” in Procedures of the 26th Yearly Universal ACM SIGIR Conference on Inquire about and Improvement in Informaion Recovery, ser. SIGIR '03. Unused York, NY, USA: Affiliation for Computing Apparatus, 2003,p.282–[Online].Available:https://doi.org/10.1145/860435.860487

[12] T. Lidy, A. Rauber, A. Pertusa, and J. Iñesta, “Combining sound and typical descriptors for music classification from audio,” 2007.

[13] C. McKay, R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga, “Ace: A system for optimizing music classification,” in ISMIR, 2005.

[14] C. McKay and I. Fujinaga, “Musical class classification: Is it worth seeking after and how can it be improved?” in ISMIR, 2006.

[15] T. Nkambule and R. Ajoodha, “Classification of music by class utilizing probabilistic graphical models and profound learning models,” in 6th Worldwide Congress on Data and Communication Innovation (6th ICICT). Springer, 2021, pp. 1–6.

[16] R. Ajoodha, R. Klein, and B. Rosman, “Single-labelled music class classification utilizing content-based features,” in 2015 Design Acknowledgment Affiliation of South Africa and Mechanical autonomy and Mechatronics Universal Conference (PRASA-RobMech), 2015, pp. 66–71.

[17] R. Ajoodha, R. Klein, and M. Jakovljevic, “Using measurable models and developmental calculations in algorithmic music composition,” in Reference book of Data Science and Innovation, Third Version. IGI Worldwide, 2015, pp. 6050–6062.

[18] A. Olteanu. Gtzan dataset - music sort classification. [Online]. Available: https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Pretten hofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, “Scikit- learn: Machine learning in python,” 2018

CB5_Conference_paper.pdf

ORIGINALITY REPORT

9%	8%	8%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	riteshajoodha.co.za Internet Source	4%
2	git.cse.iitb.ac.in Internet Source	1%
3	Mousumi Chaudhury, Amin Karami, Mustansar Ali Ghazanfar. "Large-Scale Music Genre Analysis and Classification Using Machine Learning with Apache Spark", Electronics, 2022 Publication	1%
4	Submitted to Aston University Student Paper	1%
5	D. Pradeep, S. Deva Prasath, J. Jerome Edwin, P. Kumaravel. "Chapter 6 Fetal Cardiac Detection Using Deep Learning from Echocardiographic Image–A Survey", Springer Science and Business Media LLC, 2023 Publication	1%
6	dr.ntu.edu.sg Internet Source	1%

7	pubmed.ncbi.nlm.nih.gov Internet Source	1%
8	www.jenrs.com Internet Source	1%
9	ijsrcseit.com Internet Source	<1%
10	ijritcc.org Internet Source	<1%
11	www.mdpi.com Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		